Introduction to Robotics

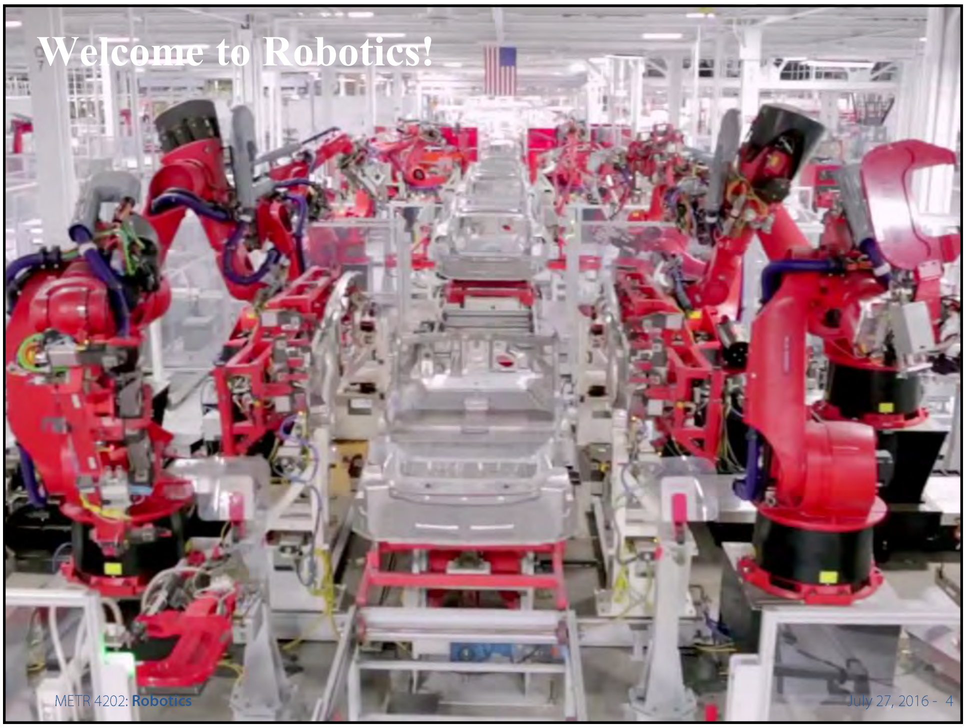
METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 1

July 27, 2016

metr4202@itee.uq.edu.au

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



METR 4202: **Robotics**
July 27, 2016 - 5

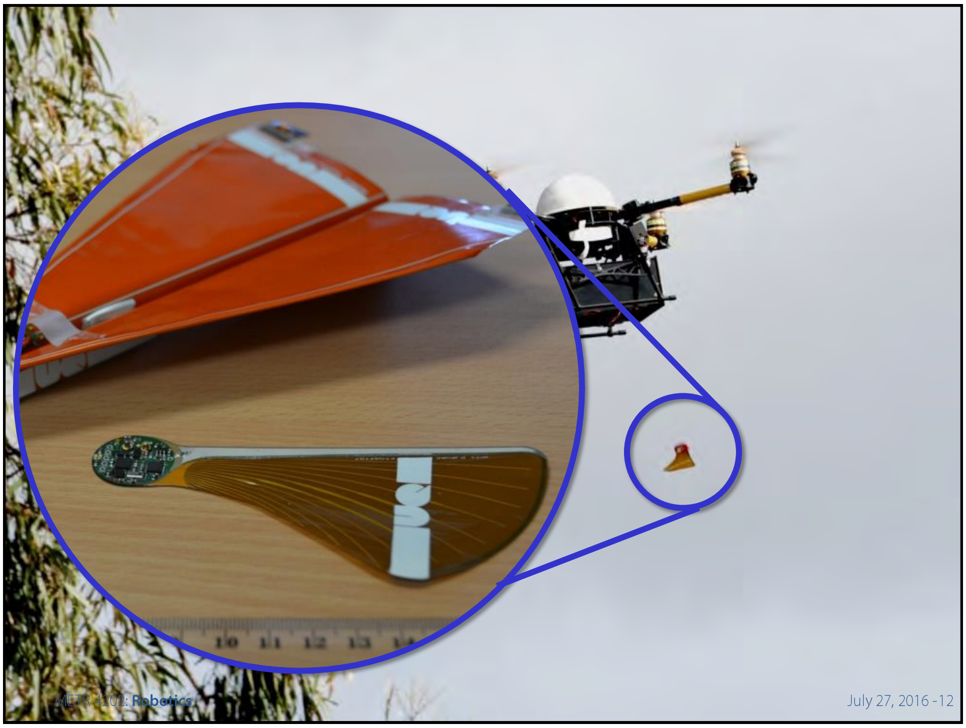
METR 4202: **Robotics**
July 27, 2016 - 6


METR 4202: Robotics
July 27, 2016 - 7


METR 4202: Robotics
July 27, 2016 - 8

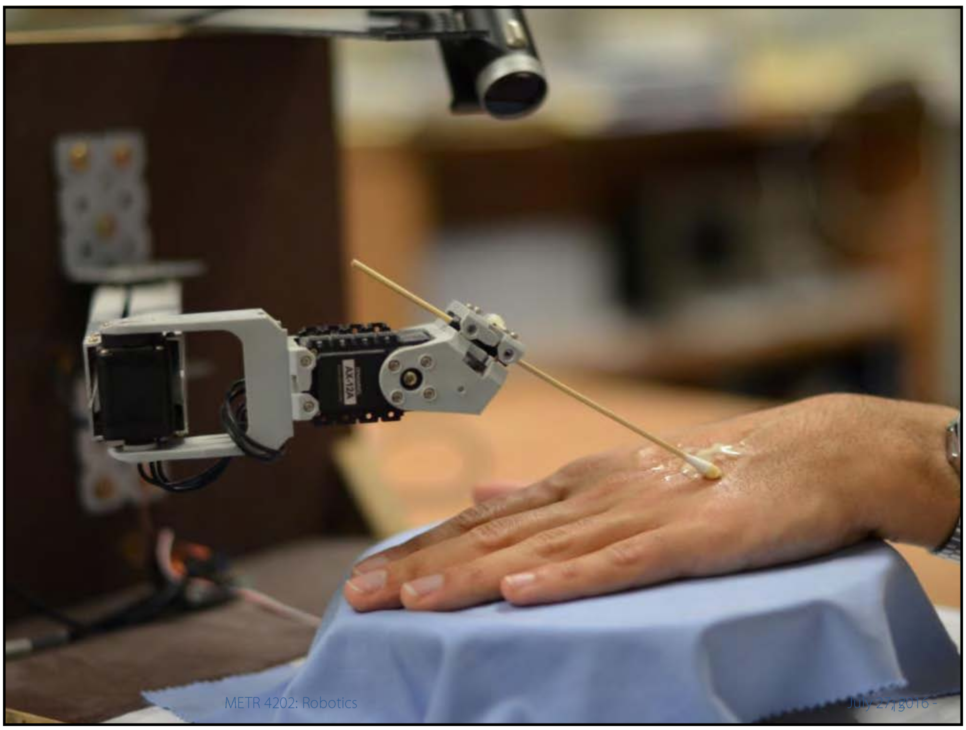

METR 4202: Robotics
July 27, 2016 - 10


METR 4202: Robotics

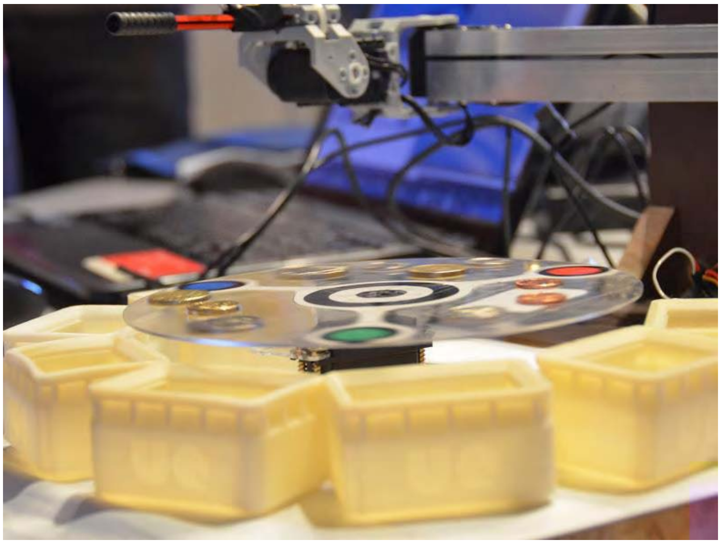

METR 4202: Robotics
July 27, 2016 - 12


METR 4202: Robotics


Magicarms
METR 4202: Robotics
July 27, 2016 - 14


AX-12A
METR 4202: Robotics

Change. The Future!



METR 4202: **Robotics** July 27, 2016 - 16

Win. The (DARPA Robotics) Challenge!

METR 4202: **Robotics** July 27, 2016 - 17

Robotics & Automation Has Limits Too

Cars: Software/Robots With 4 Wheels

So *What is a Robot* ?????

- A “Smart” Machine …
- A “General Purpose” (Adaptive) “Smart” Machine…

Plan → Sense → Control → Act

“Learning”

Robotics Definition

- Many, depends on context…

“A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.”
(Robotics Institute of America)

It is a machine which has some ability to interact with physical objects and to be given electronic programming to do a specific task or to do a whole range of tasks or actions.
(Wikipedia)

Programmable electro-mechanical systems that adapt to identify and leverage a **structural characteristic** of the environment
(Surya)

Types of Robotics Systems

- Manipulators

- Mobile

- Adaptive

Enabling Mathematics:

- Computational Kinematics
- Operational Space

- Behaviour based "Reflexive" control rules

-Probabilistic methods

Types of Robotics Systems ➔ Textbooks

- Manipulators

- Roth
- Craig
- S&S
- Asada & Slotine
- Tsai

- Mobile

- Corke
- Dillman
- Choset, Thrun, *et al.*
- [SLAM]

- Adaptive

- LaValle
- Thrun
- [[Model] **Predictive** Operations]

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	**27-Jul**	**Introduction**
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (& *Ekka Day*)
4	17-Aug	Robot Dynamics
5	24-Aug	Robot Sensing: Perception
6	31-Aug	Robot Sensing: Multiple View Geometry
7	7-Sep	Robot Sensing: Feature Detection (as Linear Observers)
8	14-Sep	Probabilistic Robotics: Localization
9	21-Sep	Probabilistic Robotics: SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	State-Space Modelling
12	19-Oct	Shaping the Dynamic Response
13	26-Oct	LQR + Course Review

Assessment

- Kinematics Lab (12.5%):
  - Proprioception
  - Arm design and operation (with Lego)
- Sensing & Control Lab (25%):
  - Exteroception
  - Camera operation and calibration (with a Kinect)
- Advanced Controls & Robotics Systems Lab (50%):
  - All together!
- **Exam** (Open-Book/closed Internet/Friends! -- 12.5%) ☺

Lectures

- Wednesdays from 12:05 – 1:50 pm

- Lectures will be posted to the course website **after** the lecture (so please attend)
  - Slides are like dessert – enjoy afterwards!

- Please ask questions (preferably about the material ☺)

METR 4202: **Robotics** July 27, 2016 - 27

Tutorials & Labs

- Labs:
  - Thursdays from 3:00 pm – 6:00 pm **xor** Mondays from 2:00 pm – 5:00 pm
  - in the Axon Learning Lab (47-104)
  - Meeting Weeks 2-9 (**not this week!**)

- Tutorials:
  - Fridays 11:00 – 11:50 am in the Axon Learning Lab (47-104)
  - Meeting: Weeks 1-13 (day after tomorrow!)

METR 4202: **Robotics** July 27, 2016 - 28

Textbook

Peter Corke

Robotics, Vision and Control

Fundamental Algorithms in MATLAB®

With 393 Images

Additional material is provided at www.petercorke.com/RVC

Springer

Robotics, Vision and Control Fundamental Algorithms in MATLAB

By:
Peter Corke

Available online (on campus) via SpingerLink

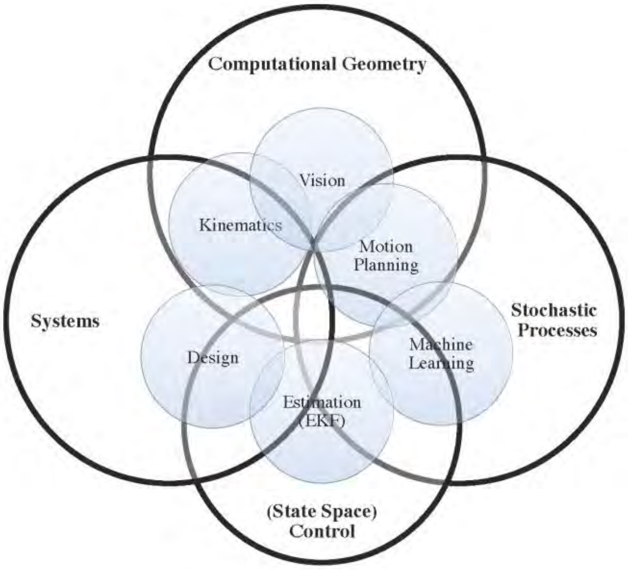
E-mail & website

metr4202 @ itee. uq . edu . au

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

Please use **metr4202** e-mail for class matters!

Course Organization



The Point of the Course

- Introduction to terminology/semantics
- An appreciation of how to frame problems in an engineering context
- Modeling and learning to trust the model
- Ability to identify critical details from the problem (separate information from trivia)

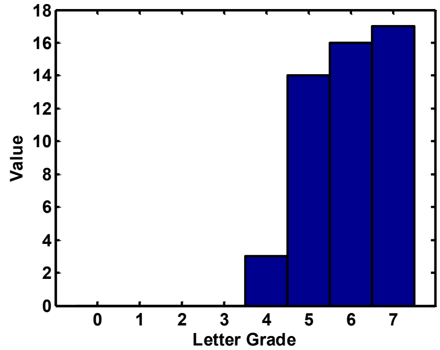
Course Objectives

1. Be familiar with sensor technologies relevant to robotic systems
2. Understand homogeneous transformations and be able to apply them to robotic systems,
3. Understand conventions used in robot kinematics and dynamics
4. Understand the dynamics of mobile robotic systems and how they are modelled
5. Understand state-space and its applications to the control of structured systems (e.g., manipulator arms)
6. Have implemented sensing and control algorithms on a practical robotic system
7. Apply a systematic approach to the design process for robotic system
8. Understand the practical application of robotic systems in to intelligent mechatronics applications (e.g., manufacturing, automobile systems and assembly systems)
9. Develop the capacity to think creatively and independently about new design problems; and,
10. Undertake independent research and analysis and to think creatively about engineering problems.

Grade Descriptors

Grade	Level	Descriptor
Fail	(<50%)	**Work not of acceptable standard**. Work may fail for any or all of the following reasons: unacceptable level of paraphrasing; irrelevance of content; presentation, grammar or structure so sloppy it cannot be understood; submitted very late without extension; not meeting the University's values with regards to academic honesty.
Pass	(50-64%)	**Work of acceptable standard**. Work meets basic requirements in terms of reading and research and demonstrates a reasonable understanding of subject matter. Able to solve relatively simple problems involving direct application of particular components of the unit of study.
Credit	(65-74%)	**Competent work**. Evidence of extensive reading and initiative in research, sound grasp of subject matter and appreciation of key issues and context. Engages critically and creatively with the question and attempts an analytical evaluation of material. Goes beyond solving of simple problems to seeing how material in different parts of the unit of study relate to each other by solving problems drawing on concepts and ideas from other parts of the unit of study.
Distinction	(75-84%)	**Work of superior standard**. Work demonstrates initiative in research, complex understanding and original analysis of subject matter and its context, both empirical and theoretical; shows critical understanding of the principles and values underlying the unit of study.
High Distinction	(85%+)	**Work of exceptional standard**. Work demonstrates initiative and ingenuity in research, pointed and critical analysis of material, thoroughness of design, and innovative interpretation of evidence. Demonstrates a comprehensive understanding of the unit of study material and its relevance in a wider context.

Last Year's Grade Statistics



- ~ 67 % received D or HD
- Worry about **learning**, not about marks [Seriously!]

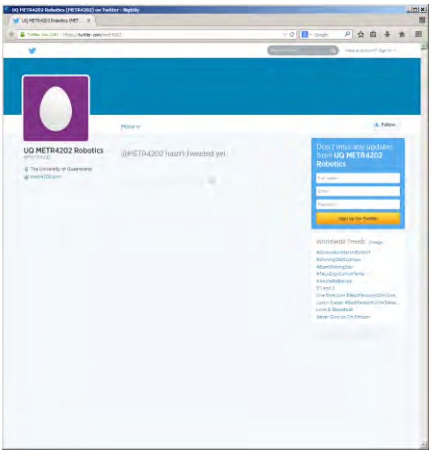
- Though a "7" might be bit more exclusive this year!

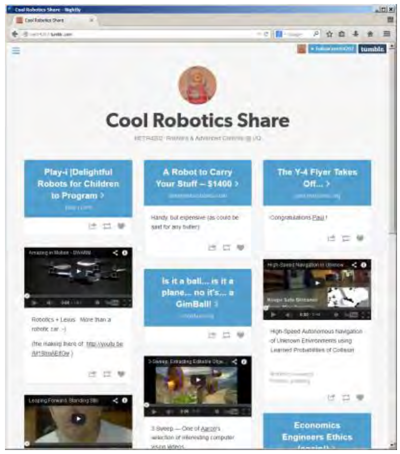
What I expect from you

- Lectures:
  - Participate - ask questions
  - Turn up (hence the attendance marks)
  - Take an interest in the material being presented
- Tutorials:
  - Work on questions before tutorials
  - Use tutorials to clarify and enhance
  - Assignments to be submitted on time

Twitter & Tumblr too!

- https://twitter.com/metr4202
- http://metr4202.tumblr.com/





What's the Magic?

Structure!

(And Some Clever Mechatronics Design)

Robotics: Exploiting the hidden structure…

- Robot working in an "unstructured" environment

➔ Does not have to be dirty to use "field robotics" technology …

➔ Robotics is about exploiting the **structure** …

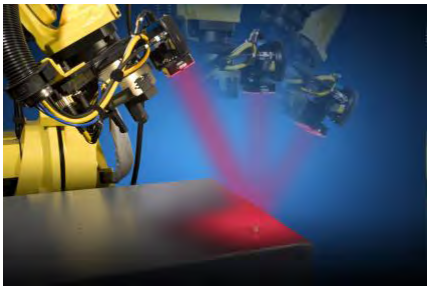
Either by:

- Putting it in from the design (mechanical structure)
- "Learning" it as the system progresses (structure is the data!)

First Let's Review the
Sense → Control → Act
Loop!

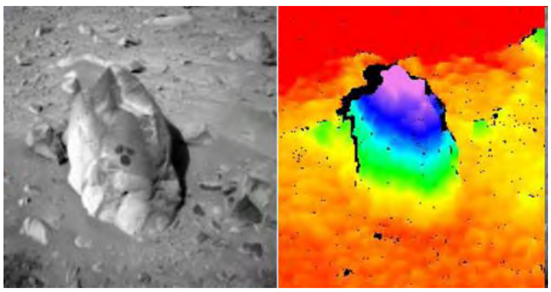
Sensing

Perception: Vision



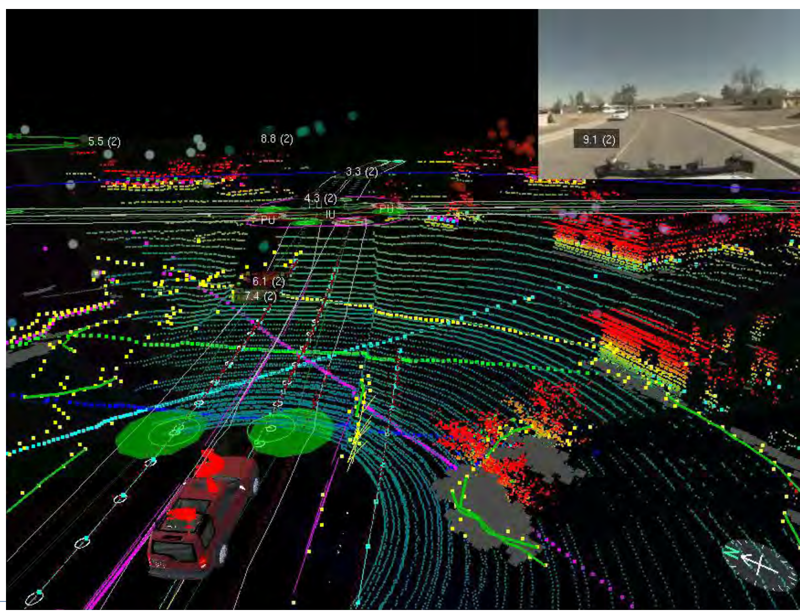
Edges, Segments, Colour, Texture

3D Stereo Vision



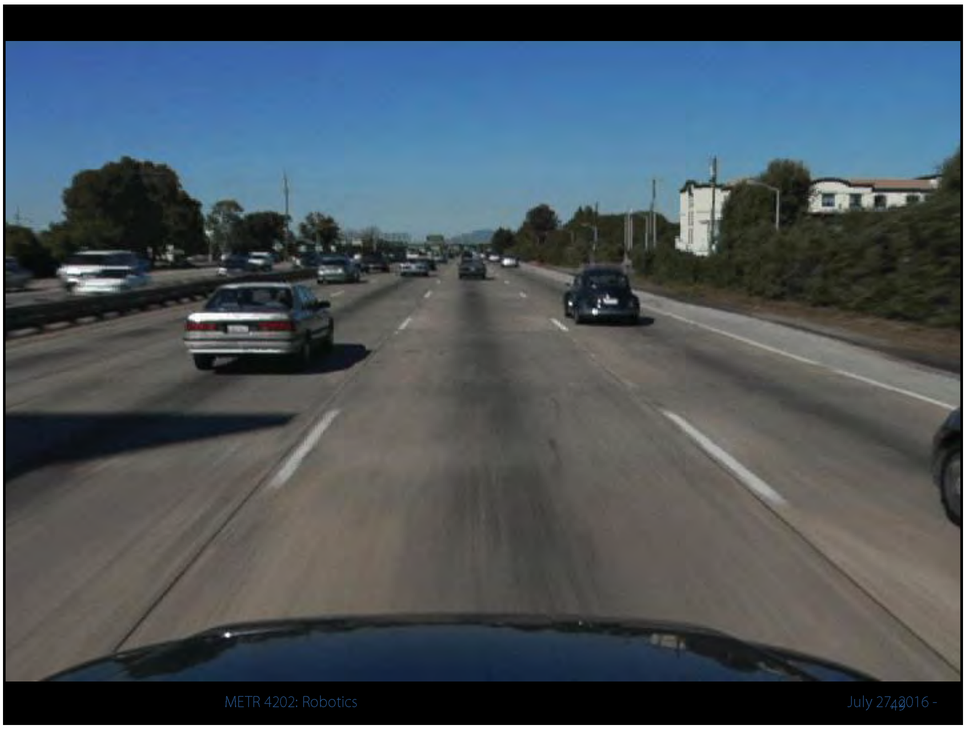
METR 4202: **Robotics** July 27, 2016 -46

Laser Sensors

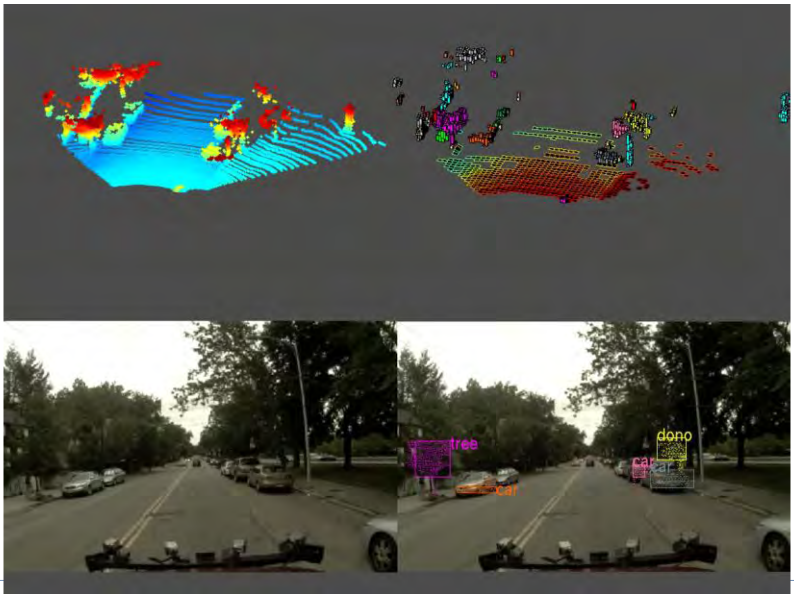


METR 4202: **Robotics** July 27, 2016 -47

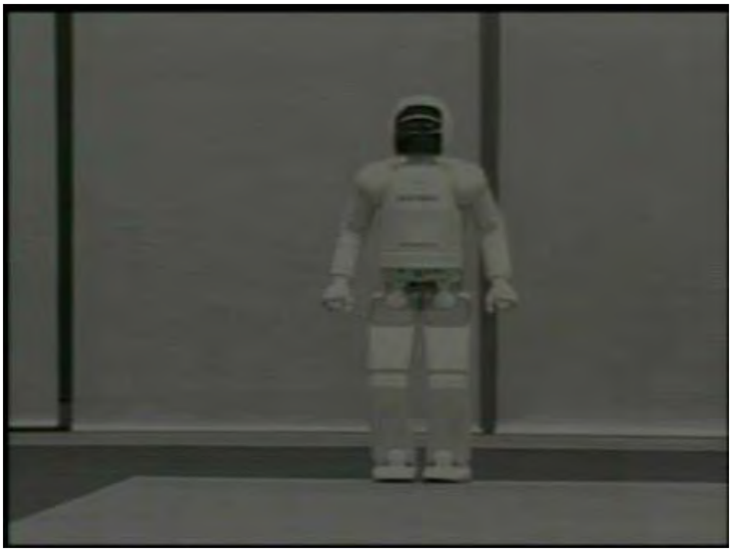
Control (Processing) …



Environment Understanding



Honda Asimov Humanoid



Act(ion)

Robot Sniper Training Robots

METR 4202: **Robotics**

July 27, 2016 -54

METR 4202: Robotics July 27, 2016 -

Extending Our Reach…

(what's hard is not what you expect…)

Throwing and Catching

METR 4202: Robotics July 27, 2016 -57

Making Iced Tea

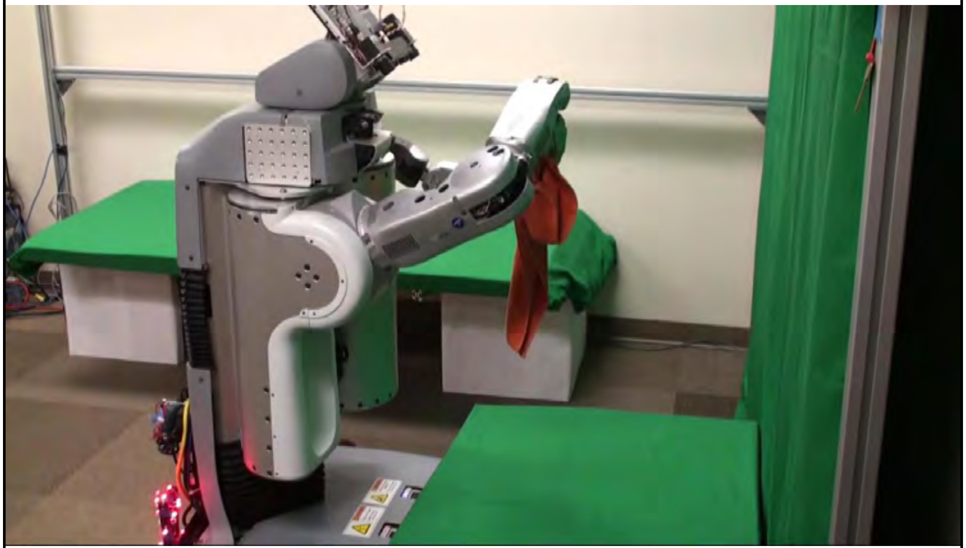
METR 4202: Robotics July 27, 2016 -58

People and Robots?

http://www.abc.net.au/radionational/image/4560736-4x3-340x255.jpg

People & Robots: Let Each Do Its Best!

Shirt-Folding (30x speed up)…



METR 4202: **Robotics** July 27, 2016 -61

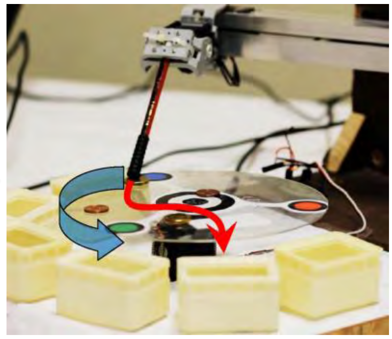
Shirt-Folding (1/3 Speed!)

METR 4202: **Robotics** July 27, 2016 -62

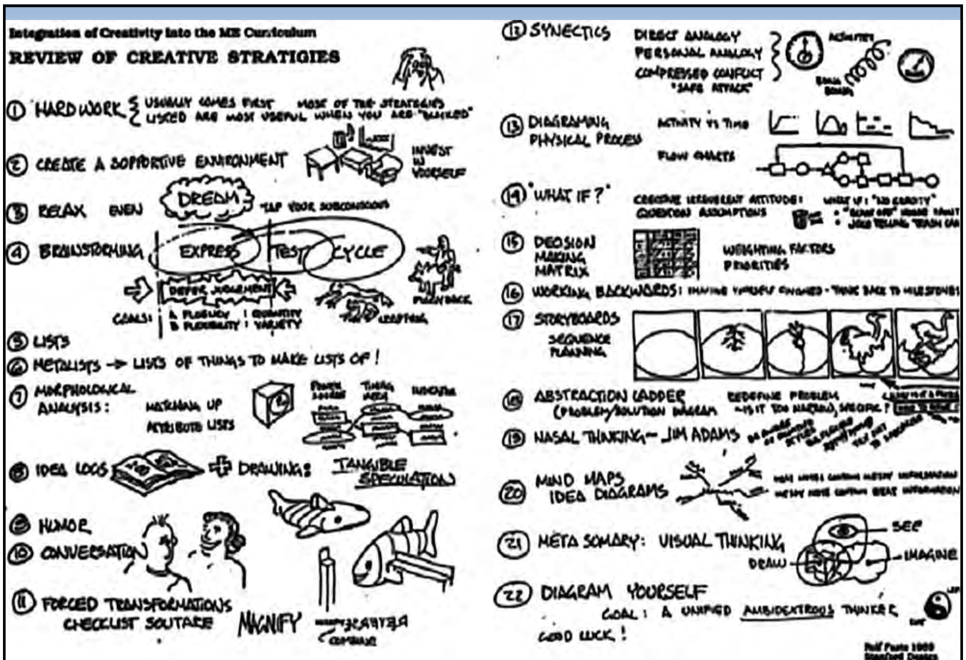
Parallel-Parking…

Parallel Parking…

The Project!
"Robotics: Domino Effect"



Next Week ☺



First thing about structure

→ **Space**

Representing Position
& Orientation & State

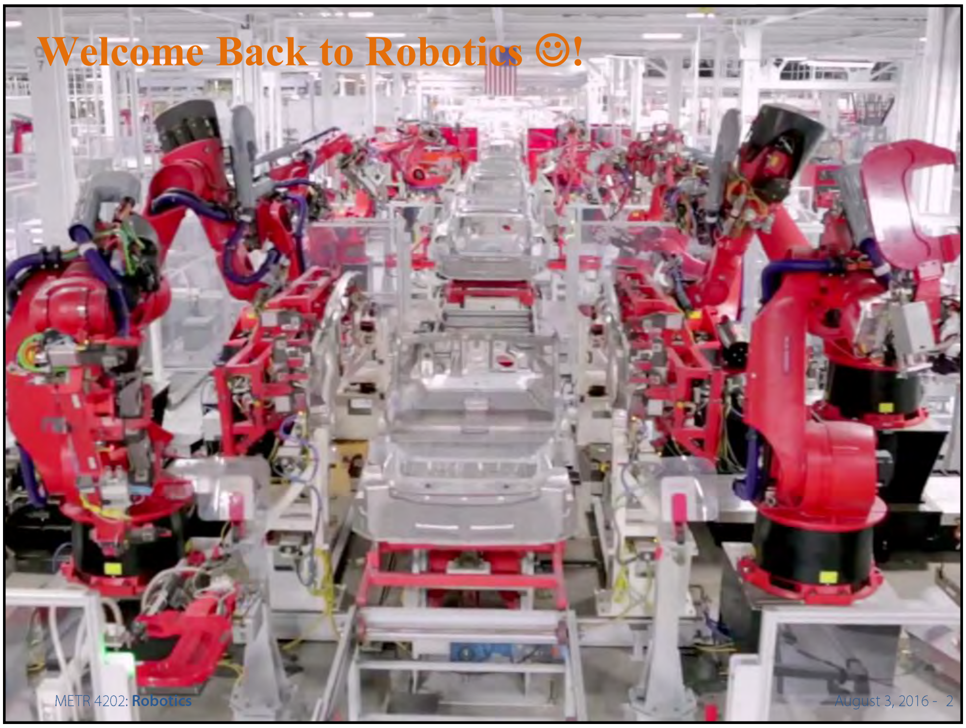
METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 2

August 3, 2016

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

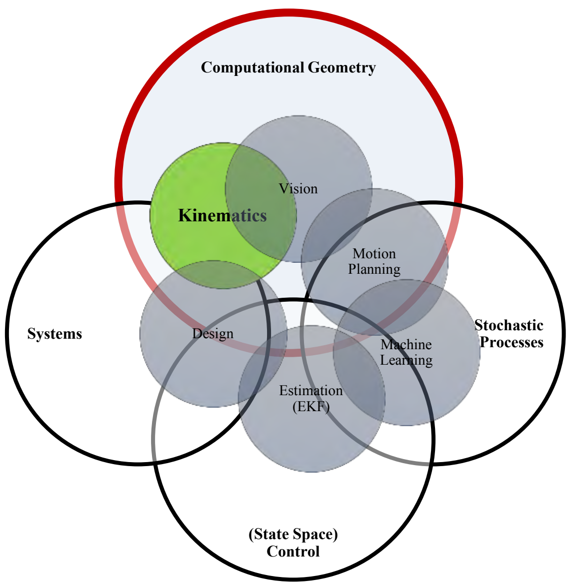
[http://**metr4202.com**]



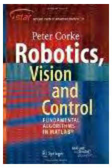
Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	**3-Aug**	**Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)**
3	10-Aug	Robot Kinematics Review (& *Ekka Day*)
4	17-Aug	Robot Dynamics
5	24-Aug	Robot Sensing: Perception
6	31-Aug	Robot Sensing: Multiple View Geometry
7	7-Sep	Robot Sensing: Feature Detection (as Linear Observers)
8	14-Sep	Probabilistic Robotics: Localization
9	21-Sep	Probabilistic Robotics: SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	State-Space Modelling
12	19-Oct	Shaping the Dynamic Response
13	26-Oct	LQR + Course Review

Course Organization



Follow Along Reading:



[Robotics, Vision & Control](#)
by [Peter Corke](#)

Also online:[*SpringerLink*](#)

[UQ Library eBook: 364220144X](#)

Today

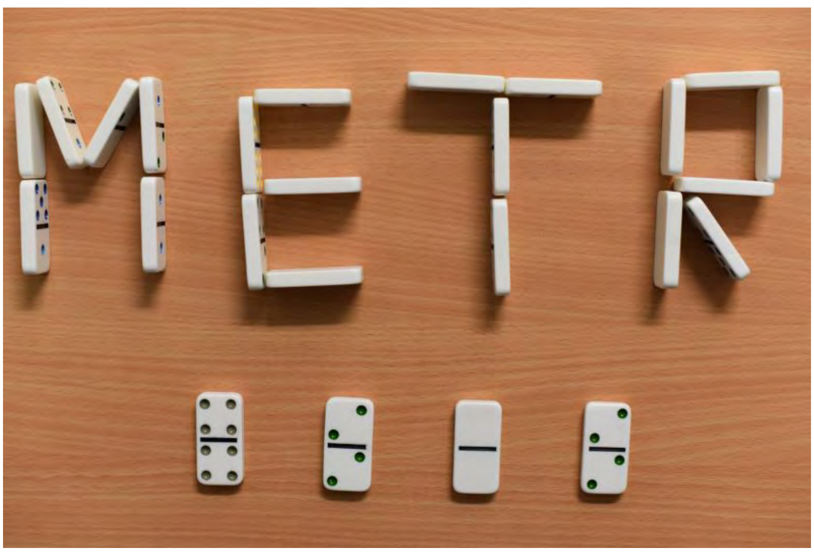
➔ **Representing Position** 🡸

- RVC
  – Ch. 2: Representing Position & Orientation

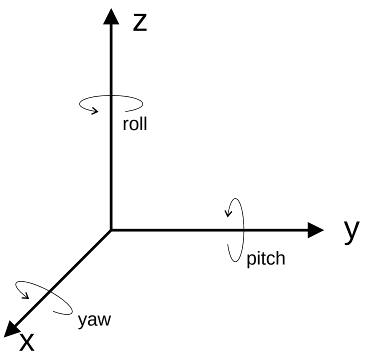
- Kinematics
  – RVC
  – Chapter 7: Robot Arm Kinematics

Next Time

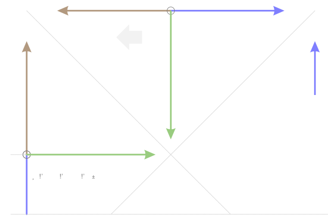
The Project!
"Robotics: Domino Effect"



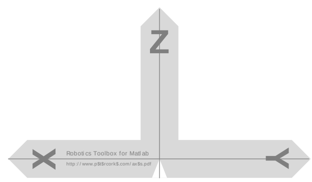
Today's Lecture is about: Frames & Their Mathematics



- Make one (online):
  - SpnS Template

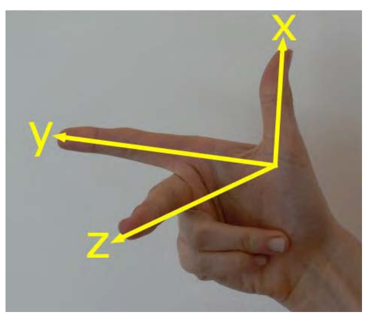


  - Peter Corke's template



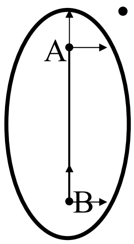
Don't Confuse a Frame with a Point

- Points
  - Position Only – Doesn't Encode Orientation
- Frame
  - Encodes both position and orientation
  - Has a "handedness"

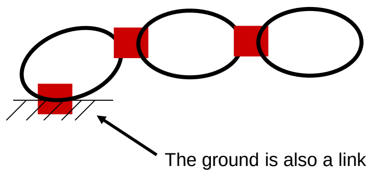


Kinematics Definition

- **Kinematics**: The study of motion in space (without regard to the forces which cause it)



- Assume:
  - Points with ***right-hand Frames***
  - *Rigid-bodies* in 3D-space (6-dof)
  - 1-dof joints: Rotary (R) or Prismatic (P) (5 constraints)

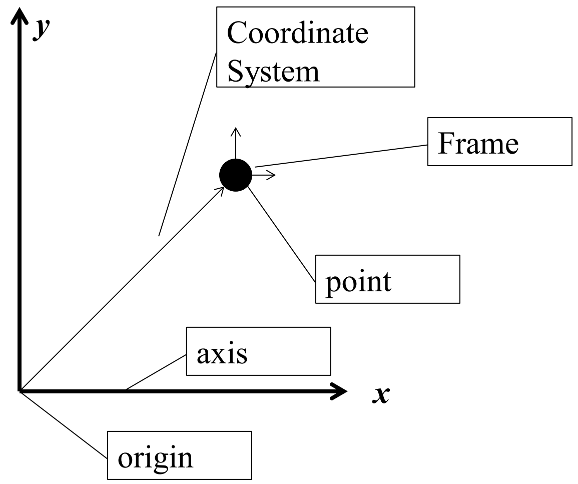


N links
M joints
→DOF = 6N-5M
➔ If N=M, then DOF=N.

Kinematics

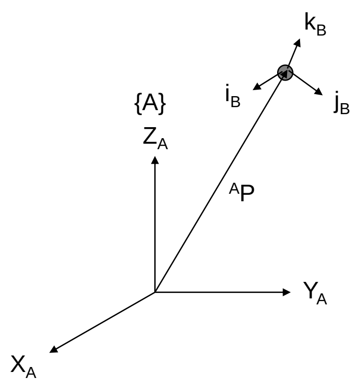
- Kinematic modelling is one of the most important analytical tools of robotics.
- Used for modelling mechanisms, actuators and sensors
- Used for on-line control and off-line programming and simulation
- In mobile robots kinematic models are used for:
  - steering (control, simulation)
  - perception (image formation)
  - sensor head and communication antenna pointing
  - world modelling (maps, object models)
  - terrain following (control feedforward)
  - gait control of legged vehicles

Basic Terminology



Coordinate System

- The position and orientation as specified only make sense with respect to some coordinate system



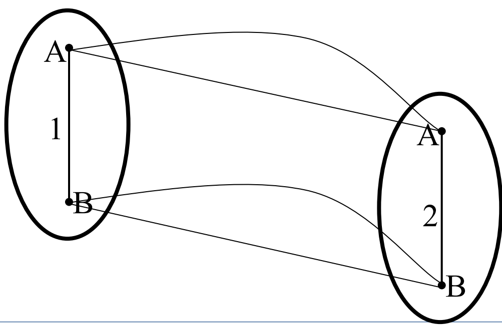
Frames of Reference

- A frame of reference defines a coordinate system relative to some point in space
- It can be specified by a position and orientation relative to other frames
- The *inertial frame* is taken to be a point that is assumed to be fixed in space

- Two types of motion:
  - Translation
  - Rotation

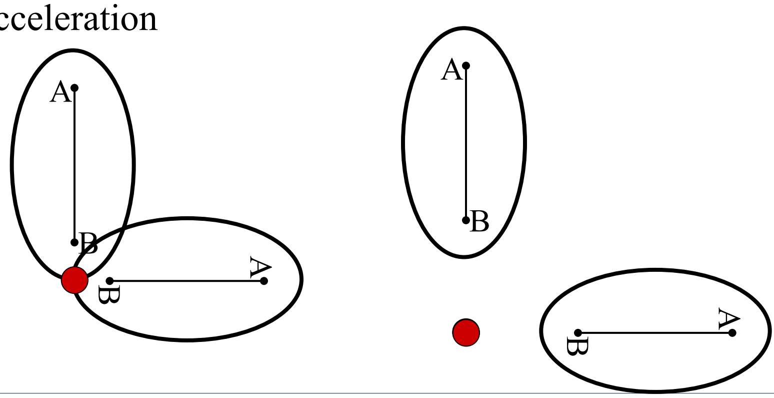
Translation

- A motion in which a straight line with in the body keeps the same direction during the
  - **Rectilinear Translation:** Along straight lines
  - **Curvilinear Translation**: Along curved lines



Rotation

- The particles forming the rigid body move in parallel planes along circles centered around the same fixed axis (called the **axis of rotation**).
- Points on the axis of rotation have zero velocity and acceleration

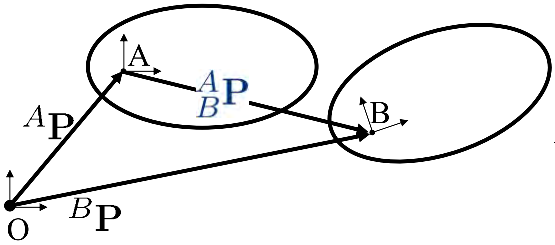


Rotation: Representations

- Orientation are not "Cartesian"
  - Non-commutative
  - Multiple representations
- Some representations:
  - **Rotation Matrices**: Homegenous Coordinates
  - Euler Angles: 3-sets of rotations in sequence
  - Quaternions: a 4-paramameter representation that exploits ½ angle properties
  - Screw-vectors (from Charles Theorem) : a canonical representation, its reciprocal is a "wrench" (forces)

Position and Orientation [1]

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



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

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

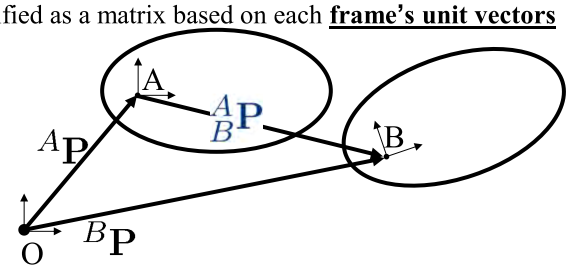
$$^{A}\mathbf{P}^{B} = {}^{A}\mathbf{P}_{B} = {}^{A}_{B}\mathbf{P} = \begin{bmatrix} {}^{B}p_x \\ {}^{B}p_y \\ {}^{B}p_z \end{bmatrix} - \begin{bmatrix} {}^{A}p_x \\ {}^{A}p_y \\ {}^{A}p_z \end{bmatrix}$$

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

METR 4202: **Robotics** August 3, 2016 - 17

Position and Orientation [2]

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



- 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}"

$$^{A}\mathbf{R}_{B} = {}^{A}_{B}\mathbf{R} = \begin{bmatrix} {}^{A}\hat{\imath}_{B} & {}^{A}\hat{\jmath}_{B} & {}^{A}\hat{k}_{B} \end{bmatrix}$$

- **Columns** are **{B} written in {A}**

METR 4202: **Robotics** August 3, 2016 - 18

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

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

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

Position and Orientation [4]

- Rotation is orthonormal

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

- The of a rotation matrix inverse = the transpose

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

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

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

Position and Orientation [5]: A note on orientations

- Orientations, as defined earlier, are represented by three orthonormal vectors
- Only three of these values are unique and we often wish to define a particular rotation using three values (it's easier than specifying 9 orthonormal values)
- There isn't a unique method of specifying the angles that define these transformations

Position and Orientation [7]

- Shortcut Notation:

$$\cos(\theta_a) = c\theta_a = \mathbf{c_a}$$
$$\sin(\theta_a) = s\theta_a = \mathbf{s_a}$$

$$\cos(\theta_a + \theta_b) = \mathbf{c_{ab}}$$

$$\therefore \mathbf{s_{ab}} = \boxed{\qquad\qquad ?}$$

Position and Orientation [8]

- Rotation Formula about the 3 Principal Axes by θ

X:
$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Y:
$$\mathbf{R}_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

Z:
$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Euler Angles

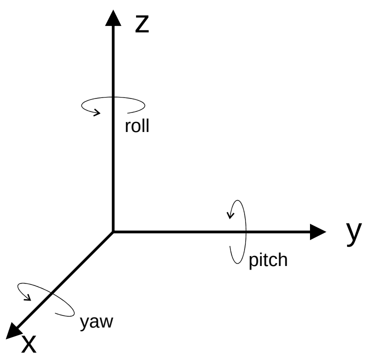
- Minimal representation of orientation (α,β,γ)
- Represent a rotation about an axis of a **moving** coordinate frame
  → ${}^{A}_{B}\mathbf{R}$: Moving frame **B** w/r/t fixed A
- The location of the axis of each successive rotation depends on the previous one! …
- So, Order Matters (12 combinations, 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°)

Fixed Angles

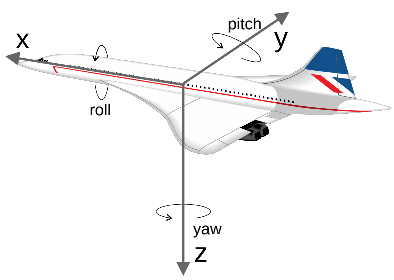
- Represent a rotation about an axis of a **fixed** coordinate frame.
- Again 12 different orders
- Interestingly:
  3 rotations about 3 axes of a **fixed** frame define the same orientation as the same 3 rotations taken in the **opposite order** of the **moving** frame
- For X-Y-Z:
  - ψ: rotation about $\mathbf{x}_A$ (sometimes called "yaw")
  - θ: rotation about $\mathbf{y}_A$ (sometimes called "pitch")
  - φ: rotation about $\mathbf{z}_A$ (sometimes called "roll")

Roll – Pitch – Yaw

- In many Kinematics References:



- In many Engineering Applications:



→ Be careful:
This name is given to other conventions too!

Euler Angles [1]: **X-Y-Z Fixed Angles** (Roll-Pitch-Yaw)

- One method of describing the orientation of a Frame {B} is:
  - Start with the frame coincident with a known reference {A}. Rotate {B} first about X_A by an angle γ, then about Y_A by an angle β and finally about Z_A by an angle α.

$$
\begin{aligned}
{}^{A}R_{BXYZ}(\gamma,\beta,\alpha) &= R_Z(\alpha)R_Y(\beta)R_X(\gamma) \\
&= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix} \\
&= \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix}
\end{aligned}
$$

METR 4202: **Robotics** August 3, 2016 - 27

Euler Angles [2]: Z-Y-X Euler Angles

- Another method of describing the orientation of {B} is:
  - Start with the frame coincident with a known reference {A}. Rotate {B} first about Z_B by an angle α, then about Y_B by an angle β and finally about X_B by an angle γ.

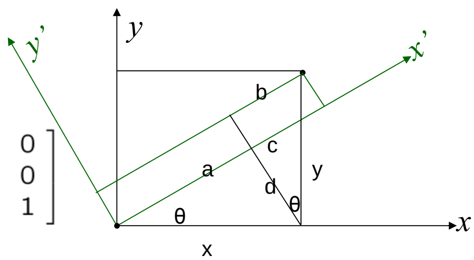
$$
\begin{aligned}
{}^{A}R_{BZ'Y'X'}(\gamma,\beta,\alpha) &= R_Z(\alpha)R_Y(\beta)R_X(\gamma) \\
&= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix} \\
&= \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix}
\end{aligned}
$$

METR 4202: **Robotics** August 3, 2016 - 28

Position and Orientation [6]: "Proof" of Principal Rotation Matrix Terms

- Geometric:

$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$a = x\cos\theta,\ b = y\sin\theta$

$c = y\cos\theta,\ d = x\sin\theta$

Thus:

$x' = x\cos\theta + y\sin\theta$

$y' = -x\sin\theta + y\cos\theta$

Unit Quaternion (ϵ_0, ϵ_1, ϵ_2, ϵ_3) [1]

- Does not suffer from singularities

$$\epsilon \equiv \epsilon_0 + \left(\epsilon_1 \hat{\mathbf{i}} + \epsilon_2 \hat{\mathbf{j}} + \epsilon_3 \hat{\mathbf{k}}\right)$$

- Uses a "4-number" to represent orientation

$$ii = jj = kk = -1$$
$$ij = k, jk = i, ki = j, ji = -k, kj = -1, ik = -j$$

- Product:

$$\begin{aligned} \mathbf{ab} \ = \ & (a_0 b_0 - a_1 b_1 - a_2 b_2 + a_3 b_3) \\ & + (a_0 b_1 + a_1 b_0 + a_2 b_3 - a_3 b_2)\,\hat{i} \\ & + (a_0 b_2 + a_2 b_0 + a_3 b_1 + a_1 b_3)\,\hat{j} \\ & + (a_0 b_3 + a_3 b_0 + a_1 b_2 - a_2 b_1)\,\hat{k} \end{aligned}$$

- Conjugate:

$$\tilde{\epsilon} \equiv \epsilon_0 - \epsilon_1 \hat{\mathbf{i}} - \epsilon_2 \hat{\mathbf{j}} - \epsilon_3 \hat{\mathbf{k}}$$
$$\epsilon\tilde{\epsilon} = \tilde{\epsilon}\epsilon = \epsilon_0^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2$$

Unit Quaternion [2]: Describing Orientation

- Set $\epsilon_0 = 0$
  Then $\mathbf{p}=(p_x,p_y,p_z)$ → $\mathbf{p} = p_x\hat{\mathbf{i}} + p_y\hat{\mathbf{j}} + p_z\hat{\mathbf{k}}$

- Then given ϵ
  the operation $\epsilon \mathbf{p} \tilde{\epsilon}$: rotates **p** about $(\epsilon_1, \epsilon_2, \epsilon_3)$

- Unit Quaternion → Rotation Matrix

$$\mathbf{R} = \begin{pmatrix} 1-2\left(\epsilon_2^2+\epsilon_3^2\right) & 2\left(\epsilon_1\epsilon_2-\epsilon_0\epsilon_3\right) & 2\left(\epsilon_1\epsilon_3-\epsilon_0\epsilon_2\right) \\ 2\left(\epsilon_1\epsilon_2-\epsilon_0\epsilon_3\right) & 1-2\left(\epsilon_1^2+\epsilon_3^2\right) & 2\left(\epsilon_2\epsilon_3-\epsilon_0\epsilon_1\right) \\ 2\left(\epsilon_1\epsilon_3-\epsilon_0\epsilon_2\right) & 2\left(\epsilon_2\epsilon_3-\epsilon_0\epsilon_1\right) & 1-2\left(\epsilon_1^2+\epsilon_2^2\right) \end{pmatrix}$$

Direction Cosine

- Uses the Direction Cosines (read dot products) of the Coordinate Axes of the moving frame with respect to the fixed frame

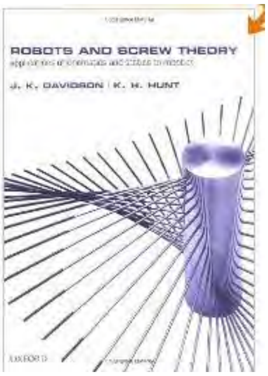
$$^{A}\mathbf{u} = u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k}$$
$$^{A}\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$$
$$^{A}\mathbf{w} = w_x\mathbf{i} + w_y\mathbf{j} + w_z\mathbf{k}$$

- It forms a rotation matrix!

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

Screw Displacements

- Comes from the notion that all motion can be viewed as a rotation (Rodrigues formula)
- Define a vector along the axis of motion (screw vector)
  - Rotation (screw angle)
  - Translation (pitch)
  - Summations → via the screw triangle!



Generalizing

Special Orthogonal & Special Euclidean Lie Algebras

- SO(n): R**o**tations

$$SO(n) = \{R \in \mathbb{R}^{n\times n} : RR^T = I, \det R = +1\}.$$

$$\exp(\widehat{\omega}\theta) = e^{\widehat{\omega}\theta} = I + \theta\widehat{\omega} + \frac{\theta^2}{2!}\widehat{\omega}^2 + \frac{\theta^3}{3!}\widehat{\omega}^3 + \dots$$

- SE(n): Transformations of EUCLIDEAN space

$$SE(n) := \mathbb{R}^n \times SO(n).$$

$$SE(3) = \{(p, R) : p \in \mathbb{R}^3, R \in SO(3)\} = \mathbb{R}^3 \times SO(3).$$

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, **I**, **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*

Homogenous Coordinates

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

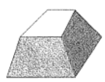
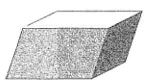
- ρ is a scaling value

Homogenous Transformation

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

- γ is a projective transformation
- The Homogenous Transformation is a **linear operation** (even if projection is not)

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.

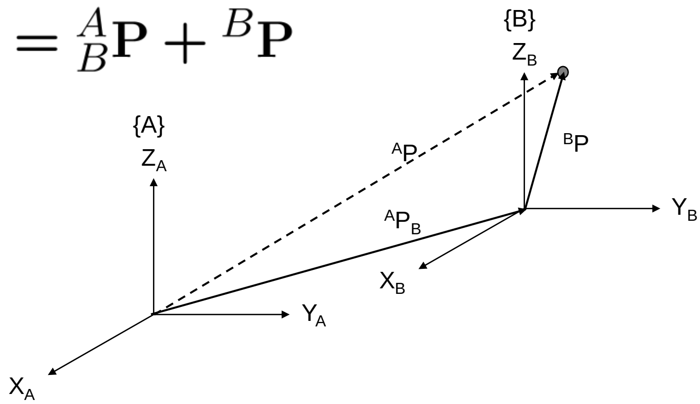
p.78, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

Coordinate Transformations [1]

- Translation Again:

  If {B} is translated with respect to {A} **without rotation**, then it is a vector sum

$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{P} + {}^{B}\mathbf{P}$$



Coordinate Transformations [2]

- Rotation Again:

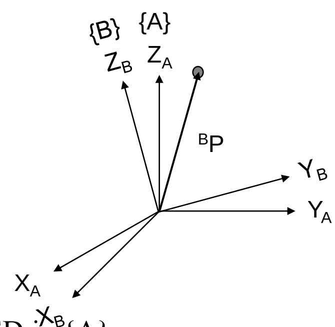
  {B} is rotated with respect to {A} then
  use rotation matrix to determine new components

$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}$$

- NOTE:
  – The Rotation matrix's ***subscript*** matches the position vector's **superscript**

$$^{A}\mathbf{P} = {}^{A}_{[B]}\mathbf{R}\,{}^{[B]}\mathbf{P}$$

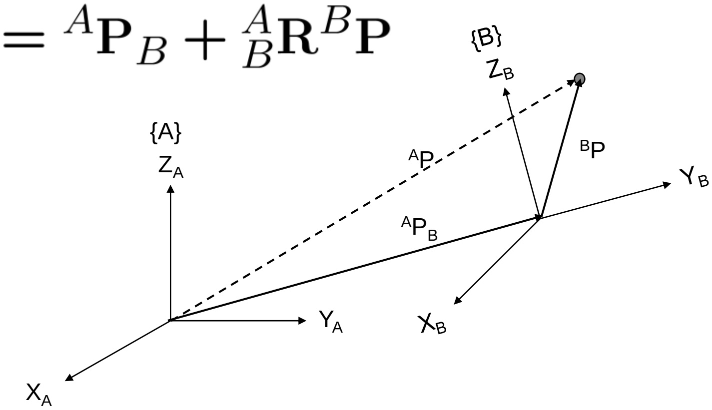
  – This gives Point Positions of {B} ORIENTED in {A}



Coordinate Transformations [3]

- Composite transformation:
  {B} is moved with respect to {A}:

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



General Coordinate Transformations [1]

- A compact representation of the translation and rotation is known as the **Homogeneous Transformation**

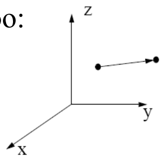
$$ {}^{A}_{B}\mathbf{T} = \begin{bmatrix} & {}^{A}_{B}\mathbf{R} & & {}^{A}\mathbf{P}_{B} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- This allows us to cast the rotation and translation of the general transform in a single matrix form

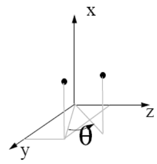
$$ \begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = {}^{A}_{B}\mathbf{T} \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix} $$

General Coordinate Transformations [2]

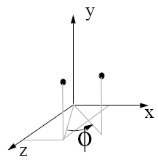
- Similarly, fundamental orthonormal transformations can be represented in this form too:



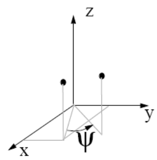
$$Trans(u, v, w) = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Rotx(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & 0 \\ 0 & s\theta & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Roty(\phi) = \begin{bmatrix} c\phi & 0 & s\phi & 0 \\ 0 & 1 & 0 & 0 \\ -s\phi & 0 & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

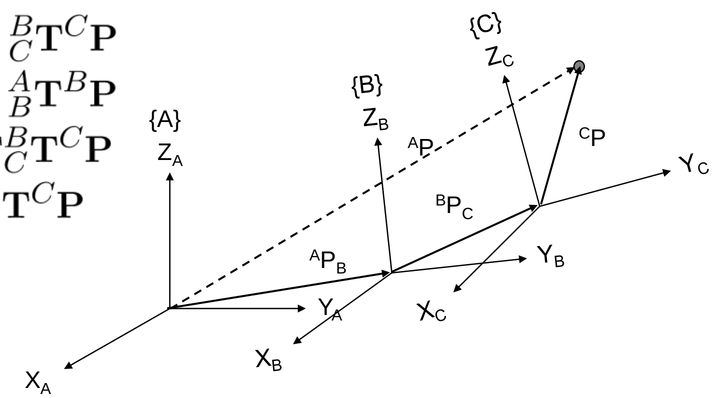


$$Rotz(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 & 0 \\ s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Coordinate Transformations [3]

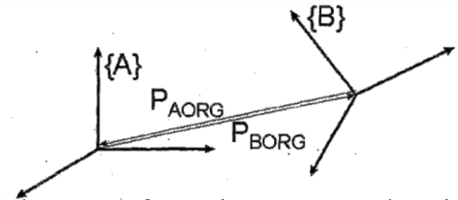
- Multiple transformations compounded as a chain

$$
\begin{aligned}
{}^{B}\mathbf{P} &= {}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P} \\
{}^{A}\mathbf{P} &= {}^{A}_{B}\mathbf{T}\,{}^{B}\mathbf{P} \\
&= {}^{A}_{B}\mathbf{T}\,{}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P} \\
&= {}^{A}_{C}\mathbf{T}\,{}^{C}\mathbf{P}
\end{aligned}
$$



$$ {}^{A}_{C}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R}\,{}^{B}_{C}\mathbf{R} & {}^{A}\mathbf{P}_B + {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}_C \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} $$

Inverse of a Homogeneous Transformation Matrix



- The inverse of the transform is **not** equal to its transpose because this 4×4 matrix is not orthonormal ($T^{-1} \neq T^T$)
- Invert by parts to give:

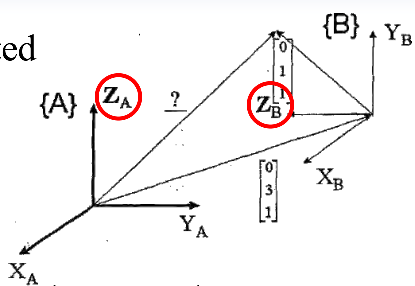
$$ {}^A_BT = \begin{bmatrix} & {}^A_BR & & {}^A\mathbf{p}_{Borg/O_A} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^A_BT^{-1} = {}^B_AT = \begin{bmatrix} & {}^A_BR^T & & -{}^A_BR^T \cdot {}^A\mathbf{p}_{Borg/O_A} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & {}^B_AR & & {}^B\mathbf{p}_{Aorg/O_B} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

METR 4202: **Robotics** August 3, 2016 -45

Tutorial Problem

The origin of frame *{B}* is translated to a position [0 3 1] with respect to frame *{A}*.



We would like to find:

1. The homogeneous transformation between the two frames in the figure.
2. For a point *P* defined as as [0 1 1] in frame *{B},* we would like to find the vector describing this point with respect to frame *{A}*.

METR 4202: **Robotics** August 3, 2016 -46

Tutorial Solution

• The matrix ${}_BT^A$ is formed as defined earlier:

• Since P in the frame is:

• We find vector **p** in frame *{A}* using the relationship

➔

- The matrix ${}_BT^A$ is formed as defined earlier:

$$
{}^A_BT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

- Since P in the frame is: $^B\mathbf{p} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

- We find vector **p** in frame *{A}* using the relationship

$$
{}^Ap = {}^A_BT\,{}^Bp
$$

➔ $^A\mathbf{p} = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 1 \end{bmatrix}$

Cool Robotics Share

http://www.kinemasystems.com/

Part II:
Forward & Inverse Kinematics

1. Forward Kinematics ($\theta \rightarrow x$)
2. Inverse Kinematics ($x \rightarrow \theta$)
3. Denavit Hartenberg [DH] Notation
4. Affine Transformations &
5. Theoretical (General) Kinematics

METR 4202: **Robotics** August 3, 2016 -49

Forward Kinematics [1]

- Forward kinematics is the process of chaining homogeneous transforms together. For example to:
  - Find the articulations of a mechanism, or
  - the fixed transformation between two frames which is known in terms of linear and rotary parameters.
- Calculates the final position from the **machine** (**joint variables**)
- Unique for an open kinematic chain (**serial arm**)
- "Complicated" (multiple solutions, etc.) for a closed kinematic chain (**parallel arm**)

METR 4202: **Robotics** August 3, 2016 -50

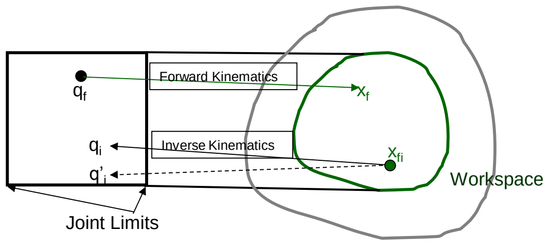
Forward Kinematics [2]

- Can think of this as "spaces":
  - Workspace (x,y,z,α,β,γ):
    The robot's position & orientation

$$\vec{\mathrm{x}} = \begin{bmatrix} \vec{\mathrm{p}} \\ \vec{\Theta} \end{bmatrix}$$

  - Joint space ($\theta_1 \ldots \theta_n$):
    A state-space vector of joint variables

$$\vec{\mathrm{q}} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$

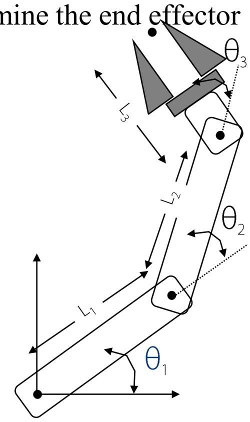


Forward Kinematics [3]

- Consider a planar RRR manipular
- Given the joint angles and link lengths, we can determine the end effector pose:

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

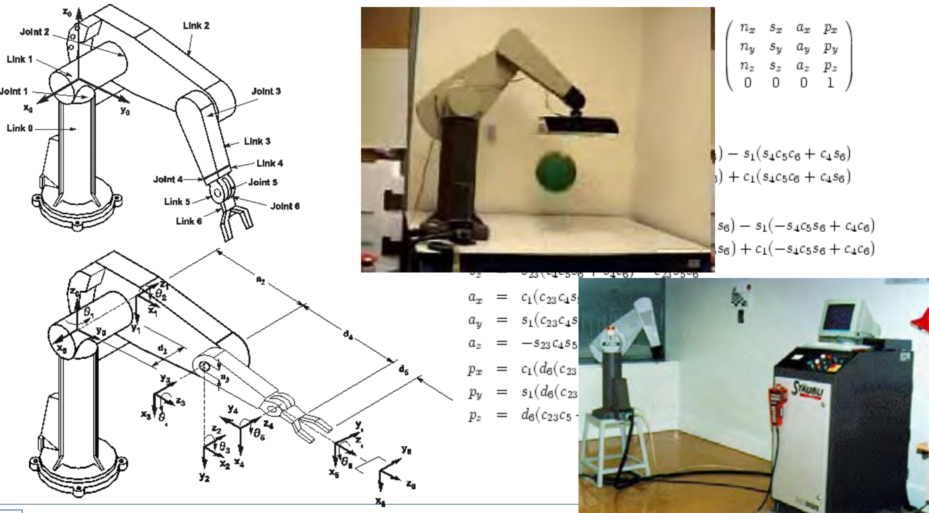
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + \ldots \\ L_3 \sin(\theta_1 + \theta_2 + \theta_3)$$



- This isn't too difficult to determine
  for a simple, planar manipulator. BUT …

Forward Kinematics [4]: The PUMA 560!

- What about a more complicated mechanism?

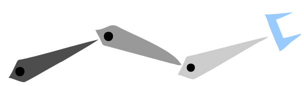


Inverse Kinematics

- Forward: angles → position

  $\mathbf{x} = f(\boldsymbol{\theta})$
- Inverse: position → angles

  $\boldsymbol{\theta} = f^{-1}(\mathbf{x})$
- Analytic Approach



- Numerical Approaches:
  - Jacobian:
  - J^T Approximation:
    - Slotine & Sheridan method
  - Cyclical Coordinate Descent

$$J = \frac{\delta x}{\delta q} \rightarrow \delta q \approx J^{-1} \delta x$$

$$\tau = J^T \cdot \mathbf{F} \rightarrow \Delta q \approx J^T \Delta x$$

Inverse Kinematics

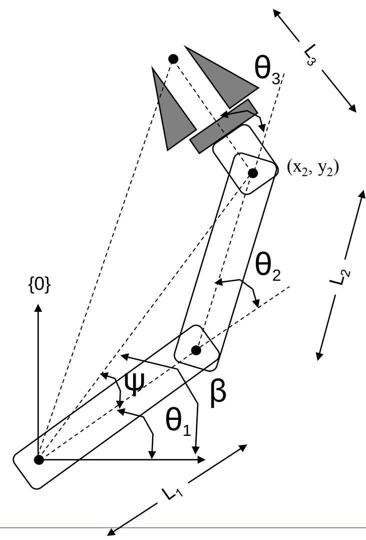
- Inverse Kinematics is the problem of finding the joint parameters given only the values of the homogeneous transforms which model the mechanism (i.e., the pose of the end effector)
- Solves the problem of where to drive the joints in order to get the hand of an arm or the foot of a leg in the right place
- In general, this involves the solution of a set of simultaneous, non-linear equations
- Hard for serial mechanisms, easy for parallel

Solution Methods

- Unlike with systems of linear equations, there are no general algorithms that may be employed to solve a set of nonlinear equation
- **Closed-form** and **numerical** methods exist
- Many exist: Most general solution to a 6R mechanism is Raghavan and Roth (1990)
- Three methods of obtaining a solution are popular: (1) **geometric** | (2) **algebraic** | (3) **DH**

Inverse Kinematics: Geometrical Approach

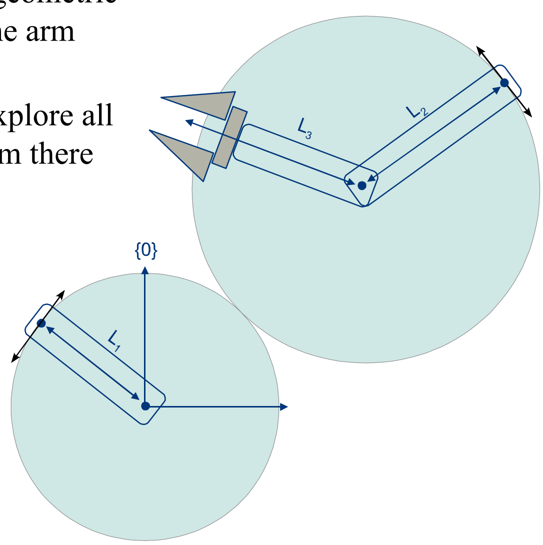
- We can also consider the geometric relationships defined by the arm



METR 4202: **Robotics** August 3, 2016 -57

Inverse Kinematics: Geometrical Approach [2]

- We can also consider the geometric relationships defined by the arm
- Start with what is fixed, explore all geometric possibilities from there

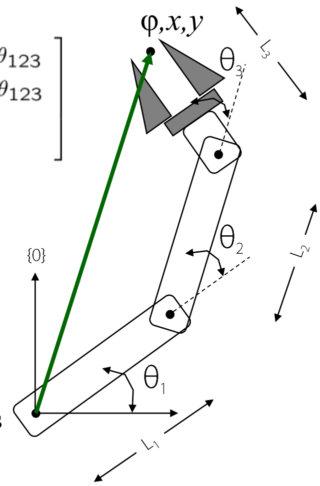


METR 4202: **Robotics** August 3, 2016 -58

Inverse Kinematics: Algebraic Approach

- We have a series of equations which define this system
- Recall, from Forward Kinematics:

$$ {}^0T_3 = \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1c_{\theta_1} + L_2c_{\theta_{12}} + L_3c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1s_{\theta_1} + L_2s_{\theta_{12}} + L_3s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



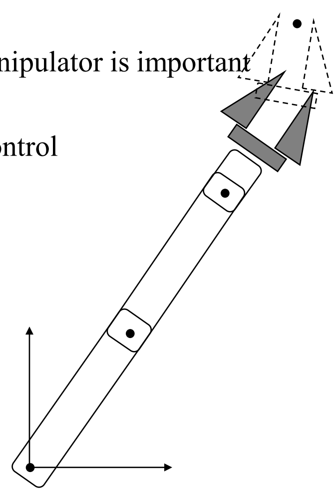
- The end-effector pose is given by

$$ {}^0T_3 = \begin{bmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- Equating terms gives us a set of algebraic relationships

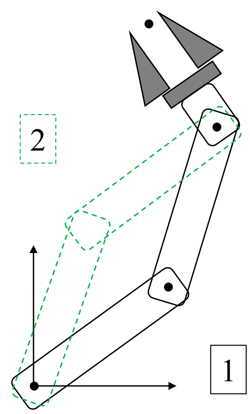
No Solution - Singularity

- Singular positions:
- An understanding of the workspace of the manipulator is important
- There will be poses that are not achievable
- There will be poses where there is a loss of control
- Singularities also occur when the manipulator loses a DOF
  - This typically happens when joints are aligned
  - **det[Jacobian]=0**



Multiple Solutions

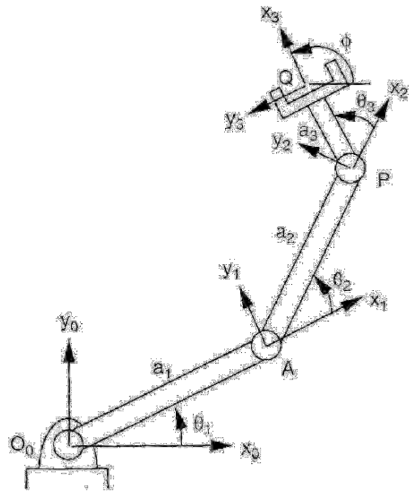
- There will often be multiple solutions for a particular inverse kinematic analysis
- Consider the three link manipulator shown. Given a particular end effector pose, two solutions are possible
- The choice of solution is a function of proximity to the current pose, limits on the joint angles and possible obstructions in the workspace



Inverse Kinematics [More Generally]

- Freudenstein (1973) referred to the inverse kinematics problem of the most general **6R** manipulator as the "Mount Everest" of kinematic problems.
- Tsai and Morgan (1985) and Primrose (1986) proved that this has at most 16 real solutions.
- Duffy and Crane (1980) derived a closed-form solution for the general **7R** single-loop spatial mechanism.
  - The solution was obtained in the form of a 16 x 16 delerminant in which every element is a second-degree polynomial in one joint variable. The determinant, when expended, should yield a 32nd-degree polynomial equation and hence confirms the upper limit predicted by Roth *et al*. (1973).
- Tsai and Morgan (1985) used the homotopy continuation method to solve the inverse kinematics of the general 6R manipulator and found only 16 solutions
- Raghavan and Roth (1989, 1990) used the dyalitic elimination method to derive a 16th-degree polynomial for the general 6R inverse kinematics problem.

Example: FK/IK of a 3R Planar Arm



- Derived from Tsai (p. 63)

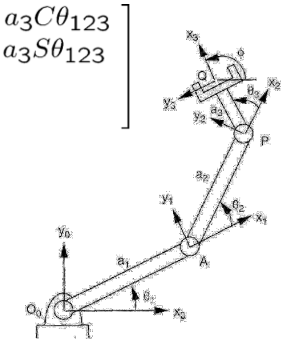
Example: 3R Planar Arm [2]

Position Analysis: 3·Planar 1-R Arm rotating about **Z** **[Ⓩ]**

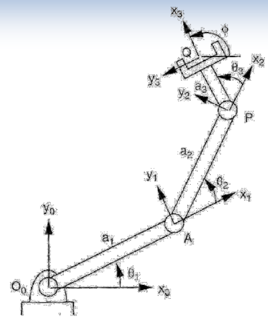
$$ {}^{0}A_3 = \; {}^{0}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 $$

Substituting gives:

$$ {}^{0}A_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 & a_1C\theta_1 + a_2C\theta_{12} + a_3C\theta_{123} \\ S\theta_{123} & C\theta_{123} & 0 & a_1S\theta_1 + a_2S\theta_{12} + a_3S\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



Example: 3R Planar Arm [2]



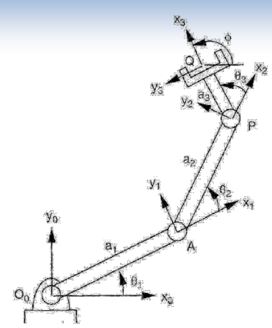
Forward Kinematics
(solve for **x given θ** → $\mathbf{x} = f(\boldsymbol{\theta})$)

Fairly straight forward:

$$^{0}R_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 \\ S\theta_{123} & C\theta_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$^{0}P_3 = \begin{bmatrix} a_1 C\theta_1 + a_2 C\theta_{12} + a_3 C\theta_{123} \\ a_1 S\theta_1 + a_2 S\theta_{12} + a_3 S\theta_{123} \\ 0 \end{bmatrix}$$

Example: 3R Planar Arm [3]



Inverse Kinematics
(solve for **θ given x** → $\mathbf{x} = f(\boldsymbol{\theta})$)

- Start with orientation φ:

$C\theta_{123} = C\phi,\ S\theta_{123} = S\phi$

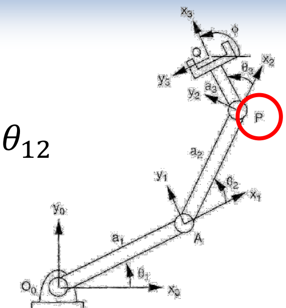
$\Rightarrow \theta_{123} = \theta_1 + \theta_2 + \theta_3 = \phi$

- Get overall position $\boldsymbol{q} = [q_x \quad q_y]$:

$q_x - a_3 C\phi = a_1 C\theta_1 + a_2 C\theta_{12}$

$q_y - a_3 S\phi = a_1 S\theta_1 + a_2 S\theta_{12}$ …

Example: 3R Planar Arm [4]



- Introduce $\boldsymbol{p} = [p_x \quad p_y]$ before "wrist"

$$p_x = a_1 C\theta_1 + a_2 C\theta_{12}, p_y = a_1 S\theta_1 + a_2 S\theta_{12}$$
$$\Rightarrow p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1 a_2 C\theta_2$$

- Solve for θ_2:

$$\theta_2 = \cos^{-1}\kappa, \kappa = \frac{p_x^2+p_y^2-a_1^2-a_2^2}{2a_1a_2}$$ (2 ℝ roots if $|\kappa|<1$)

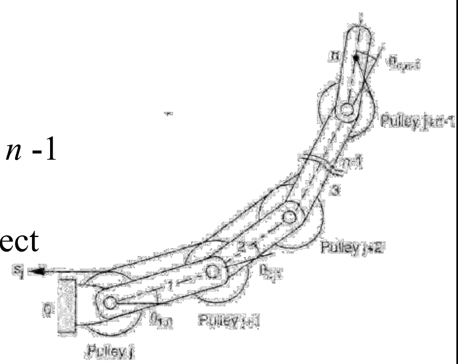
- Solve for θ_1:

$$C\theta_1 = \frac{p_x(a_1+a_2C\theta_2)+p_y a_2 S\theta_2}{a_1^2+a_2^2+2a_1a_2C\theta_2}, S\theta_1 = \frac{-p_x a_2 S\theta_2 + p_y(a_1+a_2C\theta_2)}{a_1^2+a_2^2+2a_1a_2C\theta_2}$$

$$\theta_1 = atan2(S\theta_1, C\theta_1)$$

Advanced Concept: Tendon-Driven Manipulators

- Tendons may be modelled as a transmission line
- in which the links are labeled sequentially from 0 to n and the pulleys are labeled from *j* to *j* + *n* -1
- Let θ_{ji} denote the angular displacement of link *j* with respect to link *i*.
- We can write a circuit equation once for each pulley pair as follows:



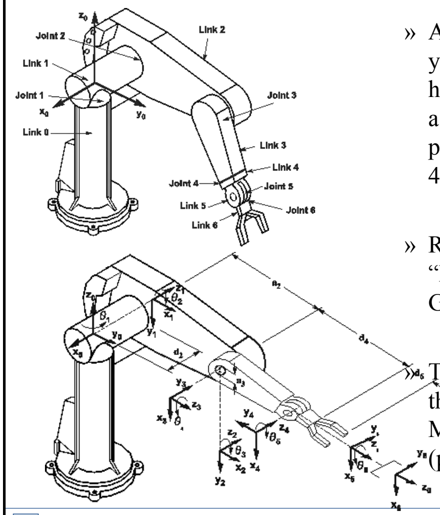
$$r_{j+i-1}\theta_{j+i-1,i} = \pm r_{j+i}\theta_{j+i,i} \qquad \text{for } i = 1, 2, \ldots, n-1.$$

$$\theta_{j+i-1,i} = \theta_{j+i-1,i-1} - \theta_{i,i-1} \qquad \text{for } i = 1, 2, \ldots, n.$$

$$\theta_{j,0} = \theta_{1,0} \pm (r_{j+1}/r_j)\theta_{2,1} \pm \cdots \pm (r_{j+n-1}/r_j)\theta_{n,n-1}.$$

Inverse Kinematics

- What about a more complicated mechanism?



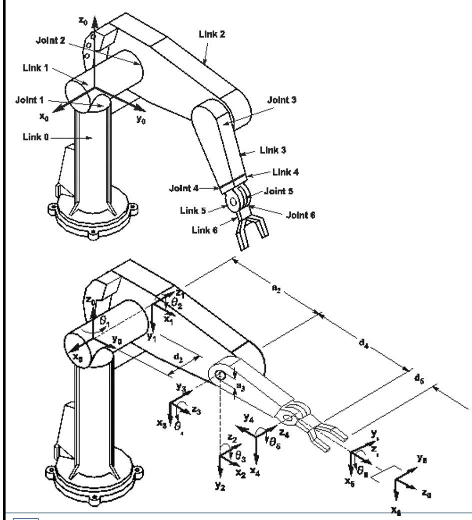
» A sufficient condition for a serial manipulator to yield a closed-form inverse kinematics solution is to have any three consecutive joint axes intersecting at a common point or any three consecutive joint axes parallel to each other. (Pieper and Roth (1969) via 4×4 matrix method)

» Raghavan and Roth 1990
"Kinematic Analysis of the 6R Manipulator of General Geometry"

» Tsai and Morgan 1985, "Solving the Kinematics of the Most General Six and Five-Dcgree-of-Freedom Manipulators by Continuation Methods"
(posted online)

Inverse Kinematics

- What about a more complicated mechanism?



$$
{}^0\boldsymbol{T}_6 = {}^0\boldsymbol{T}_1\,{}^1\boldsymbol{T}_2\,{}^2\boldsymbol{T}_3\,{}^3\boldsymbol{T}_4\,{}^4\boldsymbol{T}_5\,{}^5\boldsymbol{T}_6 = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
\begin{aligned}
n_x &= c_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) - s_1(s_4c_5c_6 + c_4s_6) \\
n_y &= s_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) + c_1(s_4c_5c_6 + c_4s_6) \\
n_z &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \\
s_x &= c_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) - s_1(-s_4c_5s_6 + c_4c_6) \\
s_y &= s_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) + c_1(-s_4c_5s_6 + c_4c_6) \\
s_z &= s_{23}(c_4c_5s_6 + s_4c_6) - c_{23}s_5s_6 \\
a_x &= c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 \\
a_y &= s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5 \\
a_z &= -s_{23}c_4s_5 + c_{23}c_5) \\
p_x &= c_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) - s_1(d_6s_4s_5 + d_2) \\
p_y &= s_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) + c_1(d_6s_4s_5 + d_2) \\
p_z &= d_6(c_{23}c_5 - s_{23}c_4s_5) + c_{23}d_4 - a_3s_{23} - a_2s_2
\end{aligned}
$$

Denavit Hartenberg [DH] Notation

- J. Denavit and R. S. Hartenberg first proposed the use of homogeneous transforms for articulated mechanisms

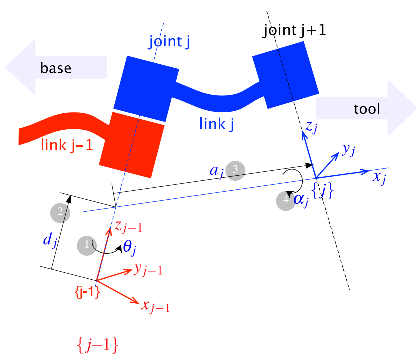
  (But B. Roth, introduced it to robotics)

- A kinematics "short-cut" that reduced the number of parameters by adding a structure to frame selection

- For two frames positioned in space, the first can be moved into coincidence with the second by a sequence of 4 operations:
  - rotate around the x_{i-1} axis by an angle α_i
  - translate along the x_{i-1} axis by a distance a_i
  - translate along the new z axis by a distance d_i
  - rotate around the new z axis by an angle θ_i

Denavit-Hartenberg Convention

- link length a_i the offset distance between the z_{i-1} and z_i axes along the x_i axis;
- link twist α_i the angle from the z_{i-1} axis to the z_i axis about the x_i axis;



Art c/o P. Corke

- link offset d_i the distance from the origin of frame i-1 to the x_i axis along the z_{i-1} axis;
- joint angle θ_i the angle between the x_{i-1} and x_i axes about the z_{i-1} axis.

DH: Where to place frame?

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

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

3. θ (rot *z*) → d (trans *z*) → a (trans *x*) → α (rot *x*)

METR 4202: **Robotics** August 3, 2016 - 73

Denavit-Hartenberg → Rotation Matrix

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

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

METR 4202: **Robotics** August 3, 2016 - 74

DH Example [1]: RRR Link Manipulator

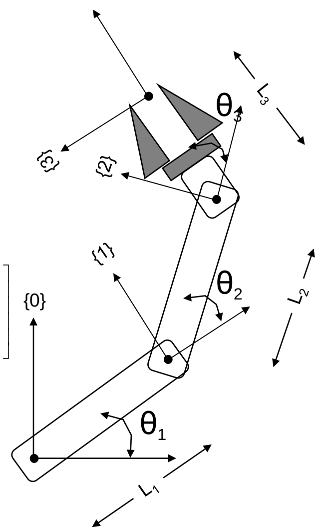
1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	$\boldsymbol{\theta_1}$
2	L_2	0	0	$\boldsymbol{\theta_2}$
3	L_3	0	0	$\boldsymbol{\theta_3}$

$$^0A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1 c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1 s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^1A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2 s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^2A_3 = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & L_3 c_{\theta_3} \\ s_{\theta_3} & c_{\theta_3} & 0 & L_3 s_{\theta_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^0T_3 = {}^0A_1 \, {}^1A_2 \, {}^2A_3$$

$$= \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1 c_{\theta_1} + L_2 c_{\theta_{12}} + L_3 c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1 s_{\theta_1} + L_2 s_{\theta_{12}} + L_3 s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



DH Example [2]: RRP Link Manipulator

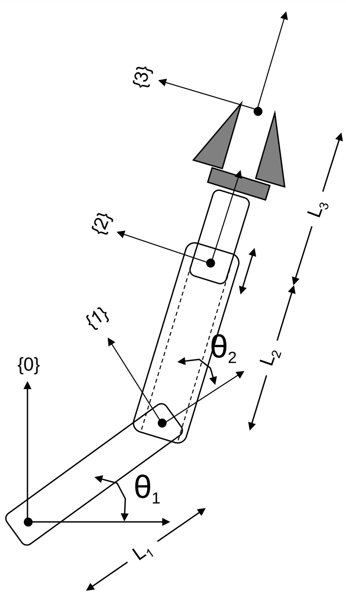
1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	$\boldsymbol{\theta_1}$
2	L_2	0	0	$\boldsymbol{\theta_2}$
3	$\mathbf{L_3}$	0	0	0

$$^0A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1 c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1 s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^1A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2 s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; ^2A_3 = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^0T_3 = {}^0A_1 \, {}^1A_2 \, {}^2A_3$$

$$= \begin{bmatrix} c_{\theta_{12}} & -s_{\theta_{12}} & 0 & L_1 c_{\theta_1} + (L_2 + L_3) c_{\theta_{12}} \\ s_{\theta_{12}} & c_{\theta_{12}} & 0 & L_1 s_{\theta_1} + (L_2 + L_3) s_{\theta_{12}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



DH Example [3]: Puma 560

- "Simple" 6R robot exercise for the reader …

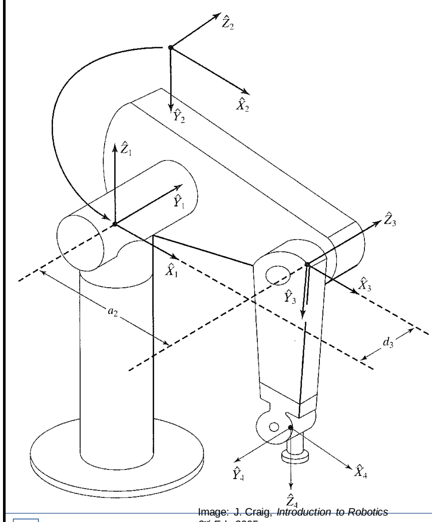
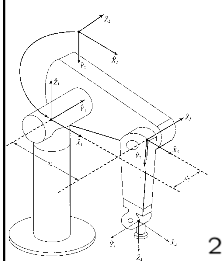


Image: J. Craig, *Introduction to Robotics* 3rd Ed., 2005

Link	a_i	α_i	d_i	θ_i
1	0	0	0	$\boldsymbol{\theta_1}$
2	0	$-\pi/2$	0	$\boldsymbol{\theta_2}$
3	L_2	0	D_3	$\boldsymbol{\theta_3}$
4	L_3	$-\pi/2$	D_4	$\boldsymbol{\theta_4}$
5	0	$\pi/2$	0	$\boldsymbol{\theta_5}$
6	0	$-\pi/2$	0	$\boldsymbol{\theta_6}$

DH Example [3]: Puma 560 [2]



$$^0A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^1A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

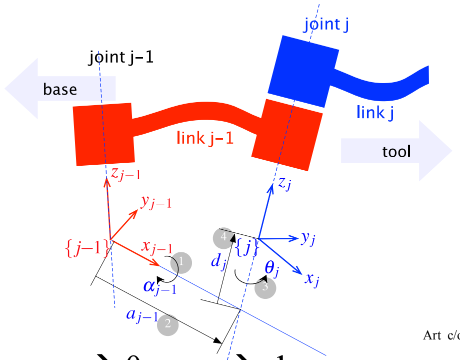
$$^2A_3 = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^3A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & L_3 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^4A_5 = \begin{bmatrix} c_4 & -s_5 & 0 & L_3 \\ 0 & 0 & 1 & d_4 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^5A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & L_3 \\ 0 & 0 & -1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^0T_6 = {}^0A_1{}^1A_2{}^2A_3{}^3A_4{}^4A_5{}^5A_6$$

Modified DH

- Made "popular" by Craig's *Intro. to Robotics* book
- Link coordinates attached to the near by joint

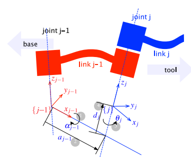


Art c/o P. Corke

- a $_{\text{(trans } x\text{-}1)}$ → α $_{\text{(rot } x\text{-}1)}$ → θ $_{\text{(rot } z)}$ → d $_{\text{(trans } z)}$

Modified DH [2]

- Gives a similar result
  (but it's not commutative)



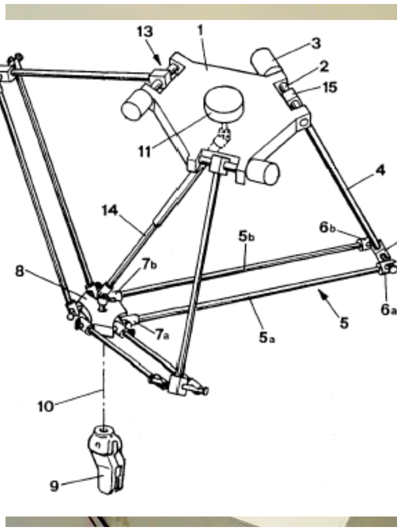
$$\Rightarrow {}^{i-1}A_i = R_x\left(\alpha_{i-1}\right) T_x\left(a_{i-1}\right) R_z\left(\theta_i\right) T_x\left(d_i\right)$$

- Refactoring Standard → to Modified

$$\underbrace{\{R_z(\theta_1)\,T_z(d_1)\,T_x(a_1)\,R_x(\alpha_1)\}}_{\text{DH}_1}\cdot\underbrace{\{R_z(\theta_2)\,T_z(d_2)\,T_x(a_2)\,R_x(\alpha_2)\}}_{\text{DH}_2}\cdot\underbrace{\{R_z(\theta_3)\,T_z(d_3)\}}_{\text{End Effector}}$$

$$=\underbrace{\{R_z(\theta_1)\,T_z(d_1)\}}_{\text{Base}}\cdot\underbrace{\{T_x(a_1)\,R_x(\alpha_1)\,R_z(\theta_2)\,T_z(d_2)\}}_{\text{MDH}_1}\cdot\underbrace{\{T_x(a_2)\,R_x(\alpha_2)\,R_z(\theta_3)\,T_z(d_3)\}}_{\text{MDH}_2}$$

Parallel Manipulators



Sources: Wikipedia, “Delta Robot”, ParallelMic.Org, “Delta Parallel Robot”, and US Patent 4,976,582

- The “central” Kinematic structure is made up of closed-loop chain(s)
- Compared to Serial Mechanisms:
  - \+ Higher Stiffness
  - \+ Higher Payload
  - \+ Less Inertia
  - – Smaller Workspace
  - – Coordinated Drive System
  - – More Complex & $$$

Symmetrical Parallel Manipulator

A sub-class of Parallel Manipulator:

- # Limbs (***m***) = # DOF (***F***)
- The joints are arranged in an identical pattern
- The # and location of actuated joints are the same

Thus:

- Number of Loops (L): One less than # of limbs

$$L = m - 1 = F - 1$$

- Connectivity (C_k)

$$\sum_{k=1}^{m} C_k = (\lambda + 1)\, F - \lambda$$

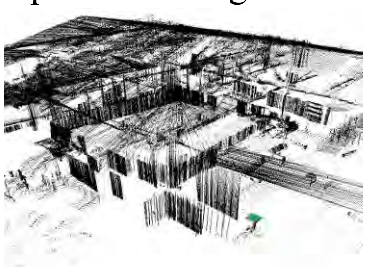
Where: λ: The DOF of the space that the system is in (e.g., λ=6 for 3D space).

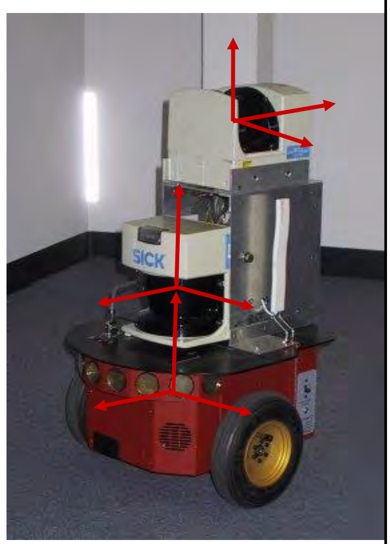
Mobile Platforms

- The preceding kinematic relationships are also important in mobile applications
- When we have sensors mounted on a platform, we need the ability to translate from the sensor frame into some world frame in which the vehicle is operating
- Should we just treat this as a P(*) mechanism?

Mobile Platforms [2]

- We typically assign a frame to the base of the vehicle
- Additional frames are assigned to the sensors
- We will develop these techniques in coming lectures



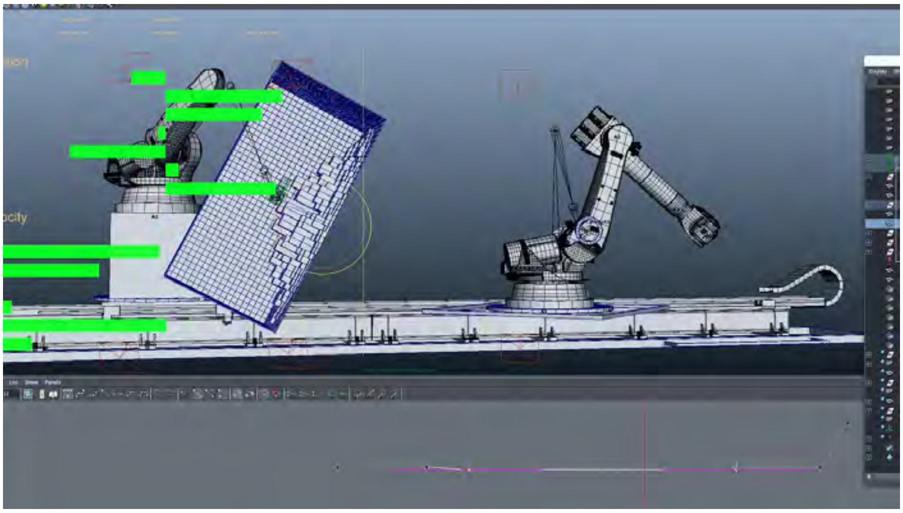


Summary

- Many ways to view a rotation
  - Rotation matrix
  - Euler angles
  - Quaternions
  - Direction Cosines
  - Screw Vectors
- Homogenous transformations
  - Based on homogeneous coordinates

METR 4202: **Robotics** August 3, 2016 -86

Cool Robotics Share



METR 4202: **Robotics** August 3, 2016 -87

Representing Position & Orientation & State

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 3 (Ekka Day)

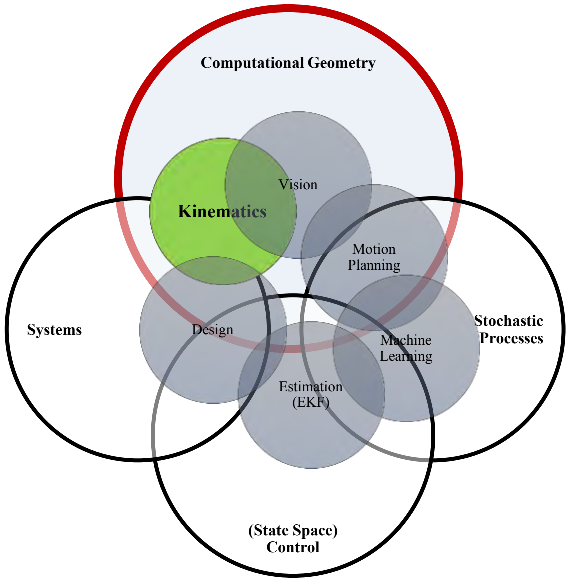
August 10, 2016

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

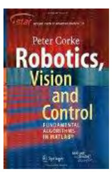
Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	**10-Aug**	**Robot Kinematics Review (&** *Ekka Day*)
4	17-Aug	Robot Dynamics
5	24-Aug	Robot Sensing: Perception
6	31-Aug	Robot Sensing: Multiple View Geometry
7	7-Sep	Robot Sensing: Feature Detection (as Linear Observers)
8	14-Sep	Probabilistic Robotics: Localization
9	21-Sep	Probabilistic Robotics: SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	State-Space Modelling
12	19-Oct	Shaping the Dynamic Response
13	26-Oct	LQR + Course Review

Course Organization



Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

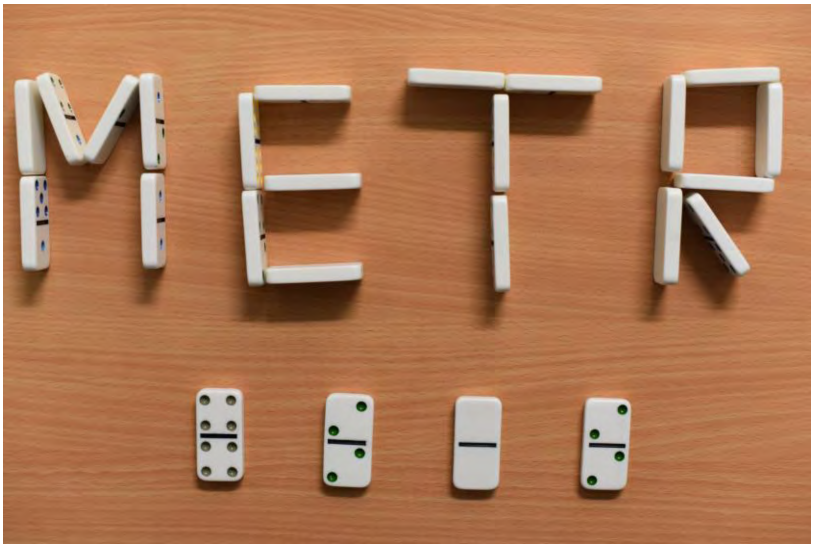
➔ Representing Space 🡐

- RVC
  - Chapter 7: Robot Arm Kinematics
  -

- Inverse Kinematics
  - RVC
    §7.3: Robot Arm Kinematics

Next Time

The Project!
"Robotics: Domino Effect"



Generalizing

Special Orthogonal & Special Euclidean Lie Algebras

- SO(n): R**o**tations

$$SO(n) = \{R \in \mathbb{R}^{n\times n} : RR^T = I, \det R = +1\}.$$

$$\exp(\widehat{\omega}\theta) = e^{\widehat{\omega}\theta} = I + \theta\widehat{\omega} + \frac{\theta^2}{2!}\widehat{\omega}^2 + \frac{\theta^3}{3!}\widehat{\omega}^3 + \ldots$$

- SE(n): Transformations of EUCLIDEAN space

$$SE(n) := \mathbb{R}^n \times SO(n).$$

$$SE(3) = \{(p, R) : p \in \mathbb{R}^3, R \in SO(3)\} = \mathbb{R}^3 \times SO(3).$$

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*

Homogenous Coordinates

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

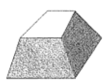
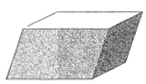
- ρ is a scaling value

Homogenous Transformation

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

- γ is a projective transformation
- The Homogenous Transformation is a **linear operation** (even if projection is not)

Projective Transformations & Other Transformations of 3D Space

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

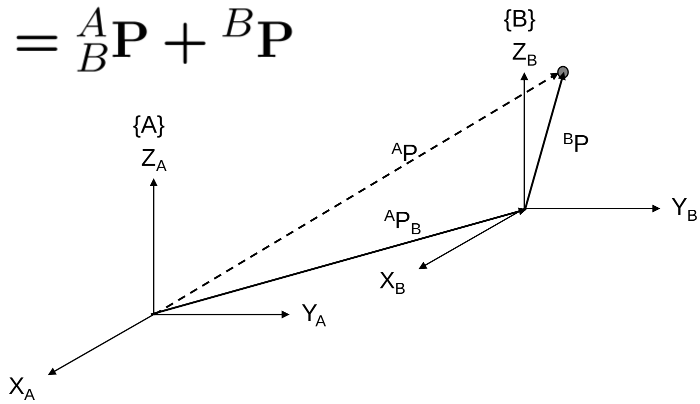
p.78, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

Coordinate Transformations [1]

- Translation Again:

  If {B} is translated with respect to {A} **without rotation**, then it is a vector sum

$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{P} + {}^{B}\mathbf{P}$$



Coordinate Transformations [2]

- Rotation Again:

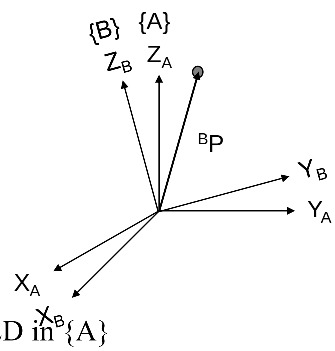
  {B} is rotated with respect to {A} then
  use rotation matrix to determine new components

$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}$$

- NOTE:
  - The Rotation matrix's ***subscript*** matches the position vector's **superscript**

$$^{A}\mathbf{P} = {}^{A}_{[B]}\mathbf{R}\,{}^{[B]}\mathbf{P}$$

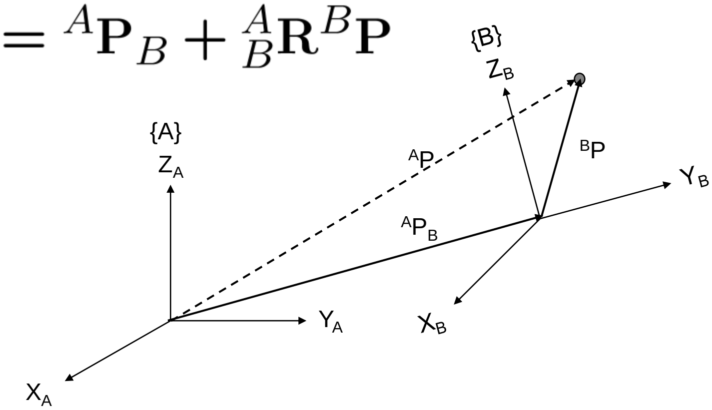
  - This gives Point Positions of {B} ORIENTED in {A}



Coordinate Transformations [3]

- Composite transformation:
  {B} is moved with respect to {A}:

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



METR 4202: **Robotics** August 3, 201613

General Coordinate Transformations [1]

- A compact representation of the translation and rotation is known as the **Homogeneous Transformation**

$$^{A}_{B}\mathbf{T} = \begin{bmatrix} & {}^{A}_{B}\mathbf{R} & & {}^{A}\mathbf{P}_{B} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

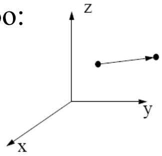
- This allows us to cast the rotation and translation of the general transform in a single matrix form

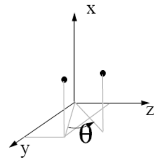
$$\begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = {}^{A}_{B}\mathbf{T} \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix}$$

METR 4202: **Robotics** August 3, 201614

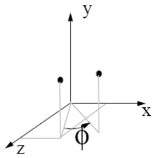
General Coordinate Transformations [2]

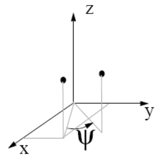
- Similarly, fundamental orthonormal transformations can be represented in this form too:





$$Trans(u, v, w) = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Rotx(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & 0 \\ 0 & s\theta & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



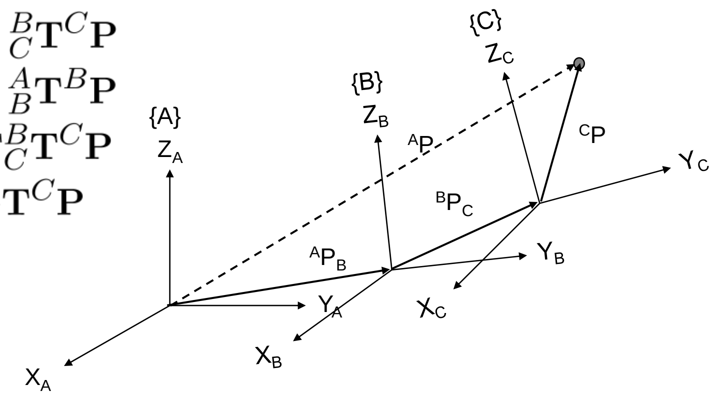


$$Roty(\phi) = \begin{bmatrix} c\phi & 0 & s\phi & 0 \\ 0 & 1 & 0 & 0 \\ -s\phi & 0 & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Rotz(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 & 0 \\ s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Coordinate Transformations [3]

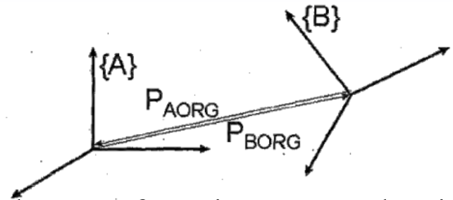
- Multiple transformations compounded as a chain

$$^{B}\mathbf{P} = {}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P}$$
$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{T}\,{}^{B}\mathbf{P}$$
$$= {}^{A}_{B}\mathbf{T}\,{}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P}$$
$$= {}^{A}_{C}\mathbf{T}\,{}^{C}\mathbf{P}$$



$$^{A}_{C}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R}\,{}^{B}_{C}\mathbf{R} & {}^{A}\mathbf{P}_{B} + {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}_{C} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Inverse of a Homogeneous Transformation Matrix



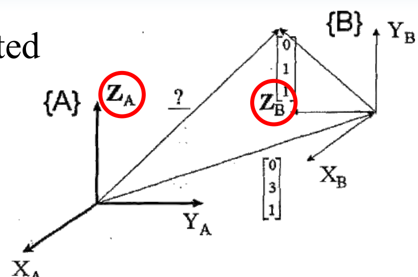
- The inverse of the transform is **<u>not</u>** equal to its transpose because this 4×4 matrix is not orthonormal ($T^{-1} \neq T^T$)
- Invert by parts to give:

$$ {}^{A}_{B}T = \begin{bmatrix} & {}^{A}_{B}R & & {}^{A}\mathbf{p}_{Borg/O_A} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{A}_{B}T^{-1} = {}^{B}_{A}T = \begin{bmatrix} & {}^{A}_{B}R^T & & -{}^{A}_{B}R^T \cdot {}^{A}\mathbf{p}_{Borg/O_A} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & {}^{B}_{A}R & & {}^{B}\mathbf{p}_{Aorg/O_B} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Tutorial Problem

The origin of frame *{B}* is translated to a position [0 3 1] with respect to frame *{A}*.



We would like to find:

1. The homogeneous transformation between the two frames in the figure.
2. For a point *P* defined as as [0 1 1] in frame *{B}*, we would like to find the vector describing this point with respect to frame *{A}*.

Tutorial Solution

• The matrix ${}_BT^A$ is formed as defined earlier:

- The matrix ${}_BT^A$ is formed as defined earlier:

$$^A_BT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

• Since P in the frame is:

• We find vector **p** in frame *{A}* using the relationship

➔

- Since P in the frame is: $^B\mathbf{p} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

- We find vector **p** in frame *{A}* using the relationship

$$^Ap = {}^A_BT\,{}^Bp$$

➔ $^A\mathbf{p} = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 1 \end{bmatrix}$

Cool Robotics Share

http://www.kinemasystems.com/

Looking in Detail: Forward & Inverse Kinematics

1. Forward Kinematics ($\theta \rightarrow x$)
2. Inverse Kinematics ($x \rightarrow \theta$)
3. Denavit Hartenberg [DH] Notation
4. Affine Transformations &
5. Theoretical (General) Kinematics

Forward Kinematics

Forward Kinematics [1]

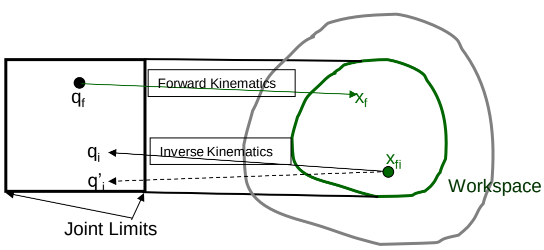
- Forward kinematics is the process of chaining homogeneous transforms together. For example to:
  - Find the articulations of a mechanism, or
  - the fixed transformation between two frames which is known in terms of linear and rotary parameters.
- Calculates the final position from the **machine** (**joint variables**)
- Unique for an open kinematic chain (**serial arm**)
- "Complicated" (multiple solutions, etc.) for a closed kinematic chain (**parallel arm**)

Forward Kinematics [2]

- Can think of this as "spaces":
  - Workspace $(x,y,z,\alpha,\beta,\gamma)$: The robot's position & orientation
  - Joint space $(\theta_1 \ldots \theta_n)$: A state-space vector of joint variables

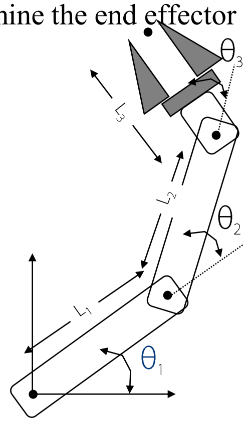
$$\vec{x} = \begin{bmatrix} \vec{p} \\ \vec{\Theta} \end{bmatrix}$$

$$\vec{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$



Forward Kinematics [3]

- Consider a planar RRR manipular
- Given the joint angles and link lengths, we can determine the end effector pose:



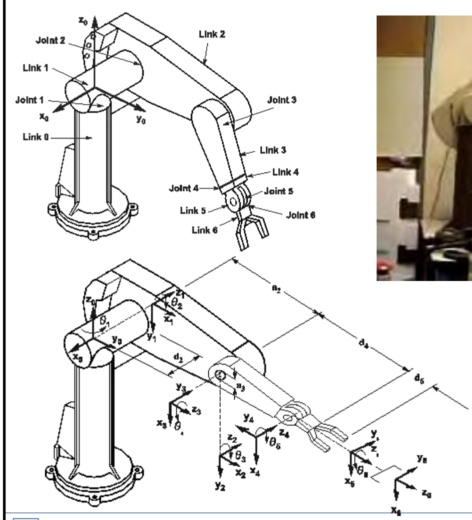
$$
\begin{aligned}
x =& L_1 \cos\theta_1 + L_2 \cos(\theta_1+\theta_2) + \ldots \\
& L_3 \cos(\theta_1+\theta_2+\theta_3)
\end{aligned}
$$

$$
\begin{aligned}
y =& L_1 \sin\theta_1 + L_2 \sin(\theta_1+\theta_2) + \ldots \\
& L_3 \sin(\theta_1+\theta_2+\theta_3)
\end{aligned}
$$

- This isn't too difficult to determine for a simple, planar manipulator. BUT …

Forward Kinematics [4]: The PUMA 560!

- What about a more complicated mechanism?



$$
\begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Denavit Hartenberg [DH] Notation

Denavit Hartenberg [DH] Notation

- J. Denavit and R. S. Hartenberg first proposed the use of homogeneous transforms for articulated mechanisms

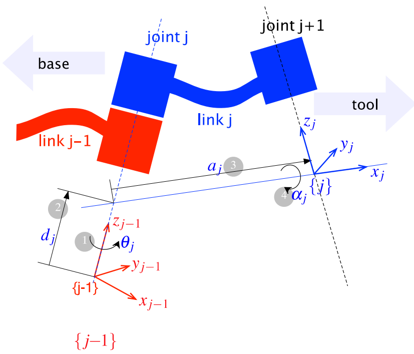
  (But B. Roth, introduced it to robotics)

- A kinematics "short-cut" that reduced the number of parameters by adding a structure to frame selection

- For two frames positioned in space, the first can be moved into coincidence with the second by a sequence of 4 operations:
  - rotate around the x_{i-1} axis by an angle α_i
  - translate along the x_{i-1} axis by a distance a_i
  - translate along the new z axis by a distance d_i
  - rotate around the new z axis by an angle θ_i

Denavit-Hartenberg Convention

- link length a_i the offset distance between the z_{i-1} and z_i axes along the x_i axis;
- link twist α_i the angle from the z_{i-1} axis to the z_i axis about the x_i axis;



Art c/o P. Corke

- link offset d_i the distance from the origin of frame i-1 to the x_i axis along the z_{i-1} axis;
- joint angle θ_i the angle between the x_{i-1} and x_i axes about the z_{i-1} axis.

DH: Where to place frame?

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)

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

METR 4202: **Robotics** August 3, 201631

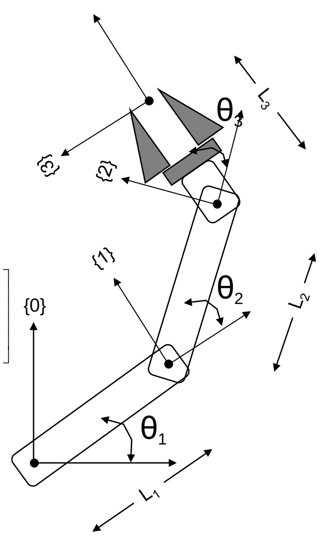
DH Example [1]: RRR Link Manipulator

1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	θ_3

$$
{}^0A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1 c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1 s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
{}^1A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2 s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
{}^2A_3 = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & L_3 c_{\theta_3} \\ s_{\theta_3} & c_{\theta_3} & 0 & L_3 s_{\theta_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0T_3 = {}^0A_1 \, {}^1A_2 \, {}^2A_3
= \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1 c_{\theta_1} + L_2 c_{\theta_{12}} + L_3 c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1 s_{\theta_1} + L_2 s_{\theta_{12}} + L_3 s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$



METR 4202: **Robotics** August 3, 201632

DH Example [2]: RRP Link Manipulator

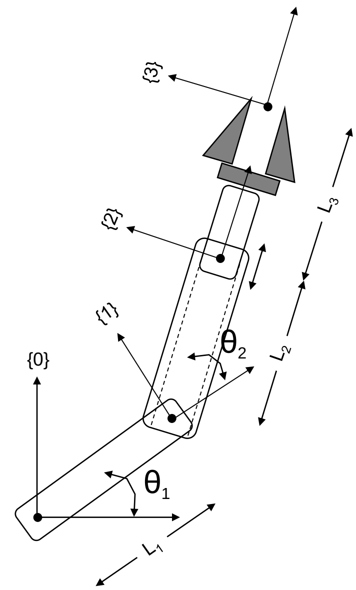
1. Assign the frames at the joints …
2. Fill DH Table …

Link	a_i	α_i	d_i	θ_i
1	L_1	0	0	$\boldsymbol{\theta_1}$
2	L_2	0	0	$\boldsymbol{\theta_2}$
3	$\mathbf{L_3}$	0	0	0

$$
{}^0A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & L_1 c_{\theta_1} \\ s_{\theta_1} & c_{\theta_1} & 0 & L_1 s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\;{}^1A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2 s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\;{}^2A_3 = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0T_3 = {}^0A_1\,{}^1A_2\,{}^2A_3
$$

$$
= \begin{bmatrix} c_{\theta_{12}} & -s_{\theta_{12}} & 0 & L_1 c_{\theta_1} + (L_2 + L_3) c_{\theta_{12}} \\ s_{\theta_{12}} & c_{\theta_{12}} & 0 & L_1 s_{\theta_1} + (L_2 + L_3) s_{\theta_{12}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$



DH Example [3]: Puma 560

- "Simple" 6R robot exercise for the reader …

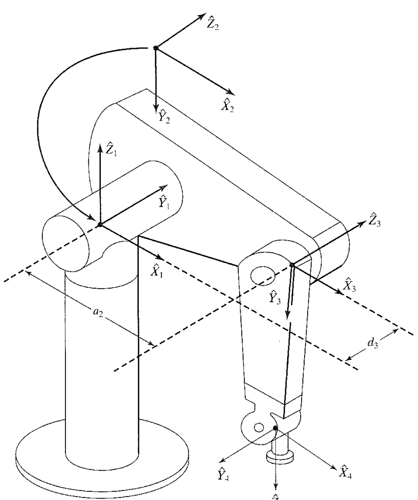
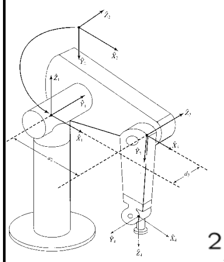


Image: J. Craig, *Introduction to Robotics* 3rd Ed., 2005

Link	a_i	α_i	d_i	θ_i
1	0	0	0	$\boldsymbol{\theta_1}$
2	0	-π/2	0	$\boldsymbol{\theta_2}$
3	L_2	0	D_3	$\boldsymbol{\theta_3}$
4	L_3	-π/2	D_4	$\boldsymbol{\theta_4}$
5	0	π/2	0	$\boldsymbol{\theta_5}$
6	0	-π/2	0	$\boldsymbol{\theta_6}$

DH Example [3]: Puma 560 [2]



$$^{0}A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

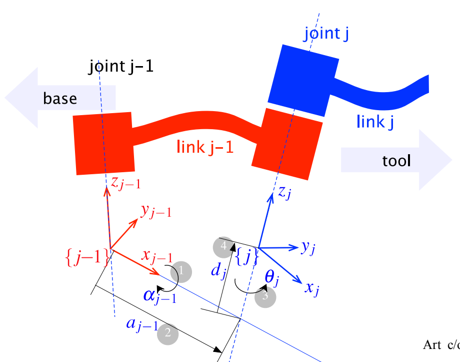
$$^{2}A_3 = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{3}A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & L_3 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}A_5 = \begin{bmatrix} c_4 & -s_5 & 0 & L_3 \\ 0 & 0 & 1 & d_4 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & L_3 \\ 0 & 0 & -1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{0}T_6 = {}^{0}A_1 {}^{1}A_2 {}^{2}A_3 {}^{3}A_4 {}^{4}A_5 {}^{5}A_6$$

Modified DH

- Made “popular” by Craig’s *Intro. to Robotics* book
- Link coordinates attached to the near by joint

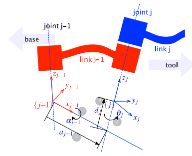


Art c/o P. Corke

- a (trans *x-1*) → α (rot *x-1*) → θ (rot *z*) → d (trans *z*)

Modified DH [2]

- Gives a similar result
  (but it's not commutative)



$$\Rightarrow {}^{i-1}A_i = R_x(\alpha_{i-1})\, T_x(a_{i-1})\, R_z(\theta_i)\, T_x(d_i)$$

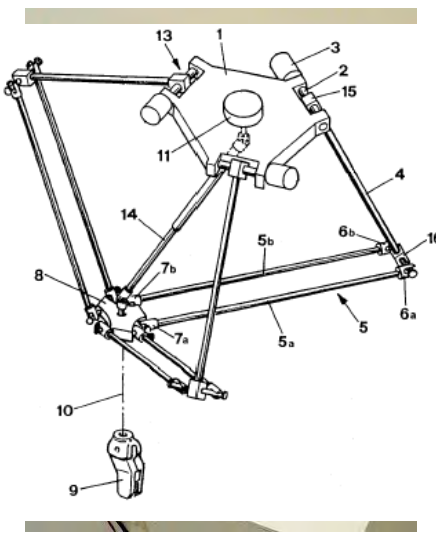
- Refactoring Standard → to Modified

$$\underbrace{\{R_z(\theta_1)\,T_z(d_1)\,T_x(a_1)\,R_x(\alpha_1)\}}_{\text{DH}_1}\cdot\underbrace{\{R_z(\theta_2)\,T_z(d_2)\,T_x(a_2)\,R_x(\alpha_2)\}}_{\text{DH}_2}\cdot\underbrace{\{R_z(\theta_3)\,T_z(d_3)\}}_{\text{End Effector}}$$

$$=\underbrace{\{R_z(\theta_1)\,T_z(d_1)\}}_{\text{Base}}\cdot\underbrace{\{T_x(a_1)\,R_x(\alpha_1)\,R_z(\theta_2)\,T_z(d_2)\}}_{\text{MDH}_1}\cdot\underbrace{\{T_x(a_2)\,R_x(\alpha_2)\,R_z(\theta_3)\,T_z(d_3)\}}_{\text{MDH}_2}$$

METR 4202: **Robotics** August 3, 201638

Parallel Manipulators



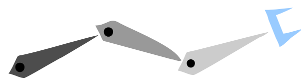
Sources: Wikipedia, "Delta Robot", ParallelMic.Org, "Delta Parallel Robot", and US Patent 4,976,582

- The "central" Kinematic structure is made up of closed-loop chain(s)
- Compared to Serial Mechanisms:
  + Higher Stiffness
  + Higher Payload
  + Less Inertia
  – Smaller Workspace
  – Coordinated Drive System
  – More Complex & $$$

METR 4202: **Robotics** August 3, 201639

Inverse Kinematics

- Forward: angles → position
  $\mathbf{x} = f(\boldsymbol{\theta})$
- Inverse: position → angles
  $\boldsymbol{\theta} = f^{-1}(\mathbf{x})$
- Analytic Approach



- Numerical Approaches:
  - Jacobian: $J = \frac{\delta x}{\delta q} \rightarrow \delta q \approx J^{-1}\delta x$
  - J^T Approximation: $\tau = J^T \cdot \mathbf{F} \rightarrow \Delta q \approx J^T \Delta x$
    - Slotine & Sheridan method
  - Cyclical Coordinate Descent

Inverse Kinematics

- Inverse Kinematics is the problem of finding the joint parameters given only the values of the homogeneous transforms which model the mechanism (i.e., the pose of the end effector)
- Solves the problem of where to drive the joints in order to get the hand of an arm or the foot of a leg in the right place
- In general, this involves the solution of a set of simultaneous, non-linear equations
- Hard for serial mechanisms, easy for parallel

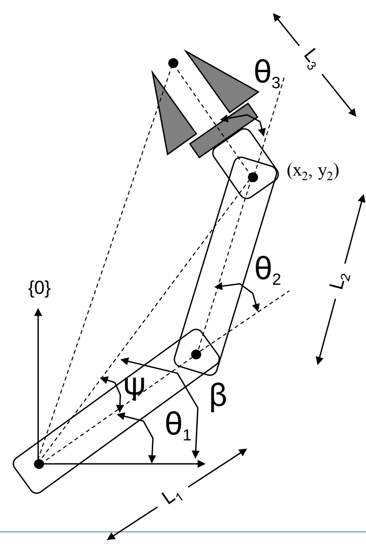
Solution Methods

- Unlike with systems of linear equations, there are no general algorithms that may be employed to solve a set of nonlinear equation
- **Closed-form** and **numerical** methods exist
- Many exist: Most general solution to a 6R mechanism is Raghavan and Roth (1990)
- Three methods of obtaining a solution are popular:
  (1) **geometric** | (2) **algebraic** | (3) **DH**

METR 4202: **Robotics** August 3, 201642

Inverse Kinematics: Geometrical Approach

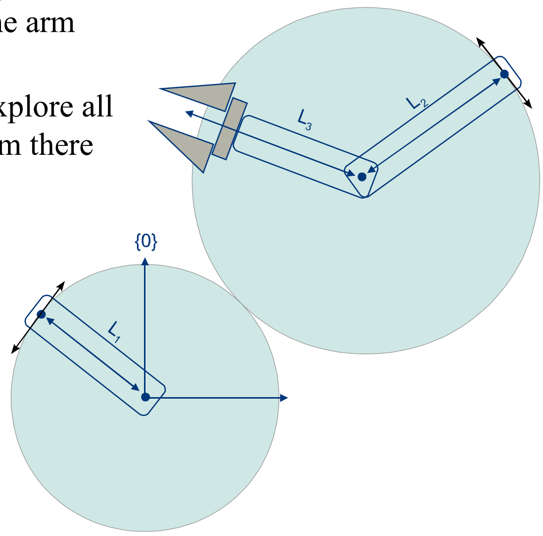
- We can also consider the geometric relationships defined by the arm



METR 4202: **Robotics** August 3, 201643

Inverse Kinematics: Geometrical Approach [2]

- We can also consider the geometric relationships defined by the arm
- Start with what is fixed, explore all geometric possibilities from there



Inverse Kinematics: Algebraic Approach

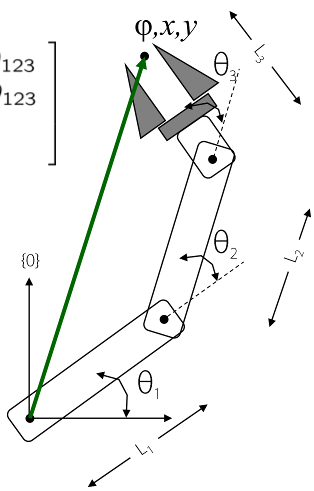
- We have a series of equations which define this system
- Recall, from Forward Kinematics:

$$ {}^0T_3 = \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & L_1c_{\theta_1} + L_2c_{\theta_{12}} + L_3c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & L_1s_{\theta_1} + L_2s_{\theta_{12}} + L_3s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The end-effector pose is given by

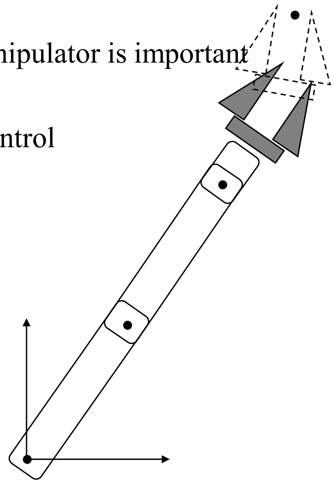
$$ {}^0T_3 = \begin{bmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- Equating terms gives us a set of algebraic relationships



No Solution - Singularity

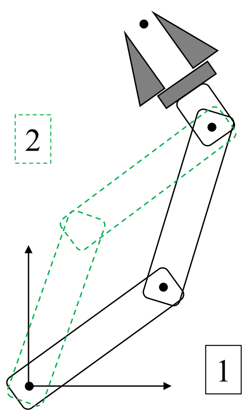
- Singular positions:
- An understanding of the workspace of the manipulator is important
- There will be poses that are not achievable
- There will be poses where there is a loss of control
- Singularities also occur when the manipulator loses a DOF
  - This typically happens when joints are aligned
  - **det[Jacobian]=0**



METR 4202: **Robotics** August 3, 201646

Multiple Solutions

- There will often be multiple solutions for a particular inverse kinematic analysis
- Consider the three link manipulator shown. Given a particular end effector pose, two solutions are possible
- The choice of solution is a function of proximity to the current pose, limits on the joint angles and possible obstructions in the workspace



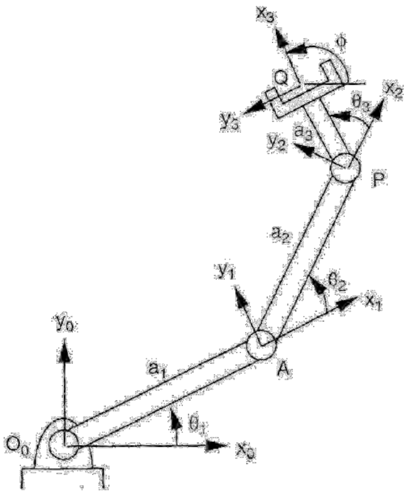
METR 4202: **Robotics** August 3, 201647

Inverse Kinematics

Inverse Kinematics [More Generally]

- Freudenstein (1973) referred to the inverse kinematics problem of the most general **6R** manipulator as the "Mount Everest" of kinematic problems.
- Tsai and Morgan (1985) and Primrose (1986) proved that this has at most 16 real solutions.
- Duffy and Crane (1980) derived a closed-form solution for the general **7R** single-loop spatial mechanism.
  - The solution was obtained in the form of a 16 x 16 delerminant in which every element is a second-degree polynomial in one joint variable. The determinant, when expended, should yield a 32nd-degree polynomial equation and hence confirms the upper limit predicted by Roth *et al*. (1973).
- Tsai and Morgan (1985) used the homotopy continuation method to solve the inverse kinematics of the general 6R manipulator and found only 16 solutions
- Raghavan and Roth (1989, 1990) used the dyalitic elimination method to derive a 16th-degree polynomial for the general 6R inverse kinematics problem.

Example: FK/IK of a 3R Planar Arm



- Derived from Tsai (p. 63)

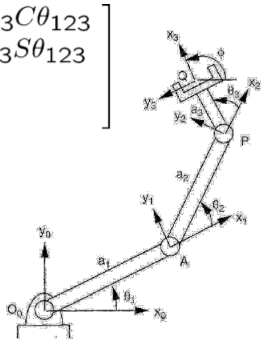
Example: 3R Planar Arm [2]

Position Analysis: 3·Planar 1-R Arm rotating about **Z** **[Ⓩ]**

$${}^0A_3 = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3$$

Substituting gives:

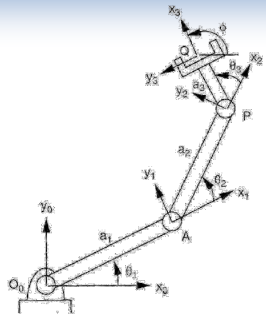
$${}^0A_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 & a_1C\theta_1 + a_2C\theta_{12} + a_3C\theta_{123} \\ S\theta_{123} & C\theta_{123} & 0 & a_1S\theta_1 + a_2S\theta_{12} + a_3S\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Example: 3R Planar Arm [2]

Forward Kinematics

(solve for **x given θ** → $\mathbf{x} = f(\boldsymbol{\theta})$)



Fairly straight forward:

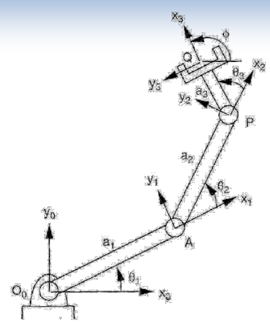
$$
{}^0R_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 \\ S\theta_{123} & C\theta_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0P_3 = \begin{bmatrix} a_1C\theta_1 + a_2C\theta_{12} + a_3C\theta_{123} \\ a_1S\theta_1 + a_2S\theta_{12} + a_3S\theta_{123} \\ 0 \end{bmatrix}
$$

Example: 3R Planar Arm [3]

Inverse Kinematics

(solve for **θ given x** → $\mathbf{x} = f(\boldsymbol{\theta})$)



- Start with orientation φ:

$C\theta_{123} = C\phi,\ S\theta_{123} = S\phi$

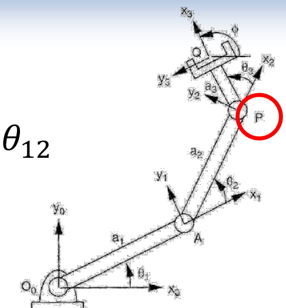
$\Rightarrow \theta_{123} = \theta_1 + \theta_2 + \theta_3 = \phi$

- Get overall position $\boldsymbol{q} = [q_x \quad q_y]$:

$q_x - a_3C\phi = a_1C\theta_1 + a_2C\theta_{12}$

$q_y - a_3S\phi = a_1S\theta_1 + a_2S\theta_{12}$ …

Example: 3R Planar Arm [4]



- Introduce $\boldsymbol{p} = [p_x \quad p_y]$ before "wrist"

$p_x = a_1 C\theta_1 + a_2 C\theta_{12}, p_y = a_1 S\theta_1 + a_2 S\theta_{12}$
$\Rightarrow p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1 a_2 C\theta_2$

- Solve for θ_2:

$\theta_2 = \cos^{-1}\kappa, \kappa = \frac{p_x^2+p_y^2-a_1^2-a_2^2}{2a_1 a_2}$ (2 $\mathbb{R}$ roots if $|\kappa|<1$)

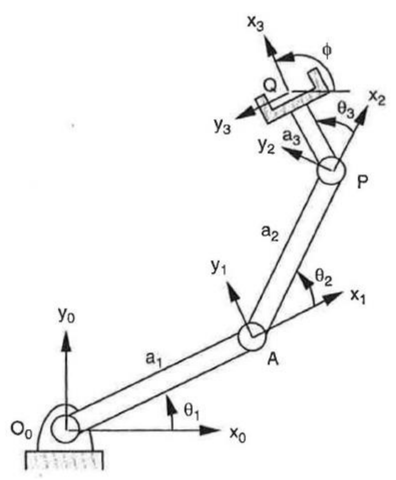
- Solve for θ_1:

$C\theta_1 = \frac{p_x(a_1+a_2C\theta_2)+p_y a_2 S\theta_2}{a_1^2+a_2^2+2a_1a_2C\theta_2}, S\theta_1 = \frac{-p_x a_2 S\theta_2+p_y(a_1+a_2C\theta_2)}{a_1^2+a_2^2+2a_1a_2C\theta_2}$

$\theta_1 = atan2(S\theta_1, C\theta_1)$

Inverse Kinematics: Example I

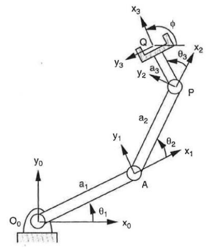
Planar Manipulator:



Inverse Kinematics: Example I

- Forward Kinematics:

[For the Frame {Q} at the end effector]:



$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

$\because$

$${}^0A_3 = \begin{bmatrix} c\theta_{123} & -s\theta_{123} & 0 & a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ s\theta_{123} & c\theta_{123} & 0 & a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

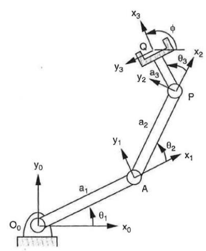
- For an arbitrary point **G** in the end effector: ${}^3\mathbf{g} = [g_u, g_v, 0, 1]^T$

$$\begin{bmatrix} g_x \\ g_y \\ g_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} g_u \\ g_v \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} g_u c\theta_{123} - g_v s\theta_{123} + a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ g_u s\theta_{123} + g_v c\theta_{123} + a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

Inverse Kinematics: Example I

- Forward Kinematics:

[For the Frame {Q} at the end effector]:



$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

$\because$

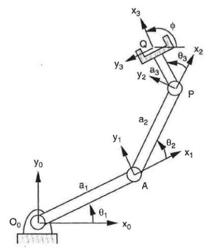
$${}^0A_3 = \begin{bmatrix} c\theta_{123} & -s\theta_{123} & 0 & a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ s\theta_{123} & c\theta_{123} & 0 & a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- For an arbitrary point **G** in the end effector: ${}^3\mathbf{g} = [g_u, g_v, 0, 1]^T$

$$\begin{bmatrix} g_x \\ g_y \\ g_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} g_u \\ g_v \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} g_u c\theta_{123} - g_v s\theta_{123} + a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ g_u s\theta_{123} + g_v c\theta_{123} + a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

Inverse Kinematics: Example I

- Inverse Kinematics:
  - Set the final position equal to the Forward Transformation Matrix $^{0}\mathbf{A}_3$:



$$
{}^{0}A_3 = \begin{bmatrix} c\phi & -s\phi & 0 & q_x \\ s\phi & c\phi & 0 & q_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

- The solution strategy is to equate the elements of $^{0}\mathbf{A}_3$ to that of the given position (q_x, q_y) and orientation ϕ

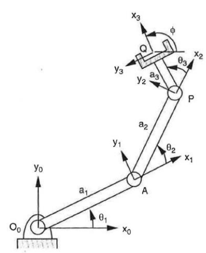
Inverse Kinematics: Example I

- Orientation (ϕ):

$$c\theta_{123} = c\phi,$$

$$s\theta_{123} = s\phi.$$

$$\theta_{123} = \theta_1 + \theta_2 + \theta_3 = \phi.$$



- Now Position of the 2DOF point **P**:

$$p_x = a_1 c\theta_1 + a_2 c\theta_{12},$$

$$p_y = a_1 s\theta_1 + a_2 s\theta_{12},$$

$\therefore \quad p_x = q_x - a_3 c\phi \qquad p_y = q_y - a_3 s\phi$

- Substitute: θ_3 disappears and now we can eliminate θ_1:

$$p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1 a_2 c\theta_2.$$

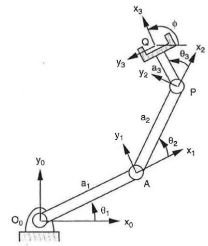
Inverse Kinematics: Example I

- we can eliminate θ_1…

  $p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1a_2c\theta_2.$

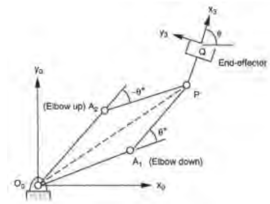
- Then solve for θ_{12}:

  $$\theta_2 = \cos^{-1}\kappa, \qquad \kappa = \frac{p_x^2 + p_y^2 - a_1^2 - a_2^2}{2a_1a_2}$$



  – This gives 2 real ($\mathbb{R}$) roots if $|\kappa| < 1$
  – One double root if $|\kappa| = 1$
  – No real roots if $|\kappa| > 1$

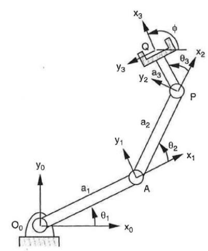
- Elbow up/down:
  – In general, **if** θ_2 is a solution **then** $-\theta_2$ is a solution



Inverse Kinematics: Example I

- Solving for θ_1…
  – Corresponding to each θ_2, we can solve θ_1

  $$(a_1 + a_2c\theta_2)c\theta_1 - (a_2s\theta_2)s\theta_1 = p_x$$
  $$(a_2s\theta_2)c\theta_1 + (a_1 + a_2c\theta_2)s\theta_1 = p_y$$



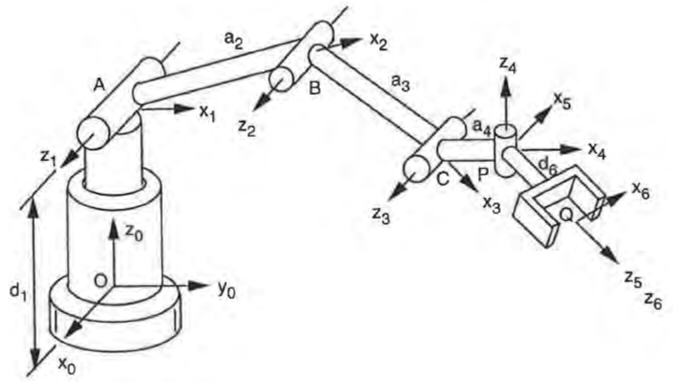
  $$c\theta_1 = \frac{p_x(a_1 + a_2c\theta_2) + p_ya_2s\theta_2}{\Delta},$$
  $$s\theta_1 = \frac{-p_xa_2s\theta_2 + p_y(a_1 + a_2c\theta_2)}{\Delta}$$

  $$\Delta = a_1^2 + a_2^2 + 2a_1a_2c\theta_2$$

  $$\boxed{\theta_1 = \mathrm{Atan2}(s\theta_1, c\theta_1)}$$

Inverse Kinematics: Example II

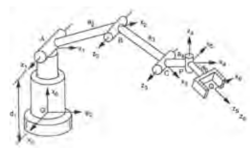
Elbow Manipulator:



Inverse Kinematics: Example II

- Target Position:

$\mathbf{u} = [u_x, u_y, u_z]^{\mathrm{T}}, \quad \mathbf{v} = [v_x, v_y, v_z]^{\mathrm{T}}, \quad \mathbf{w} = [w_x, w_y, w_z]^{\mathrm{T}}$, and
$\mathbf{p} = [p_x, p_y, p_z]^{\mathrm{T}}$.



- Transformation Matrices:

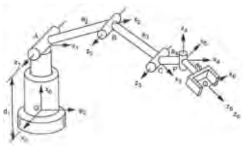
$$A_1 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad ({}^0A_1)^{-1} = \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & 0 \\ -s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ s\theta_2 & 0 & c\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} c\theta_3 & 0 & -s\theta_3 & a_2(1 - c\theta_3) \\ 0 & 1 & 0 & 0 \\ s\theta_3 & 0 & c\theta_3 & -a_2 s\theta_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4 = \begin{bmatrix} c\theta_4 & 0 & -s\theta_4 & (a_2 + a_3)(1 - c\theta_4) \\ 0 & 1 & 0 & 0 \\ s\theta_4 & 0 & c\theta_4 & -(a_2 + a_3)s\theta_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5 = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & (a_2 + a_3 + a_4)(1 - c\theta_5) \\ s\theta_5 & c\theta_5 & 0 & -(a_2 + a_3 + a_4)s\theta_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics: Example II

- Key Matrix Products:

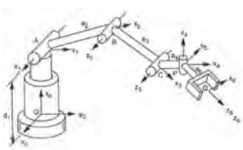


$$A_2A_3A_4 = \begin{bmatrix} c\theta_{234} & 0 & -s\theta_{234} & a_2c\theta_2 + a_3c\theta_{23} - (a_2+a_3)c\theta_{234} \\ 0 & 1 & 0 & 0 \\ s\theta_{234} & 0 & c\theta_{234} & a_2s\theta_2 + a_3s\theta_{23} - (a_2+a_3)s\theta_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1A_2A_3A_4 = \begin{bmatrix} c\theta_1c\theta_{234} & -s\theta_1 & -c\theta_1s\theta_{234} & c\theta_1[a_2c\theta_2 + a_3c\theta_{23} - (a_2+a_3)c\theta_{234}] \\ s\theta_1c\theta_{234} & c\theta_1 & -s\theta_1s\theta_{234} & s\theta_1[a_2c\theta_2 + a_3c\theta_{23} - (a_2+a_3)c\theta_{234}] \\ s\theta_{234} & 0 & c\theta_{234} & [a_2s\theta_2 + a_3s\theta_{23} - (a_2+a_3)s\theta_{234}] \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics: Example II

- Inverse Kinematics:



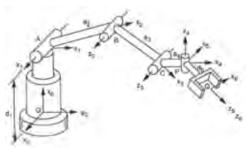
$$\mathbf{p} = A_1A_2A_3A_4\mathbf{p}_0.$$

$$A_1^{-1}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = A_2A_3A_4\begin{bmatrix} a_2+a_3+a_4 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$p_xc\theta_1 + p_ys\theta_1 = a_2c\theta_2 + a_3c\theta_{23} + a_4c\theta_{234},$$
$$-p_xs\theta_1 + p_yc\theta_1 = 0,$$
$$p_z = a_2s\theta_2 + a_3s\theta_{23} + a_4s\theta_{234}.$$

Inverse Kinematics: Example II

- Solving the System:



$$\theta_1 = \tan^{-1}\frac{p_y}{p_x}.$$

$$\theta_5 = \sin^{-1}(-w_x s\theta_1 + w_y c\theta_1).$$

$$\theta_{234} = \mathrm{Atan2}\left[w_z/c\theta_5,\ (w_x c\theta_1 + w_y s\theta_1)/c\theta_5\right].$$

$$a_2 c\theta_2 + a_3 c\theta_{23} = k_1, \qquad k_1 = p_x c\theta_1 + p_y s\theta_1 - a_4 c\theta_{234}$$
$$a_2 s\theta_2 + a_3 s\theta_{23} = k_2, \qquad k_2 = p_z - a_4 s\theta_{234}$$

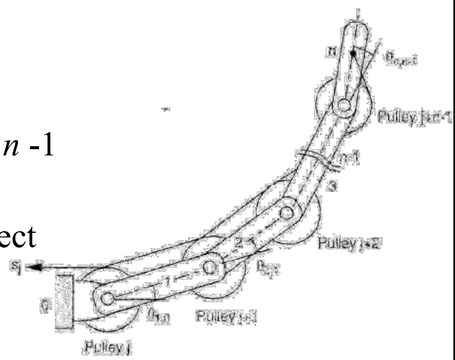
$$a_2^2 + a_3^2 + 2a_2 a_3 c\theta_3 = k_1^2 + k_2^2.$$

$$\theta_3 = \cos^{-1}\frac{k_1^2 + k_2^2 - a_2^2 - a_3^2}{2a_2 a_3}.$$

$$\theta_6 = \mathrm{Atan2}(s\theta_6, c\theta_6).$$

Advanced Concept: Tendon-Driven Manipulators

- Tendons may be modelled as a transmission line
- in which the links are labeled sequentially from 0 to n and the pulleys are labeled from *j* to *j* + *n* -1
- Let θ_{ji} denote the angular displacement of link *j* with respect to link *i*.
- We can write a circuit equation once for each pulley pair as follows:



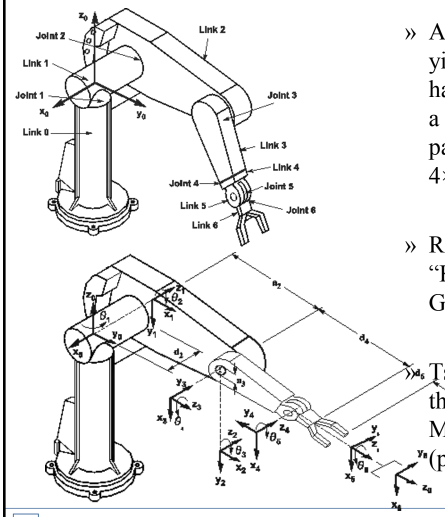
$$r_{j+i-1}\theta_{j+i-1,i} = \pm r_{j+i}\theta_{j+i,i} \qquad \text{for } i = 1, 2, \ldots, n-1.$$

$$\theta_{j+i-1,i} = \theta_{j+i-1,i-1} - \theta_{i,i-1} \qquad \text{for } i = 1, 2, \ldots, n.$$

$$\theta_{j,0} = \theta_{1,0} \pm (r_{j+1}/r_j)\theta_{2,1} \pm \cdots \pm (r_{j+n-1}/r_j)\theta_{n,n-1}.$$

Inverse Kinematics

- What about a more complicated mechanism?



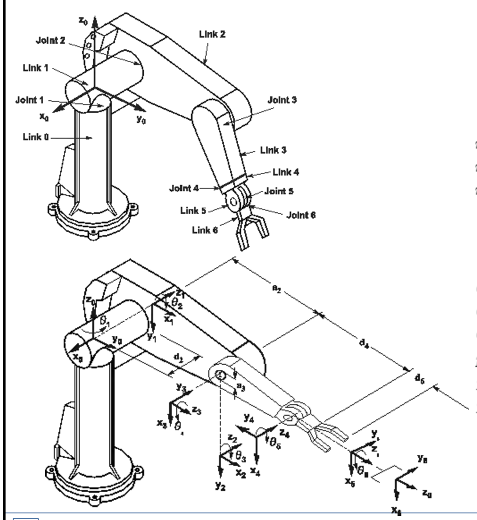
» A sufficient condition for a serial manipulator to yield a closed-form inverse kinematics solution is to have any three consecutive joint axes intersecting at a common point or any three consecutive joint axes parallel to each other. (Pieper and Roth (1969) via 4×4 matrix method)

» Raghavan and Roth 1990
"Kinematic Analysis of the 6R Manipulator of General Geometry"

» Tsai and Morgan 1985, "Solving the Kinematics of the Most General Six and Five-Dcgree-of-Freedom Manipulators by Continuation Methods" (posted online)

Inverse Kinematics

- What about a more complicated mechanism?



$$
{}^0\boldsymbol{T}_6 = {}^0\boldsymbol{T}_1\,{}^1\boldsymbol{T}_2\,{}^2\boldsymbol{T}_3\,{}^3\boldsymbol{T}_4\,{}^4\boldsymbol{T}_5\,{}^5\boldsymbol{T}_6 = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
\begin{aligned}
n_x &= c_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) - s_1(s_4c_5c_6 + c_4s_6) \\
n_y &= s_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) + c_1(s_4c_5c_6 + c_4s_6) \\
n_z &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \\
s_x &= c_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) - s_1(-s_4c_5s_6 + c_4c_6) \\
s_y &= s_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) + c_1(-s_4c_5s_6 + c_4c_6) \\
s_z &= s_{23}(c_4c_5s_6 + s_4c_6) - c_{23}s_5s_6 \\
a_x &= c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 \\
a_y &= s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5 \\
a_z &= -s_{23}c_4s_5 + c_{23}c_5) \\
p_x &= c_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) - s_1(d_6s_4s_5 + d_2) \\
p_y &= s_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) + c_1(d_6s_4s_5 + d_2) \\
p_z &= d_6(c_{23}c_5 - s_{23}c_4s_5) + c_{23}d_4 - a_3s_{23} - a_2s_2
\end{aligned}
$$

Symmetrical Parallel Manipulator

A sub-class of Parallel Manipulator:

- o # Limbs (***m***) = # DOF (***F***)
- o The joints are arranged in an identical pattern
- o The # and location of actuated joints are the same

Thus:

- o Number of Loops (L): One less than # of limbs

$$L = m - 1 = F - 1$$

- o Connectivity (C_k)

$$\sum_{k=1}^{m} C_k = (\lambda + 1)\, F - \lambda$$

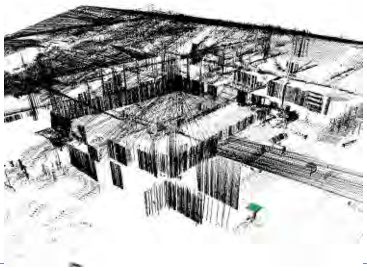
Where: λ: The DOF of the space that the system is in (e.g., λ=6 for 3D space).

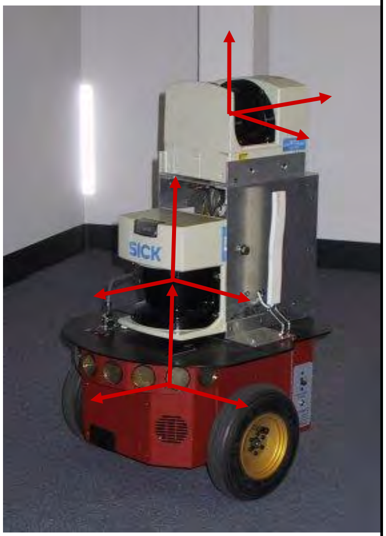
Mobile Platforms

- The preceding kinematic relationships are also important in mobile applications
- When we have sensors mounted on a platform, we need the ability to translate from the sensor frame into some world frame in which the vehicle is operating
- Should we just treat this as a P(*) mechanism?

Mobile Platforms [2]

- We typically assign a frame to the base of the vehicle
- Additional frames are assigned to the sensors
- We will develop these techniques in coming lectures

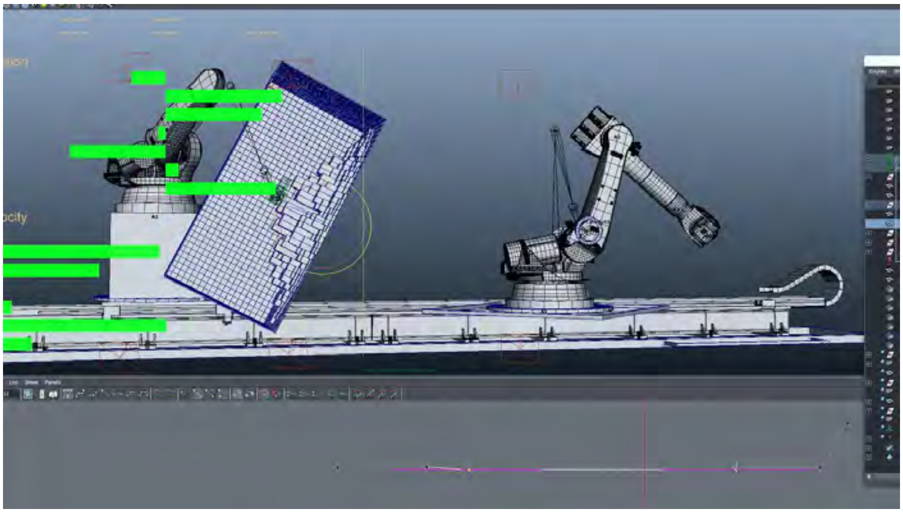




Summary

- Many ways to view a rotation
  - Rotation matrix
  - Euler angles
  - Quaternions
  - Direction Cosines
  - Screw Vectors
- Homogenous transformations
  - Based on homogeneous coordinates

Cool Robotics Share



METR 4202: **Robotics** August 3, 201676

Inverse Kinematics & Kinetics

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 4

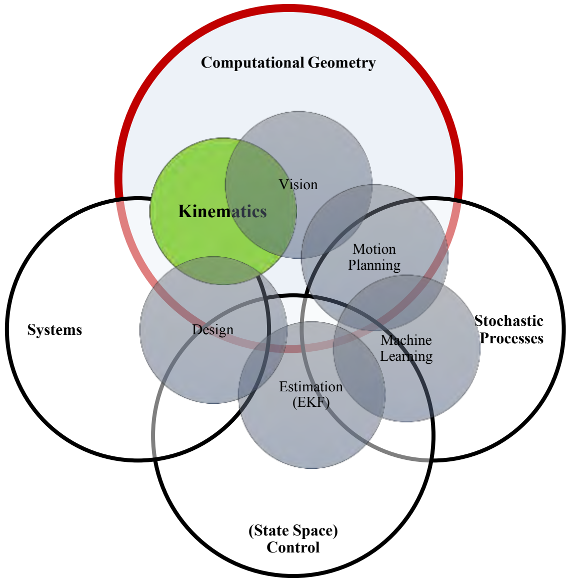
August 17, 2016

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

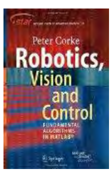
Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	**17-Aug**	**Robot Inverse Kinematics & Kinetics**
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Multiple View Geometry & Feature Detection
8	14-Sep	Probabilistic Robotics: Localization
9	21-Sep	Probabilistic Robotics: SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	State-Space Modelling
12	19-Oct	Shaping the Dynamic Response
13	26-Oct	LQR + Course Review

Course Organization



Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

➔ Representing Space ←

- RVC
  - Chapter 7: Robot Arm Kinematics
  -

- Inverse Kinematics
  - RVC
    §7.3: Robot Arm Kinematics

Next Time

Reference Material

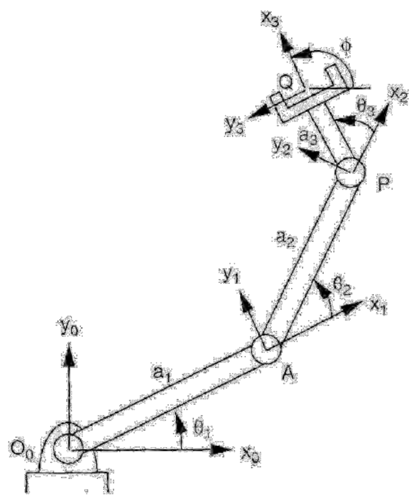
Online:
http://ruina.tam.cornell.edu/Book/RuinaPratap1-15-13.pdf

Inverse Kinematics

Inverse Kinematics [More Generally]

- Freudenstein (1973) referred to the inverse kinematics problem of the most general **6R** manipulator as the "Mount Everest" of kinematic problems.
- Tsai and Morgan (1985) and Primrose (1986) proved that this has at most 16 real solutions.
- Duffy and Crane (1980) derived a closed-form solution for the general **7R** single-loop spatial mechanism.
  - The solution was obtained in the form of a 16 x 16 delerminant in which every element is a second-degree polynomial in one joint variable. The determinant, when expended, should yield a 32nd-degree polynomial equation and hence confirms the upper limit predicted by Roth *et al*. (1973).
- Tsai and Morgan (1985) used the homotopy continuation method to solve the inverse kinematics of the general 6R manipulator and found only 16 solutions
- Raghavan and Roth (1989, 1990) used the dyalitic elimination method to derive a 16th-degree polynomial for the general 6R inverse kinematics problem.

Example: FK/IK of a 3R Planar Arm



- Derived from Tsai (p. 63)

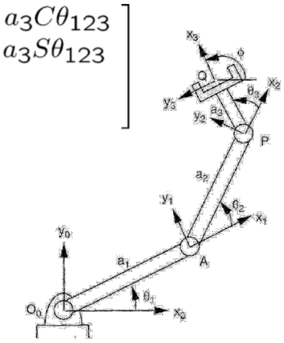
Example: 3R Planar Arm [2]

Position Analysis: 3·Planar 1-R Arm rotating about **Z** **[Ⓩ]**

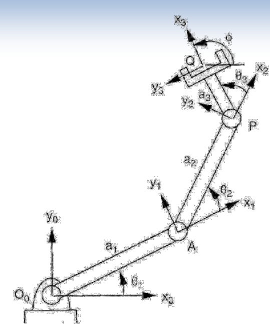
$$^{0}A_3 = \ ^{0}A_1 \cdot ^{1}A_2 \cdot ^{2}A_3$$

Substituting gives:

$$^{0}A_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 & a_1C\theta_1 + a_2C\theta_{12} + a_3C\theta_{123} \\ S\theta_{123} & C\theta_{123} & 0 & a_1S\theta_1 + a_2S\theta_{12} + a_3S\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Example: 3R Planar Arm [2]



Forward Kinematics

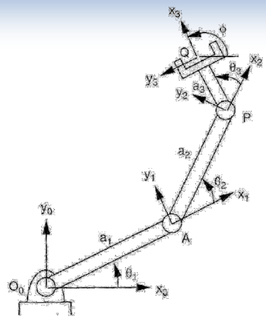
(solve for **x** **given θ** → **x** = f (**θ**))

Fairly straight forward:

$$^{0}R_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 \\ S\theta_{123} & C\theta_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$^{0}P_3 = \begin{bmatrix} a_1C\theta_1 + a_2C\theta_{12} + a_3C\theta_{123} \\ a_1S\theta_1 + a_2S\theta_{12} + a_3S\theta_{123} \\ 0 \end{bmatrix}$$

Example: 3R Planar Arm [3]



Inverse Kinematics
(solve for **θ given x** → $\mathbf{x} = f(\boldsymbol{\theta})$)

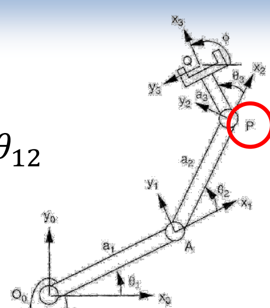
- Start with orientation φ:

$$C\theta_{123} = C\phi,\ S\theta_{123} = S\phi$$
$$\Rightarrow \theta_{123} = \theta_1 + \theta_2 + \theta_3 = \phi$$

- Get overall position $\boldsymbol{q} = [q_x \quad q_y]$:

$$q_x - a_3 C\phi = a_1 C\theta_1 + a_2 C\theta_{12}$$
$$q_y - a_3 S\phi = a_1 S\theta_1 + a_2 S\theta_{12} \ \ldots$$

Example: 3R Planar Arm [4]



- Introduce $\boldsymbol{p} = [p_x \quad p_y]$ before "wrist"

$$p_x = a_1 C\theta_1 + a_2 C\theta_{12}, p_y = a_1 S\theta_1 + a_2 S\theta_{12}$$
$$\Rightarrow p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1 a_2 C\theta_2$$

- Solve for θ_2:

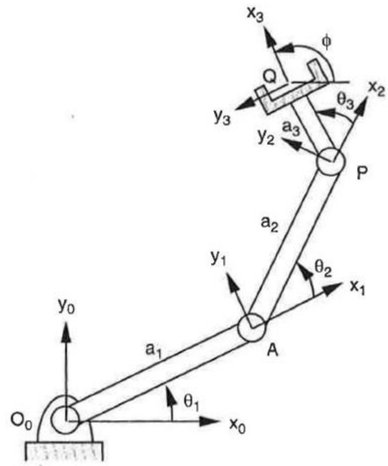
$$\theta_2 = \cos^{-1}\kappa,\ \kappa = \frac{p_x^2 + p_y^2 - a_1^2 - a_2^2}{2a_1 a_2}$$ (2 ℝ roots if |κ|<1)

- Solve for θ_1:

$$C\theta_1 = \frac{p_x(a_1 + a_2 C\theta_2) + p_y a_2 S\theta_2}{a_1^2 + a_2^2 + 2a_1 a_2 C\theta_2},\ S\theta_1 = \frac{-p_x a_2 S\theta_2 + p_y(a_1 + a_2 C\theta_2)}{a_1^2 + a_2^2 + 2a_1 a_2 C\theta_2}$$
$$\theta_1 = atan2(S\theta_1, C\theta_1)$$

Inverse Kinematics: Example I

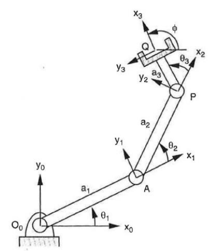
Planar Manipulator:



Inverse Kinematics: Example I

- Forward Kinematics:

[For the Frame {Q} at the end effector]:



$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

$\because$ $${}^0A_3 = \begin{bmatrix} c\theta_{123} & -s\theta_{123} & 0 & a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ s\theta_{123} & c\theta_{123} & 0 & a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

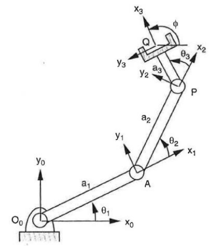
- For an arbitrary point **G** in the end effector: ${}^3\mathbf{g} = [g_u, g_v, 0, 1]^T$

$$\begin{bmatrix} g_x \\ g_y \\ g_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} g_u \\ g_v \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} g_u c\theta_{123} - g_v s\theta_{123} + a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ g_u s\theta_{123} + g_v c\theta_{123} + a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

Inverse Kinematics: Example I

- Forward Kinematics:

[For the Frame {Q} at the end effector]:



$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

$\because$

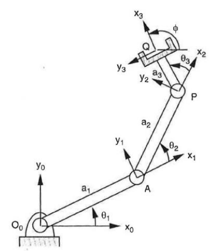
$${}^0A_3 = \begin{bmatrix} c\theta_{123} & -s\theta_{123} & 0 & a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ s\theta_{123} & c\theta_{123} & 0 & a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- For an arbitrary point **G** in the end effector: ${}^3\mathbf{g} = [g_u, g_v, 0, 1]^T$

$$\begin{bmatrix} g_x \\ g_y \\ g_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} g_u \\ g_v \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} g_u c\theta_{123} - g_v s\theta_{123} + a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \\ g_u s\theta_{123} + g_v c\theta_{123} + a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \\ 0 \\ 1 \end{bmatrix}$$

Inverse Kinematics: Example I

- Inverse Kinematics:
  - Set the final position equal to the Forward Transformation Matrix $\mathbf{{}^0A_3}$:



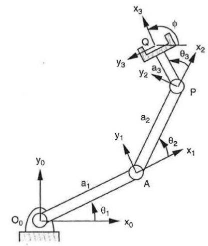
$${}^0A_3 = \begin{bmatrix} c\phi & -s\phi & 0 & q_x \\ s\phi & c\phi & 0 & q_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The solution strategy is to equate the elements of $\mathbf{{}^0A_3}$ to that of the given position (q_x, q_y) and orientation ϕ

Inverse Kinematics: Example I

- Orientation (ϕ):

$$c\theta_{123} = c\phi,$$
$$s\theta_{123} = s\phi.$$
$$\theta_{123} = \theta_1 + \theta_2 + \theta_3 = \phi.$$



- Now Position of the 2DOF point **P**:

$$p_x = a_1 c\theta_1 + a_2 c\theta_{12},$$
$$p_y = a_1 s\theta_1 + a_2 s\theta_{12},$$

$\therefore \quad p_x = q_x - a_3 c\phi \qquad p_y = q_y - a_3 s\phi$

- Substitute: θ_3 disappears and now we can eliminate θ_1:

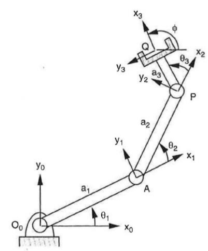
$$p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1 a_2 c\theta_2.$$

METR 4202: **Robotics** August 17, 2016 - 17

Inverse Kinematics: Example I

- we can eliminate θ_1…

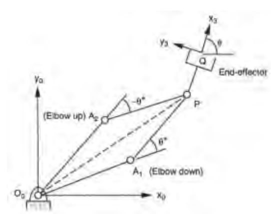
$$p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1 a_2 c\theta_2.$$



- Then solve for θ_{12}:

$$\theta_2 = \cos^{-1}\kappa, \qquad \kappa = \frac{p_x^2 + p_y^2 - a_1^2 - a_2^2}{2a_1 a_2}$$

  - This gives 2 real ($\mathbb{R}$) roots if $|\kappa| < 1$
  - One double root if $|\kappa| = 1$
  - No real roots if $|\kappa| > 1$
- Elbow up/down:
  - In general, **if** θ_2 is a solution **then** $-\theta_2$ is a solution



METR 4202: **Robotics** August 17, 2016 - 18

Inverse Kinematics: Example I

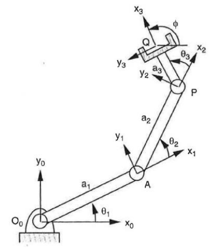
- Solving for θ_1…
  - Corresponding to each θ_2, we can solve θ_1

$$(a_1 + a_2 c\theta_2)c\theta_1 - (a_2 s\theta_2)s\theta_1 = p_x$$
$$(a_2 s\theta_2)c\theta_1 + (a_1 + a_2 c\theta_2)s\theta_1 = p_y$$

$$c\theta_1 = \frac{p_x(a_1 + a_2 c\theta_2) + p_y a_2 s\theta_2}{\Delta},$$
$$s\theta_1 = \frac{-p_x a_2 s\theta_2 + p_y(a_1 + a_2 c\theta_2)}{\Delta}$$

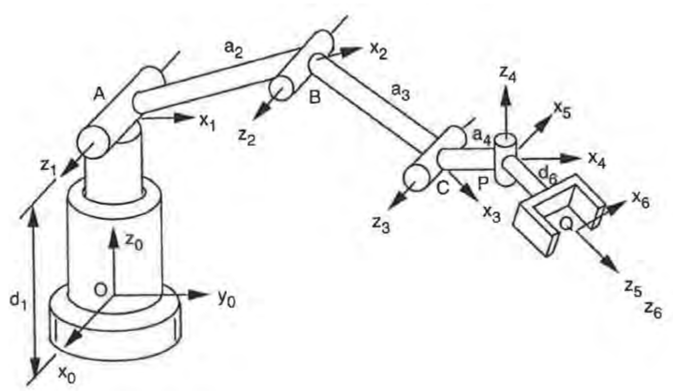
$$\Delta = a_1^2 + a_2^2 + 2a_1 a_2 c\theta_2$$

$$\boxed{\theta_1 = \text{Atan2}(s\theta_1, c\theta_1).}$$



Inverse Kinematics: Example II

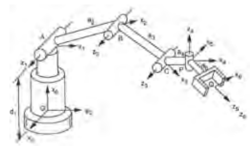
Elbow Manipulator:



Inverse Kinematics: Example II

- Target Position:

$\mathbf{u} = [u_x, u_y, u_z]^{\mathrm{T}}$, $\mathbf{v} = [v_x, v_y, v_z]^{\mathrm{T}}$, $\mathbf{w} = [w_x, w_y, w_z]^{\mathrm{T}}$, and $\mathbf{p} = [p_x, p_y, p_z]^{\mathrm{T}}$.



- Transformation Matrices:

$$A_1 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad ({}^0A_1)^{-1} = \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & 0 \\ -s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

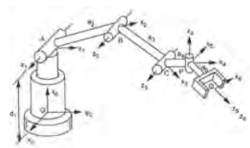
$$A_2 = \begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ s\theta_2 & 0 & c\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} c\theta_3 & 0 & -s\theta_3 & a_2(1 - c\theta_3) \\ 0 & 1 & 0 & 0 \\ s\theta_3 & 0 & c\theta_3 & -a_2 s\theta_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} c\theta_4 & 0 & -s\theta_4 & (a_2 + a_3)(1 - c\theta_4) \\ 0 & 1 & 0 & 0 \\ s\theta_4 & 0 & c\theta_4 & -(a_2 + a_3)s\theta_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5 = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & (a_2 + a_3 + a_4)(1 - c\theta_5) \\ s\theta_5 & c\theta_5 & 0 & -(a_2 + a_3 + a_4)s\theta_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics: Example II

- Key Matrix Products:

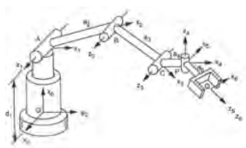


$$A_2A_3A_4 = \begin{bmatrix} c\theta_{234} & 0 & -s\theta_{234} & a_2c\theta_2 + a_3c\theta_{23} - (a_2 + a_3)c\theta_{234} \\ 0 & 1 & 0 & 0 \\ s\theta_{234} & 0 & c\theta_{234} & a_2s\theta_2 + a_3s\theta_{23} - (a_2 + a_3)s\theta_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1A_2A_3A_4 = \begin{bmatrix} c\theta_1c\theta_{234} & -s\theta_1 & -c\theta_1s\theta_{234} & c\theta_1[a_2c\theta_2 + a_3c\theta_{23} - (a_2 + a_3)c\theta_{234}] \\ s\theta_1c\theta_{234} & c\theta_1 & -s\theta_1s\theta_{234} & s\theta_1[a_2c\theta_2 + a_3c\theta_{23} - (a_2 + a_3)c\theta_{234}] \\ s\theta_{234} & 0 & c\theta_{234} & [a_2s\theta_2 + a_3s\theta_{23} - (a_2 + a_3)s\theta_{234}] \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics: Example II

- Inverse Kinematics:



$$\mathbf{p} = A_1 A_2 A_3 A_4 \mathbf{p}_0.$$

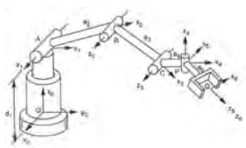
$$A_1^{-1}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = A_2 A_3 A_4 \begin{bmatrix} a_2 + a_3 + a_4 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$p_x \mathrm{c}\theta_1 + p_y \mathrm{s}\theta_1 = a_2 \mathrm{c}\theta_2 + a_3 \mathrm{c}\theta_{23} + a_4 \mathrm{c}\theta_{234},$$
$$-p_x \mathrm{s}\theta_1 + p_y \mathrm{c}\theta_1 = 0,$$
$$p_z = a_2 \mathrm{s}\theta_2 + a_3 \mathrm{s}\theta_{23} + a_4 \mathrm{s}\theta_{234}.$$

METR 4202: **Robotics** August 17, 2016 - 23

Inverse Kinematics: Example II

- Solving the System:



$$\theta_1 = \tan^{-1}\frac{p_y}{p_x}.$$

$$\theta_5 = \sin^{-1}(-w_x \mathrm{s}\theta_1 + w_y \mathrm{c}\theta_1).$$

$$\theta_{234} = \mathrm{Atan2}\left[w_z/\mathrm{c}\theta_5, (w_x \mathrm{c}\theta_1 + w_y \mathrm{s}\theta_1)/\mathrm{c}\theta_5\right].$$

$$a_2 \mathrm{c}\theta_2 + a_3 \mathrm{c}\theta_{23} = k_1, \qquad k_1 = p_x \mathrm{c}\theta_1 + p_y \mathrm{s}\theta_1 - a_4 \mathrm{c}\theta_{234}$$
$$a_2 \mathrm{s}\theta_2 + a_3 \mathrm{s}\theta_{23} = k_2, \qquad k_2 = p_z - a_4 \mathrm{s}\theta_{234}$$

$$a_2^2 + a_3^2 + 2a_2 a_3 \mathrm{c}\theta_3 = k_1^2 + k_2^2.$$

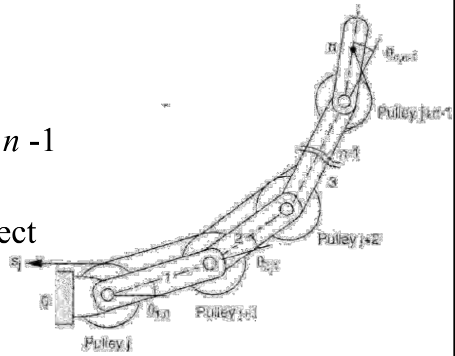
$$\theta_3 = \cos^{-1}\frac{k_1^2 + k_2^2 - a_2^2 - a_3^2}{2a_2 a_3}.$$

$$\theta_6 = \mathrm{Atan2}(\mathrm{s}\theta_6, \mathrm{c}\theta_6).$$

METR 4202: **Robotics** August 17, 2016 - 24

Advanced Concept: Tendon-Driven Manipulators

- Tendons may be modelled as a transmission line
- in which the links are labeled sequentially from 0 to n and the pulleys are labeled from j to $j + n - 1$
- Let θ_{ji} denote the angular displacement of link j with respect to link i.
- We can write a circuit equation once for each pulley pair as follows:



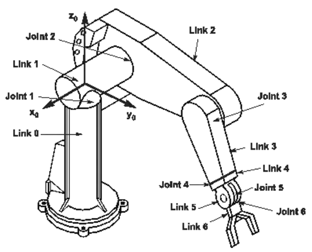
$$r_{j+i-1}\theta_{j+i-1,i} = \pm r_{j+i}\theta_{j+i,i} \quad \text{for } i = 1, 2, \ldots, n-1.$$

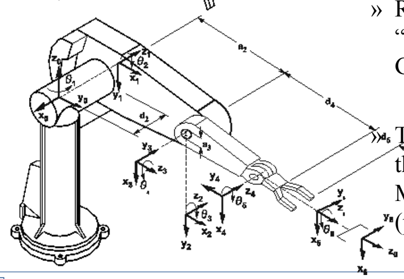
$$\theta_{j+i-1,i} = \theta_{j+i-1,i-1} - \theta_{i,i-1} \quad \text{for } i = 1, 2, \ldots, n.$$

$$\theta_{j,0} = \theta_{1,0} \pm (r_{j+1}/r_j)\theta_{2,1} \pm \cdots \pm (r_{j+n-1}/r_j)\theta_{n,n-1}.$$

Inverse Kinematics

- What about a more complicated mechanism?





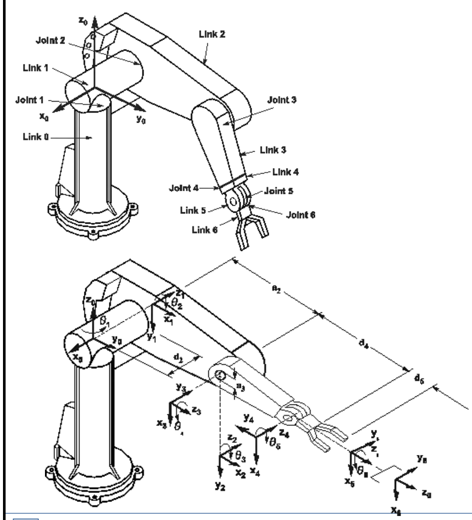
» A sufficient condition for a serial manipulator to yield a closed-form inverse kinematics solution is to have any three consecutive joint axes intersecting at a common point or any three consecutive joint axes parallel to each other. (Pieper and Roth (1969) via 4×4 matrix method)

» Raghavan and Roth 1990
"Kinematic Analysis of the 6R Manipulator of General Geometry"

» Tsai and Morgan 1985, "Solving the Kinematics of the Most General Six and Five-Dcgree-of-Freedom Manipulators by Continuation Methods" (posted online)

Inverse Kinematics

- What about a more complicated mechanism?



$$ {}^0\boldsymbol{T}_6 = {}^0\boldsymbol{T}_1\,{}^1\boldsymbol{T}_2\,{}^2\boldsymbol{T}_3\,{}^3\boldsymbol{T}_4\,{}^4\boldsymbol{T}_5\,{}^5\boldsymbol{T}_6 = \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$
\begin{aligned}
n_x &= c_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) - s_1(s_4c_5c_6 + c_4s_6) \\
n_y &= s_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) + c_1(s_4c_5c_6 + c_4s_6) \\
n_z &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \\
s_x &= c_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) - s_1(-s_4c_5s_6 + c_4c_6) \\
s_y &= s_1(-c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6) + c_1(-s_4c_5s_6 + c_4c_6) \\
s_z &= s_{23}(c_4c_5s_6 + s_4c_6) - c_{23}s_5s_6 \\
a_x &= c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 \\
a_y &= s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5 \\
a_z &= -s_{23}c_4s_5 + c_{23}c_5) \\
p_x &= c_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) - s_1(d_6s_4s_5 + d_2) \\
p_y &= s_1(d_6(c_{23}c_4s_5 + s_{23}c_5) + s_{23}d_4 + a_3c_{23} + a_2c_2) + c_1(d_6s_4s_5 + d_2) \\
p_z &= d_6(c_{23}c_5 - s_{23}c_4s_5) + c_{23}d_4 - a_3s_{23} - a_2s_2
\end{aligned}
$$

Symmetrical Parallel Manipulator

A sub-class of Parallel Manipulator:

- # Limbs (***m***) = # DOF (***F***)
- The joints are arranged in an identical pattern
- The # and location of actuated joints are the same

Thus:

- Number of Loops (L): One less than # of limbs

$$L = m - 1 = F - 1$$

- Connectivity (C_k)

$$\sum_{k=1}^{m} C_k = (\lambda + 1)\,F - \lambda$$

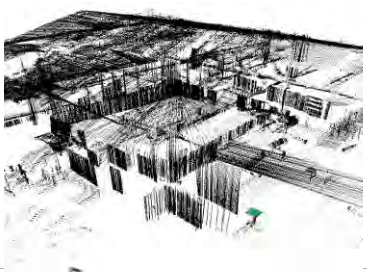
Where: λ: The DOF of the space that the system is in (e.g., λ=6 for 3D space).

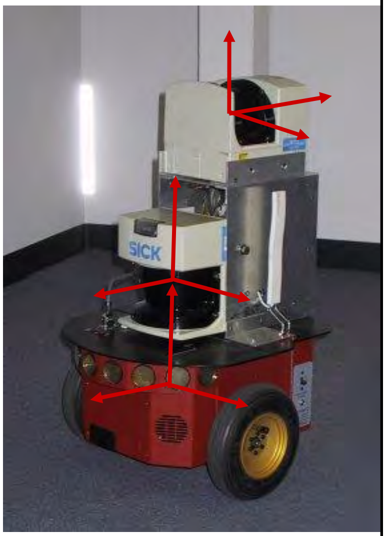
Mobile Platforms

- The preceding kinematic relationships are also important in mobile applications
- When we have sensors mounted on a platform, we need the ability to translate from the sensor frame into some world frame in which the vehicle is operating
- Should we just treat this as a P(*) mechanism?

Mobile Platforms [2]

- We typically assign a frame to the base of the vehicle
- Additional frames are assigned to the sensors
- We will develop these techniques in coming lectures





Summary

- Many ways to view a rotation
  - Rotation matrix
  - Euler angles
  - Quaternions
  - Direction Cosines
  - Screw Vectors

- Homogenous transformations
  - Based on homogeneous coordinates

Generalizing

Special Orthogonal & Special Euclidean Lie Algebras

- SO(n): R**o**tations

$$SO(n) = \{R \in \mathbb{R}^{n\times n} : RR^T = I, \det R = +1\}.$$

$$\exp(\widehat{\omega}\theta) = e^{\widehat{\omega}\theta} = I + \theta\widehat{\omega} + \frac{\theta^2}{2!}\widehat{\omega}^2 + \frac{\theta^3}{3!}\widehat{\omega}^3 + \ldots$$

- SE(n): Transformations of EUCLIDEAN space

$$SE(n) := \mathbb{R}^n \times SO(n).$$

$$SE(3) = \{(p, R) : p \in \mathbb{R}^3, R \in SO(3)\} = \mathbb{R}^3 \times SO(3).$$

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, **I**, **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*

Homogenous Coordinates

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

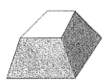
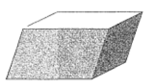
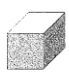
- ρ is a scaling value

Homogenous Transformation

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

- γ is a projective transformation
- The Homogenous Transformation is a **linear operation** (even if projection is not)

Projective Transformations & Other Transformations of 3D Space

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

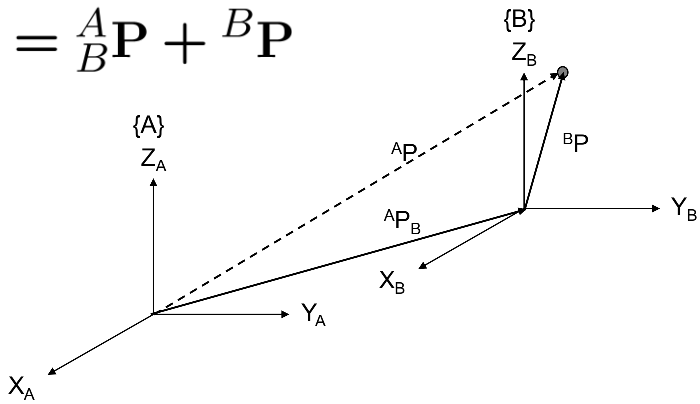
p.78, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

Coordinate Transformations [1]

- Translation Again:

  If {B} is translated with respect to {A} **without rotation**, then it is a vector sum

$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{P} + {}^{B}\mathbf{P}$$



Coordinate Transformations [2]

- Rotation Again:

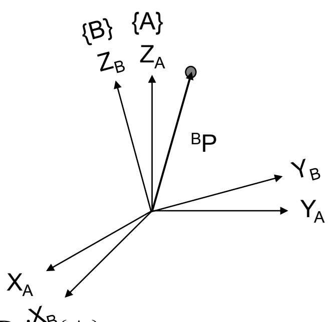
  {B} is rotated with respect to {A} then
  use rotation matrix to determine new components

$$^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}$$

- NOTE:
  - The Rotation matrix's ***subscript*** matches the position vector's **superscript**

$$^{A}\mathbf{P} = {}^{A}_{[B]}\mathbf{R}\,{}^{[B]}\mathbf{P}$$

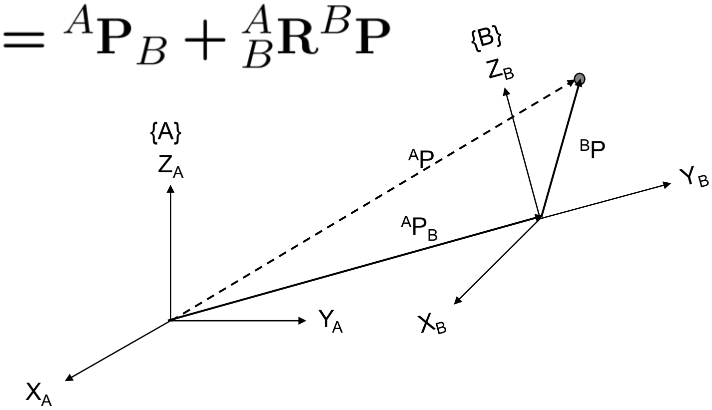
  - This gives Point Positions of {B} ORIENTED in {A}



Coordinate Transformations [3]

- Composite transformation:
  {B} is moved with respect to {A}:

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



General Coordinate Transformations [1]

- A compact representation of the translation and rotation is known as the **Homogeneous Transformation**

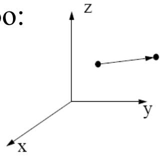
$$^{A}_{B}\mathbf{T} = \begin{bmatrix} & {}^{A}_{B}\mathbf{R} & & {}^{A}\mathbf{P}_{B} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This allows us to cast the rotation and translation of the general transform in a single matrix form

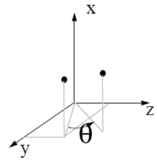
$$\begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = {}^{A}_{B}\mathbf{T} \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix}$$

General Coordinate Transformations [2]

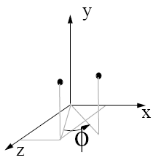
- Similarly, fundamental orthonormal transformations can be represented in this form too:



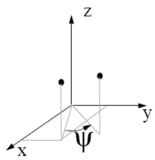
$$Trans(u,v,w) = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Rotx(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & 0 \\ 0 & s\theta & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Roty(\phi) = \begin{bmatrix} c\phi & 0 & s\phi & 0 \\ 0 & 1 & 0 & 0 \\ -s\phi & 0 & c\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

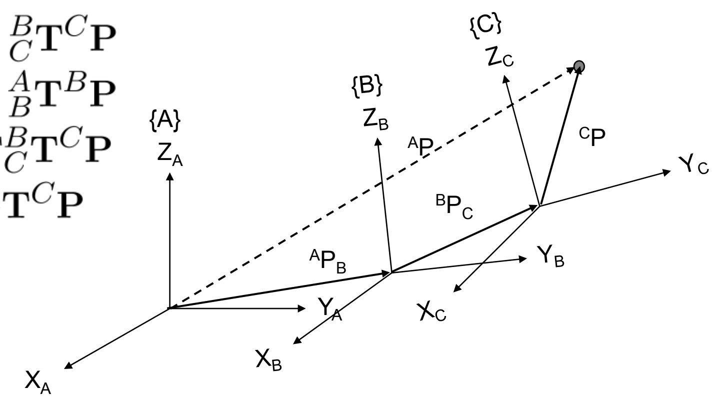


$$Rotz(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 & 0 \\ s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Coordinate Transformations [3]

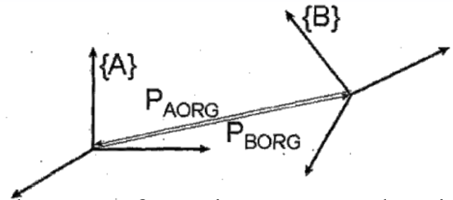
- Multiple transformations compounded as a chain

$$\begin{aligned} {}^{B}\mathbf{P} &= {}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P} \\ {}^{A}\mathbf{P} &= {}^{A}_{B}\mathbf{T}\,{}^{B}\mathbf{P} \\ &= {}^{A}_{B}\mathbf{T}\,{}^{B}_{C}\mathbf{T}\,{}^{C}\mathbf{P} \\ &= {}^{A}_{C}\mathbf{T}\,{}^{C}\mathbf{P} \end{aligned}$$



$$ {}^{A}_{C}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R}\,{}^{B}_{C}\mathbf{R} & {}^{A}\mathbf{P}_B + {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}_C \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Inverse of a Homogeneous Transformation Matrix



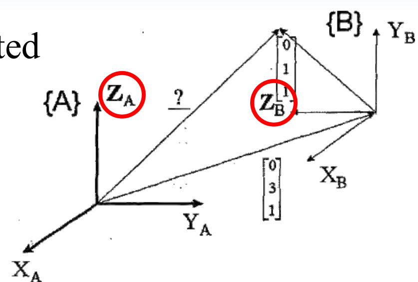
- The inverse of the transform is **not** equal to its transpose because this 4×4 matrix is not orthonormal ($T^{-1} \neq T^T$)
- Invert by parts to give:

$$ {}^A_B T = \begin{bmatrix} {}^A_B R & {}^A\mathbf{p}_{Borg/O_A} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} $$

$$ {}^A_B T^{-1} = {}^B_A T = \begin{bmatrix} {}^A_B R^T & -{}^A_B R^T \cdot {}^A\mathbf{p}_{Borg/O_A} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^B_A R & {}^B\mathbf{p}_{Aorg/O_B} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} $$

Tutorial Problem

The origin of frame *{B}* is translated to a position [0 3 1] with respect to frame *{A}*.



We would like to find:

1. The homogeneous transformation between the two frames in the figure.
2. For a point *P* defined as as [0 1 1] in frame *{B}*, we would like to find the vector describing this point with respect to frame *{A}*.

Tutorial Solution

• The matrix $_{B}T^{A}$ is formed as defined earlier:

- The matrix $_{B}T^{A}$ is formed as defined earlier:

$$ {}^{A}_{B}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

• Since P in the frame is:

• We find vector **p** in frame *{A}* using the relationship

➔

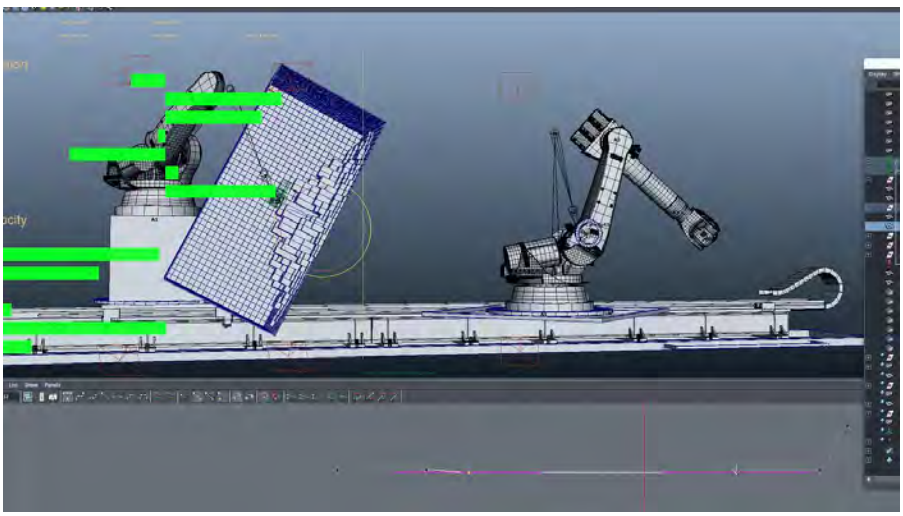
- Since P in the frame is: $ {}^{B}\mathbf{p} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} $

- We find vector **p** in frame *{A}* using the relationship

$$ {}^{A}p = {}^{A}_{B}T\,{}^{B}p $$

➔ $ {}^{A}\mathbf{p} = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 1 \end{bmatrix} $

Cool Robotics Share



Robot Dynamics

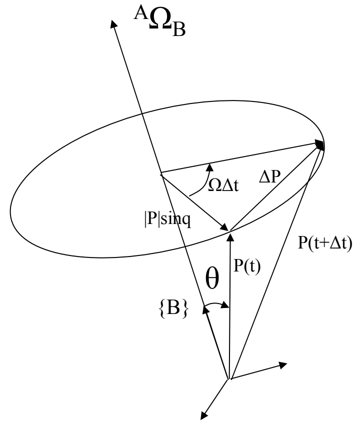
METR 4202: **Robotics** August 17, 2016 -48

Robot Dynamics

METR 4202: **Robotics** August 17, 2016 -49

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

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

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

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

METR 4202: **Robotics** August 17, 2016 -52

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

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

METR 4202: **Robotics** August 17, 2016 -53

Manipulator Velocities

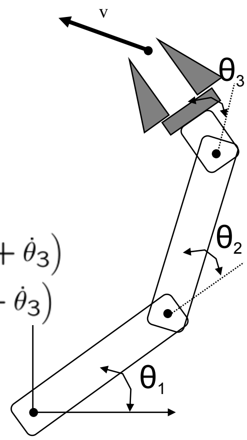
- Consider again the schematic of the planar manipulator shown. We found that the end effector position is given by

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

- Differentiating w/r/t to t

$$\dot{x} = -L_1\, s_1\, \dot{\theta}_1 - L_2\, s_{12} \left(\dot{\theta}_1 + \dot{\theta}_2\right) - L_3\, s_{123} \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)$$
$$\dot{y} = L_1\, c_1\, \dot{\theta}_1 + L_2\, c_{12} \left(\dot{\theta}_1 + \dot{\theta}_2\right) + L_3\, c_{123} \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)$$

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



Manipulator Velocities [2]

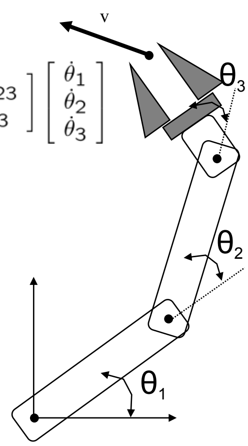
- Rearranging, we can recast this relation in matrix form

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -L_1\, s_1 - L_2\, s_{12} - L_3\, s_{123} & -L_2\, s_{12} - L_3\, s_{123} & -L_3\, s_{123} \\ L_1\, c_1 + L_2\, c_{12} + L_3\, c_{123} & L_2\, c_{12} + L_3\, c_{123} & L_3\, c_{123} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

- 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**

Summary

- Many ways to handle motion
  - Direct Kinematics
  - Dynamics
- Homogenous transformations
  - Based on homogeneous coordinates

Robot Dynamics (Jacobians)

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 5

August 24, 2016

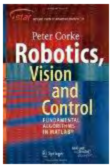
metr4202@itee.uq.edu.au

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

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	**24-Aug**	**Robot Dynamics (Jacobians)**
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Multiple View Geometry & Feature Detection
8	14-Sep	Probabilistic Robotics: Localization
9	21-Sep	Probabilistic Robotics: SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	State-Space Modelling
12	19-Oct	Shaping the Dynamic Response
13	26-Oct	LQR + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook: 364220144X

Today

→ Representing Space ←

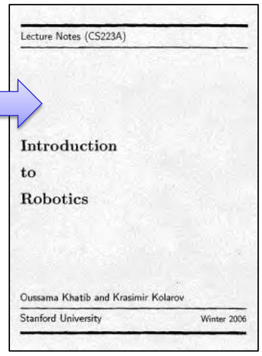
- RVC
  – Chapter 9: Dynamics and Control
- Khatib (pp. 81-150)
  – Chapter 4: Jacobian
  – Chapter 5: Dynamics

- Vision
  – Chapter 11: Image Formation
  – (optimally) Chapter 10: Light & Color

Next Time

Robot Dynamics

Reference Material


Lecture Notes (CS223A)

Introduction
to
Robotics

Oussama Khatib and Krasimir Kolarov
Stanford University
Winter 2006

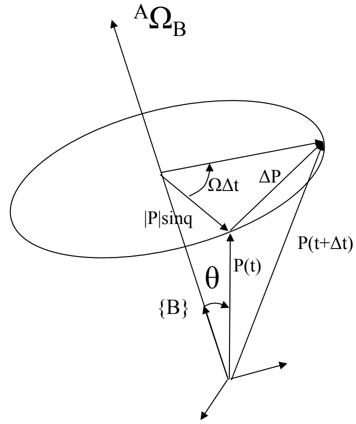
On class webpage
Password: metr4202

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

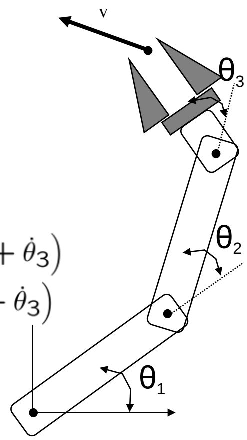
- Consider again the schematic of the planar manipulator shown. We found that the end effector position is given by

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

- 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]

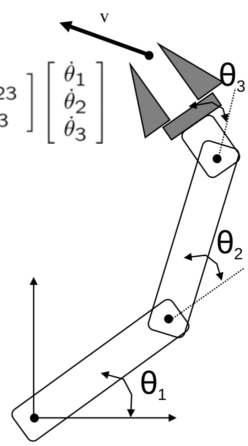
- Rearranging, we can recast this relation in matrix form

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

- 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]

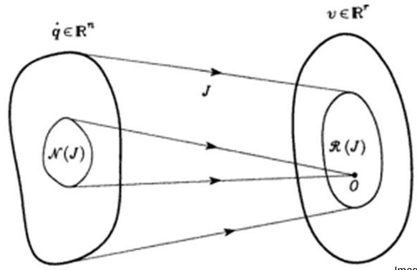


Image:. Sciavicco and Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed, 2000

- 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

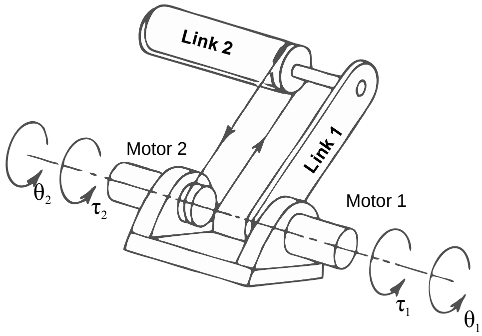
$$J = \begin{bmatrix} \frac{\partial x_P}{\partial q_1} & \cdots & \frac{\partial x_P}{\partial q_n} \\ \bar{\varepsilon}_1 z_1 & \cdots & \bar{\varepsilon}_1 z_n \end{bmatrix}$$

- 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

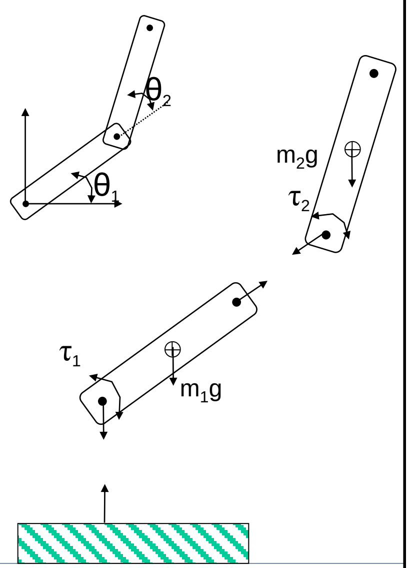
METR 4202: **Robotics** August 24, 2016 -23

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



METR 4202: **Robotics** August 24, 2016 -24

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
- $M(\Theta)$ 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

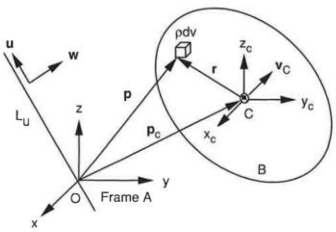
- Notice that all of these terms depend on Θ so the dynamics varies as the manipulator move

METR 4202: **Robotics** August 24, 2016 -25

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

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

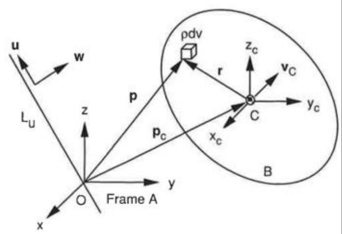
METR 4202: **Robotics** August 24, 2016 -26

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

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

→Thus: the kinetic energy and gravity terms become

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

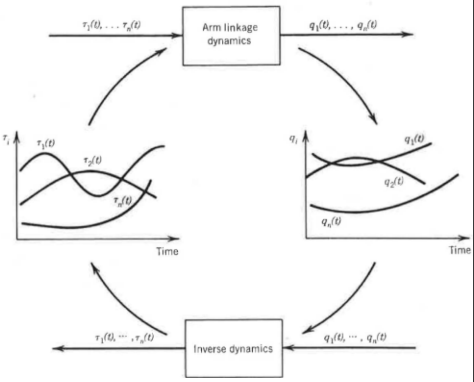
where: $\mathbf{H}^* = \left(\mathbf{J}^{-1}\right)^T \mathbf{H}\mathbf{J}^{-1}$

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:

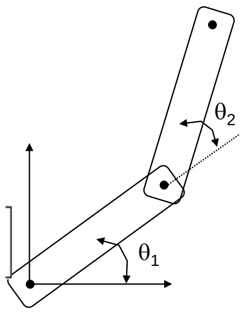
$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

- The Jacobian for this is

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

- Taking the inverse of the Jacobian yields

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



- 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

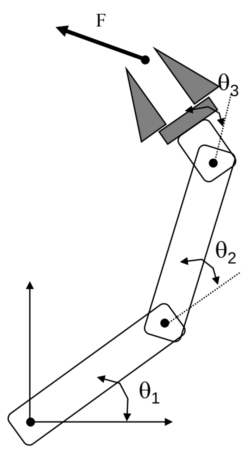
$$\delta \mathbf{X} = \mathbf{J} \delta\theta$$

- So that

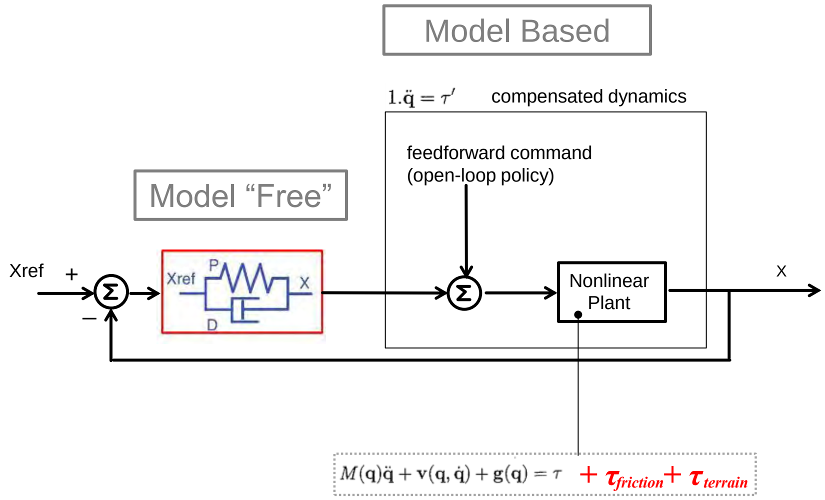
$$F^T \mathbf{J} = \tau^T$$

- Or

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



Operation Space (Computed Torque)

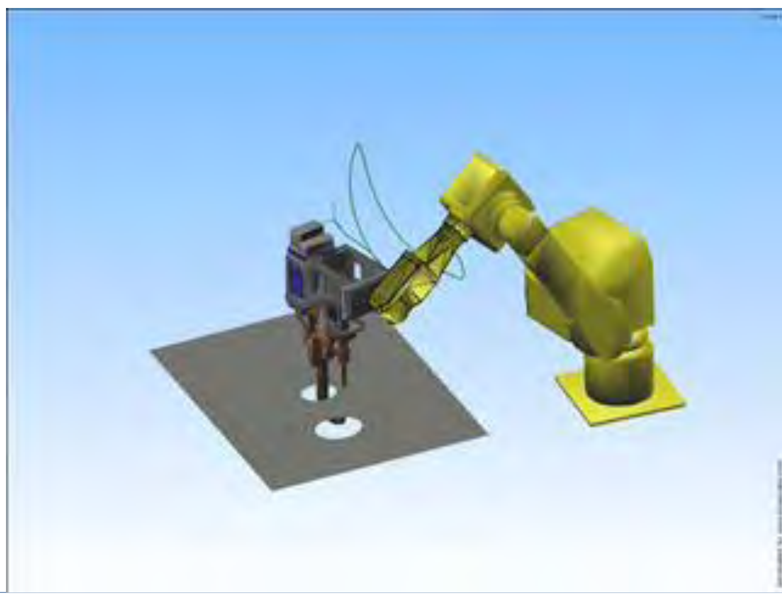


Compensated Manipulation

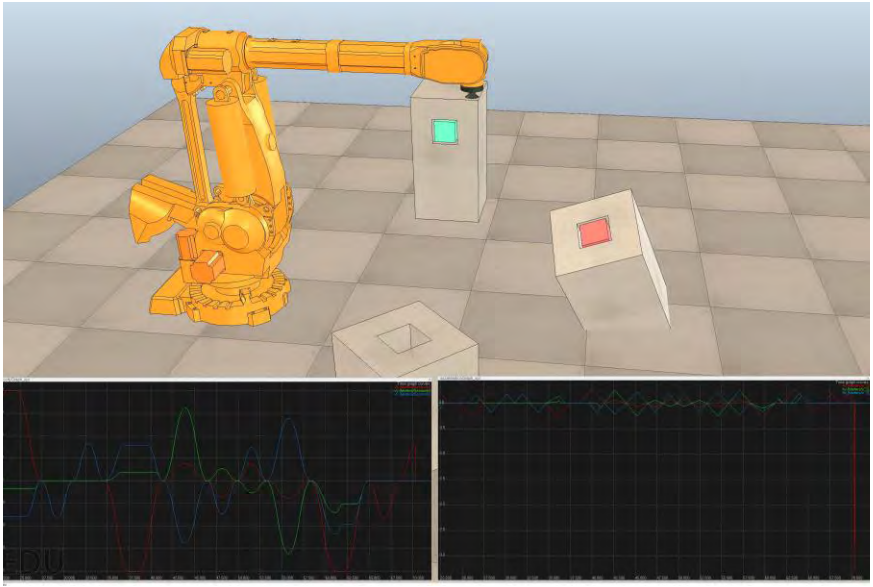
Dynamics of Parallel Manipulators

- 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

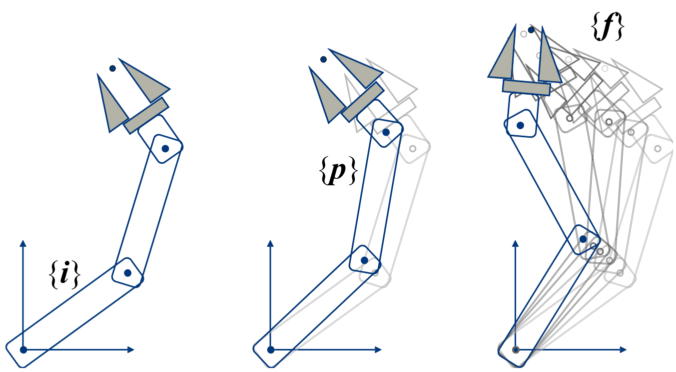


Trajectory Generation & Planning



Trajectory Generation

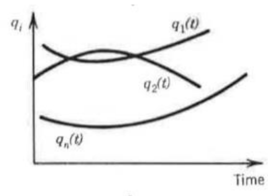
- The goal is to get from an initial position $\{\boldsymbol{i}\}$ to a final position $\{\boldsymbol{f}\}$ via a path points $\{\boldsymbol{p}\}$



Joint Space

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

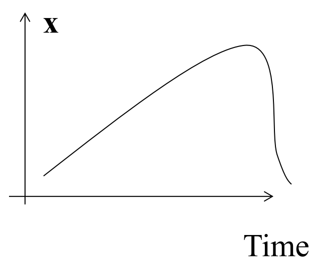
- + Since we control the joints, this is more direct
- -- If we want to follow a particular trajectory, not easy
  - at best lots of intermediate points
  - No guarantee that you can solve the Inverse Kinematics for all path points



Cartesian Workspace

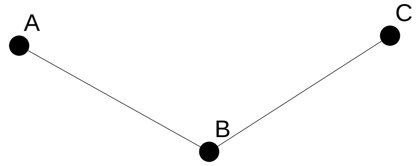
Consider the **Cartesian positions** as a function of time

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



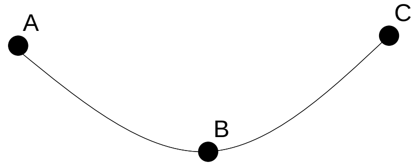
Polynomial Trajectories

- 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"

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

Dynamic Simulation Software

- v-rep virtual robot experimentation platform
- Reflexxes

http://www.coppeliarobotics.com/

http://www.reflexxes.com/

Cool Robotics Share

Cool Robotics Share (II)

Source: Youtube: Wired, How the Tesla Model S is Made

METR 4202: **Robotics** August 24, 2016 -49

Cool Robotics Share (III)

Source: New Kinova Arm in the Robotics Design Lab!

METR 4202: **Robotics** August 24, 2016 -50

Robot Sensing
Perception & Linear Observers

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 6

August 31, 2016

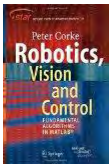
metr4202@itee.uq.edu.au

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

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	**31-Aug**	**Robot Sensing: Perception & Linear Observers**
7	7-Sep	Robot Sensing: Multiple View Geometry & Feature Detection
8	14-Sep	Probabilistic Robotics: Localization
9	21-Sep	Probabilistic Robotics: SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	State-Space Modelling
12	19-Oct	Shaping the Dynamic Response
13	26-Oct	LQR + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

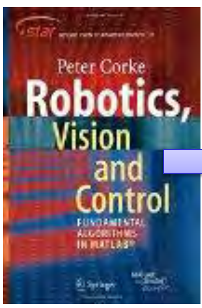
➔ **Sensing and Vision** 🡐

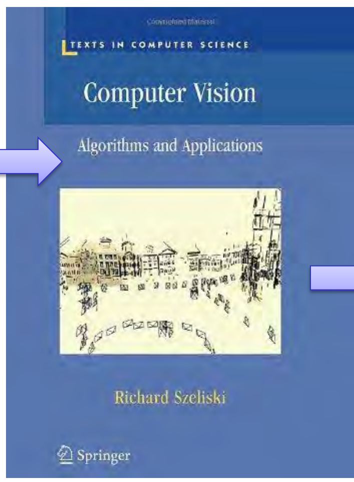
- Vision
  - Chapter 11: Image Formation
  - Chapter 14: Using Multiple Images
    § 14.2 Geometry of Multiple Views

- Multiple View Geometry
  - Hartley & Zisserman:
    Chapter 6: Camera Models
    Chapter 7: Camera Matrix (P)

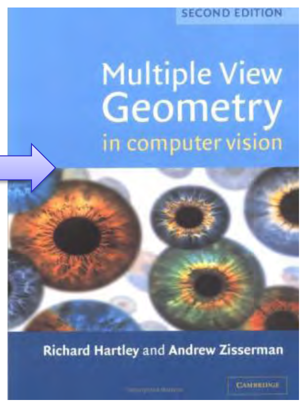
Next Time

Reference Material





UQ Library/ SpringerLink



UQ Library
(ePDF)

Robot Dynamics (leftovers!)

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

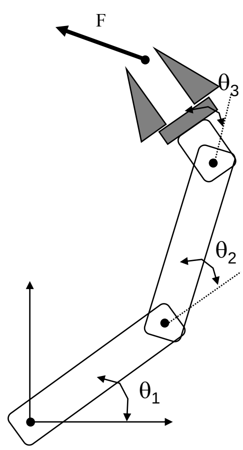
$$\delta \mathbf{X} = \mathbf{J} \delta \theta$$

- So that

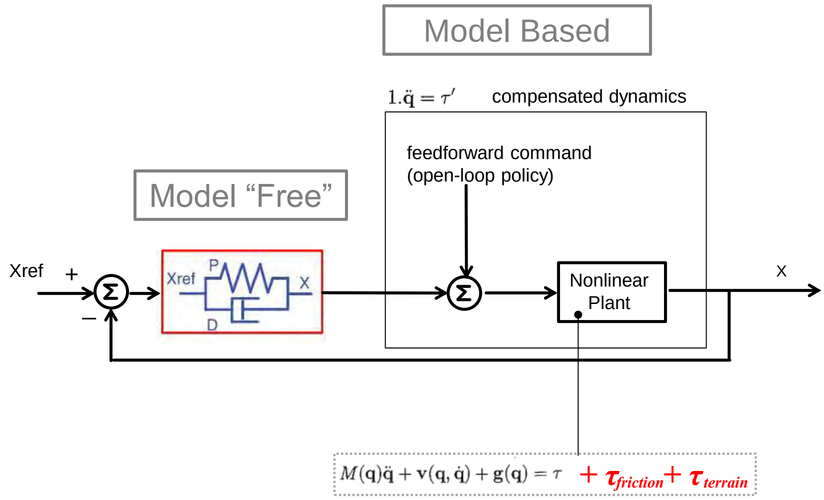
$$F^T \mathbf{J} = \tau^T$$

- Or

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



Operation Space (Computed Torque)



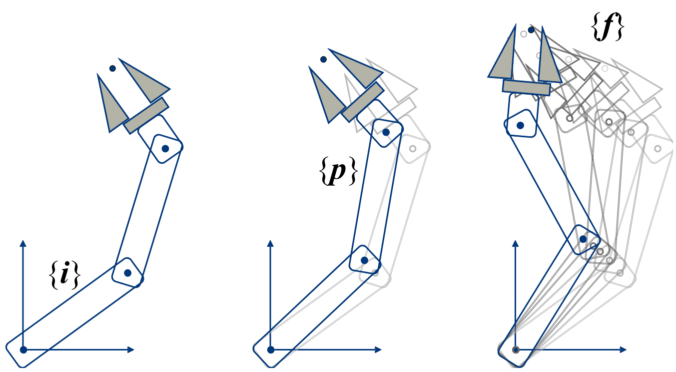
Compensated Manipulation

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

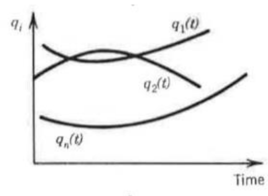
- The goal is to get from an initial position $\{i\}$ to a final position $\{f\}$ via a path points $\{p\}$



Joint Space

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

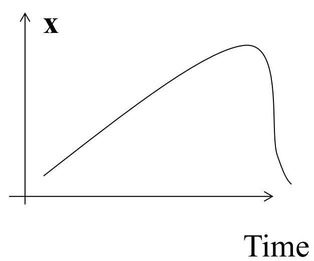
- + Since we control the joints, this is more direct
- -- If we want to follow a particular trajectory, not easy
  - at best lots of intermediate points
  - No guarantee that you can solve the Inverse Kinematics for all path points



Cartesian Workspace

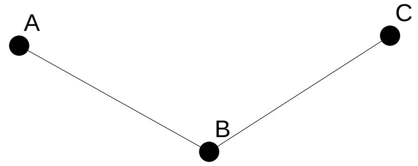
Consider the **Cartesian positions** as a function of time

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



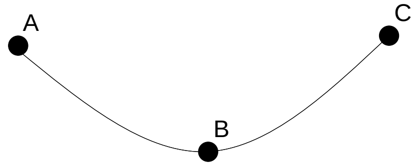
Polynomial Trajectories

- Straight line Trajectories



- Simpler

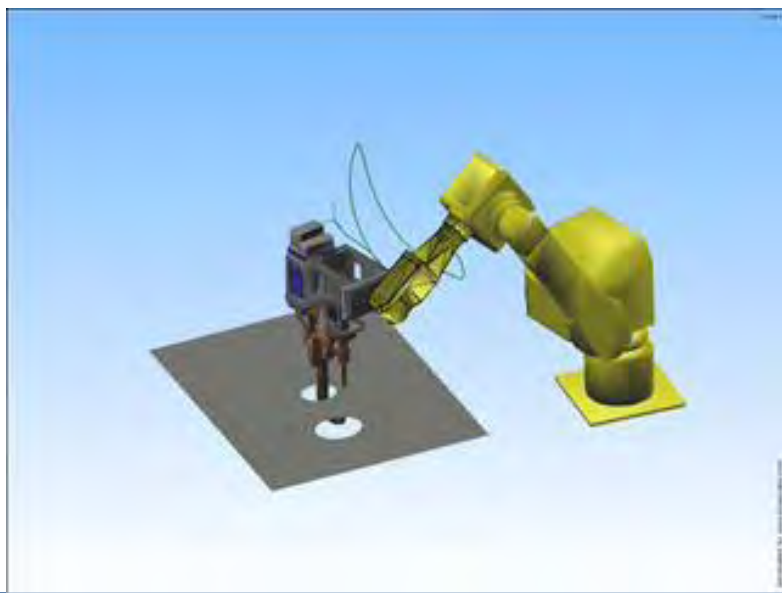
- Polynomial Trajectories



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

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

Trajectory Generation & Planning



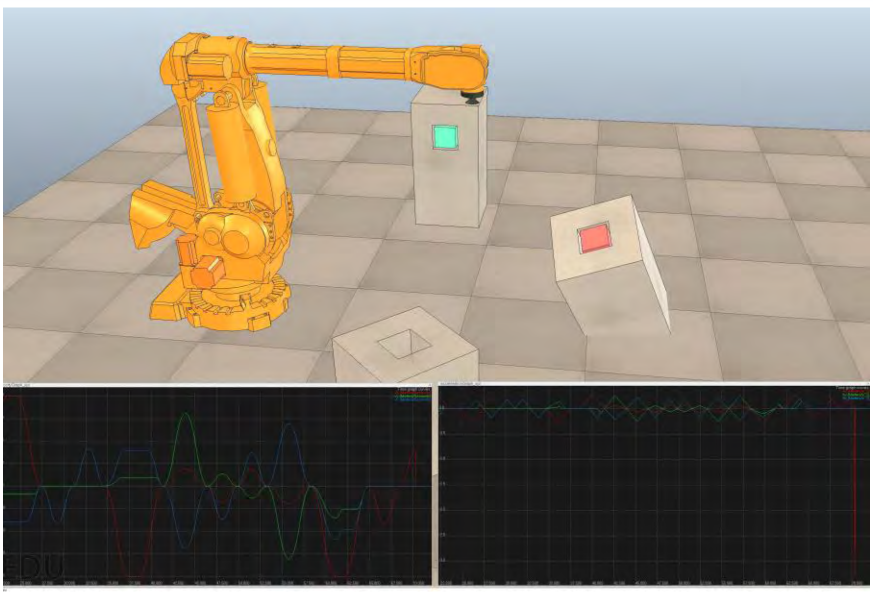
Dynamic Simulation Software

- v-rep virtual robot experimentation platform
- Reflexxes

http://www.coppeliarobotics.com/

http://www.reflexxes.com/

Trajectory Generation & Planning



Sensing:
Image Formation / Single-View Geometry

Quick Outline

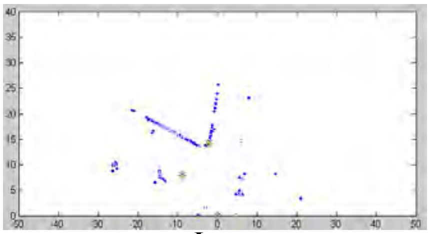
- Frames
- Kinematics

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

1. <u>Perception ➔ Camera Sensors</u>

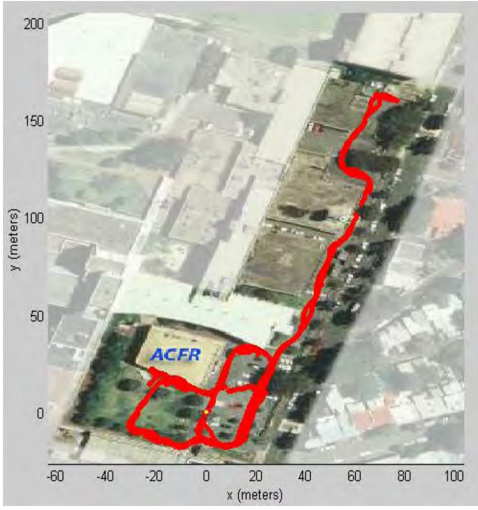
1. Image Formation
   ➔ "Computational Photography"
2. Calibration
3. Features
4. Stereopsis and depth
5. Optical flow

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



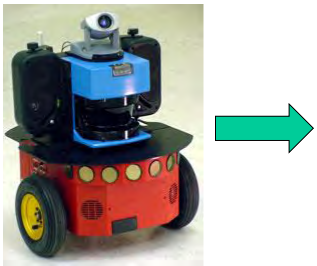
Laser

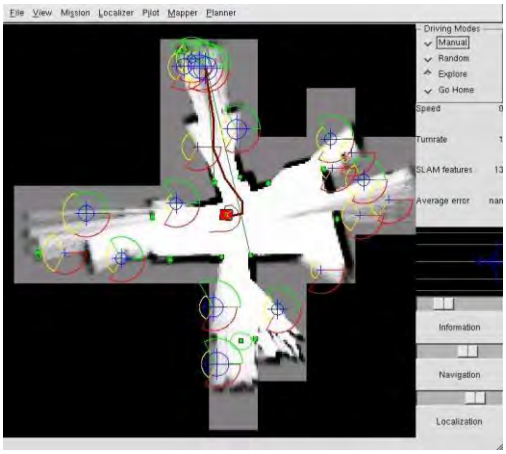
Vision/Cameras



GPS

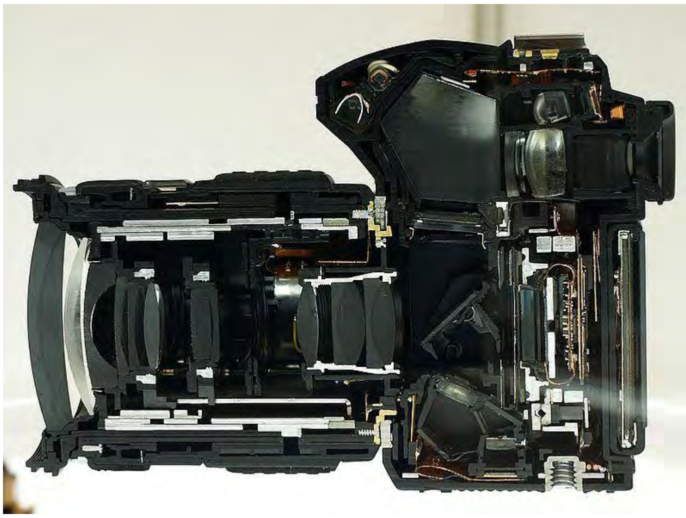
Mapping: Indoor robots





ACFR, IROS 2002

Cameras



Wikipedia, E-30-Cutmodel

METR 4202: **Robotics** August 31, 2016 - 21

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

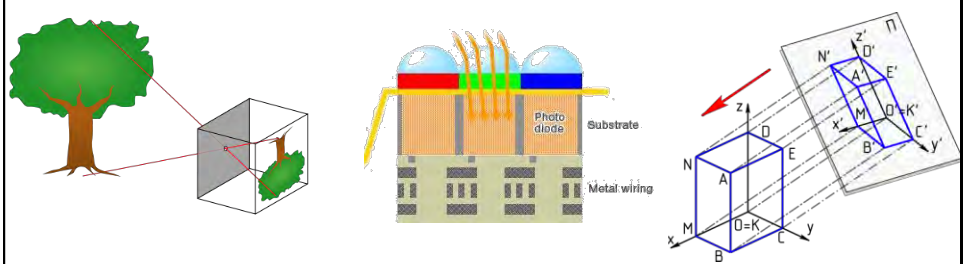
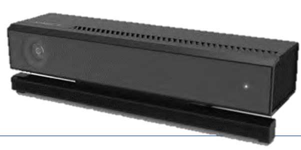
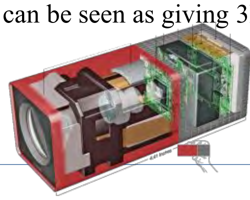


Image Formation ⇒ Image Sensing ⇒ (Re)Projection

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

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

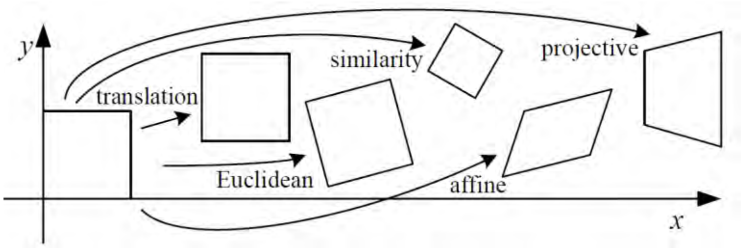




METR 4202: **Robotics** August 31, 2016 - 22

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

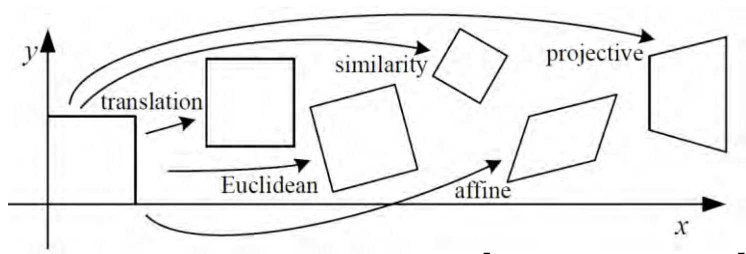
Transformations



- ***x'*: New Image & $\underline{x}$: 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 ||)
- Projective:
  (straight lines preserved)
  H: Homogenous 3x3 Matrix

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \underline{x}$$

$$x' = \mathbf{H}\underline{x}$$

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$$

$$y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

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

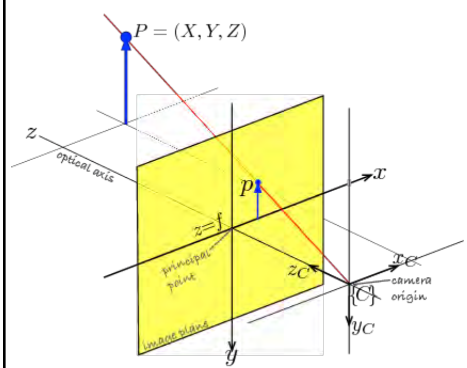
2-D Transformations

➔ 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



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

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

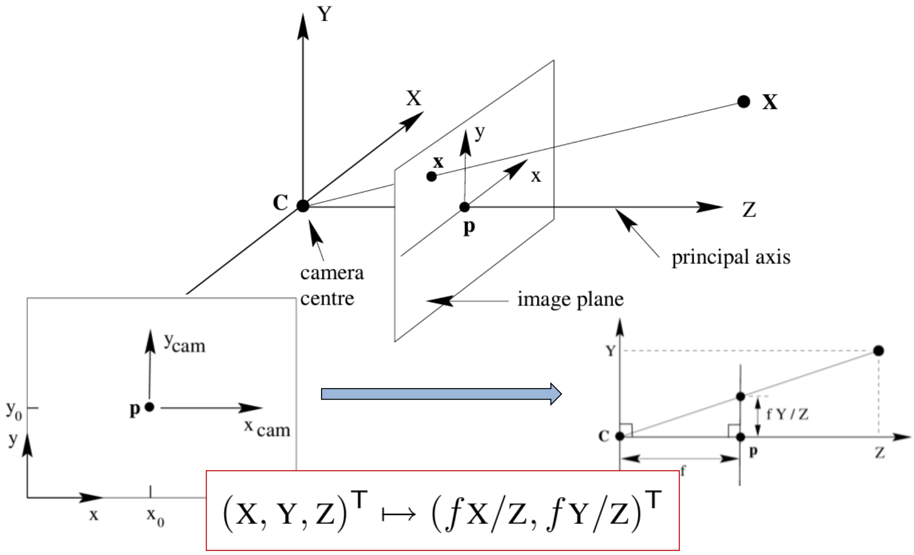
$$(X, Y, Z) \mapsto (x, y)$$

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

Image and Slide from: Corke, Ch. 11

METR 4202: **Robotics** August 31, 2016 -32

Image Formation – Single View Geometry [I]



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

Hartly & Zisserman, Ch. 6

METR 4202: **Robotics** August 31, 2016 -33

Image Formation – Single View Geometry [II]

➔ Camera Projection Matrix

$$\begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} f\mathrm{X} + \mathrm{Z}p_x \\ f\mathrm{Y} + \mathrm{Z}p_y \\ \mathrm{Z} \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix}$$

world camera Intrinsics

- x = Image point
- $\mathbf{X}$ = World point
- K = Camera Calibration Matrix

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

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

➔Perspective Camera as:

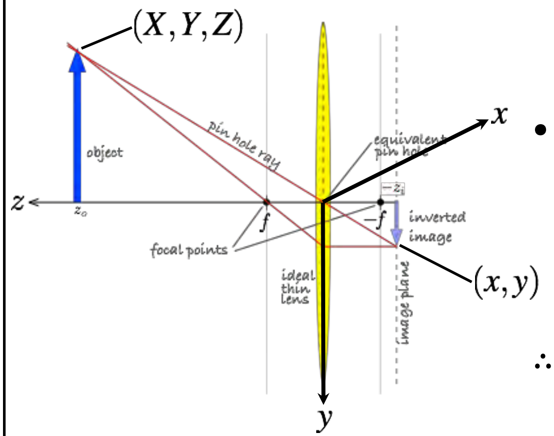
$$\mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathbf{t}]$$

where: P is **3**×4 and of **rank 3**

METR 4202: **Robotics** August 31, 2016 -34

Image Formation: (Thin-Lens) Projection model

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



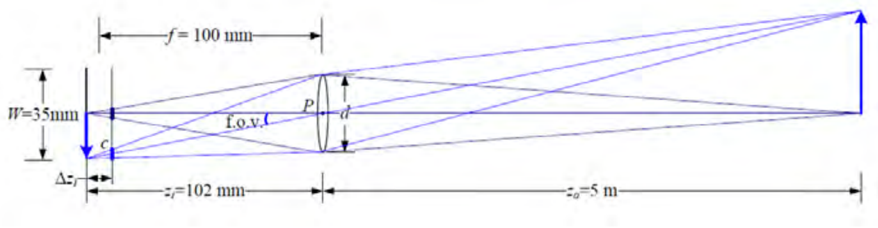
- $\frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f}$

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

Image and Slide from: Corke, Ch. 11

METR 4202: **Robotics** August 31, 2016 -35

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*

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

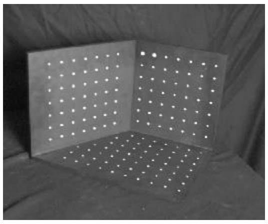
See: *Camera Calibration Toolbox for Matlab*
(http://www.vision.caltech.edu/bouguetj/calib_doc/)

- 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

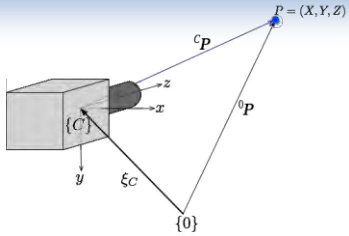
Camera calibration

- 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

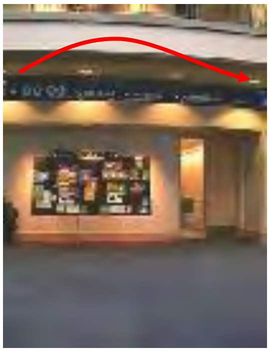
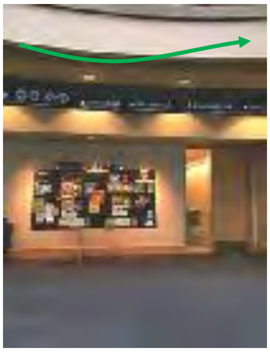
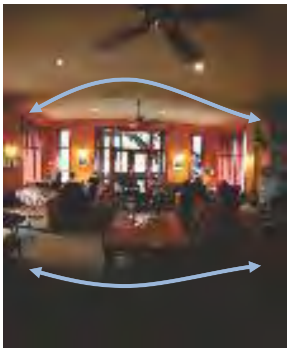
Image and Slide from: Corke, Ch. 11

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

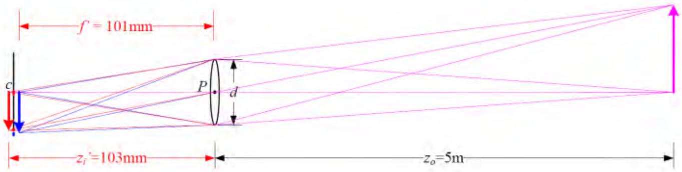
Barrel	Pincushion	Fisheye
		

➔ 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:

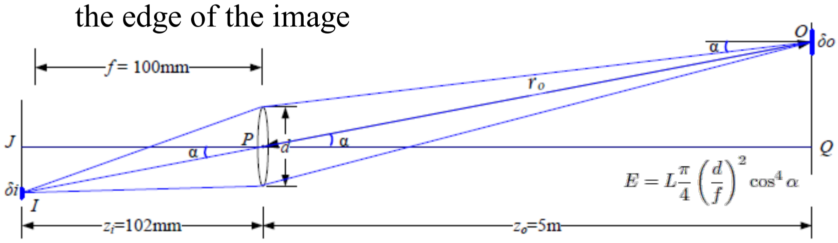


  – In a lens subject to chromatic aberration, light at different wavelengths (e.g., the red and blur arrows) is focused with a different focal length f' and hence a different depth z_i, resulting in both a geometric (in-plane) displacement and a loss of focus

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

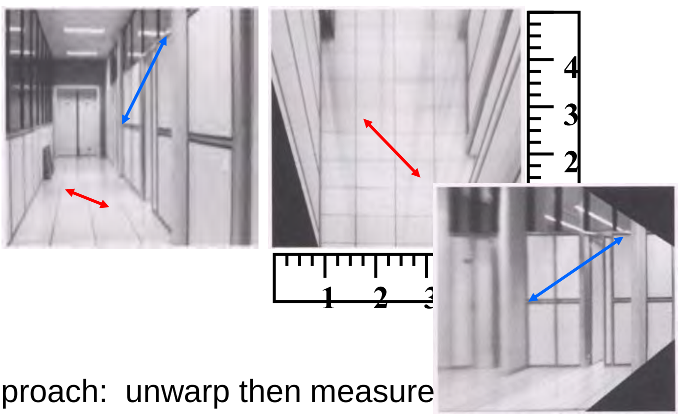


$$E = L\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4 \alpha$$

  - 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!)

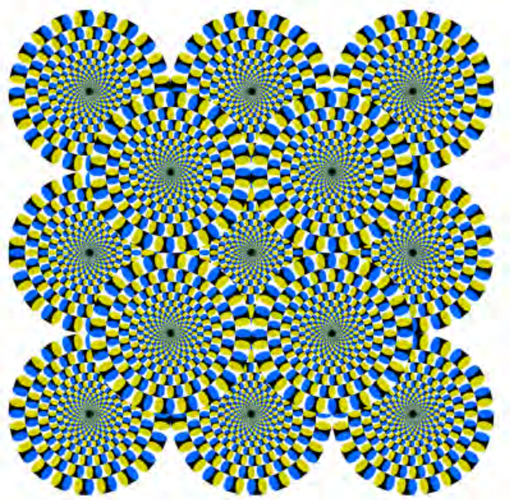


Approach: unwarp then measure

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

Perception

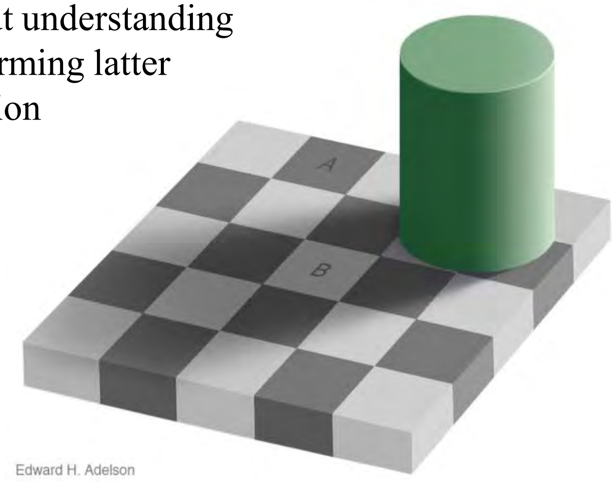
- Making Sense from Sensors



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

Perception

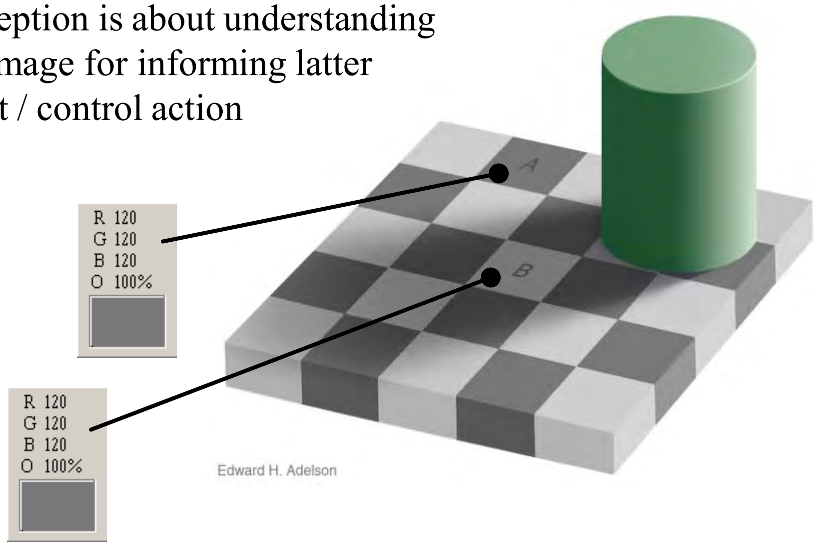
- Perception is about understanding the image for informing latter robot / control action



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

Basic Features:

~~Colour~~

Edges & Lines

Features -- Colour Features

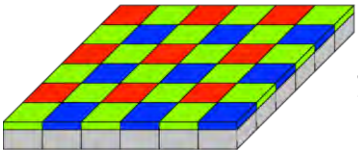
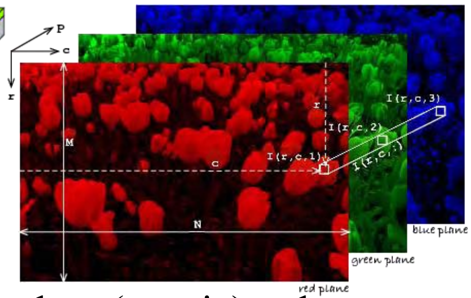


Fig: Ch. 10, *Robotics Vision and Control*

Bayer Patterns



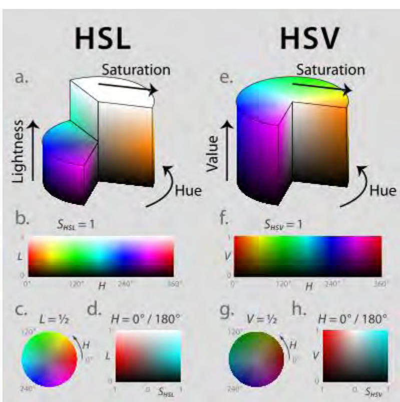
- RGB is **NOT** an absolute (metric) colour space

Also!

- **RGB** (display or additive colour) does not map to **CYMK** (printing or subtractive colour) without calibration
- **Y-Cr-Cb** or **HSV** does not solve this either

Colour Spaces

- HSV



Source: Wikipedia – HSV and YCrCb

- YCrCb

→ Gamma Corrected Luma (Y) + Chrominance

→ BW → Colour TVs : Just add the Chrominance

→ γ Correction: CRTs γ=2.2-2.5

$$\begin{aligned} Y' &= 16 + (65.481 \cdot R' + 128.553 \cdot G' + 24.966 \cdot B') \\ C_B &= 128 + (-37.797 \cdot R' - 74.203 \cdot G' + 112.0 \cdot B') \\ C_R &= 128 + (112.0 \cdot R' - 93.786 \cdot G' - 18.214 \cdot B') \end{aligned}$$

- L*ab

How to get the Features? Still MANY Ways

- Canny edge detector:

Subtractive (CMYK) & Uniform (L*ab) Color Spaces

- $C = W - R$
- $M = W - G$
- $Y = W - B$
- $K = -W$ ☺

- A Uniform color space is one in which the distance in coordinate space is a fair guide to the significance of the difference between the two colors
- Start with RGB → CIE XYZ (Under Illuminant D65)

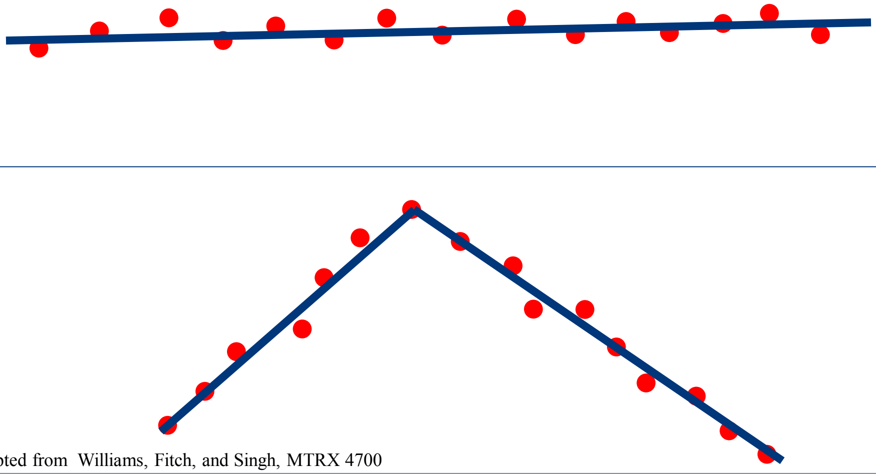
$$L^\star = 116(Y/Y_n)^{(1/3)} - 16$$
$$a^\star = 500\left[(X/X_n)^{(1/3)} - (Y/Y_n)^{(1/3)}\right]$$
$$b^\star = 200\left[(Y/Y_n)^{(1/3)} - (Z/Z_n)^{(1/3)}\right]$$

Edge Detection

- Canny edge detector:
  - Pepsi Sequence:

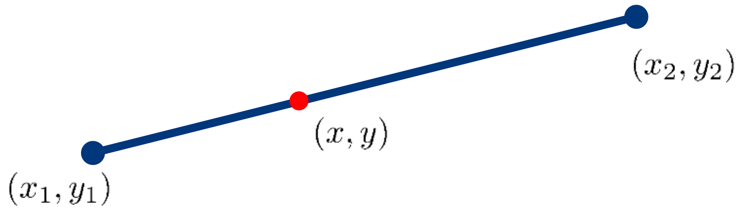
Image Data: http://www.cs.brown.edu/~black/mixtureOF.html and Szeliski, CS223B-L9
See also: Use of Temporal information to aid segmentation:
http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html

Line Extraction and Segmentation



Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Formula

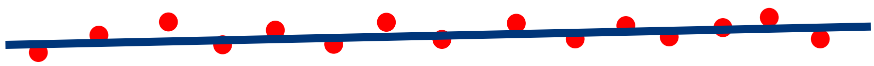


$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\mathbf{y} = m\mathbf{x} + b$$

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Estimation

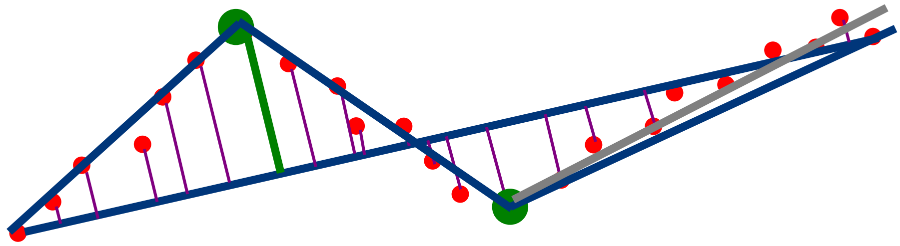


Least squares minimization of the line:

- Line Equation: $\mathbf{y} - m\mathbf{x} - b = 0$
- Error in Fit: $\sum_i (y_i - mx_i - b)^2$
- Solution: $\begin{pmatrix} \bar{x}\bar{y} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} \overline{x^2} & \bar{x} \\ \bar{x} & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix}$

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Splitting / Segmentation



- What about corners?

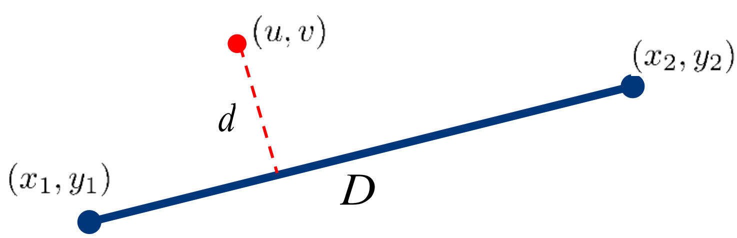
➔Split into multiple lines (via expectation maximization)

1. Expect (assume) a number of lines N (say 3)
2. Find "breakpoints" by finding nearest neighbours upto a threshold or simply at random (RANSAC)
3. How to know N? (Also RANSAC)

Adopted from Williams, Fitch, and Singh, MTRX 4700

METR 4202: **Robotics** August 31, 2016 -59

⊥ of a Point from a Line Segment



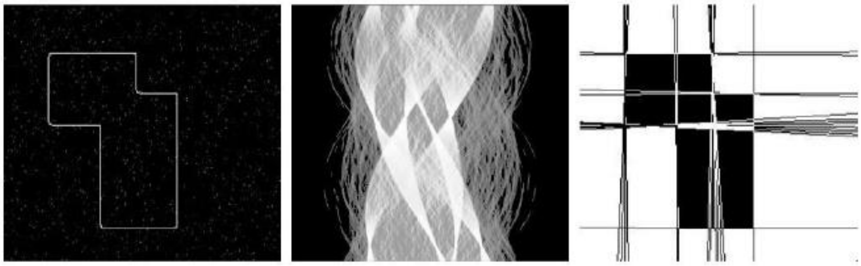
$$r = u(y_1 - y_2) + v(x_2 - x_1) + y_2 x_1 - y_1 x_2$$

$$d = \frac{r}{D}$$

Adopted from Williams, Fitch, and Singh, MTRX 4700

METR 4202: **Robotics** August 31, 2016 -60

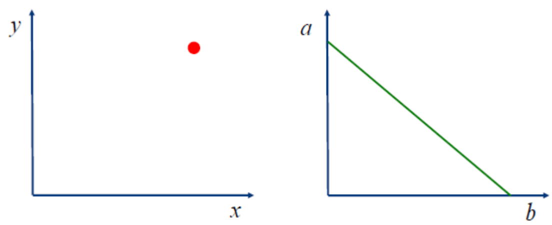
Hough Transform



- Uses a voting mechanism
- Can be used for other lines and shapes (not just straight lines)

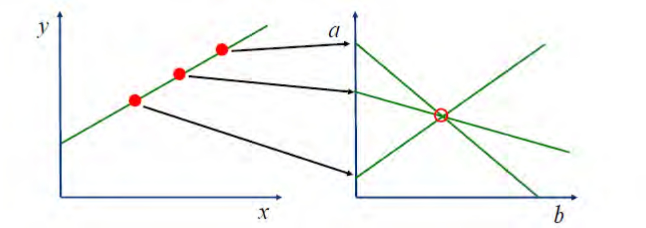
Hough Transform: Voting Space

$y = ax + b$ $\qquad$ $a = -\frac{1}{x}b + \frac{y}{x}$



- Count the number of lines that can go through a point and move it from the "*x*-*y*" plane to the "a-b" plane
- There is only a one-"infinite" number (a line!) of solutions (not a two-"infinite" set – a plane)

Hough Transform: Voting Space



- In practice, the polar form is often used

  $$a = x \cos a + y \sin b$$

- This avoids problems with lines that are nearly vertical

Hough Transform: Algorithm

1. Quantize the parameter space appropriately.

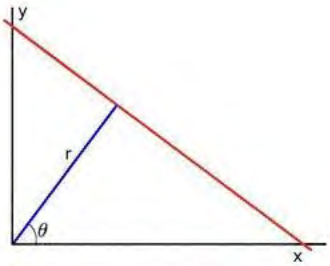
2. Assume that each cell in the parameter space is an accumulator. Initialize all cells to zero.

3. For each point (x,y) in the (visual & range) image space, increment by 1 each of the accumulators that satisfy the equation.

4. Maxima in the accumulator array correspond to the parameters of model instances.

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:

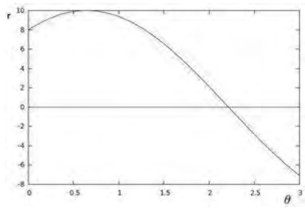
$$r = x_1 \cos\theta + y_1 \sin\theta$$

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

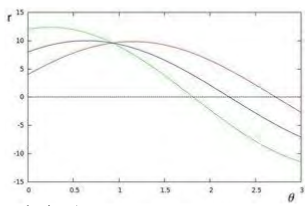
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [2]

- Thus a given point gives a sinusoid



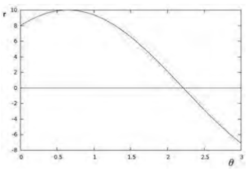
- Repeating for all points on the image



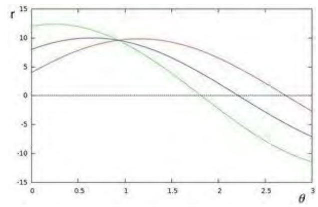
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [3]

- Thus a given point gives a sinusoid



- Repeating for all points on the image

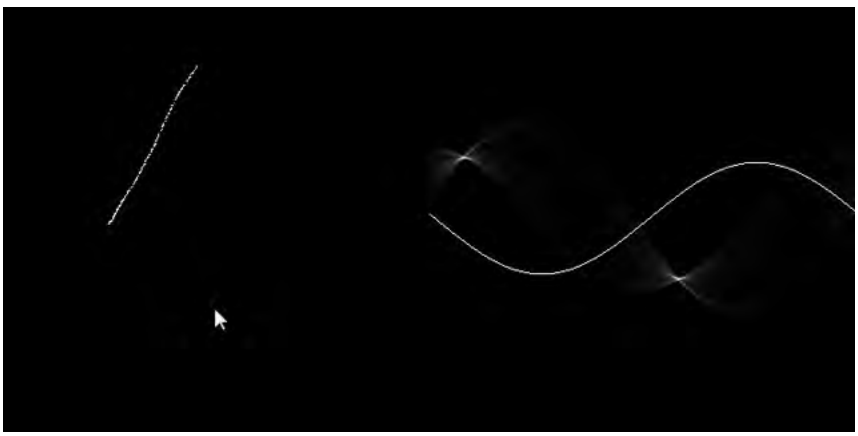


- NOTE that an intersection of sinusoids represents (**a point**) represents **a line** in which pixel points lay.

➔ Thus, a line can be *detected* by finding the number of Intersections between curves

See also OpenCV documentation (cv::HoughLines)

"Cool Robotics Share" -- Hough Transform



- http://www.activovision.com/octavi/doku.php?id=hough_transform

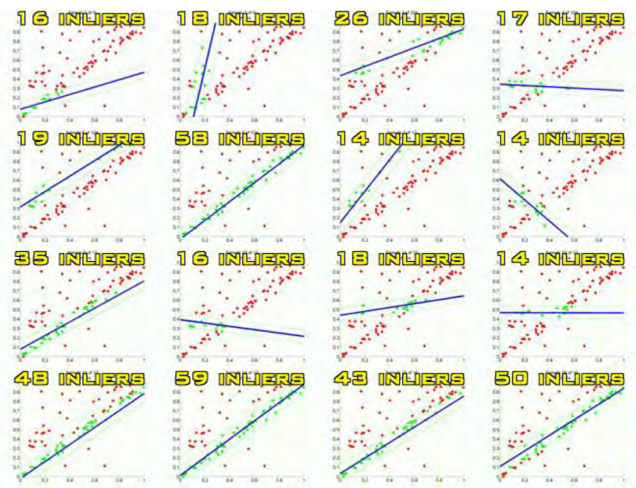
RANdom SAmple Consensus

1. Repeatedly select a small (minimal) subset of correspondences
2. Estimate a solution (in this case a the line)
3. Count the number of "inliers", $|e|<\Theta$ (for LMS, estimate $\text{med}(|e|)$)
4. Pick the *best* subset of inliers
5. Find a complete least-squares solution

- Related to least median squares
- See also:
  MAPSAC (Maximum *A Posteriori* SAmple Consensus)

From Szeliski, *Computer Vision: Algorithms and Applications*

Cool Robotics Share Time!



D. Wedge, *The RANSAC Song*

Robot Sensing: Multiple View Geometry & Feature Detection

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 7

September 7, 2016

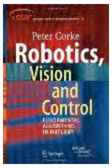
metr4202@itee.uq.edu.au

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

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	**7-Sep**	**Robot Sensing: Multiple View Geometry & Feature Detection**
8	14-Sep	Probabilistic Robotics: Localization
9	21-Sep	Probabilistic Robotics: SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	State-Space Modelling
12	19-Oct	Shaping the Dynamic Response
13	26-Oct	LQR + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

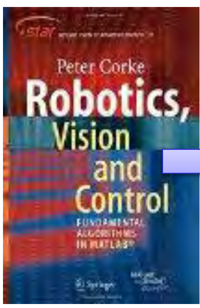
➔ **Sensing and Vision** ←

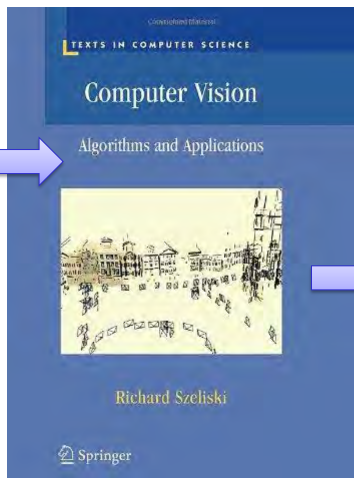
- Multiple View Geometry
  - Chapter 14: Using Multiple Images
    § 14.2 Geometry of Multiple Views
- Multiple View Geometry
  - Hartley & Zisserman:
    Chapter 6: Camera Models
    Chapter 7: Camera Matrix

- Localization
  - Chapter 6: Localization

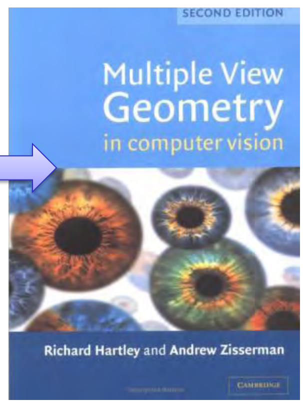
Next Time

Reference Material





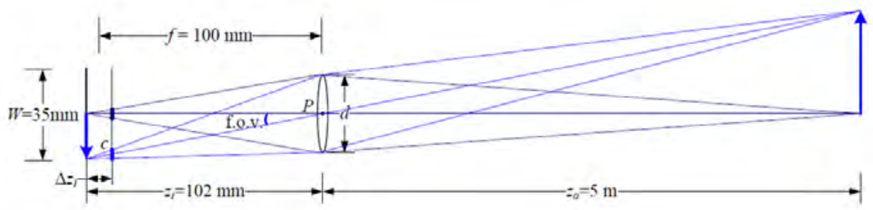
UQ Library/ SpringerLink



UQ Library
(ePDF)

Sensing:
Image Formation /
Single-View Geometry

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]

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

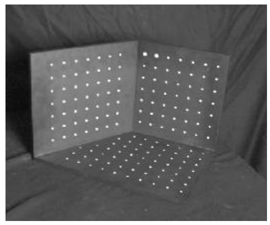
See: *Camera Calibration Toolbox for Matlab*
(http://www.vision.caltech.edu/bouguetj/calib_doc/)

- 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

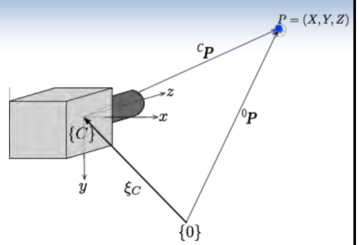
Camera calibration

- 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

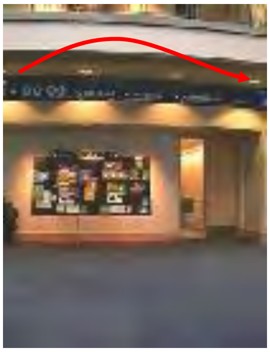
Camera Image Formation "Aberrations"[I]: Lens Optics (Aperture / Depth of Field)

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

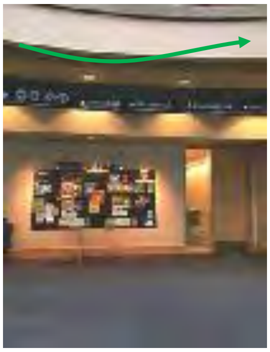
http://en.wikipedia.org/wiki/File:Aperture_in_Canon_50mm_f1.8_II_lens.jpg

Camera Image Formation "Aberrations"[II]: Lens Distortions

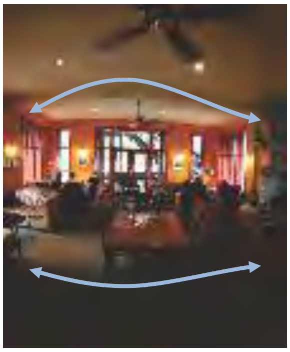
Barrel



Pincushion



Fisheye

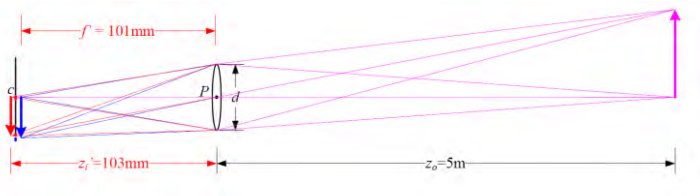


➔ Explore these with `visualize_distortions` in the Camera Calibration Toolbox

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

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

- Chromatic Aberration:

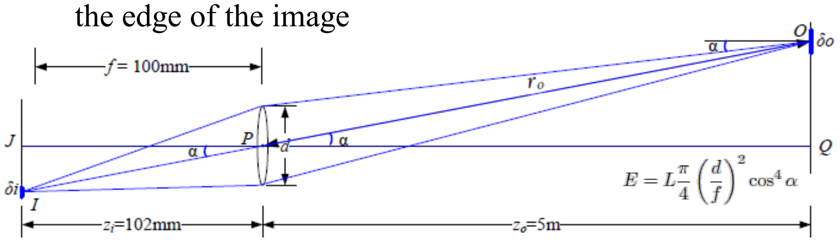


  – In a lens subject to chromatic aberration, light at different wavelengths (e.g., the red and blur arrows) is focused with a different focal length f' and hence a different depth z_i, resulting in both a geometric (in-plane) displacement and a loss of focus

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

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

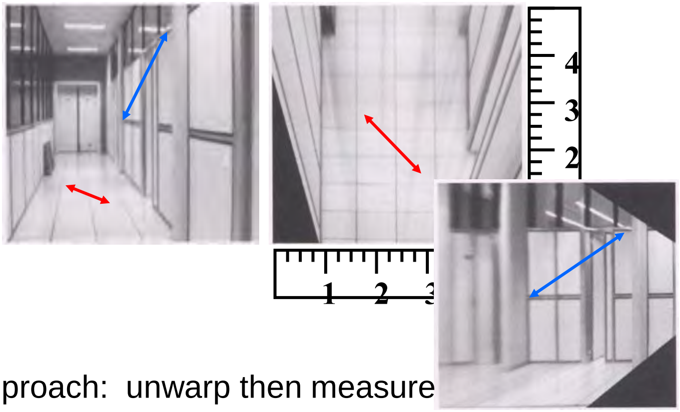
- Vignetting:
  – The tendency for the brightness of the image to fall off towards the edge of the image



  – 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!)

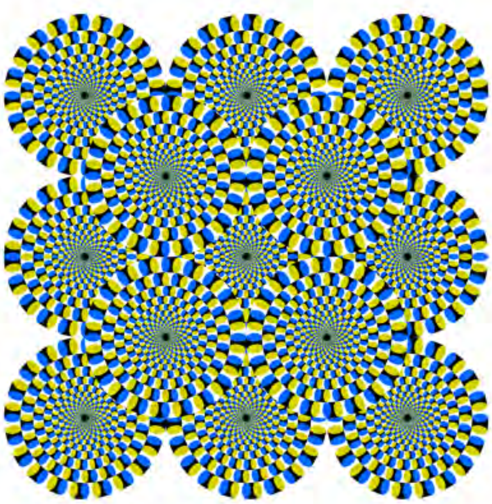


Approach: unwarp then measure

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

Perception

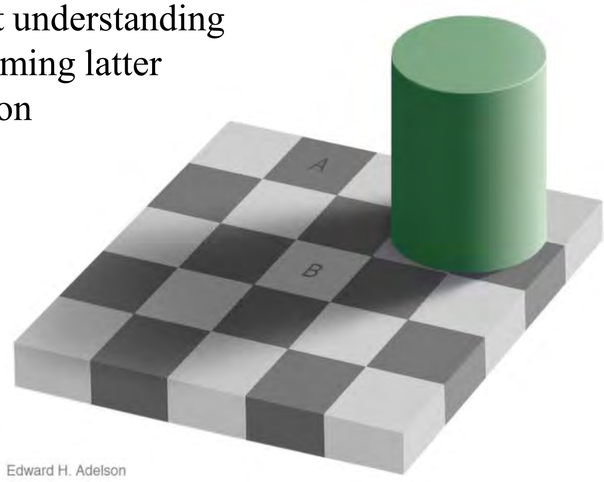
- Making Sense from Sensors



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

Perception

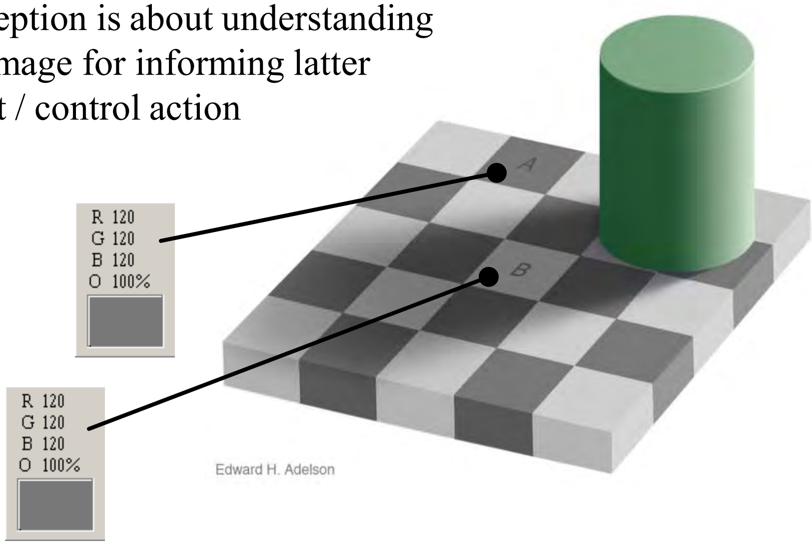
- Perception is about understanding the image for informing latter robot / control action



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

Basic Features:
~~Colour~~
Edges & Lines

METR 4202: **Robotics** September 7, 2016 - 20

Features -- Colour Features

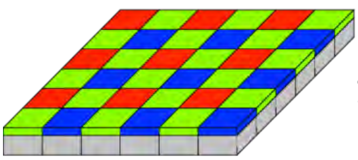
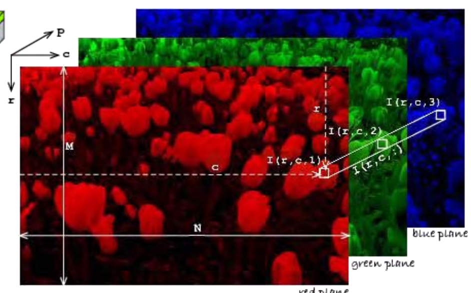


Fig: Ch. 10, *Robotics Vision and Control*

Bayer Patterns



- RGB is **NOT** an absolute (metric) colour space

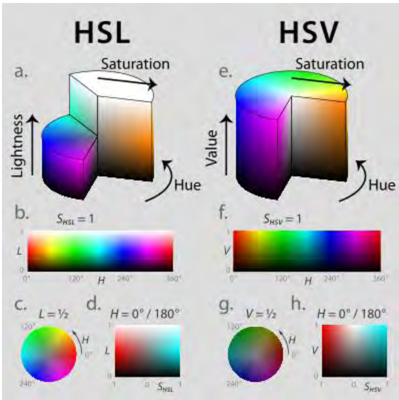
Also!

- **RGB** (display or additive colour) does not map to **CYMK** (printing or subtractive colour) without calibration
- **Y-Cr-Cb** or **HSV** does not solve this either

METR 4202: **Robotics** September 7, 2016 - 21

Colour Spaces

- HSV



Source: Wikipedia – HSV and YCrCb

- YCrCb

➔Gamma Corrected Luma (Y) + Chrominance

  ➔BW → Colour TVs : Just add the Chrominance

  ➔γ Correction: CRTs γ=2.2-2.5

$$\begin{aligned} Y' &= 16 + (65.481 \cdot R' + 128.553 \cdot G' + 24.966 \cdot B') \\ C_B &= 128 + (-37.797 \cdot R' - 74.203 \cdot G' + 112.0 \cdot B') \\ C_R &= 128 + (112.0 \cdot R' - 93.786 \cdot G' - 18.214 \cdot B') \end{aligned}$$

- L*ab

Subtractive (CMYK) & Uniform (L*ab) Color Spaces

- $C = W - R$
- $M = W - G$
- $Y = W - B$

- $K = -W$ ☺

- A Uniform color space is one in which the distance in coordinate space is a fair guide to the significance of the difference between the two colors

- Start with RGB → CIE XYZ (Under [Illuminant D65])

$$\begin{aligned} L^\star &= 116(Y/Y_n)^{(1/3)} - 16 \\ a^\star &= 500\left[(X/X_n)^{(1/3)} - (Y/Y_n)^{(1/3)}\right] \\ b^\star &= 200\left[(Y/Y_n)^{(1/3)} - (Z/Z_n)^{(1/3)}\right] \end{aligned}$$

Colour: Illumination Variant

- Toy Image

- Toy Image **With Flash**

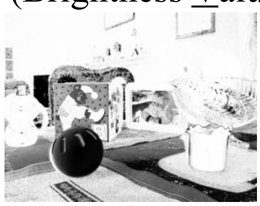
Source: %MATLABROOT%\toolbox\images\imdata\toysflash.png

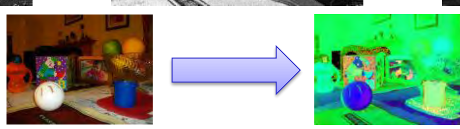
METR 4202: **Robotics** September 7, 2016 -24

Colour Spaces:

- Red | Green | Blue :

- Hue | Saturation | V (Brightness Value) :





"False-colour": Show HSV as "RGB"

METR 4202: **Robotics** September 7, 2016 -25

Lines

How to get the Features? Still MANY Ways

- Canny edge detector:

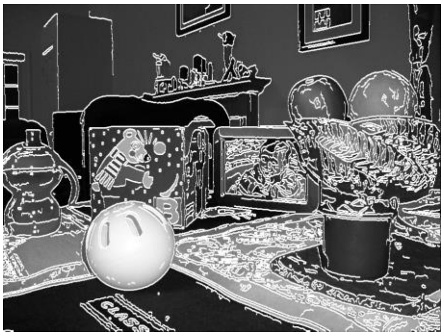
Edge Detection

- Laplacian of Gaussian
  - Gaussian (Low Pass filter)
  - Laplacian (Gradient)
- Prewitt
  - Discrete differentiation
  - Convolution

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \qquad \mathbf{G_y} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * \mathbf{A}$$

Edge Detection

- Canny edge detector
  - Finds the peak gradient magnitude orthogonal to the edge direction
    1. Apply Gaussian filter to smooth the image in order to remove the noise
    2. Find the intensity gradients of the image
    3. Apply non-maximum suppression to get rid of spurious response to edge detection
    4. Apply double threshold to determine potential edges
    5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.
  - Two Thresholds:
    - Non-maximum suppression
    - Hysteresis



Edge Detection

- Canny edge detector:
  - Pepsi Sequence:

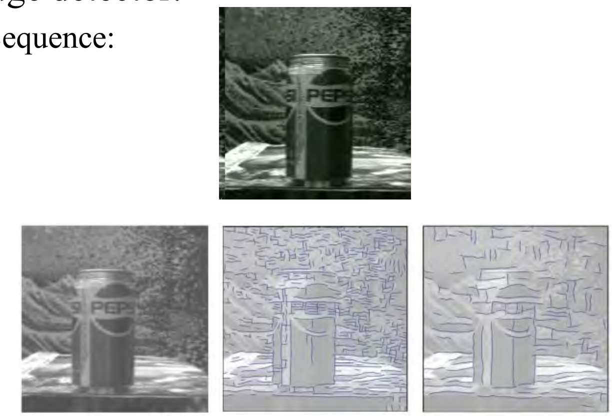
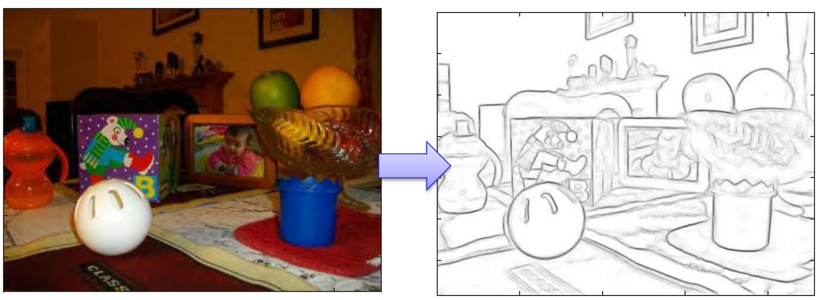


Image Data: http://www.cs.brown.edu/~black/mixtureOF.html and Szeliski, CS223B-L9
See also: Use of Temporal information to aid segmentation:
http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html

METR 4202: **Robotics** 2 September 2015 -30

Edge Detection

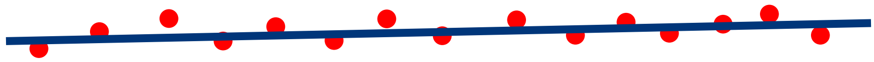
- Many, many more
  ➔ Structured Edge Detection Toolbox

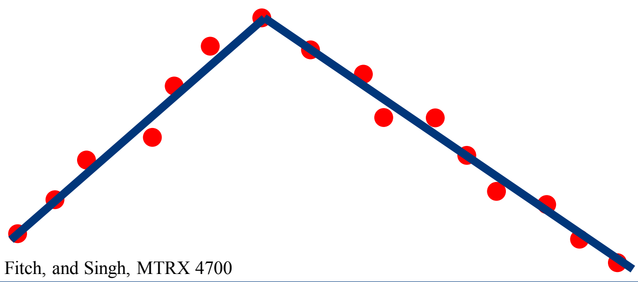


Dollár and Zitnick, Structured Forests for Fast Edge Detection, ICCV 13
https://github.com/pdollar/edges

METR 4202: **Robotics** September 7, 2016 -31

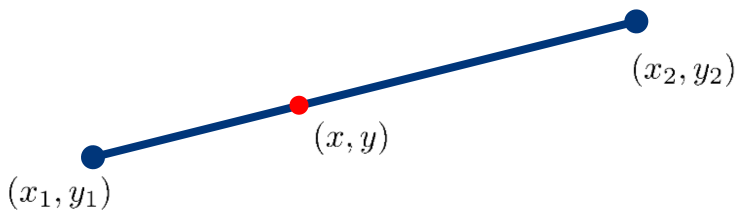
Line Extraction and Segmentation





Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Formula

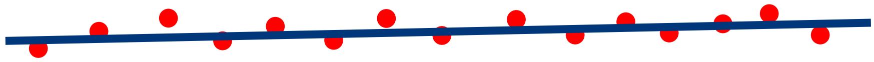


$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\mathbf{y} = m\mathbf{x} + b$$

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Estimation

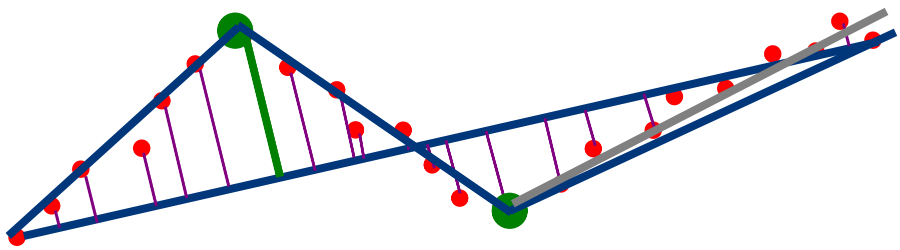


Least squares minimization of the line:

- Line Equation: $\mathbf{y} - m\mathbf{x} - b = 0$
- Error in Fit: $\sum_i (y_i - mx_i - b)^2$
- Solution: $\begin{pmatrix} \bar{x}\bar{y} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} \overline{x^2} & \bar{x} \\ \bar{x} & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix}$

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Splitting / Segmentation



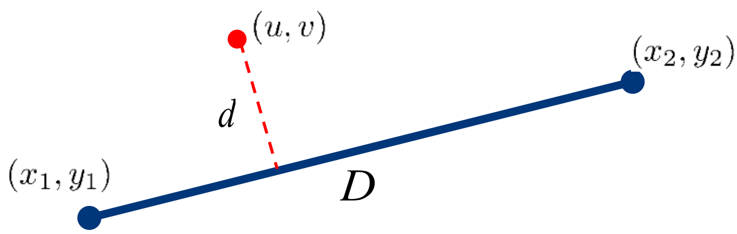
- What about corners?

➔Split into multiple lines (via expectation maximization)

1. Expect (assume) a number of lines N (say 3)
2. Find “breakpoints” by finding nearest neighbours upto a threshold or simply at random (RANSAC)
3. How to know N? (Also RANSAC)

Adopted from Williams, Fitch, and Singh, MTRX 4700

⊥ of a Point from a Line Segment



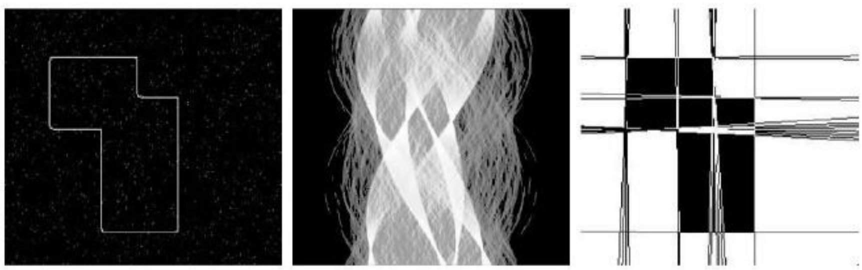
$$r = u(y_1 - y_2) + v(x_2 - x_1) + y_2 x_1 - y_1 x_2$$

$$d = \frac{r}{D}$$

Adopted from Williams, Fitch, and Singh, MTRX 4700

METR 4202: **Robotics** 2 September 2015 -36

Hough Transform

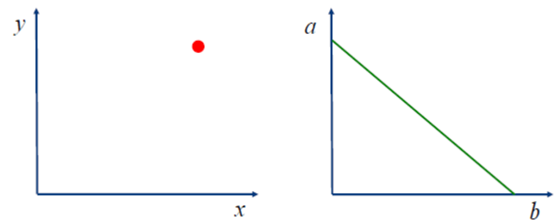


- Uses a voting mechanism
- Can be used for other lines and shapes (not just straight lines)

METR 4202: **Robotics** 2 September 2015 -37

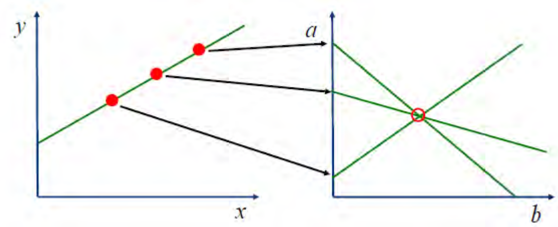
Hough Transform: Voting Space

$$y = ax + b \qquad\qquad a = -\frac{1}{x}b + \frac{y}{x}$$



- Count the number of lines that can go through a point and move it from the "*x*-*y*" plane to the "a-b" plane
- There is only a one-"infinite" number (a line!) of solutions (not a two-"infinite" set – a plane)

Hough Transform: Voting Space



- In practice, the polar form is often used

  $a = x \cos a + y \sin b$

- This avoids problems with lines that are nearly vertical

Hough Transform: Algorithm

1. Quantize the parameter space appropriately.

2. Assume that each cell in the parameter space is an accumulator. Initialize all cells to zero.

3. For each point (x,y) in the (visual & range) image space, increment by 1 each of the accumulators that satisfy the equation.

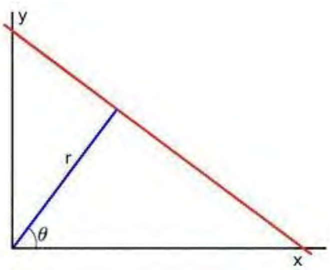
4. Maxima in the accumulator array correspond to the parameters of model instances.

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:

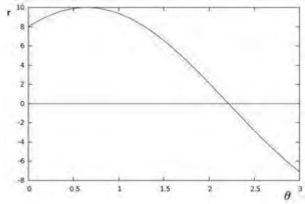
$$r = x_1 \cos\theta + y_1 \sin\theta$$

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

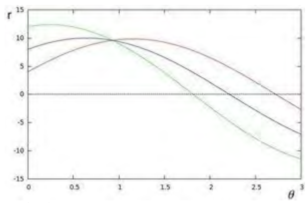
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [2]

- Thus a given point gives a sinusoid



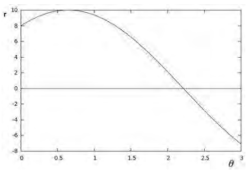
- Repeating for all points on the image



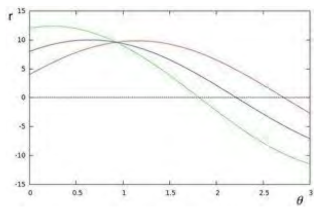
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [3]

- Thus a given point gives a sinusoid



- Repeating for all points on the image

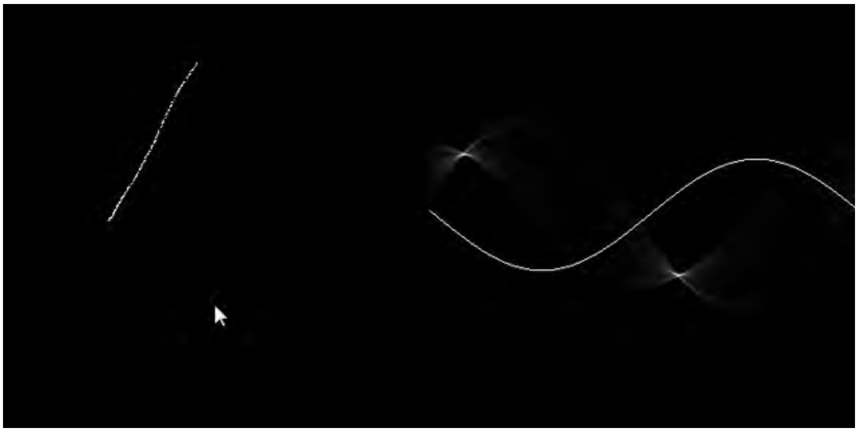


- NOTE that an intersection of sinusoids represents (**a point**) represents **a line** in which pixel points lay.

➔ Thus, a line can be *detected* by finding the number of Intersections between curves

See also OpenCV documentation (cv::HoughLines)

“Cool Robotics Share” -- Hough Transform



- http://www.activovision.com/octavi/doku.php?id=hough_transform

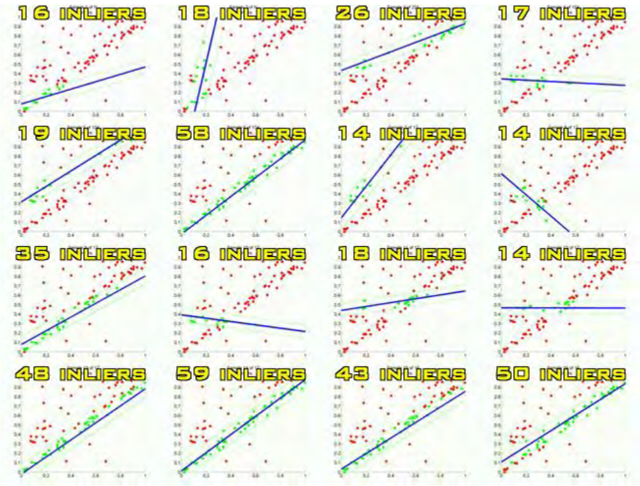
RANdom SAmple Consensus

1. Repeatedly select a small (minimal) subset of correspondences
2. Estimate a solution (in this case a the line)
3. Count the number of “inliers”, $|e|<\Theta$
   (for LMS, estimate med($|e|$))
4. Pick the *best* subset of inliers
5. Find a complete least-squares solution

- Related to least median squares
- See also:
  MAPSAC (Maximum *A Posteriori* SAmple Consensus)

From Szeliski, Computer Vision: Algorithms and Applications

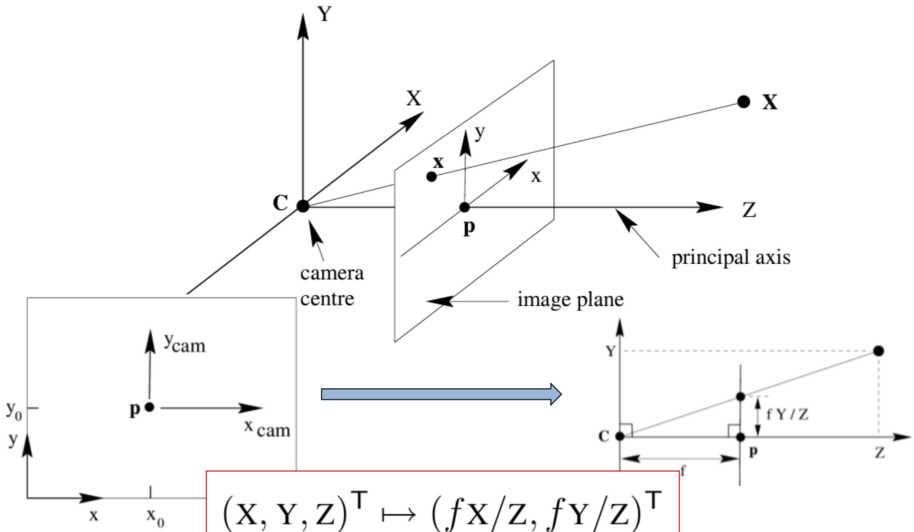
Cool Robotics Share Time!



D. Wedge, *The RANSAC Song*

Multiple View Geometry

Image Formation – Single View Geometry [I]



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

Hartly & Zisserman, Ch. 6

METR 4202: **Robotics** 2 September 2015 -48

Image Formation – Single View Geometry [II]

➔ Camera Projection Matrix

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

world camera Intrinsics

- x = Image point
- $\mathbf{X}$ = World point
- K = Camera Calibration Matrix

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

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

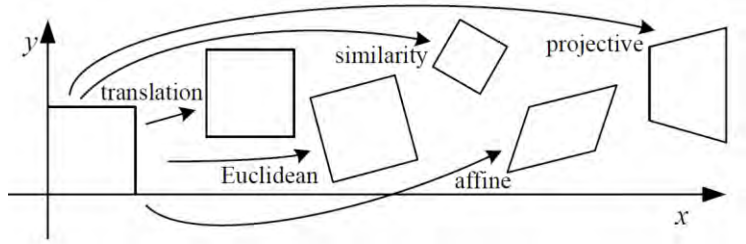
➔Perspective Camera as:

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

where: P is **3**×4 and of **rank 3**

METR 4202: **Robotics** 2 September 2015 -49

Transformations

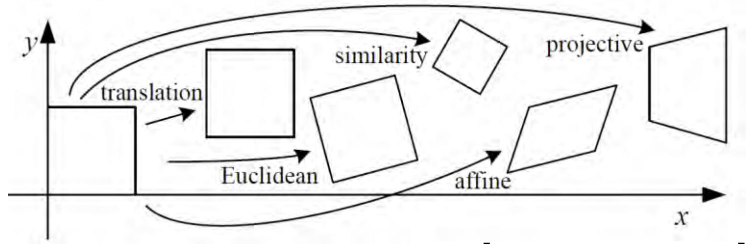


- ***x*'**: New Image & ***x*** : Old Image
- Euclidean: (Distances preserved) $x' = \begin{bmatrix} R & t \end{bmatrix} \underline{x}$
- Similarity (Scaled Rotation): (Angles preserved) $x' = \begin{bmatrix} sR & t \end{bmatrix} \underline{x}$

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

METR 4202: **Robotics** 2 September 2015 -50

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

METR 4202: **Robotics** 2 September 2015 -51

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

2-D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, I \,\middle|\, t \,\right]_{2\times 3}$	2	orientation $+ \cdots$	
rigid (Euclidean)	$\left[\, R \,\middle|\, t \,\right]_{2\times 3}$	3	lengths $+ \cdots$	
similarity	$\left[\, sR \,\middle|\, t \,\right]_{2\times 3}$	4	angles $+ \cdots$	
affine	$\left[\, A \,\right]_{2\times 3}$	6	parallelism $+ \cdots$	
projective	$\left[\, \tilde{H} \,\right]_{3\times 3}$	8	straight lines	

3D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{3\times4}$	3	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{3\times4}$	6	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{3\times4}$	7	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{3\times4}$	12	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{4\times4}$	15	straight lines	

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

Projection Models

- Orthographic
- Weak Perspective
- Affine
- Perspective
- Projective

$$\mathbf{\Pi} = \begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{\Pi} = f\begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{\Pi} = [\mathbf{R} \quad \mathbf{t}]$$

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

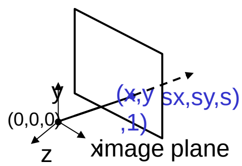
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Properties of Projection

- Preserves
  - Lines and conics
  - Incidence
  - Invariants (cross-ratio)
- Does not preserve
  - Lengths
  - Angles
  - Parallelism

Planar Projective Transformations

- Perspective projection of a plane
  - lots of names for this:
    - homography, colineation, planar projective map
  - Easily modeled using homogeneous coordinates



$$\underset{\mathbf{p'}}{\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix}} = \underset{\mathbf{H}}{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}} \underset{\mathbf{p}}{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$

To apply a homography **H**

- compute **p'** = **Hp**
- **p"** = **p'**/s normalize by dividing by third component

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

Image Formation – Two-View Geometry [Stereopsis]
➔ Fundamental Matrix

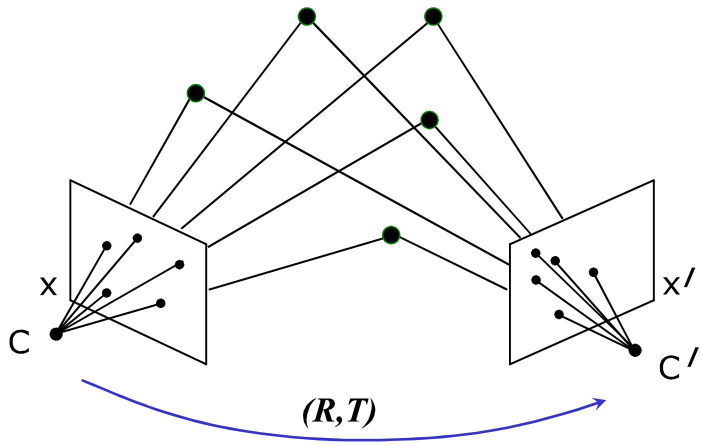
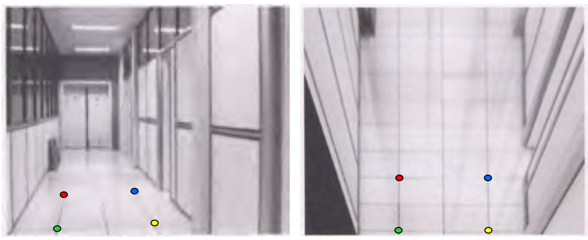


Image Rectification



To unwarp (rectify) an image

- solve for **H** given **p”** and **p**
- solve equations of the form: s**p”** = **Hp**
  - linear in unknowns: s and coefficients of **H**
  - need at least 4 points

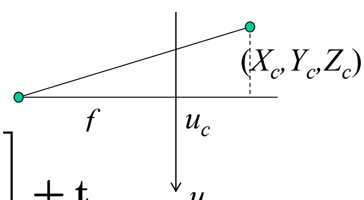
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D Projective Geometry

- These concepts generalize naturally to 3D
  - Homogeneous coordinates
    - Projective 3D points have four coords: P = (X,Y,Z,W)
  - Duality
    - A plane L is also represented by a 4-vector
    - Points and planes are dual in 3D: L P=0
  - Projective transformations
    - Represented by 4x4 matrices T: P' = TP, L' = L T-1
  - Lines are a special case…

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

3D → 2D Perspective Projection (Image Formation Equations)



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

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

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

3D → 2D Perspective Projection

- Matrix Projection (camera matrix):

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$

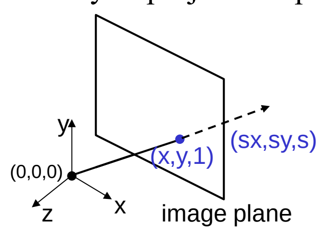
It's useful to decompose $\prod$ into $\mathbf{T} \rightarrow \mathbf{R} \rightarrow$ project $\rightarrow \mathbf{A}$

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} s_x & 0 & -t_x \\ 0 & s_y & -t_y \\ 0 & 0 & 1/f \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{orientation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{position}}$$

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

The Projective Plane

- Why do we need homogeneous coordinates?
  - Represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
  - A point in the image is a ray in projective space

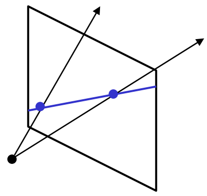


- Each *point* (x,y) on the plane is represented by a *ray* (sx,sy,s)
  - all points on the ray are equivalent: (x, y, 1) ≅ (sx, sy, s)

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

Projective Lines

- What is a line in projective space?



- A line is a *plane* of rays through origin
  - all rays (x,y,z) satisfying: ax + by + cz = 0

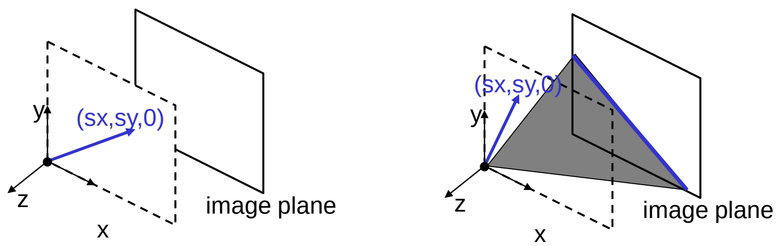
$$\text{in vector notation:} \quad 0 = \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{\mathbf{l}^T} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{p}}$$

- A line is represented as a homogeneous 3-vector **l**

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

Ideal points and lines

- Ideal point ("point at infinity")
  - p ≅ (x, y, 0) – parallel to image plane
  - It has infinite image coordinates

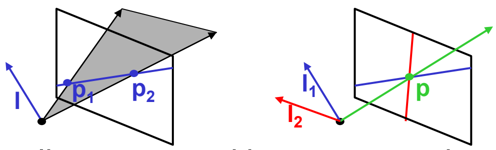


Line at infinity

- l_∞ ≅ (0, 0, 1) – parallel to image plane
- Contains all ideal points

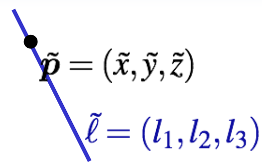
Point and Line Duality

- A line l is a homogeneous 3-vector (a ray)
- It is ⊥ to every point (ray) p on the line: lT p=0



- What is the line **l** spanned by rays $\mathbf{p_1}$ and $\mathbf{p_2}$?
  - **l** is ⊥ to $\mathbf{p_1}$ and $\mathbf{p_2}$ ⇒ $\mathbf{l} = \mathbf{p_1} \times \mathbf{p_2}$ (**l** is the plane normal)
- What is the intersection of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$?
  - **p** is ⊥ to $\mathbf{l_1}$ and $\mathbf{l_2}$ ⇒ $\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$
- Points and lines are *dual* in projective space
  - every property of points also applies to lines

Point and Line Duality [II]



Homogeneous ⇔ Cartesian

- Point:

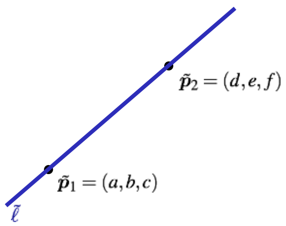
$\tilde{\boldsymbol{P}} = (\tilde{x}, \tilde{y}, \tilde{z}) \quad | \quad \boldsymbol{P} = (x, y) \quad x = \frac{\tilde{x}}{\tilde{z}},\ y = \frac{\tilde{y}}{\tilde{z}}$

- Line:
  - Is such that $\tilde{l}^T \tilde{p} = 0$
  - Point Eq of a line is: $y = mx + b$

Image/Notation from: Corke, Ch. 11

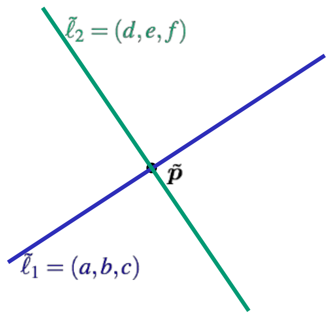
Point and Line Duality [III]

- 2 Points Make a Line



$$\tilde{\ell} = \tilde{p}_1 \times \tilde{p}_2$$

- 2 Lines Make Point!

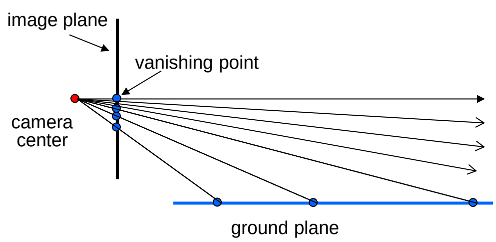


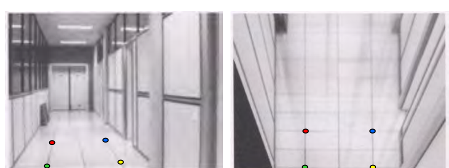
$$\tilde{p} = \tilde{\ell}_1 \times \tilde{\ell}_2$$

Image/Notation from: Corke, Ch. 11

Vanishing Points

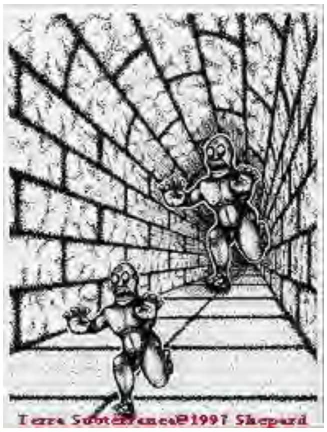
- Vanishing point
  - projection of a point at infinity
  - whiteboard capture, architecture,…





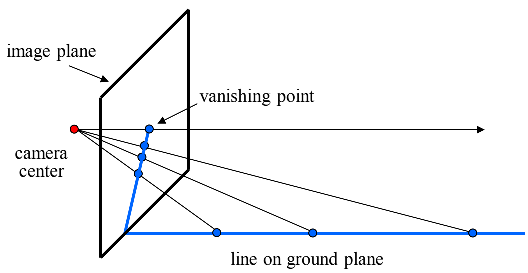
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Fun With Vanishing Points



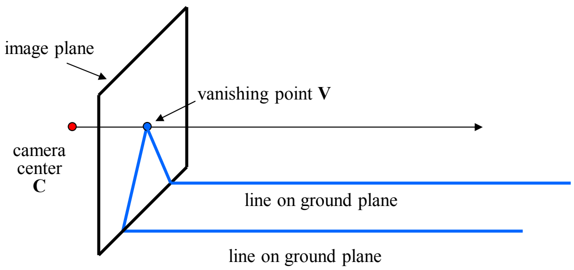
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Vanishing Points (2D)



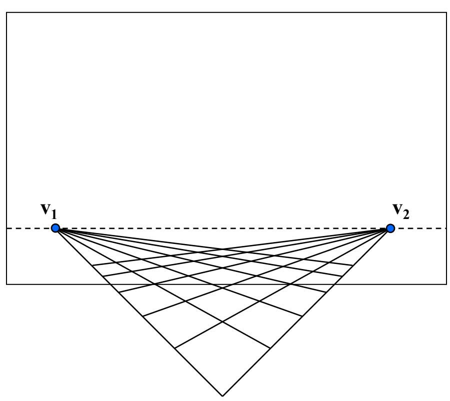
Vanishing Points

- Properties
  - Any two parallel lines have the same vanishing point
  - The ray from C through v point is parallel to the lines
  - An image may have more than one vanishing point

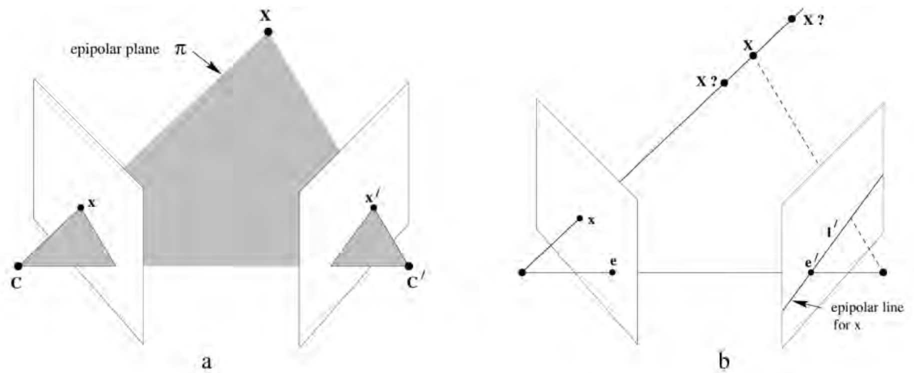


Vanishing Lines

- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the horizon line



Two-View Geometry: Epipolar Plane



- **Epipole:** The *point* of intersection of the line joining the camera centres (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera centre of the other view.
- **Epipolar plane** is a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes
- **Epipolar line** is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

Two-frame methods

- Two main variants:
- Calibrated: “Essential matrix” E
  use ray directions (xi, xi’)
- Uncalibrated: “Fundamental matrix” F
- [Hartley & Zisserman 2000]

From Szeliski, *Computer Vision: Algorithms and Applications*

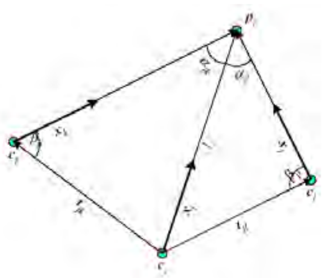
Fundamental matrix

- Camera calibrations are unknown
- x' F x = 0 with F = [e]× H = K'[t]× R K-1
- Solve for F using least squares (SVD)
  - re-scale (xi, xi') so that |xi|≈1/2 [Hartley]
- e (epipole) is still the least singular vector of F
- H obtained from the other two s.v.s
- "plane + parallax" (projective) reconstruction
- use self-calibration to determine K [Pollefeys]

From Szeliski, *Computer Vision: Algorithms and Applications*

Essential matrix

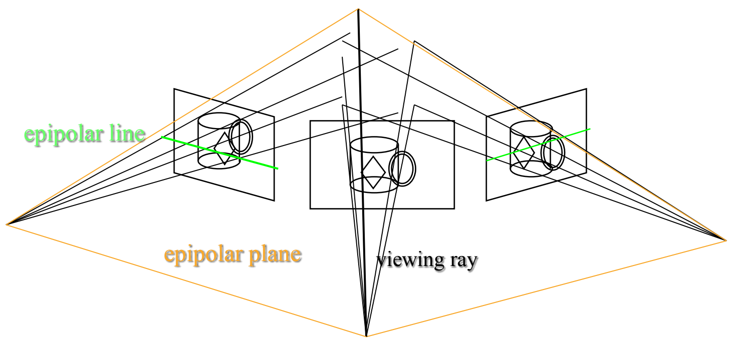
- Co-planarity constraint:
- x' ≈ R x + t
- [t]× x' ≈ [t]× R x
- x' [t]× x' ≈ x' [t]× R x
- x' E x = 0 with E =[t]× R
- Solve for E using least squares (SVD)
- t is the least singular vector of E
- R obtained from the other two s.v.s



From Szeliski, *Computer Vision: Algorithms and Applications*

Stereo: Epipolar geometry

- Match features along epipolar lines



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

Stereo: epipolar geometry

- for two images (or images with collinear camera centers), can find epipolar lines
- epipolar lines are the projection of the pencil of planes passing through the centers
- Rectification: warping the input images (perspective transformation) so that epipolar lines are horizontal

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

Fundamental Matrix

- The fundamental matrix is the algebraic representation of epipolar geometry.

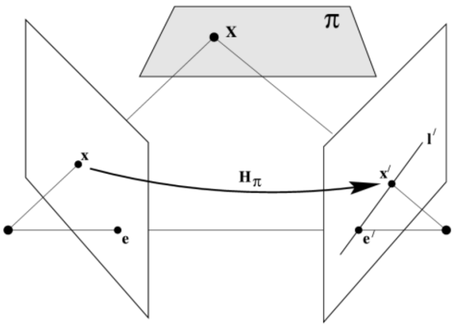


Fig. 9.5. *A point* $\mathbf{x}$ *in one image is transferred via the plane* $\boldsymbol{\pi}$ *to a matching point* $\mathbf{x}'$ *in the second image. The epipolar line through* $\mathbf{x}'$ *is obtained by joining* $\mathbf{x}'$ *to the epipole* $\mathbf{e}'$*. In symbols one may write* $\mathbf{x}' = \mathtt{H}_\pi \mathbf{x}$ *and* $\mathbf{l}' = [\mathbf{e}']_\times \mathbf{x}' = [\mathbf{e}']_\times \mathtt{H}_\pi \mathbf{x} = \mathtt{F}\mathbf{x}$ *where* $\mathtt{F} = [\mathbf{e}']_\times \mathtt{H}_\pi$ *is the fundamental matrix.*

Fundamental Matrix Example

- Suppose the camera matrices are those of a calibrated stereo rig with the world origin at the first camera

$$\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}] \qquad \mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}].$$

- Then:

$$\mathtt{P}^{+} = \begin{bmatrix} \mathtt{K}^{-1} \\ \mathbf{0}^{\mathsf{T}} \end{bmatrix} \qquad \mathbf{C} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$$

- Epipoles are at:

$$\mathbf{e} = \mathtt{P}\begin{pmatrix} -\mathtt{R}^{\mathsf{T}}\mathbf{t} \\ 1 \end{pmatrix} = \mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t} \qquad \mathbf{e}' = \mathtt{P}'\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \mathtt{K}'\mathbf{t}.$$

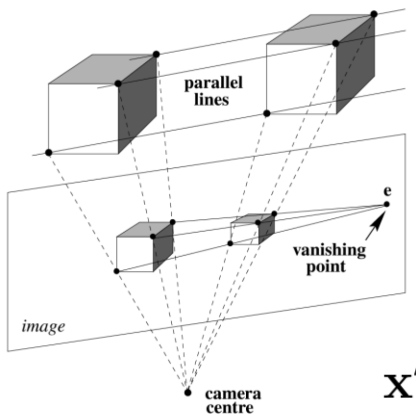
$\therefore$

$$\mathtt{F} = [\mathbf{e}']_\times \mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}[\mathtt{R}^{\mathsf{T}}\mathbf{t}]_\times \mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathbf{e}]_\times$$

Summary of fundamental matrix properties

- F is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points, then $\mathbf{x}'^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0$.
- **Epipolar lines**:
  - $\mathbf{l}' = \mathtt{F}\mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$.
  - $\mathbf{l} = \mathtt{F}^{\mathsf{T}}\mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$.
- **Epipoles**:
  - $\mathtt{F}\mathbf{e} = \mathbf{0}$.
  - $\mathtt{F}^{\mathsf{T}}\mathbf{e}' = \mathbf{0}$.
- **Computation from camera matrices** $\mathtt{P}, \mathtt{P}'$:
  - General cameras, $\mathtt{F} = [\mathbf{e}']_\times \mathtt{P}'\mathtt{P}^+$, where $\mathtt{P}^+$ is the pseudo-inverse of $\mathtt{P}$, and $\mathbf{e}' = \mathtt{P}'\mathbf{C}$, with $\mathtt{P}\mathbf{C} = \mathbf{0}$.
  - Canonical cameras, $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = [\mathtt{M} \mid \mathbf{m}]$, $\mathtt{F} = [\mathbf{e}']_\times \mathtt{M} = \mathtt{M}^{-\mathsf{T}}[\mathbf{e}]_\times$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = \mathtt{M}^{-1}\mathbf{m}$.
  - Cameras not at infinity $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$, $\mathtt{F} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}^{-1} = [\mathtt{K}'\mathbf{t}]_\times \mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t}]_\times$.

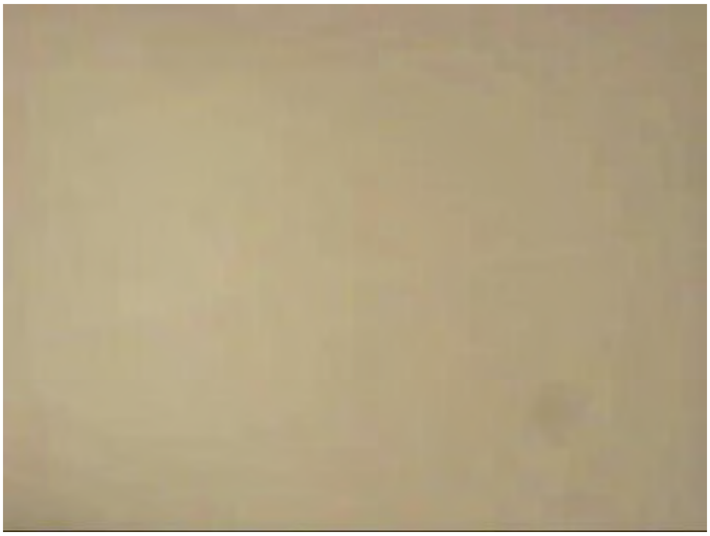
Fundamental Matrix & Motion



$$\mathbf{x}'^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0.$$

- Under a pure translational camera motion, 3D points appear to slide along parallel rails. The images of these parallel lines intersect in a vanishing point corresponding to the translation direction. The epipole **e** is the vanishing point.

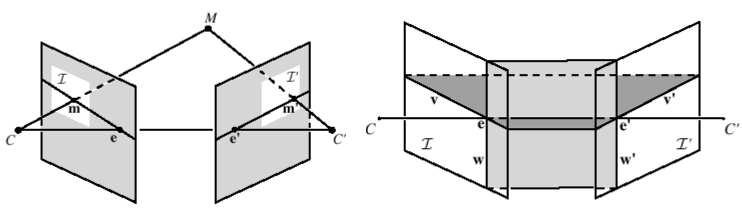
Cool Robotics Share



D. Wedge, *The Fundamental Matrix Song*

Rectification

- Project each image onto same plane, which is parallel to the epipole
- Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion



- [Zhang and Loop, MSR-TR-99-21]

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

How to get Matching Points? **Features**

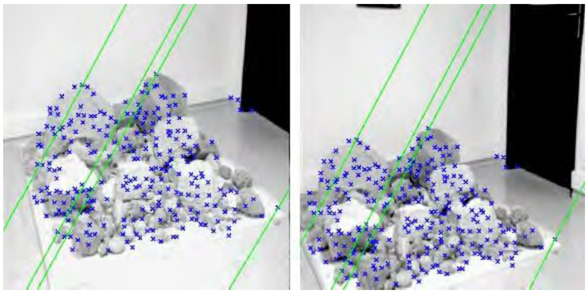
- ~~Colour~~
- Corners
- Edges
- Lines
- Statistics on Edges: SIFT, SURF, ORB…

In OpenCV: The following detector types are supported:

- "FAST" – FastFeatureDetector
- "STAR" – StarFeatureDetector
- "SIFT" – SIFT (nonfree module)
- "SURF" – SURF (nonfree module)
- "ORB" – ORB
- "BRISK" – BRISK
- "MSER" – MSER
- "GFTT" – GoodFeaturesToTrackDetector
- "HARRIS" – GoodFeaturesToTrackDetector with Harris detector enabled
- "Dense" – DenseFeatureDetector
- "SimpleBlob" – SimpleBlobDetector

Feature-based stereo

- Match "corner" (interest) points



- Interpolate complete solution

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

SFM: Structure from Motion (& Cool Robotics Share (this week))

METR 4202: **Robotics** 2 September 2015 -88

Structure [from] Motion

- Given a set of feature tracks,
  estimate the 3D structure and 3D (camera) motion.

- Assumption: orthographic projection

- Tracks: (u_{fp}, v_{fp}), f: frame, p: point
- Subtract out **mean** 2D position…

  $\mathbf{i}_f$: rotation, $\mathbf{s}_p$: position

$$u_{fp} = i_f^T s_p, v_{fp} = j_f^T s_p$$

From Szeliski, *Computer Vision: Algorithms and Applications*

METR 4202: **Robotics** 2 September 2015 -89

Structure from motion

- How many points do we need to match?
- 2 frames:
  - (R,t): 5 dof + 3n point locations ≤
  - 4n point measurements ⇒
  - n ≥ 5
- k frames:
  - 6(k–1)-1 + 3n ≤ 2kn
- always want to use many more

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

From Szeliski, Computer Vision: Algorithms and Applications

Measurement equations

- Measurement equations

$$u_{fp} = \boldsymbol{i}_f^T \boldsymbol{s}_p \qquad \boldsymbol{i}_f\text{: rotation, } \boldsymbol{s}_p\text{: position}$$
$$v_{fp} = \boldsymbol{j}_f^T \boldsymbol{s}_p$$

- Stack them up…

$$\boldsymbol{W} = \boldsymbol{R}\,\boldsymbol{S}$$
$$\boldsymbol{R} = (\boldsymbol{i}_1,\ldots,\boldsymbol{i}_F, \boldsymbol{j}_1,\ldots,\boldsymbol{j}_F)^T$$
$$\boldsymbol{S} = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_P)$$

From Szeliski, Computer Vision: Algorithms and Applications

Factorization

$$\boldsymbol{W} = \boldsymbol{R}_{2F\times3}\,\boldsymbol{S}_{3\times P}$$

SVD

$$\boldsymbol{W} = \boldsymbol{U}\,\boldsymbol{\Lambda}\,\boldsymbol{V} \quad \boldsymbol{\Lambda} \textbf{ must be rank 3}$$

$$\boldsymbol{W'} = (\boldsymbol{U}\boldsymbol{\Lambda}^{1/2})(\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}) \qquad = \boldsymbol{U'}\,\boldsymbol{V'}$$

Make $\boldsymbol{R}$ orthogonal

$$\boldsymbol{R} = \boldsymbol{QU'},\ \ \boldsymbol{S} = \boldsymbol{Q}^{-1}\boldsymbol{V'}$$

$$\boldsymbol{i}_f^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{i}_f = 1\ \ldots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Results

- Look at paper figures…

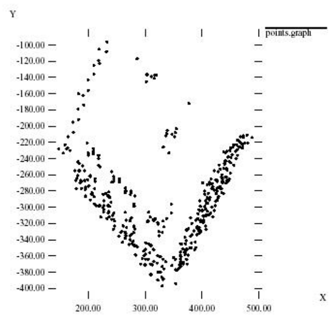


Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

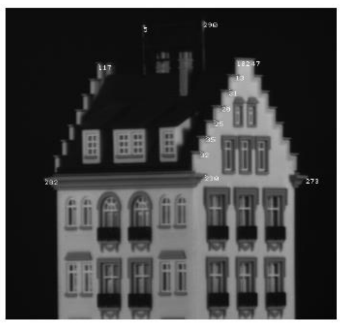


Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

From Szeliski, *Computer Vision: Algorithms and Applications*

Bundle Adjustment

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

$$\hat{u}_{ij} = f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)$$
$$\hat{v}_{ij} = g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

More Cool Robotics Share!

Robot Sensing:
Feature Detection

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 8

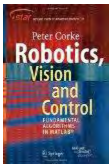
metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

September 14, 2016

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	**14-Sep**	**Robot Sensing: Multiple View Geometry & Feature Detection**
9	21-Sep	Probabilistic Robotics: Localization & SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	Planning & Control
12	19-Oct	State-Space Modelling
13	26-Oct	Shaping the Dynamic Response/LQR + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

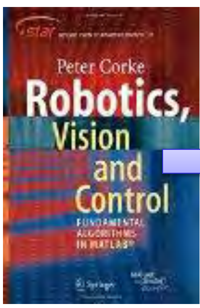
➔ Sensing and Vision ⬅

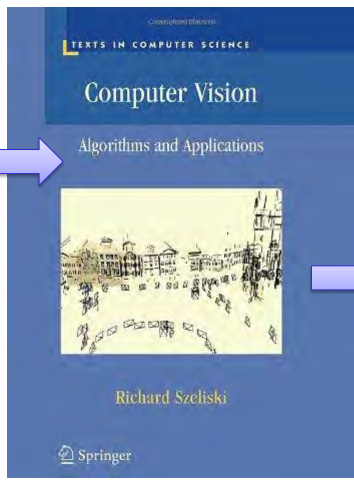
- Multiple View Geometry
  - P. 47
  - Hartley & Zisserman:
    Chapter 6: Camera Models
    Chapter 7: Camera Matrix

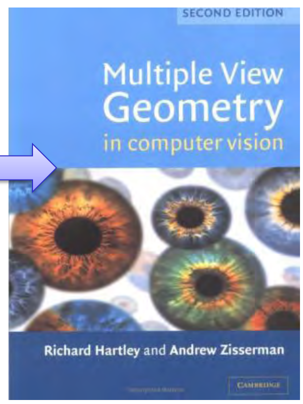
- Localization
  - Chapter 6: Localization

Next Time

Reference Material







UQ Library/ SpringerLink

UQ Library **(ePDF)**

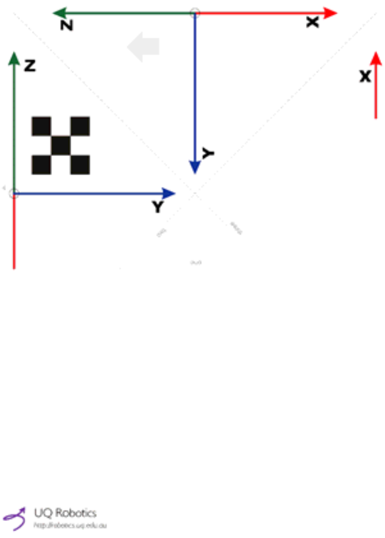
Announcement: Monday Lab → Demo Thur | Fri

- Monday, October 3:
  - Queens Birthday Public Holiday
- "Makeup" Lab
  on Friday, October 7
  from 4-6pm
- Monday Prac students
  may demo on Thursday (Oct 6)
  or Friday (Oct 7)
- Thursday Prac students to demo
  on Thursday (Oct 6)

SIFT / Corners for the {Frame} finder

To find the Frame, Consider:

- Structure
  - Corners
  - SIFT
  - ???
- Calibration Sequence
- Thought Experiment:
  How do you make this
  traceable back to the
  {camera frame}



Camera matrix calibration

- Advantages:
  - very simple to formulate and solve
  - can recover K [R | t] from M using QR decomposition [Golub & VanLoan 96]

- Disadvantages:
  - doesn't compute internal parameters
  - more unknowns than true degrees of freedom
  - need a separate camera matrix for each new view

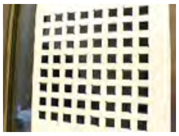
From Szeliski, *Computer Vision: Algorithms and Applications*

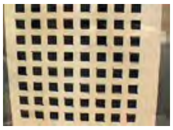
Multi-plane calibration

- Use several images of planar target held at unknown orientations [Zhang 99]
  - Compute plane homographies

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim \mathbf{HX}$$

  - Solve for K-TK-1 from Hk's
    - 1plane if only f unknown
    - 2 planes if (f,uc,vc) unknown
    - 3+ planes for full K
  - Code available from Zhang and OpenCV



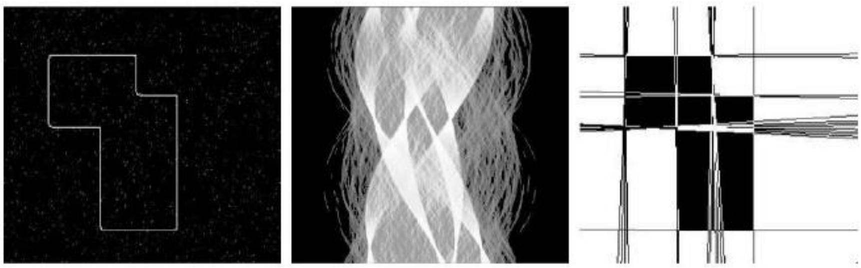


From Szeliski, *Computer Vision: Algorithms and Applications*

Lines (Recap)

METR 4202: **Robotics** September 14, 2016 - 11

Hough Transform

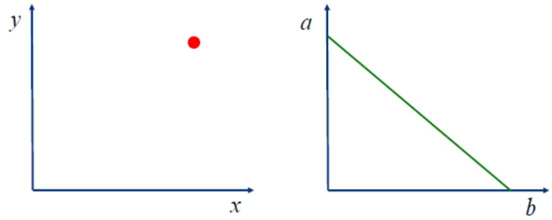


- Uses a voting mechanism
- Can be used for other lines and shapes (not just straight lines)

METR 4202: **Robotics** September 14, 2016 - 12

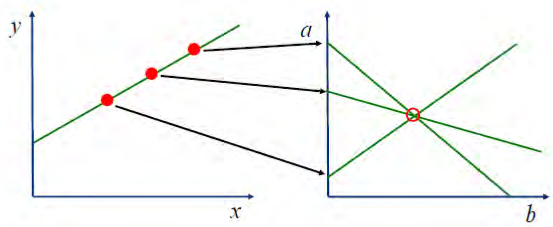
Hough Transform: Voting Space

$$y = ax + b \qquad a = -\frac{1}{x}b + \frac{y}{x}$$



- Count the number of lines that can go through a point and move it from the "*x*-*y*" plane to the "a-b" plane
- There is only a one-"infinite" number (a line!) of solutions (not a two-"infinite" set – a plane)

Hough Transform: Voting Space



- In practice, the polar form is often used

  $a = x \cos a + y \sin b$
- This avoids problems with lines that are nearly vertical

Hough Transform: Algorithm

1. Quantize the parameter space appropriately.

2. Assume that each cell in the parameter space is an accumulator. Initialize all cells to zero.

3. For each point (x,y) in the (visual & range) image space, increment by 1 each of the accumulators that satisfy the equation.

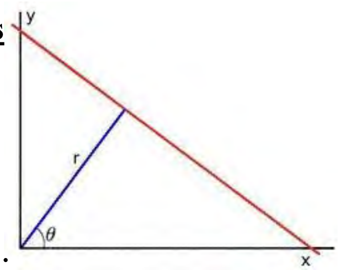
4. Maxima in the accumulator array correspond to the parameters of model instances.

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:

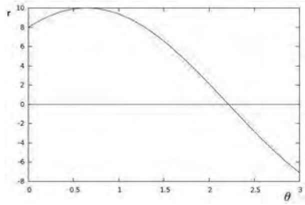
$$r = x_1 \cos\theta + y_1 \sin\theta$$

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

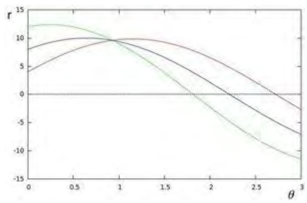
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [2]

- Thus a given point gives a sinusoid



- Repeating for all points on the image

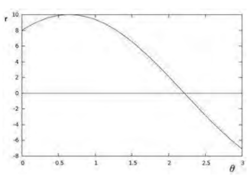


See also OpenCV documentation (cv::HoughLines)

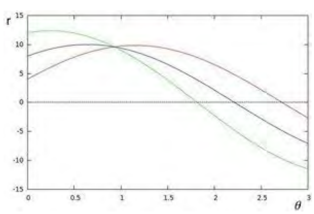
METR 4202: **Robotics** September 14, 2016 - 17

Line Detection – Hough Lines [3]

- Thus a given point gives a sinusoid



- Repeating for all points on the image



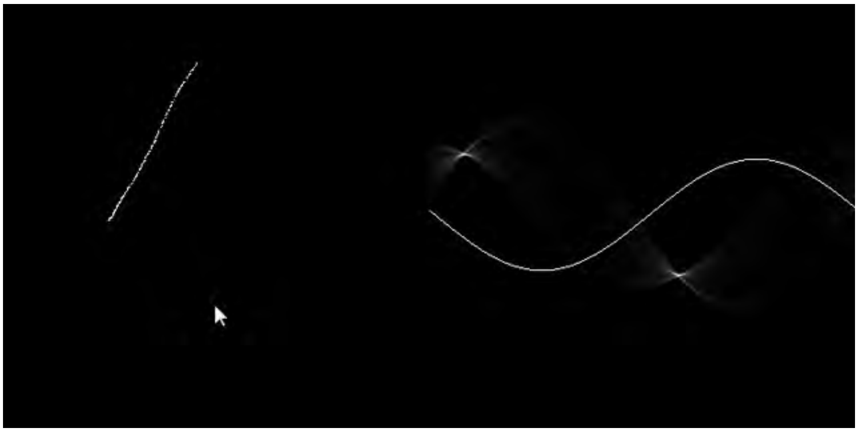
- NOTE that an intersection of sinusoids represents (**a point**) represents **a line** in which pixel points lay.

➔ Thus, a line can be *detected* by finding the number of Intersections between curves

See also OpenCV documentation (cv::HoughLines)

METR 4202: **Robotics** September 14, 2016 - 18

“Cool Robotics Share” -- Hough Transform



- http://www.activovision.com/octavi/doku.php?id=hough_transform

RANdom SAmple Consensus

1. Repeatedly select a small (minimal) subset of correspondences
2. Estimate a solution (in this case a the line)
3. Count the number of “inliers”, $|e|<\Theta$
   (for LMS, estimate med($|e|$))
4. Pick the *best* subset of inliers
5. Find a complete least-squares solution

- Related to least median squares
- See also:
  MAPSAC (Maximum *A Posteriori* SAmple Consensus)

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature Detection

METR 4202: **Robotics** September 14, 2016 - 21

"A Rose By Any Other Name?

3 8 1 7 6 7 4 7 8 3 5 9 5 3 6 3 7 4 4 6 9 3 8 7 9 0 3 6 3 2 6 6 5 6 0 3 4 2 6 8 3 8 1…
7 6 7 4 7 8 3 5 9 5 3 6 3 7 4 4 6 9 3 8 7 9 0 3 6 3 2 6 6 5 6 0 3 4 2 6 8

– SIFT

METR 4202: **Robotics** September 14, 2016 - 22

How to get Matching Points? **Features**

- ~~Colour~~
- Corners
- Edges
- Lines
- Statistics on Edges: SIFT, SURF, ORB…

In OpenCV: The following detector types are supported:

- "FAST" – FastFeatureDetector
- "STAR" – StarFeatureDetector
- "SIFT" – SIFT (nonfree module)
- "SURF" – SURF (nonfree module)
- "ORB" – ORB
- "BRISK" – BRISK
- "MSER" – MSER
- "GFTT" – GoodFeaturesToTrackDetector
- "HARRIS" – GoodFeaturesToTrackDetector with Harris detector enabled
- "Dense" – DenseFeatureDetector
- "SimpleBlob" – SimpleBlobDetector

Why extract features?

- **Object detection**
- Robot Navigation
- Scene Recognition

- Steps:
  - Extract Features
  - Match Features

Adopted drom S. Lazebnik, Gang Hua (CS 558)

Why extract features? [2]

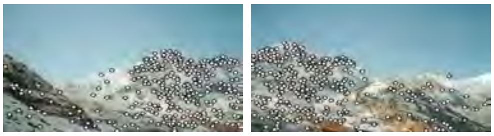
- Panorama stitching…
  - → Step 3: Align images

Adopted from S. Lazebnik, Gang Hua (CS 558)

METR 4202: **Robotics** September 14, 2016 - 25

Characteristics of good features

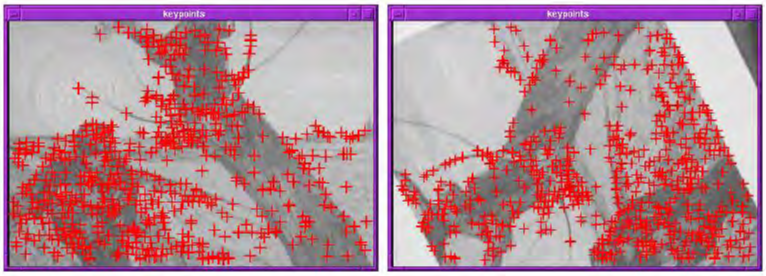
- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion



Adopted from S. Lazebnik, Gang Hua (CS 558)

METR 4202: **Robotics** September 14, 2016 - 26

Finding Corners



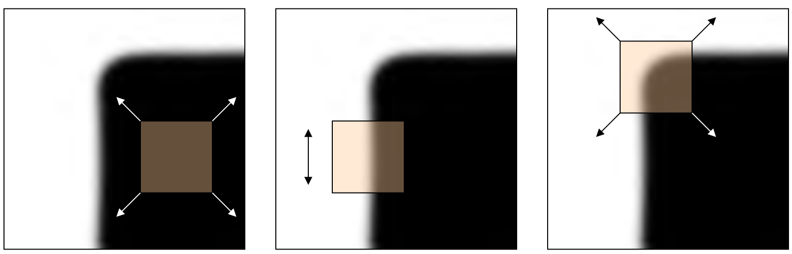
- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

C.Harris and M.Stephens. "A Combined Corner and Edge Detector.“ *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Adopted from S. Lazebnik, Gang Hua (CS 558)

Corner Detection: Basic Idea

- Look through a window
- Shifting a window in any direction should give a large change in intensity



“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions

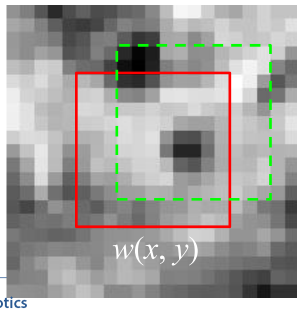
Source: A. Efros

Corner Detection: Mathematics

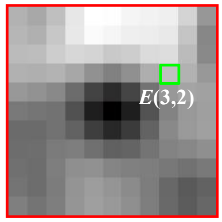
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

$I(x, y)$



$E(u, v)$



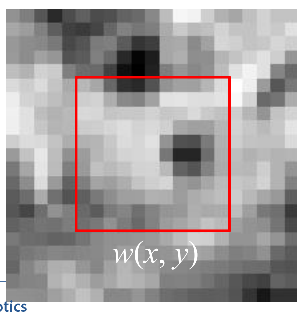
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

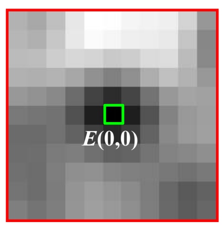
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

$I(x, y)$



$E(u, v)$



Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

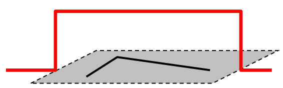
$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

Window function

Shifted intensity

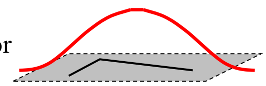
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Source: R. Szeliski

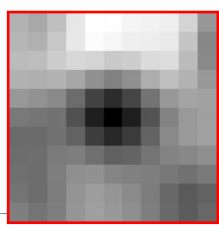
Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

We want to find out how this function behaves for small shifts

$E(u, v)$



Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

We want to find out how this function behaves for small shifts

$$E(u,v) \approx E(0,0) + [u\ \ v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u\ \ v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Local quadratic approximation of $E(u,v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + [u\ \ v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u\ \ v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E_u(u,v) = \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right]I_x(x+u,y+v)$$

$$E_{uu}(u,v) = \sum_{x,y} 2w(x,y)I_x(x+u,y+v)I_x(x+u,y+v) + \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right]I_{xx}(x+u,y+v)$$

$$E_{uv}(u,v) = \sum_{x,y} 2w(x,y)I_y(x+u,y+v)I_x(x+u,y+v) + \sum_{x,y} 2w(x,y)\left[I(x+u,y+v) - I(x,y)\right]I_{xy}(x+u,y+v)$$

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx [u \;\; v]\begin{bmatrix} \sum_{x,y} w(x,y)I_x^2(x,y) & \sum_{x,y} w(x,y)I_x(x,y)I_y(x,y) \\ \sum_{x,y} w(x,y)I_x(x,y)I_y(x,y) & \sum_{x,y} w(x,y)I_y^2(x,y) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$
$$E_u(0,0) = 0$$
$$E_v(0,0) = 0$$
$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y)I_x(x,y)I_x(x,y)$$
$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y)I_y(x,y)I_y(x,y)$$
$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y)I_x(x,y)I_y(x,y)$$

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Harris detector: Steps

- Compute Gaussian derivatives at each pixel
- Compute second moment matrix M in a Gaussian window around each pixel
- Compute corner response function R
- Threshold R
- Find local maxima of response function (nonmaximum suppression)

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

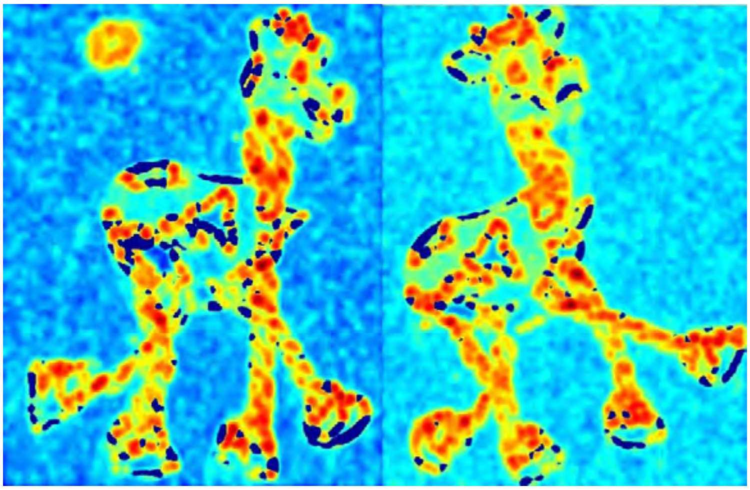
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Harris Detector: Steps

Adopted from S. Lazebnik, Gang Hua ([CS 558](CS 558))

Harris Detector: Steps

Compute corner response *R*



Adopted from S. Lazebnik, Gang Hua ([CS 558](CS 558))

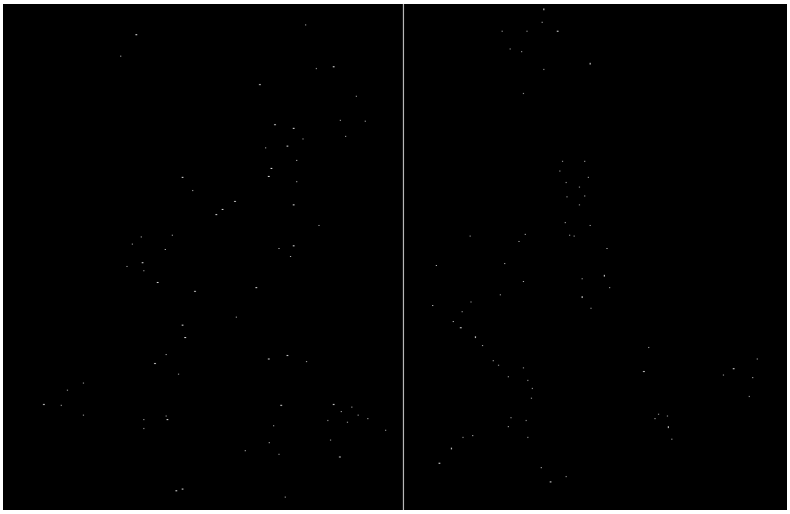
Harris Detector: Steps

Find points with large corner response: R>threshold

Adopted from S. Lazebnik, Gang Hua (CS 558)

Harris Detector: Steps

Take only the points of local maxima of R



Adopted from S. Lazebnik, Gang Hua (CS 558)

Harris Detector: Steps

Adopted from S. Lazebnik, Gang Hua (CS 558)

METR 4202: **Robotics** September 14, 2016 -41

Invariance and covariance

- We want corner locations to be invariant to photometric transformations and covariant to geometric transformations
  - Invariance: image is transformed and corner locations do not change
  - Covariance: if we have two transformed versions of the same image, features should be detected in corresponding locations

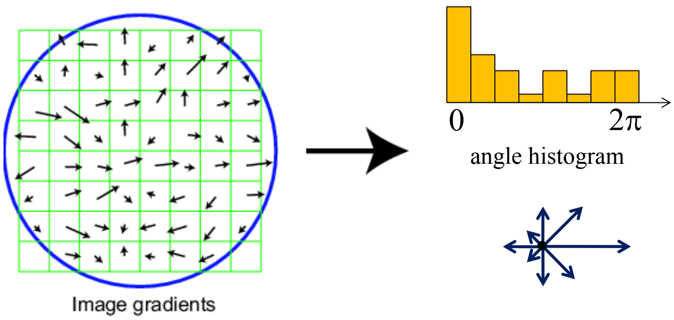
Adopted from S. Lazebnik, Gang Hua (CS 558)

METR 4202: **Robotics** September 14, 2016 -42

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

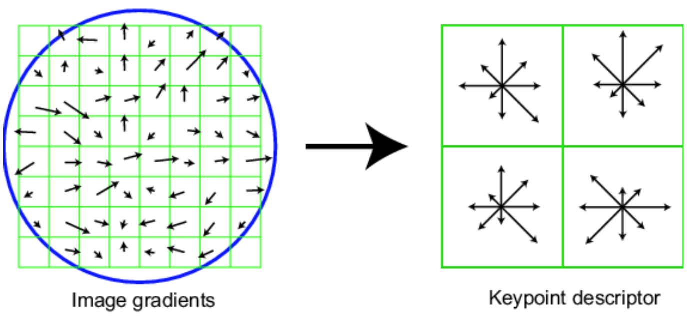


Adapted from slide by David Lowe

SIFT descriptor

Full version

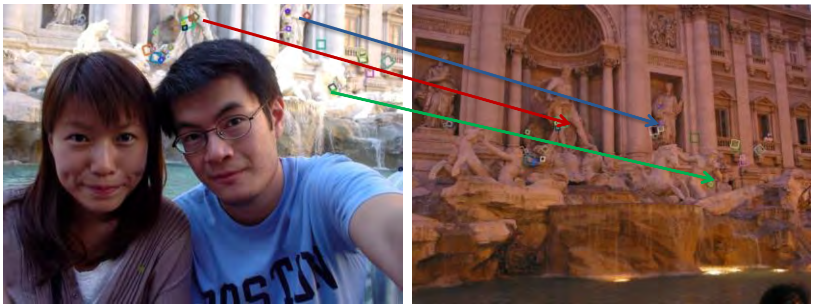
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Adapted from slide by David Lowe

Properties of SIFT

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available
    - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



From David Lowe and Szeliski, *Computer Vision: Algorithms and Applications*

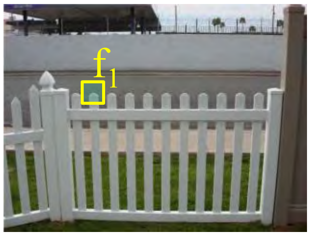
Feature matching

- Given a feature in I_1, how to find the best match in I_2?
  1. Define distance function that compares two descriptors
  2. Test all the features in I_2, find the one with min distance

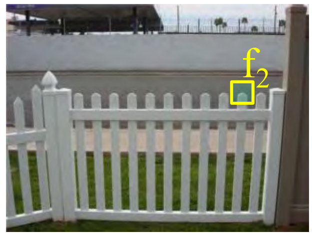
From Szeliski, *Computer Vision: Algorithms and Applications*

Feature distance

- How to define the difference between two features f_1, f_2?
  - Simple approach is SSD(f_1, f_2)
    - sum of square differences between entries of the two descriptors
    - can give good scores to very ambiguous (bad) matches



I_1

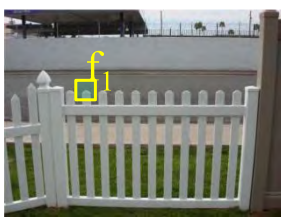


I_2

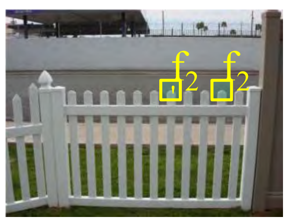
From Szeliski, *Computer Vision: Algorithms and Applications*

Feature distance

- How to define the difference between two features f_1, f_2?
  - Better approach: ratio distance = SSD(f_1, f_2) / SSD(f_1, f_2')
    - f_2 is best SSD match to f_1 in I_2
    - f_2' is 2[nd] best SSD match to f_1 in I_2
    - gives small values for ambiguous matches



I_1

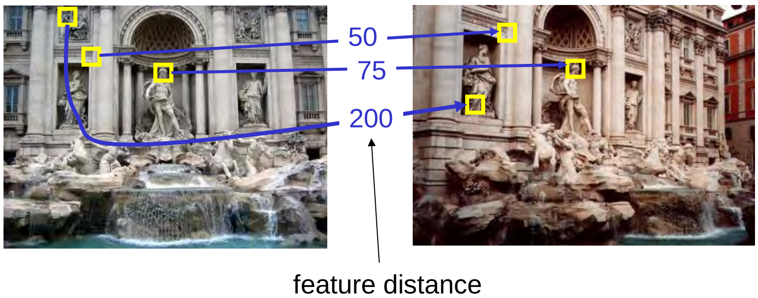


I_2

From Szeliski, *Computer Vision: Algorithms and Applications*

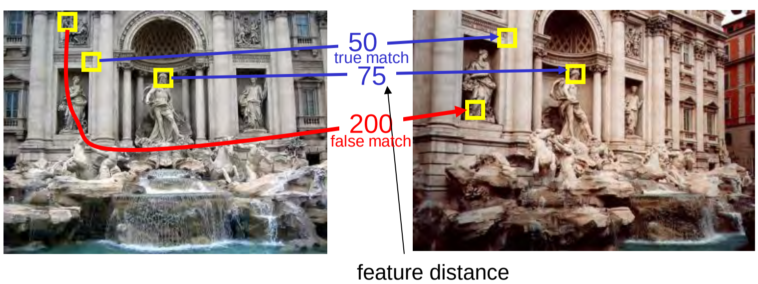
Evaluating the results

- How can we measure the performance of a feature matcher?



From Szeliski, [Computer Vision: Algorithms and Applications]()

True/false positives



- The distance threshold affects performance
  - True positives = # of detected matches that are correct
    - Suppose we want to maximize these—how to choose threshold?
  - False positives = # of detected matches that are incorrect
    - Suppose we want to minimize these—how to choose threshold?

From Szeliski, [Computer Vision: Algorithms and Applications]()

Levenberg-Marquardt

- Iterative non-linear least squares [Press'92]
  - Linearize measurement equations

$$\begin{aligned}\hat{u}_i &= f(\mathbf{m}, \mathbf{x}_i) + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m} \\ \hat{v}_i &= g(\mathbf{m}, \mathbf{x}_i) + \frac{\partial g}{\partial \mathbf{m}} \Delta \mathbf{m}\end{aligned}$$

  - Substitute into log-likelihood equation: quadratic cost function in Dm

$$\sum_i \sigma_i^{-2} (\hat{u}_i - u_i + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m})^2 + \cdots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Levenberg-Marquardt

- What if it doesn't converge?
  - Multiply diagonal by (1 + l), increase l until it does
  - Halve the step size Dm (my favorite)
  - Use line search
  - Other ideas?
- Uncertainty analysis: covariance S = A-1
- Is maximum likelihood the best idea?
- How to start in vicinity of global minimum?

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature Based Vision Extras

METR 4202: **Robotics** September 14, 2016 -53

Scale Invariant Feature Transforms

- Goal was to define an algorithm to describe an image with features
- This would enable a number of different applications:
  - Feature Matching
  - Object / Image Matching
  - Orientation / Homography Resolution

Wikipedia: Scale Invariant Feature Transforms (2014)

METR 4202: **Robotics** September 14, 2016 -54

SIFT: Feature Definition

- SIFT features are defined as the local extrema in a Difference of Gaussian (D) Scale Pyramid.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_i\sigma)$$

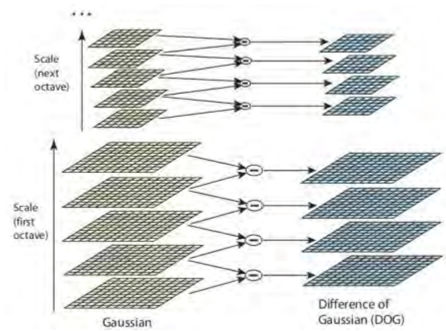
Where

$$L(x, y, k_i\sigma) = G(x, y, k\sigma) * I(x, y)$$

METR 4202: **Robotics** September 14, 2016 -55

SIFT: Scale Pyramid

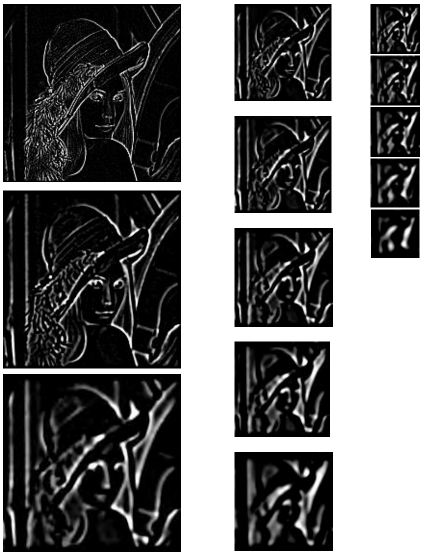
- D images are organised into a pyramid of progressively blurred images.
- Separated into octaves and scale levels per octave.
- Between octaves image is decimated by a factor of 2.

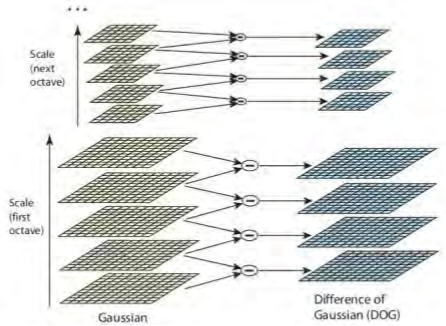


Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

METR 4202: **Robotics** September 14, 2016 -56

SIFT: Scale Pyramid

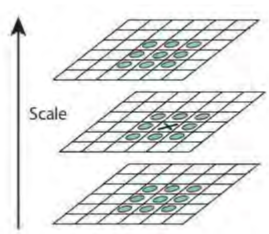




Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

SIFT: Feature Detection

- Each scale level in the image is evaluated for features.
- A feature is defined as a local maximum or minimum.
- For efficiency the 26 surrounding points are evaluated.



Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

SIFT: Feature Reduction

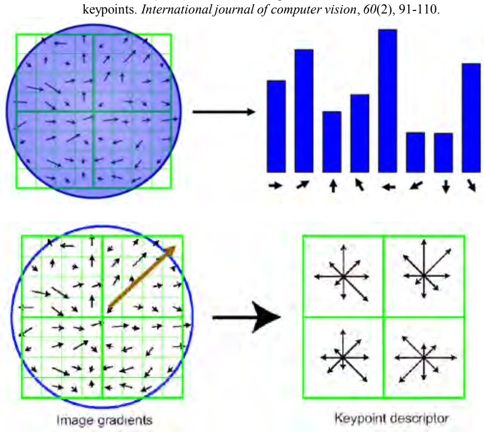
- Initial feature detection over detects features descriptive of the image.
- Initially remove features with low contrast.
- Then evaluate features to remove any edge responses.

Wikipedia: Scale Invariant Feature Transforms (2014)

SIFT: Feature Description

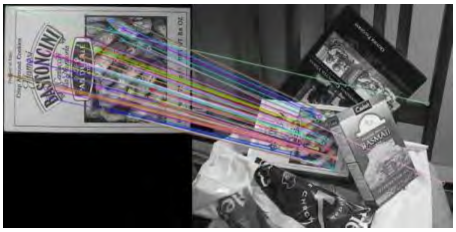
- Features are described using the pixel gradients in a 16x16 square centring on the feature point.
- These gradients are then segmented into 4x4 boxes. An 8 bin orientation histogram is created to define the box.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.



SIFT: Feature Matching

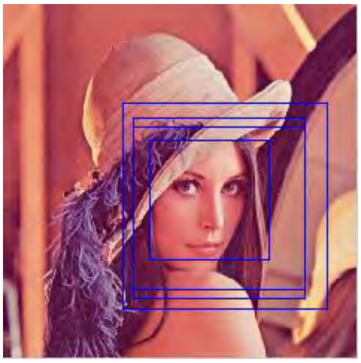
- A match is defined as a pair of features with the closest Euclidian distance to each other.
- Matches above a threshold are culled to improve match.



OpenCV: Feature Matching (2014)

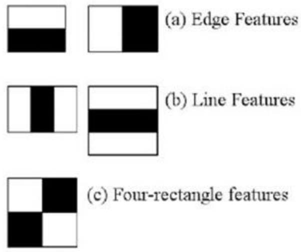
Boosted Cascade Haar-like Weak Classifiers

- Fast object detector designed primarily for use in face detection.
- Uses a cascade of weak classifiers to define object match.



Viola Jones: Feature Definition

- Feature is classified as being the difference between the average intensity of two or more image sections.
- Can be any arithmetic combination of section values.



Viola Jones: Efficient Calculation of Features

- Fast calculation of the feature value is obtained by calculating the integral image.
- This leaves at most 4 sum operations to calculate a feature.

1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

input image

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

integral image

Viola Jones: Boosting

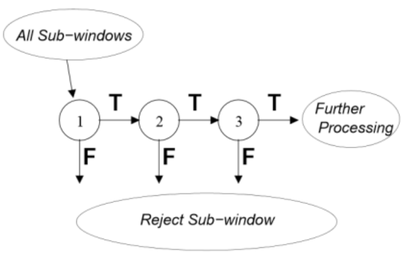
- Iteratively selects best classifier for detection.
- Assigns weights to each classifier to indicate likelihood of classifier indicating positive detection
- If the sum of the weights of positive classifier responses is above a threshold then there is a positive detection.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier , h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features

Viola Jones: Boosted Cascades

- Effective boosted classifiers require a high number of weak classifiers.
- However, simple low count classifiers offer high rejection rate.
- Solution is to use cascaded classifiers.



Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features

Robot Sensing: Multiple View Geometry

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 9

September 21, 2016

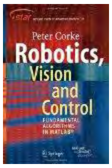
metr4202@itee.uq.edu.au

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

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	14-Sep	Robot Sensing: Feature Detection
9	**21-Sep**	**Robot Sensing: Multiple View Geometry**
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	Probabilistic Robotics: Localization & SLAM
12	19-Oct	Probabilistic Robotics: Planning & Control
13	26-Oct	State-Space Automation (Shaping the Dynamic Response/LQR) + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

➔ Sensing and Vision ⇐

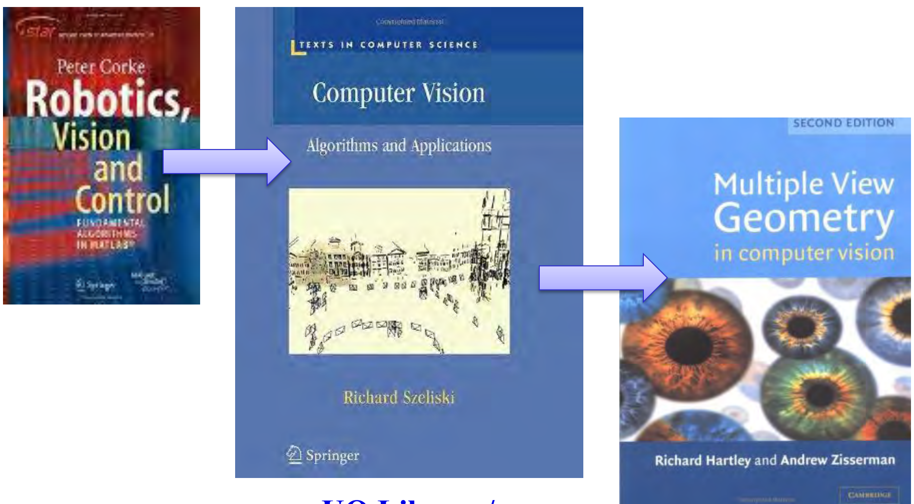
- Multiple View Geometry
  - P. 47
  - Hartley & Zisserman:
    Chapter 6: Camera Models
    Chapter 7: Camera Matrix

- Planning
  - pp. 91-103
    (Yup! That's all Peter Corke has to say on that – which explains why there is no planning at ACRV ☺).

Next Time

METR 4202: **Robotics** September 21, 2016 - 3

Reference Material



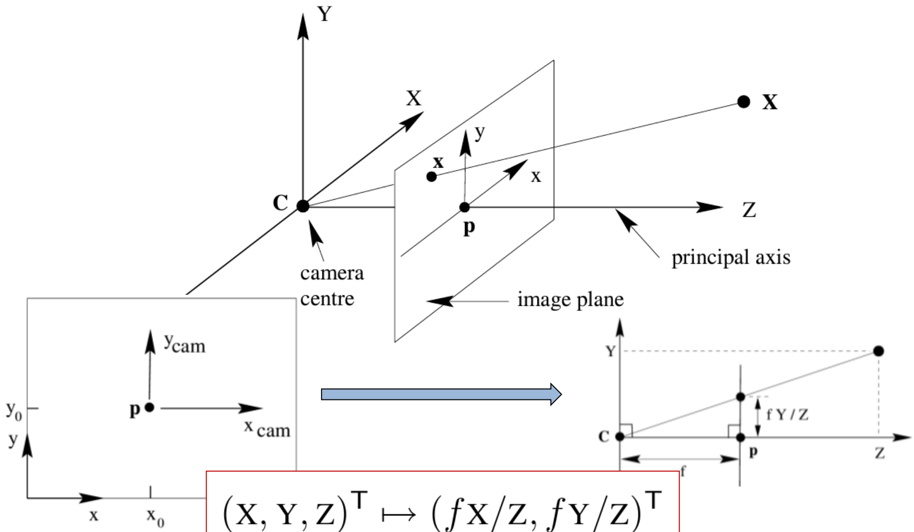
UQ Library/ SpringerLink

UQ Library (ePDF)

METR 4202: **Robotics** September 21, 2016 - 4

Multiple View Geometry

Image Formation – Single View Geometry [I]



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

Hartly & Zisserman, Ch. 6

Image Formation – Single View Geometry [II]

➔ Camera Projection Matrix

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

world camera Intrinsics

- x = Image point
- $\mathbf{X}$ = World point
- K = Camera Calibration Matrix

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

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

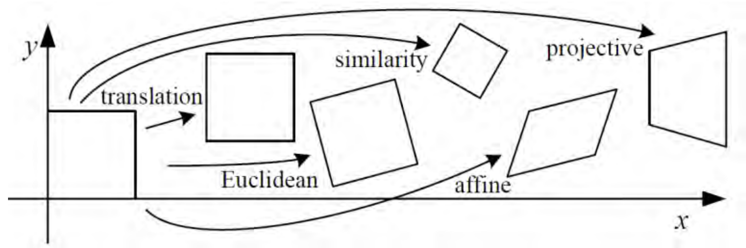
➔Perspective Camera as:

where: P is **3**×4 and of **rank 3**

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

METR 4202: **Robotics** September 21, 2016 - 7

Transformations ☆



- ***x*'**: New Image & **x̲** : Old Image
- Euclidean:
  (Distances preserved)

$$x' = \begin{bmatrix} R & t \end{bmatrix} \underline{x}$$

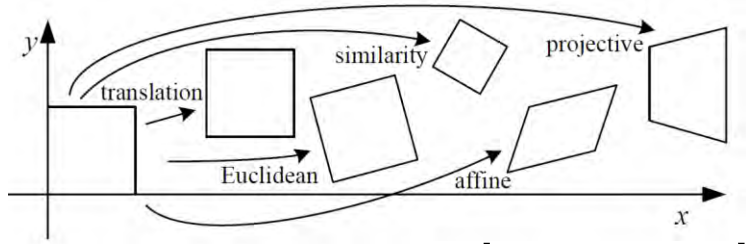
- Similarity (Scaled Rotation):
  (Angles preserved)

$$x' = \begin{bmatrix} sR & t \end{bmatrix} \underline{x}$$

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

METR 4202: **Robotics** September 21, 2016 - 8

Transformations [2]



- Affine :
  (|| lines remain ||)
- Projective:
  (straight lines preserved)
  H: Homogenous 3x3 Matrix

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \underline{x}$$

$$x' = \mathbf{H}\underline{x}$$

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$$

$$y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

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

2-D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times 3}$	8	straight lines	

3D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{3\times 4}$	3	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{3\times 4}$	6	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{3\times 4}$	7	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{3\times 4}$	12	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{4\times 4}$	15	straight lines	

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

Projection Models

- Orthographic

$$\mathbf{\Pi} = \begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Weak Perspective

$$\mathbf{\Pi} = f \begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Affine

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Perspective

$$\mathbf{\Pi} = [\mathbf{R} \quad \mathbf{t}]$$

- Projective

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

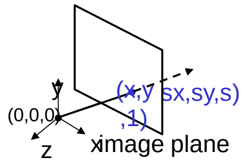
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Properties of Projection

- Preserves
  - Lines and conics
  - Incidence
  - Invariants (cross-ratio)
- Does not preserve
  - Lengths
  - Angles
  - Parallelism

Planar Projective Transformations

- Perspective projection of a plane
  - lots of names for this:
    - homography, colineation, planar projective map
  - Easily modeled using homogeneous coordinates



$$\underset{\mathbf{p'}}{\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix}} = \underset{\mathbf{H}}{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}} \underset{\mathbf{p}}{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$

To apply a homography **H**

- compute **p’** = **Hp**
- **p”** = **p’**/s normalize by dividing by third component

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

Image Formation – Two-View Geometry [Stereopsis] ➔ Fundamental Matrix

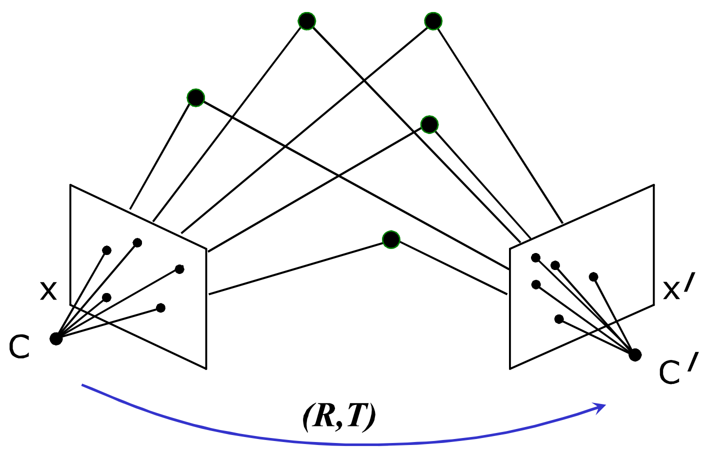
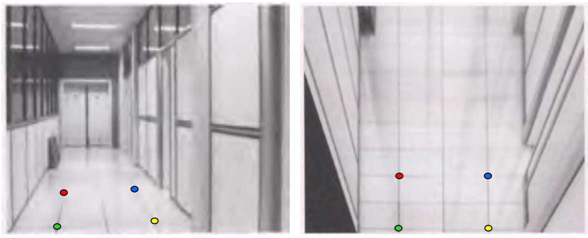


Image Rectification



To unwarp (rectify) an image

- solve for **H** given **p"** and **p**
- solve equations of the form: s**p"** = **Hp**
  - linear in unknowns: s and coefficients of **H**
  - need at least 4 points

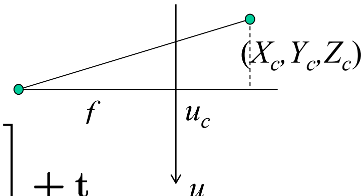
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D Projective Geometry

- These concepts generalize naturally to 3D
  - Homogeneous coordinates
    - Projective 3D points have four coords: P = (X,Y,Z,W)
  - Duality
    - A plane L is also represented by a 4-vector
    - Points and planes are dual in 3D: L P=0
  - Projective transformations
    - Represented by 4x4 matrices T: P' = TP, L' = L T-1
  - Lines are a special case…

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

3D → 2D Perspective Projection (Image Formation Equations)



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

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

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

3D → 2D Perspective Projection

- Matrix Projection (camera matrix):

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$

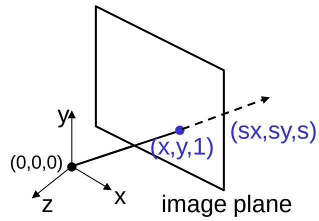
It's useful to decompose Π into $\mathbf{T} \rightarrow \mathbf{R} \rightarrow$ project $\rightarrow \mathbf{A}$

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} s_x & 0 & -t_x \\ 0 & s_y & -t_y \\ 0 & 0 & 1/f \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{orientation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{position}}$$

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

The Projective Plane

- Why do we need homogeneous coordinates?
  - Represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
  - A point in the image is a ray in projective space

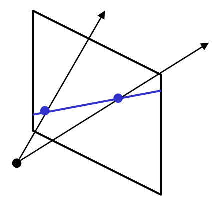


- Each *point* (x,y) on the plane is represented by a *ray* (sx,sy,s)
  - all points on the ray are equivalent: (x, y, 1) ≅ (sx, sy, s)

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

Projective Lines

- What is a line in projective space?



- A line is a *plane* of rays through origin
  - all rays (x,y,z) satisfying: ax + by + cz = 0

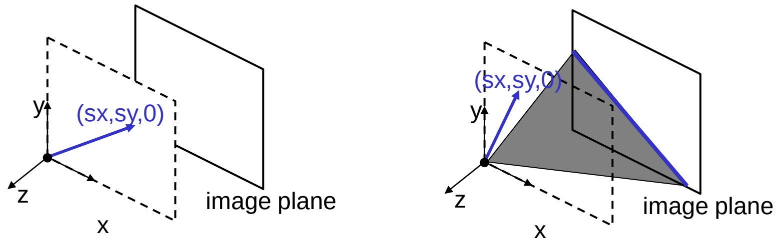
$$\text{in vector notation:}\quad 0 = \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{\mathbf{l}^T} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{p}}$$

- A line is represented as a homogeneous 3-vector **l**

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

Ideal points and lines

- Ideal point ("point at infinity")
  - p ≅ (x, y, 0) – parallel to image plane
  - It has infinite image coordinates

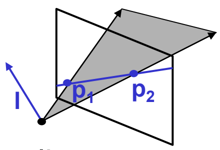


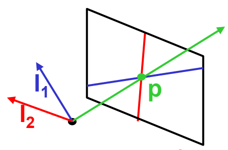
Line at infinity

- l_∞ ≅ (0, 0, 1) – parallel to image plane
- Contains all ideal points

Point and Line Duality

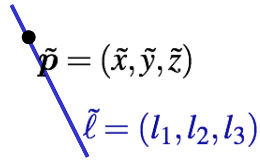
- A line l is a homogeneous 3-vector (a ray)
- It is ⊥ to every point (ray) p on the line: lT p=0





- What is the line **l** spanned by rays $\mathbf{p}_1$ and $\mathbf{p}_2$?
  - **l** is ⊥ to $\mathbf{p}_1$ and $\mathbf{p}_2$ ⇒ **l** = $\mathbf{p}_1 \times \mathbf{p}_2$ (**l** is the plane normal)
- What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?
  - **p** is ⊥ to $\mathbf{l}_1$ and $\mathbf{l}_2$ ⇒ **p** = $\mathbf{l}_1 \times \mathbf{l}_2$
- Points and lines are *dual* in projective space
  - every property of points also applies to lines

Point and Line Duality [II]



Homogeneous ⇔ Cartesian

- Point:

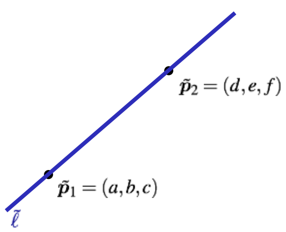
  $\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z}) \quad | \quad P = (x, y) \quad x = \frac{\tilde{x}}{\tilde{z}}, \ y = \frac{\tilde{y}}{\tilde{z}}$

- Line:
  - Is such that $\tilde{l}^T \tilde{p} = 0$
  - Point Eq of a line is: $y = mx + b$

Image/Notation from: Corke, Ch. 11

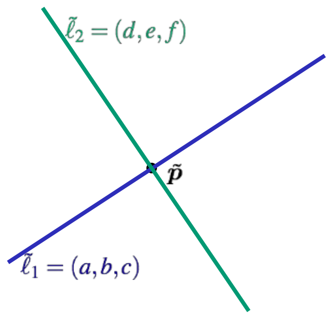
Point and Line Duality [III]

- 2 Points Make a Line



$$\tilde{\ell} = \tilde{p}_1 \times \tilde{p}_2$$

- 2 Lines Make Point!

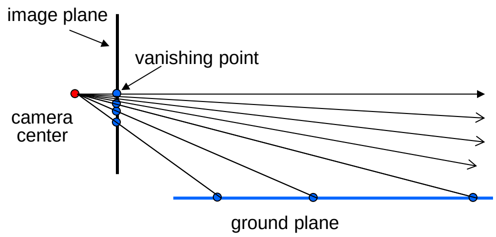


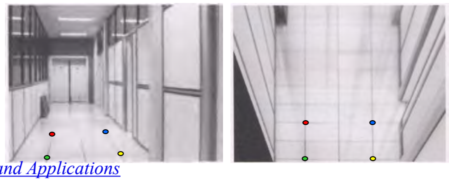
$$\tilde{p} = \tilde{\ell}_1 \times \tilde{\ell}_2$$

Image/Notation from: Corke, Ch. 11

Vanishing Points

- Vanishing point
  - projection of a point at infinity
  - whiteboard capture, architecture,…

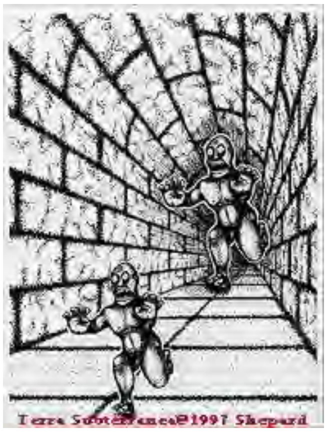




Slide from Szeliski, *Computer Vision: Algorithms and Applications*

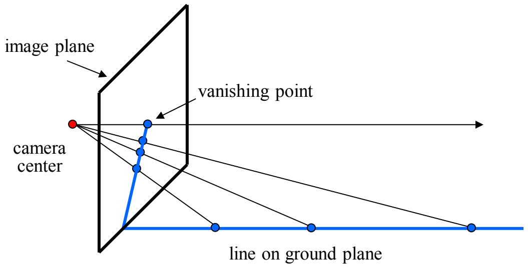
"Fundamental" Multi-View Geometry

Fun With Vanishing Points

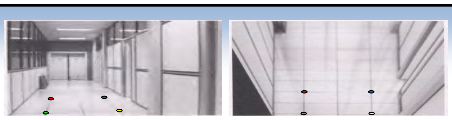


Slide from Szeliski, *Computer Vision: Algorithms and Applications*

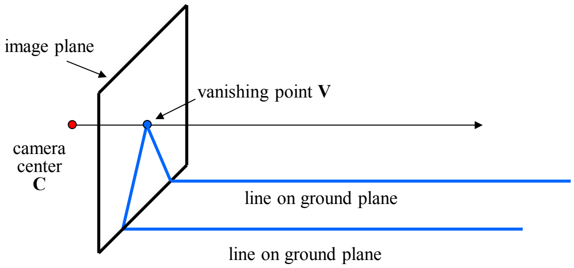
Vanishing Points (2D)



Vanishing Points

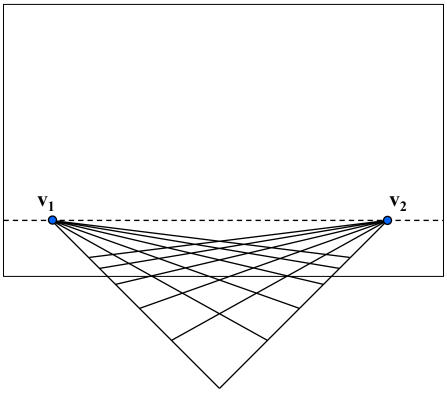


- Properties
  - Any two parallel lines have the same vanishing point
  - The ray from C through v point is parallel to the lines
  - An image may have more than one vanishing point



Vanishing Lines

- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the horizon line

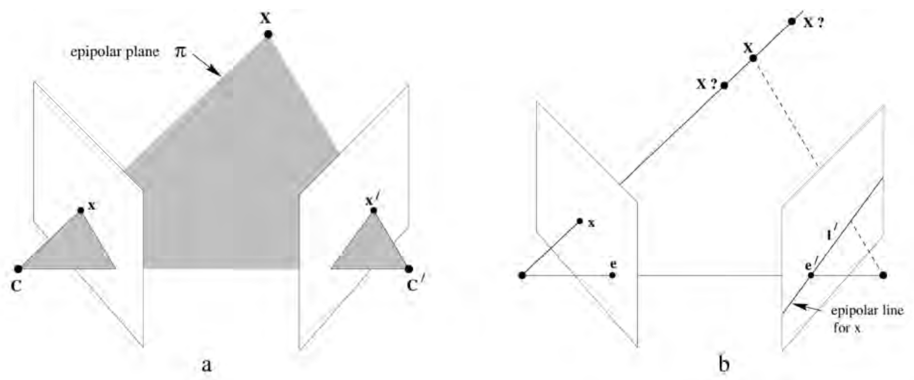


Stereo: epipolar geometry

- for two images (or images with collinear camera centers), can find epipolar lines
- epipolar lines are the projection of the pencil of planes passing through the centers
- Rectification: warping the input images (perspective transformation) so that epipolar lines are horizontal

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

Two-View Geometry: Epipolar Plane



- **Epipole:** The *point* of intersection of the line joining the camera centres (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera centre of the other view.
- **Epipolar plane** is a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes
- **Epipolar line** is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

Two-frame methods

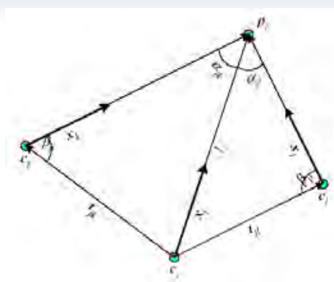
- Two main variants:
- Calibrated: "Essential matrix" E
  use ray directions (xi, xi')
- Uncalibrated: "Fundamental matrix" F

- [Hartley & Zisserman 2000]

From Szeliski, *Computer Vision: Algorithms and Applications*

Essential matrix

- Co-planarity constraint:
  - $x' \approx R\,x + t$
  - $[t]_\times\, x' \approx [t]_\times\, R\, x$
  - $x'\, [t]_\times\, x' \approx x'\, [t]_\times\, R\, x$
  - $x'\, E\, x = 0$ with $E = [t]_\times R$

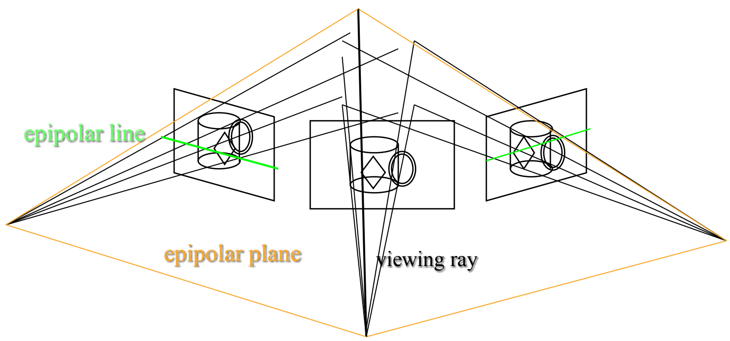


- Solve for **E** using least squares (SVD)
- t is the least singular vector of E
- R obtained from the other two s.v.s

From Szeliski, *Computer Vision: Algorithms and Applications*

Stereo: Epipolar geometry

- Match features along epipolar lines



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

Fundamental Matrix

- The fundamental matrix is the algebraic representation of epipolar geometry.

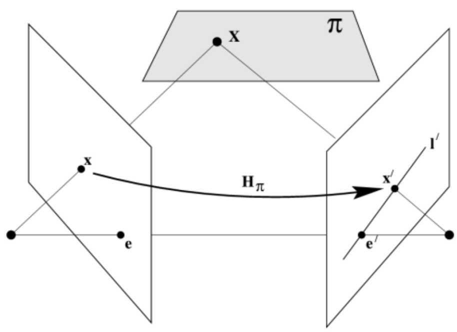


Fig. 9.5. *A point* $\mathbf{x}$ *in one image is transferred via the plane* π *to a matching point* $\mathbf{x}'$ *in the second image. The epipolar line through* $\mathbf{x}'$ *is obtained by joining* $\mathbf{x}'$ *to the epipole* $\mathbf{e}'$. *In symbols one may write* $\mathbf{x}' = \mathtt{H}_\pi \mathbf{x}$ *and* $\mathbf{l}' = [\mathbf{e}']_\times \mathbf{x}' = [\mathbf{e}']_\times \mathtt{H}_\pi \mathbf{x} = \mathtt{F}\mathbf{x}$ *where* $\mathtt{F} = [\mathbf{e}']_\times \mathtt{H}_\pi$ *is the fundamental matrix.*

Fundamental matrix

- Camera calibrations are unknown
- $x' F x = 0$ with $F = [e]\times H = K'[t]\times R K\text{-}1$
- Solve for F using least squares (SVD)
  - re-scale (xi, xi') so that |xi|≈1/2 [Hartley]
- e (epipole) is still the least singular vector of F
- H obtained from the other two s.v.s
- "plane + parallax" (projective) reconstruction
- use self-calibration to determine K [Pollefeys]

From Szeliski, *Computer Vision: Algorithms and Applications*

Fundamental Matrix Example

- Suppose the camera matrices are those of a calibrated stereo rig with the world origin at the first camera

$$P = K[I \mid \mathbf{0}] \qquad P' = K'[R \mid \mathbf{t}].$$

- Then:

$$P^{+} = \begin{bmatrix} K^{-1} \\ \mathbf{0}^{\mathsf{T}} \end{bmatrix} \qquad \mathbf{C} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$$

- Epipoles are at:

$$\mathbf{e} = P\begin{pmatrix} -R^{\mathsf{T}}\mathbf{t} \\ 1 \end{pmatrix} = KR^{\mathsf{T}}\mathbf{t} \qquad \mathbf{e}' = P'\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = K'\mathbf{t}.$$

∴

$$F = [\mathbf{e}']_{\times}K'RK^{-1} = K'^{-\mathsf{T}}[\mathbf{t}]_{\times}RK^{-1} = K'^{-\mathsf{T}}R[R^{\mathsf{T}}\mathbf{t}]_{\times}K^{-1} = K'^{-\mathsf{T}}RK^{\mathsf{T}}[\mathbf{e}]_{\times}$$

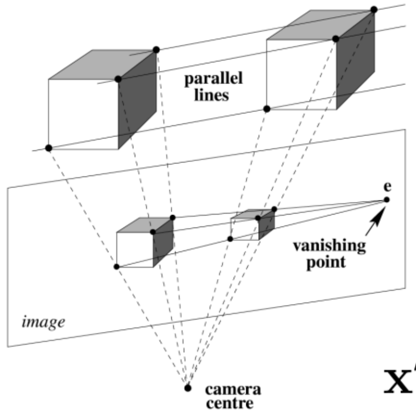
Summary of fundamental matrix properties

- F is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points, then $\mathbf{x}'^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0$.
- **Epipolar lines**:
  - ◇ $\mathbf{l}' = \mathtt{F}\mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$.
  - ◇ $\mathbf{l} = \mathtt{F}^{\mathsf{T}}\mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$.
- **Epipoles**:
  - ◇ $\mathtt{F}\mathbf{e} = \mathbf{0}$.
  - ◇ $\mathtt{F}^{\mathsf{T}}\mathbf{e}' = \mathbf{0}$.
- **Computation from camera matrices** $\mathtt{P}, \mathtt{P}'$:
  - ◇ General cameras,
    $\mathtt{F} = [\mathbf{e}']_\times \mathtt{P}'\mathtt{P}^+$, where $\mathtt{P}^+$ is the pseudo-inverse of $\mathtt{P}$, and $\mathbf{e}' = \mathtt{P}'\mathbf{C}$, with $\mathtt{P}\mathbf{C} = \mathbf{0}$.
  - ◇ Canonical cameras, $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = [\mathtt{M} \mid \mathbf{m}]$,
    $\mathtt{F} = [\mathbf{e}']_\times \mathtt{M} = \mathtt{M}^{-\mathsf{T}}[\mathbf{e}]_\times$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = \mathtt{M}^{-1}\mathbf{m}$.
  - ◇ Cameras not at infinity $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$,
    $\mathtt{F} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}^{-1} = [\mathtt{K}'\mathbf{t}]_\times \mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t}]_\times$.

Cool Robotics Share

D. Wedge, *The Fundamental Matrix Song*

Fundamental Matrix & Motion

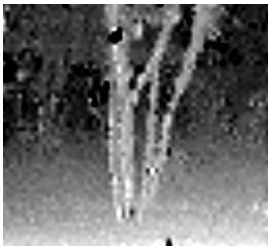


$$\mathbf{x}'^{\mathsf{T}} F \mathbf{x} = 0.$$

- Under a pure translational camera motion, 3D points appear to slide along parallel rails. The images of these parallel lines intersect in a vanishing point corresponding to the translation direction. The epipole **e** is the vanishing point.

Finding correspondences

- Apply feature matching criterion (e.g., correlation or Lucas-Kanade) at all pixels simultaneously
- Search only over epipolar lines (many fewer candidate positions)



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

Matching criteria

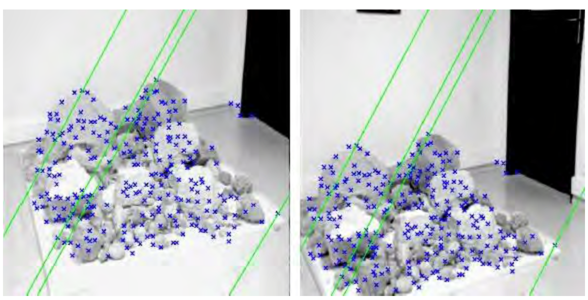
- Raw pixel values (correlation)
- Band-pass filtered images [Jones & Malik 92]
- "Corner" like features [Zhang, …]
- Edges [many people…]
- Gradients [Seitz 89; Scharstein 94]
- Rank statistics [Zabih & Woodfill 94]

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

METR 4202: **Robotics** September 21, 2016 -45

Feature-based stereo

- Match "corner" (interest) points



- Interpolate complete solution

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

METR 4202: **Robotics** September 21, 2016 -49

SFM: Structure from Motion
(& Cool Robotics Share (this week))

Structure [from] Motion

- Given a set of feature tracks,
  estimate the 3D structure and 3D (camera) motion.
- Assumption: orthographic projection
- Tracks: (u_{fp}, v_{fp}), f: frame, p: point
- Subtract out **mean** 2D position…

  $\mathbf{i}_f$: rotation, $\mathbf{s}_p$: position

$$u_{fp} = i_f^T s_p, v_{fp} = j_f^T s_p$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Structure from motion

- How many points do we need to match?
- 2 frames:
  - (R,t): 5 dof + 3n point locations ≤
  - 4n point measurements ⇒
  - n ≥ 5
- k frames:
  - 6(k–1)-1 + 3n ≤ 2kn
- always want to use many more

$$\hat{u}_{ij} = f(\boxed{\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i})$$
$$\hat{v}_{ij} = g(\boxed{\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i})$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Measurement equations

- Measurement equations

$$u_{fp} = \boldsymbol{i}_f^T \boldsymbol{s}_p \qquad \boldsymbol{i}_f\text{: rotation, } \boldsymbol{s}_p\text{: position}$$
$$v_{fp} = \boldsymbol{j}_f^T \boldsymbol{s}_p$$

- Stack them up…

$$\boldsymbol{W} = \boldsymbol{R}\,\boldsymbol{S}$$
$$\boldsymbol{R} = (\boldsymbol{i}_1,\ldots,\boldsymbol{i}_F, \boldsymbol{j}_1,\ldots,\boldsymbol{j}_F)^T$$
$$\boldsymbol{S} = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_P)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Factorization

$$\boldsymbol{W} = \boldsymbol{R}_{2F\times 3}\, \boldsymbol{S}_{3\times P}$$

SVD

$$\boldsymbol{W} = \boldsymbol{U}\,\boldsymbol{\Lambda}\,\boldsymbol{V} \quad \boldsymbol{\Lambda} \textbf{ must be rank 3}$$

$$\boldsymbol{W'} = (\boldsymbol{U}\,\boldsymbol{\Lambda}^{1/2})(\boldsymbol{\Lambda}^{1/2}\,\boldsymbol{V}) \quad = \boldsymbol{U'}\,\boldsymbol{V'}$$

Make $\boldsymbol{R}$ orthogonal

$$\boldsymbol{R} = \boldsymbol{Q}\boldsymbol{U'},\ \ \boldsymbol{S} = \boldsymbol{Q}^{-1}\boldsymbol{V'}$$

$$\boldsymbol{i}_f^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{i}_f = 1\ \ldots$$

From Szeliski, Computer Vision: Algorithms and Applications

Results

- Look at paper figures…

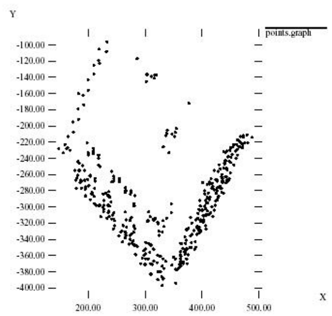


Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

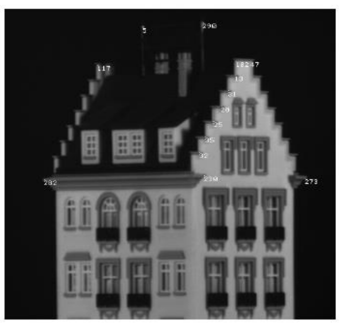


Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

From Szeliski, Computer Vision: Algorithms and Applications

Bundle Adjustment

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

$$\hat{u}_{ij} = f(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})$$
$$\hat{v}_{ij} = g(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})$$

From Szeliski, *Computer Vision: Algorithms and Applications*

More Cool Robotics Share!

Motion Planning

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 10

October 5, 2016

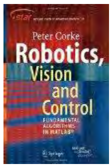
metr4202@itee.uq.edu.au

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

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	14-Sep	Robot Sensing: Feature Detection
9	21-Sep	Robot Sensing: Multiple View Geometry
	28-Sep	*Study break*
10	**5-Oct**	**Motion Planning**
11	12-Oct	Probabilistic Robotics: Localization & SLAM
12	19-Oct	Probabilistic Robotics: Planning & Control
13	26-Oct	State-Space Automation (Shaping the Dynamic Response/LQR) + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

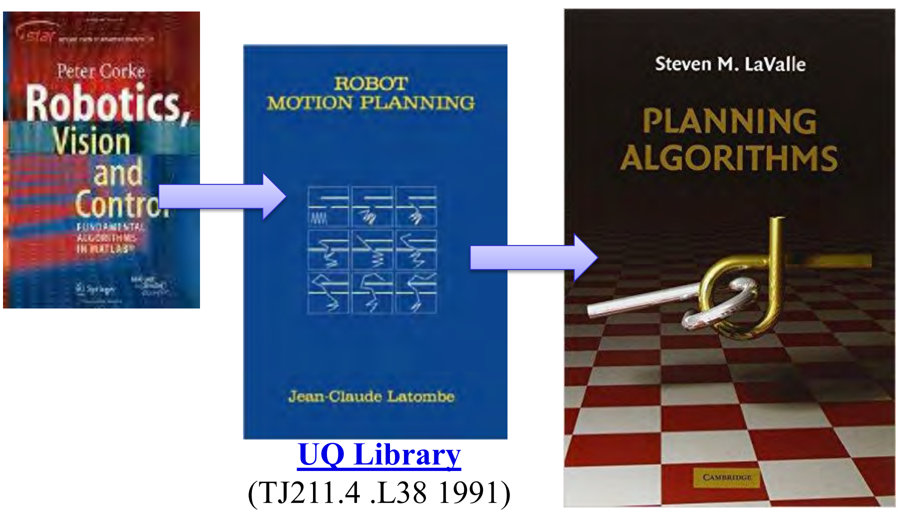
➔Planning & Control ➔

- Planning (**Global** Motion)
  - pp. 91-103
    (Yup! That’s all Peter Corke has to say! Yet there is a Chapter [15] on Visual Servoing, a **local** motion method that can’t handle obstacles).

- SLAM
  - pp. 123-4
    (§6.4-6.5)

Next Time

Reference Material



UQ Library
(TJ211.4 .L38 1991)

UQ Library / Online (PDF)

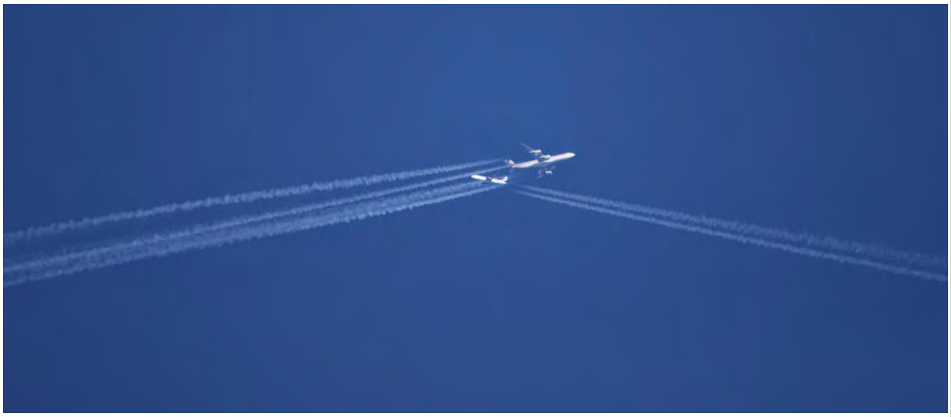
(Kinematic) Motion Planning

METR 4202: **Robotics** October 5, 2016 - 5

Motion Planning? Let's Get Moving…

METR 4202: **Robotics** October 5, 2016 - 6

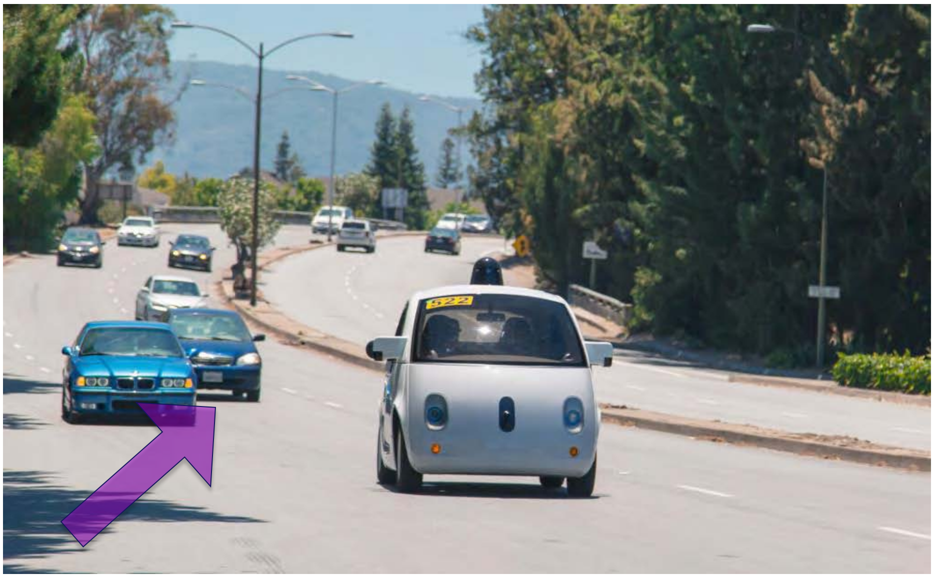
Motion Planning? Let's Get Moving?



METR 4202: **Robotics**

October 5, 2016 - 7

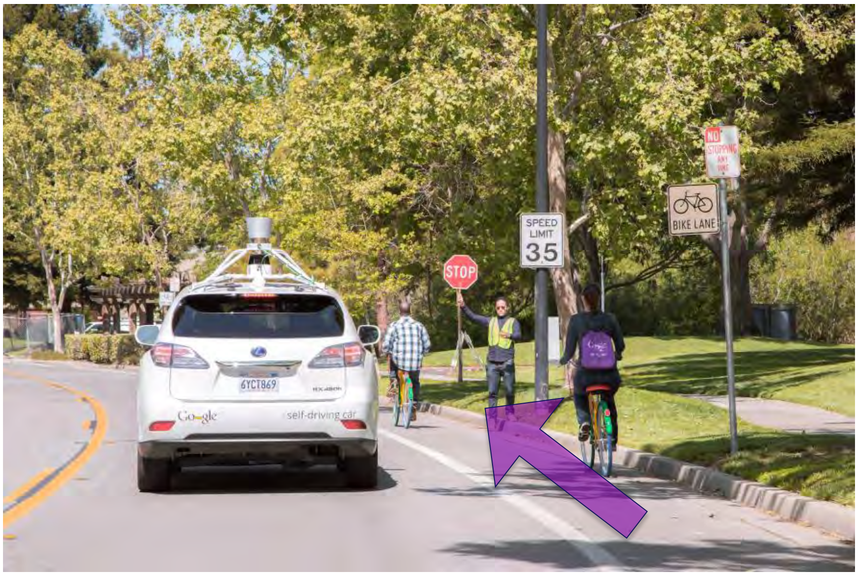
Motion Planning? The clutter can not be "ignored"



METR 4202: **Robotics**

October 5, 2016 - 8

Motion Planning: Processing the Limits

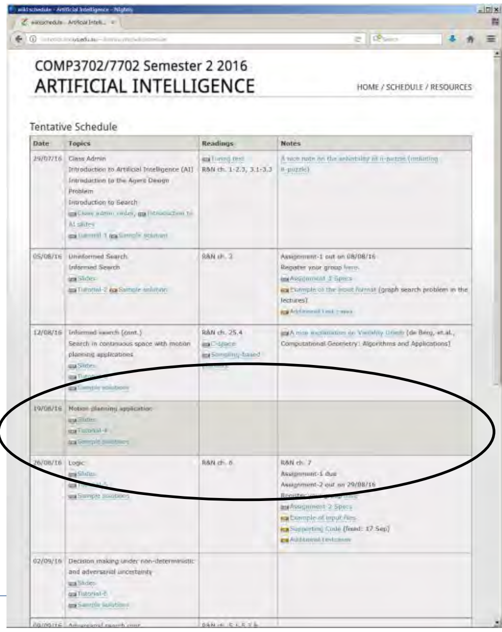


Path-Planning Approaches

- Roadmap
  Represent the connectivity of the free space by a network of 1-D curves
- Cell decomposition
  Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
- Potential field
  Define a function over the free space that has a global minimum at the goal configuration and follow its steepest descent

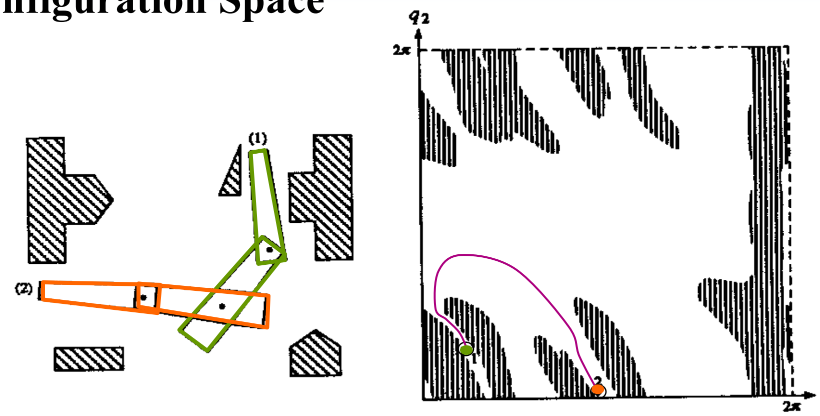
Slide from Latombe, CS326A

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



External Configuration Is Important …

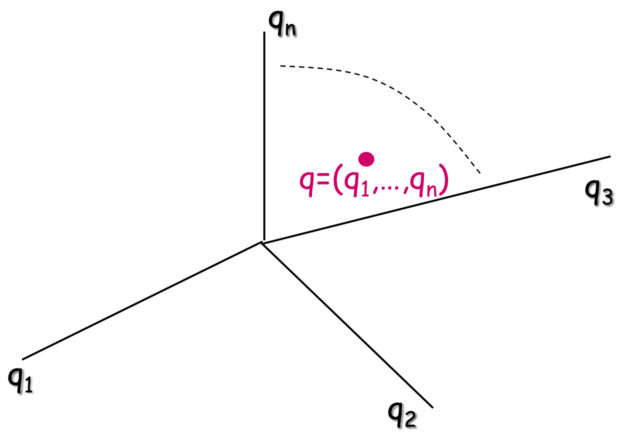
Configuration Space



- 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

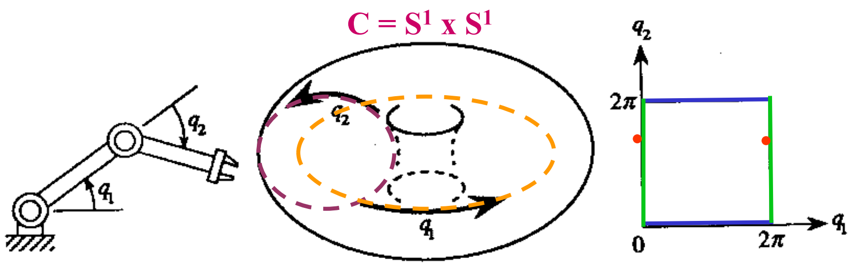
Motion Planning in C-Space



Slide from Latombe, CS326A

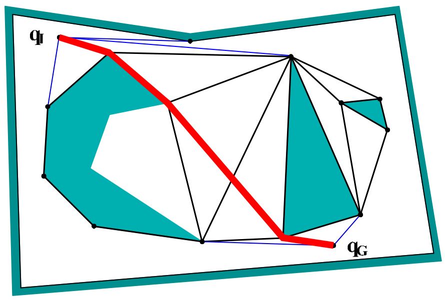
Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space



Slide from Latombe, CS326A

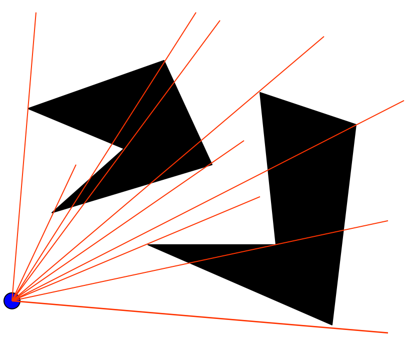
Geometric Planning Methods



- Several Geometric Methods:
  - Vertical (Trapezoidal) Cell Decomposition
  - **Roadmap Methods**
    - **Cell (Triangular) Decomposition**
    - Visibility Graphs
    - Veroni Graphs

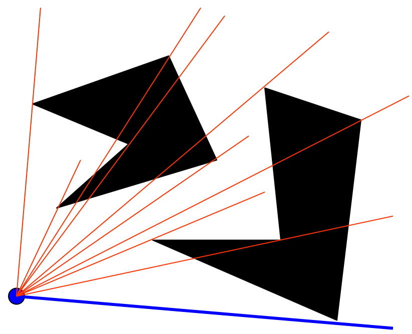
Artwork from LaValle, Ch. 6

I. Rotational Sweep



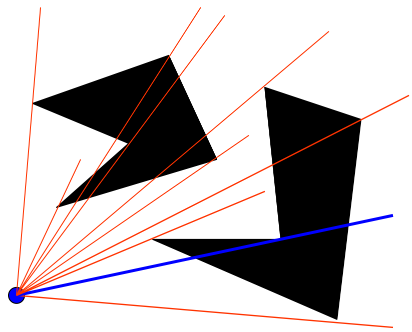
Slide from Latombe, CS326A

Rotational Sweep



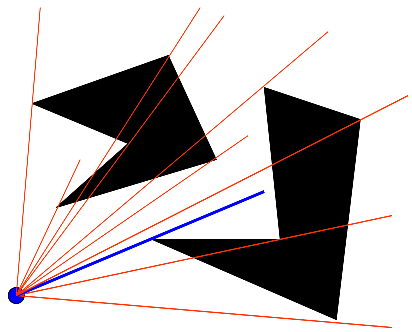
Slide from Latombe, CS326A

Rotational Sweep



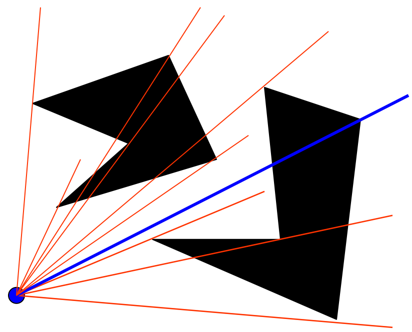
Slide from Latombe, CS326A

Rotational Sweep



Slide from Latombe, CS326A

Rotational Sweep



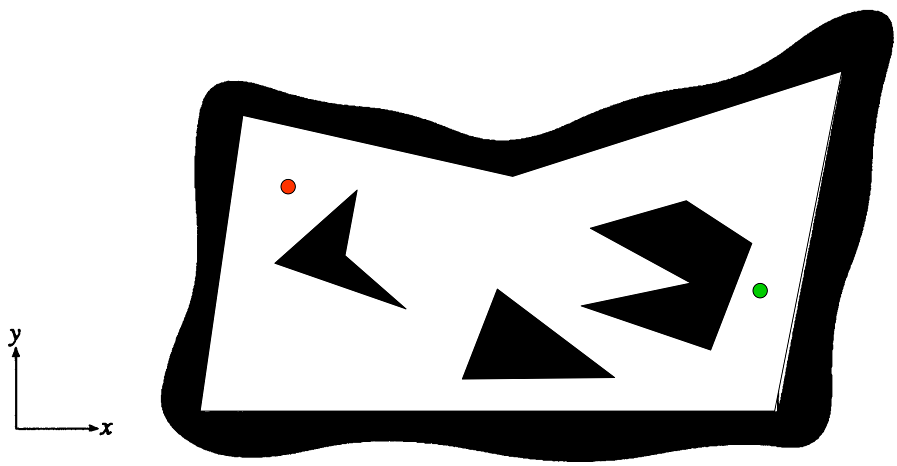
Slide from Latombe, CS326A

II. Cell-Decomposition Methods

Two classes of methods:

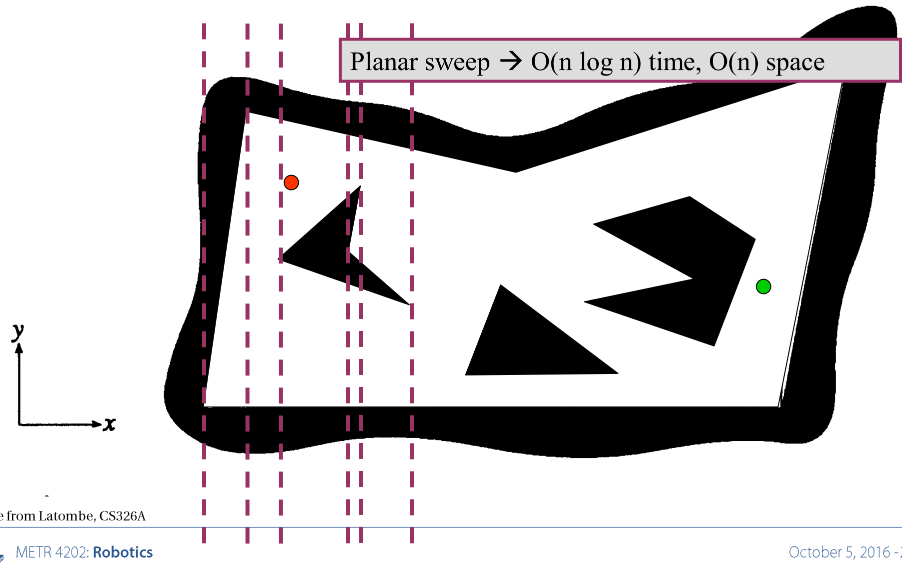
- Exact cell decomposition
  - The free space **F** is represented by a collection of non-overlapping cells whose union is exactly **F**
  - Example: trapezoidal decomposition
- Approximate cell decomposition
  - **F** is represented by a collection of non-overlapping cells whose union is contained in **F**
    Examples: quadtree, octree, 2n-tree

Trapezoidal decomposition



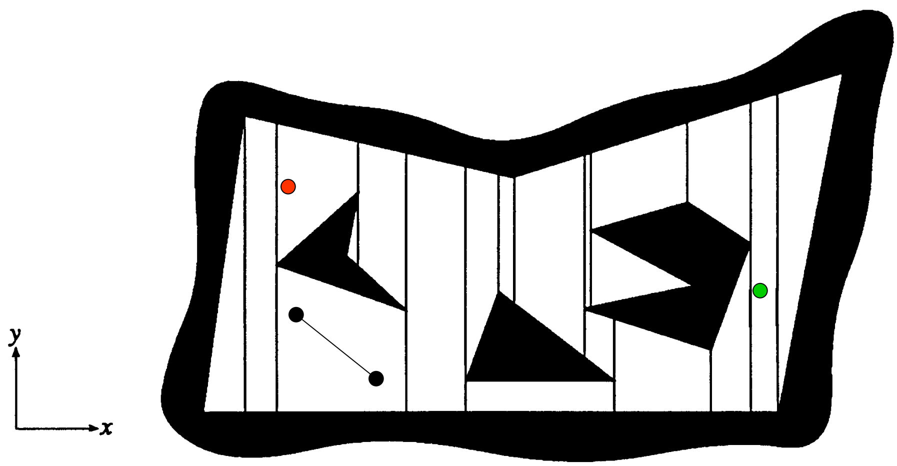
Slide from Latombe, CS326A

Trapezoidal decomposition



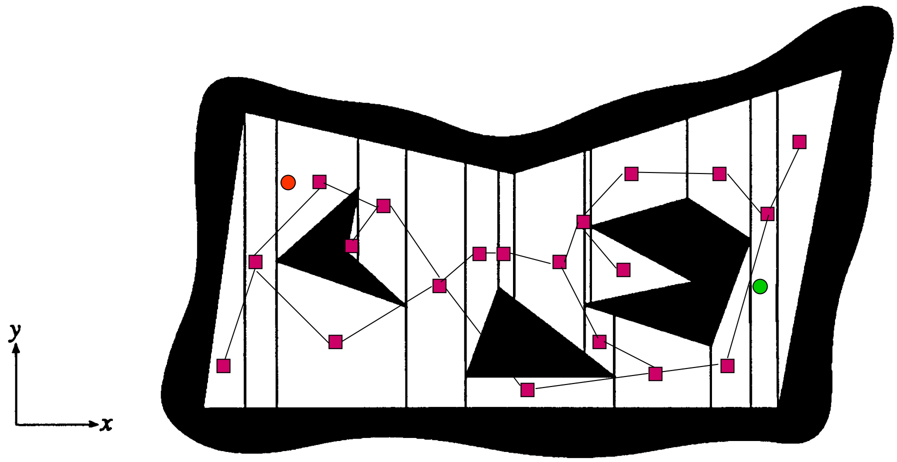
Slide from Latombe, CS326A

Trapezoidal decomposition



Slide from Latombe, CS326A

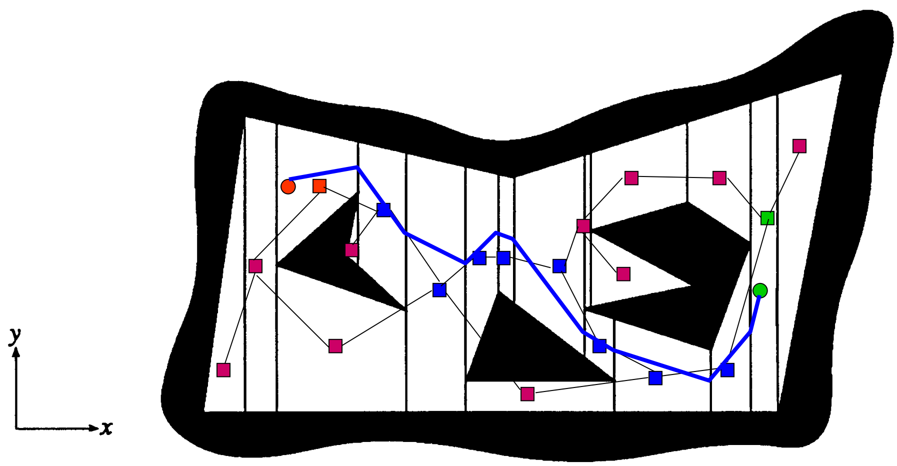
Trapezoidal decomposition



Slide from Latombe, CS326A

METR 4202: **Robotics** October 5, 2016 - 25

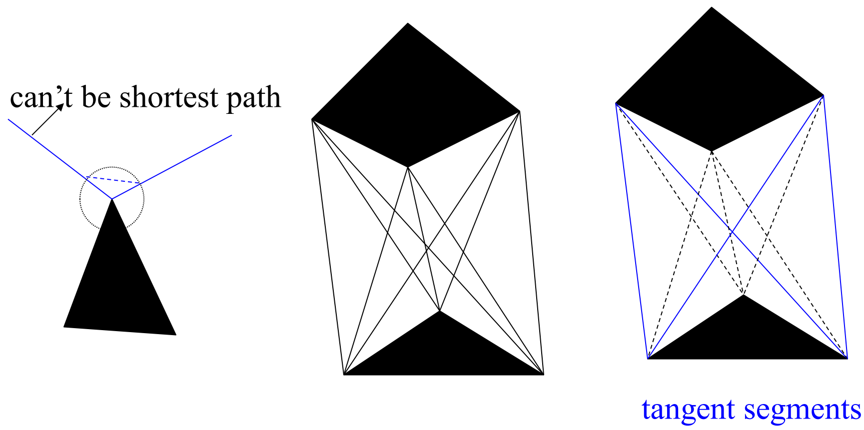
Trapezoidal decomposition



Slide from Latombe, CS326A

METR 4202: **Robotics** October 5, 2016 - 26

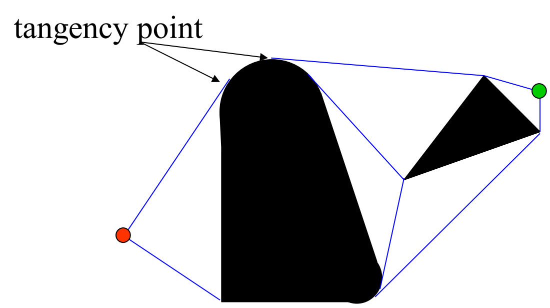
II.Visibility Graph



→ Eliminate concave obstacle vertices

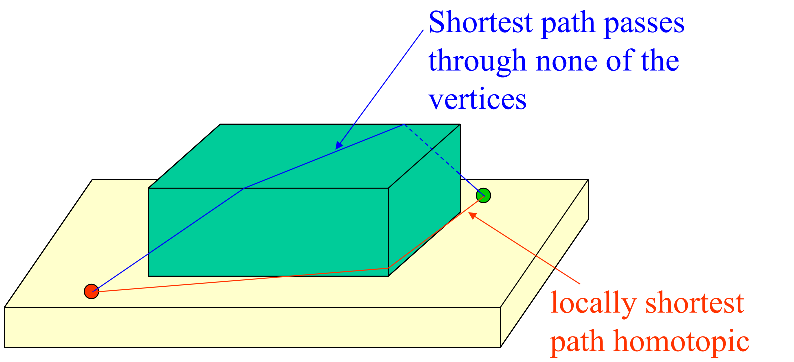
Slide from Latombe, CS326A

Generalized (Reduced) -- Visibility Graph



Slide from Latombe, CS326A

Three-Dimensional Space

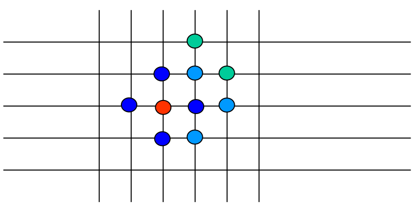


Computing the shortest collision-free path in a polyhedral space is NP-hard

Slide from Latombe, CS326A

Sketch of Grid Algorithm (with best-first search)

- Place regular grid G over space
- Search G using best-first search algorithm with potential as heuristic function



Slide from Latombe, CS326A

Simple Algorithm (for Visibility Graphs)

- Install all obstacles vertices in VG, plus the start and goal positions
- For every pair of nodes u, v in VG

    If segment(u,v) is an obstacle edge then

        insert (u,v) into VG

    else

    for every obstacle edge e

        if segment(u,v) intersects e

            then go up to segment

    insert (u,v) into VG

- Search VG using A*

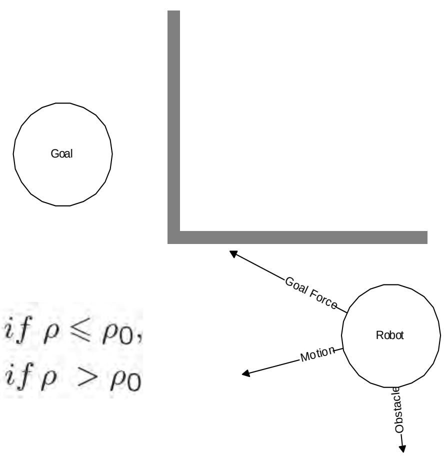
Slide based on Latombe, CS326A

III. Potential Field Methods

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

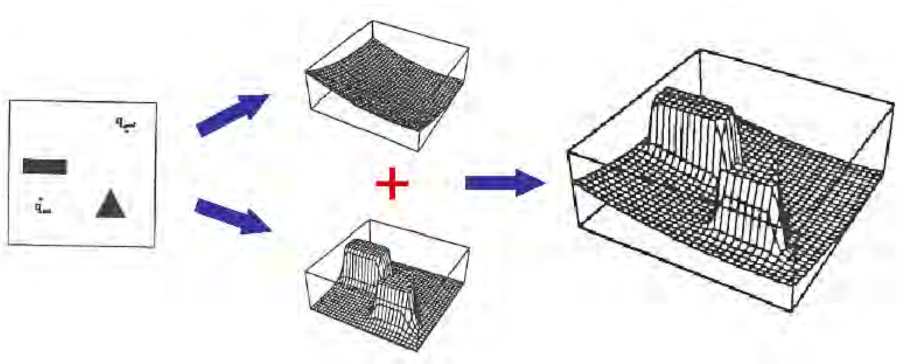
$$F_{Goal} = -k_p(x - x_{Goal})$$

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



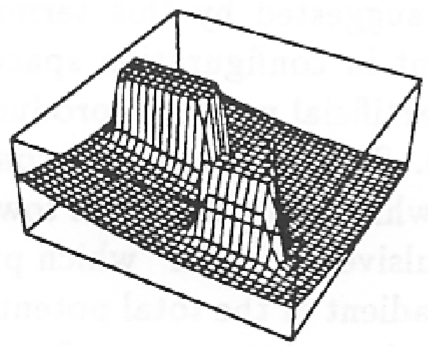
Slide based on Latombe, CS326A

Attractive and Repulsive fields



Slide from Latombe, CS326A

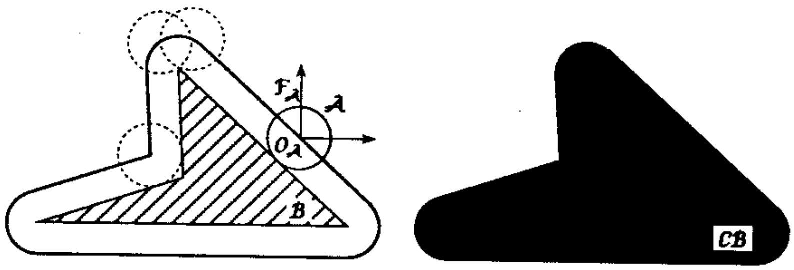
Local-Minimum Issue



- Perform best-first search (possibility of combining with approximate cell decomposition)
- Alternate descents and random walks
- Use local-minimum-free potential (navigation function)

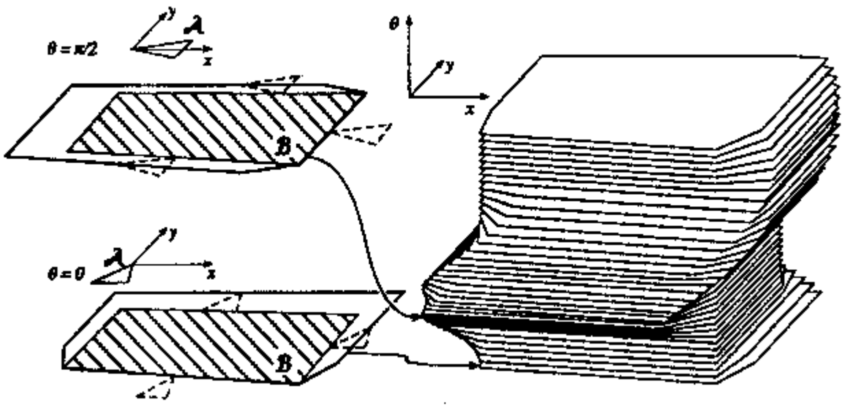
Slide from Latombe, CS326A

Disc Robot in 2-D Workspace



Slide from Latombe, CS326A

Rigid Robot Translating and Rotating in 2-D



Slide from Latombe, CS326A

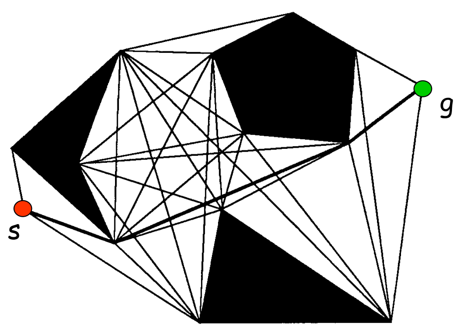
IV. Roadmap Methods

- **Visibility graph**
- **Voronoi diagram**
- Silhouette
  First complete general method that applies to spaces of any dimension and is singly exponential in # of dimensions [Canny, 87]
- **Probabilistic roadmaps (PRMS)**
  and Rapidly-exploring Randomized Trees (RRTs)

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

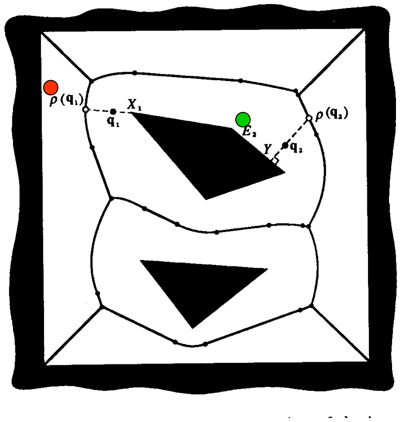


Slide from Latombe, CS326A

Roadmap Methods

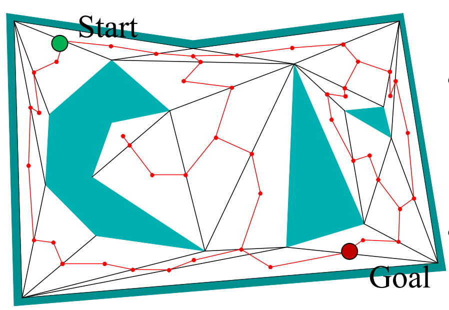
- Voronoi diagram
  Introduced by Computational Geometry researchers. Generate paths that maximizes clearance.

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



Slide from Latombe, CS326A

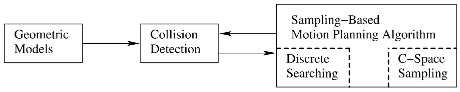
Limits of Geometric Planning Methods



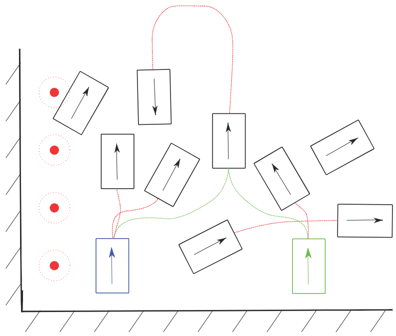
- How does this scale to high degrees of freedom?
- What about "dynamic constraints"?
- What about optimality?
- How to tie this to learning and optimization

Artwork from LaValle, Ch. 6

Sample-Based Motion Planning

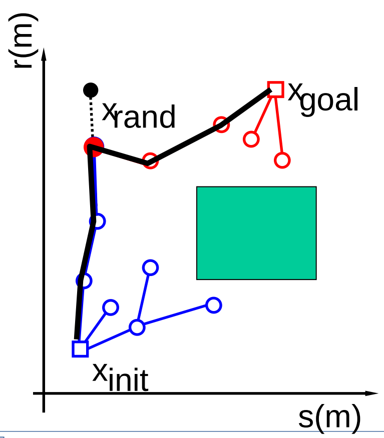


- PRMs
- RRTs

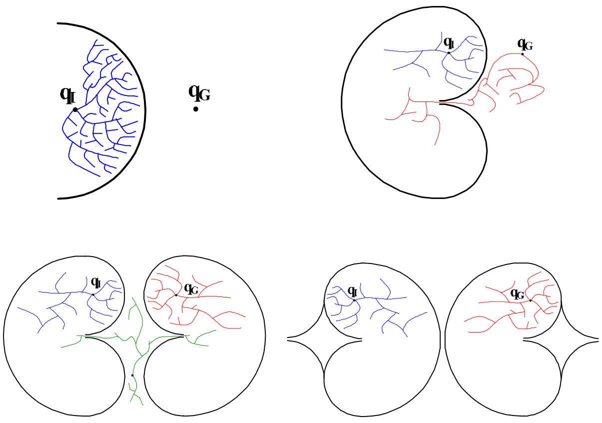


Artwork based on LaVallе, Ch. 5

Rapidly Exploring Random Trees (RRT)

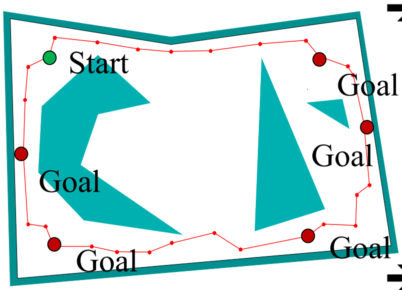


Sampling and the "Bug Trap" Problem



Artwork based on LaValle, Ch. 5

Multiple Points & Sequencing



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

➔ Travelling Salesman Problem

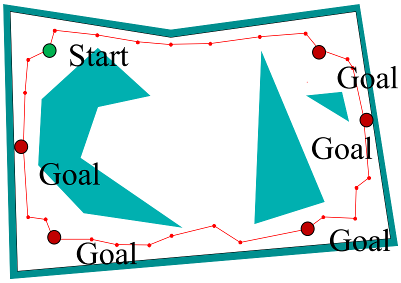
A salesman has to visit each city on a given list exactly once. In doing this, he **starts** from his home city and in the **end he has to return to his home** city. It is plausible for him to select the order in which he visits the cities so that the **total of the distances travelled** in his tour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

➔ Multi-Goal 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

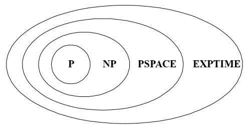
Travelling Salesman Problem



- Given a $n \times n$ distance matrix $\mathbf{C}=(c_{ij})$
- Minimize:

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

- Note that this problem is NP-Hard

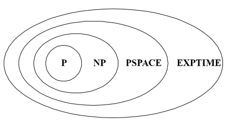


➔ BUT, Special Cases are Well-Solvable!

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) :

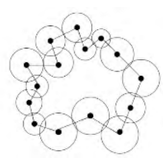
- The shortest TSP tour does not intersect itself, and thus geometry makes the problem somewhat easier.
- If all cities lie on the boundary of a convex polygon, the optimal tour is a cyclic walk along the boundary of the polygon (in clockwise or counterclockwise direction).

***The k*-line TSP**

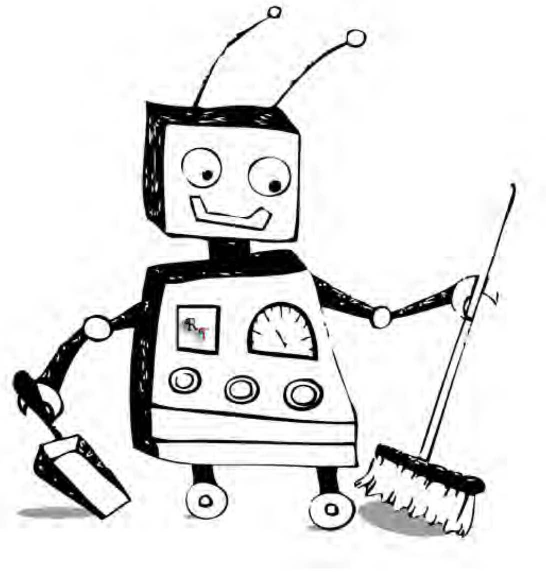
- The a special case where the cities lie on *k* parallel (or almost parallel) lines in the Euclidean plane.
- EG: Fabrication of printed circuit boards
- Solvable in **O**(n^3) time by Dynamic Programming (Rote's algorithm)

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



Cool Robotics Share



Probabilistic Robotics: Localization & SLAM

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 11

October 12, 2016

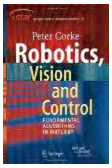
metr4202@itee.uq.edu.au

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

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	14-Sep	Robot Sensing: Feature Detection
9	21-Sep	Robot Sensing: Multiple View Geometry
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	**12-Oct**	**Probabilistic Robotics: Localization & SLAM**
12	19-Oct	Probabilistic Robotics: Planning & Control
13	26-Oct	State-Space Automation (Shaping the Dynamic Response/LQR) + Course Review

Follow Along Reading:



[Robotics, Vision & Control](#)
by [Peter Corke](#)

Also online:[SpringerLink](#)

[UQ Library eBook: 364220144X](#)

Today

➔ SLAM ⟵

- SLAM
  - pp. 123-4 (§6.4-6.5)

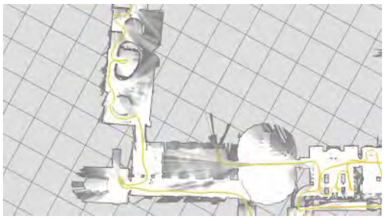
- Planning & Control
  - pp. ??

Next Time

Cool Robotics Share (It's Back!)

[Cartographer](#)

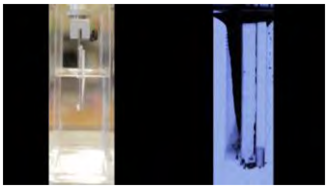
- Google Open Source SLAM



https://opensource.googleblog.com/2016/10/introducing-cartographer.html

Compliant Materials/Robotics

- Vision in ME Research



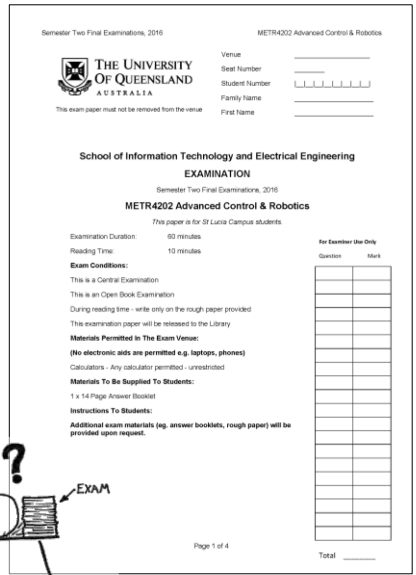
http://news.mit.edu/2016/beaver-inspired-wetsuits-surfers-1005

Final Exam!

- 4 Questions | 60 Minutes
- **Open Book**
- Similar in nature to the [2015 Quiz]

Topics:

- Position, orientation and location in space
- Robot analysis (forward/Inverse kinematics, recursive Newton-Euler formulations, etc.)
- Sensing geometry (including camera calibration)
- Multiple-view geometry
- Motion planning and control



Semester Two Final Examinations, 2016 METR4202 Advanced Control & Robotics

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

This exam paper must not be removed from the venue

Venue

Seat Number

Student Number

Family Name

First Name

School of Information Technology and Electrical Engineering

EXAMINATION

Semester Two Final Examinations, 2016

METR4202 Advanced Control & Robotics

This paper is for St Lucia Campus students.

Examination Duration: 60 minutes

Reading Time: 10 minutes

Exam Conditions:

This is a Central Examination

This is an Open Book Examination

During reading time - write only on the rough paper provided

This examination paper will be released to the Library

Materials Permitted In The Exam Venue:

(No electronic aids are permitted e.g. laptops, phones)

Calculators - Any calculator permitted - unrestricted

Materials To Be Supplied To Students:

1 x 14 Page Answer Booklet

Instructions To Students:

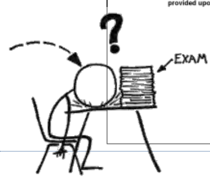
Additional exam materials (eg. answer booklets, rough paper) will be provided upon request.

For Examiner Use Only

Question	Mark

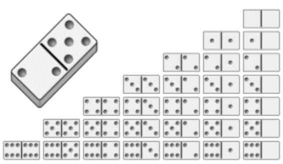
Page 1 of 4

Total



Lab 3!: Sort | Play Domino

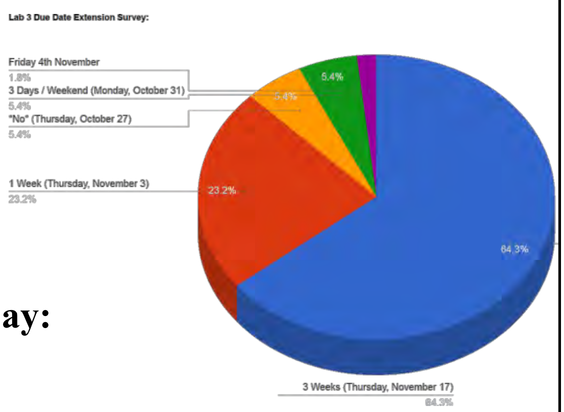
- Option 1: Sort Dominos
- Option 2: Play Dominos



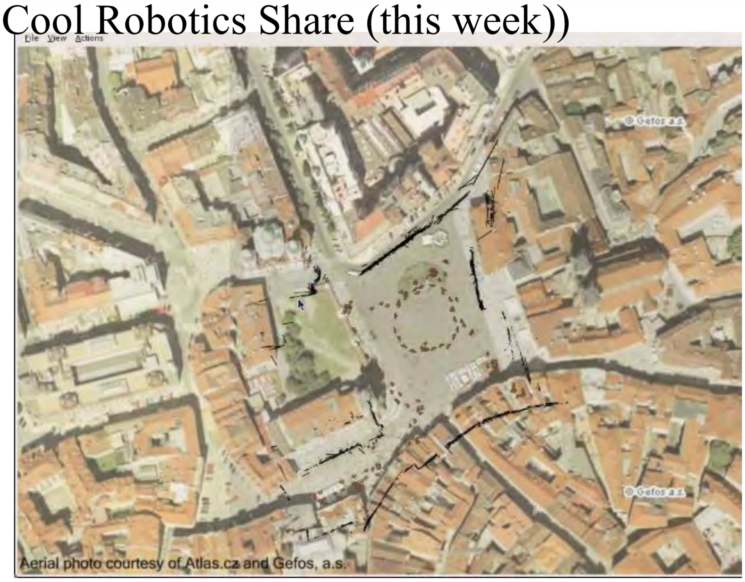
Source: https://upload.wikimedia.org/wikipedia/commons/0/04/Dominoes.jpg

Lab 3: Extension! ["Lab-ey McLabFace"]

- **Robot Grading**:
  November 3rd – 7th
- **Report:**
  November 17
- **Open-House/Demo Day:**
  November 21



SFM: Structure from Motion
(& Cool Robotics Share (this week))



Structure [from] Motion

- Given a set of feature tracks, estimate the 3D structure and 3D (camera) motion.
- Assumption: orthographic projection
- Tracks: (u_{fp},v_{fp}), f: frame, p: point
- Subtract out **mean** 2D position…

  $\mathbf{i}_f$: rotation, $\mathbf{s}_p$: position

$$u_{fp} = i_f^T s_p, v_{fp} = j_f^T s_p$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Structure from motion

- How many points do we need to match?
- 2 frames:
  - (R,t): 5 dof + 3n point locations ≤
  - 4n point measurements ⇒
  - n ≥ 5
- k frames:
  - 6(k–1)-1 + 3n ≤ 2kn
- always want to use many more

$$\hat{u}_{ij} = f(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})$$
$$\hat{v}_{ij} = g(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Measurement equations

- Measurement equations

$$u_{fp} = \boldsymbol{i}_f^T \boldsymbol{s}_p \qquad \boldsymbol{i}_f\text{: rotation, } \boldsymbol{s}_p\text{: position}$$

$$v_{fp} = \boldsymbol{j}_f^T \boldsymbol{s}_p$$

- Stack them up…

$$\boldsymbol{W} = \boldsymbol{R}\,\boldsymbol{S}$$

$$\boldsymbol{R} = (\boldsymbol{i}_1,\ldots,\boldsymbol{i}_F, \boldsymbol{j}_1,\ldots,\boldsymbol{j}_F)^T$$

$$\boldsymbol{S} = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_P)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Factorization

$$\boldsymbol{W} = \boldsymbol{R}_{2F\times3}\,\boldsymbol{S}_{3\times P}$$

SVD

$$\boldsymbol{W} = \boldsymbol{U}\,\boldsymbol{\Lambda}\,\boldsymbol{V} \quad \boldsymbol{\Lambda} \textbf{ must be rank 3}$$

$$\boldsymbol{W'} = (\boldsymbol{U}\,\boldsymbol{\Lambda}^{1/2})(\boldsymbol{\Lambda}^{1/2}\,\boldsymbol{V}) \qquad = \boldsymbol{U'}\,\boldsymbol{V'}$$

Make $\boldsymbol{R}$ orthogonal

$$\boldsymbol{R} = \boldsymbol{QU'},\ \ \boldsymbol{S} = \boldsymbol{Q}^{-1}\boldsymbol{V'}$$

$$\boldsymbol{i}_f^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{i}_f = 1 \ \ldots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Results

- Look at paper figures…

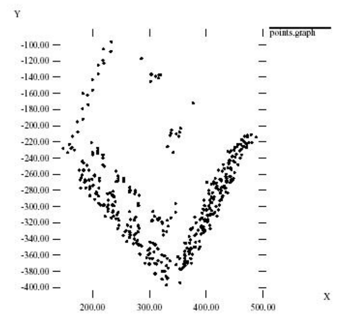


Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

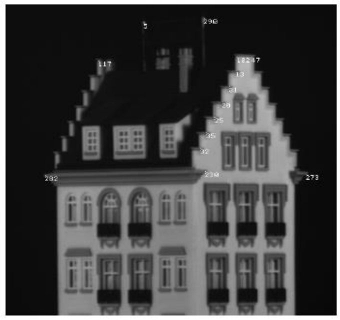


Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

From Szeliski, *Computer Vision: Algorithms and Applications*

Bundle Adjustment

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

$$\begin{aligned} \hat{u}_{ij} &= f(\boxed{\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i}) \\ \hat{v}_{ij} &= g(\boxed{\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i}) \end{aligned}$$

From Szeliski, *Computer Vision: Algorithms and Applications*

More Cool Robotics Share!

SLAM!

(Better than SMAL! ☺)

What is SLAM?

- SLAM asks the following question:

  Is it possible for an autonomous vehicle to start at an unknown location in an unknown environment and then to incrementally build a map of this environment while simultaneously using this map to compute vehicle location?

- SLAM has many indoor, outdoor, in-air and underwater applications for both manned and autonomous vehicles.

- Examples
  - Explore and return to starting point (Newman)
  - Learn trained paths to different goal locations
  - Traverse a region with complete coverage (eg, mine fields, lawns, reef monitoring)
  - …

Components of SLAM

- Localisation
  - Determine pose given a priori map
- Mapping
  - Generate map when pose is accurately known from auxiliary source.
- SLAM
  - Define some arbitrary coordinate origin
  - Generate a map from on-board sensors
  - Compute pose from this map
  - Errors in map and in pose estimate are dependent.

SLAM: 30+ Year History!

Source: Leonard (MIT) Hartley and Zisserman, Cambridge University Press, p. 437

Jenkin Building Basement, Circa 1989

Source: Leonard (MIT)

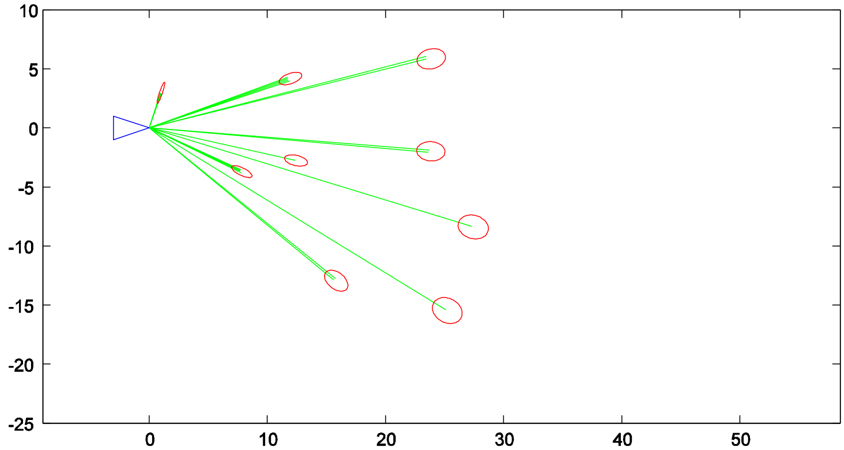
Basic SLAM Operation

METR 4202: **Robotics** October 12, 2016 - 28

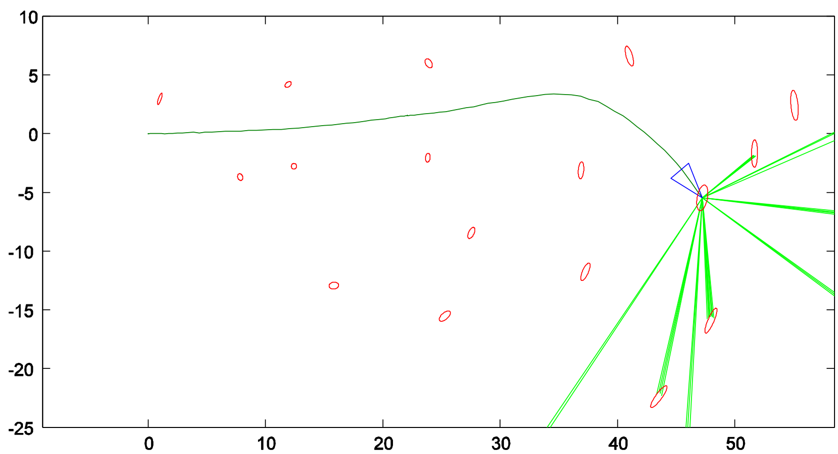
Example: SLAM in Victoria Park

METR 4202: **Robotics** October 12, 2016 - 29

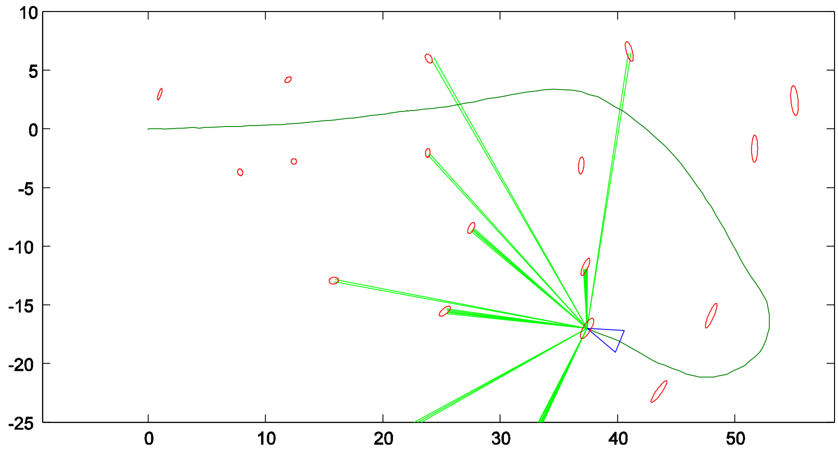
Basic SLAM Operation



Basic SLAM Operation

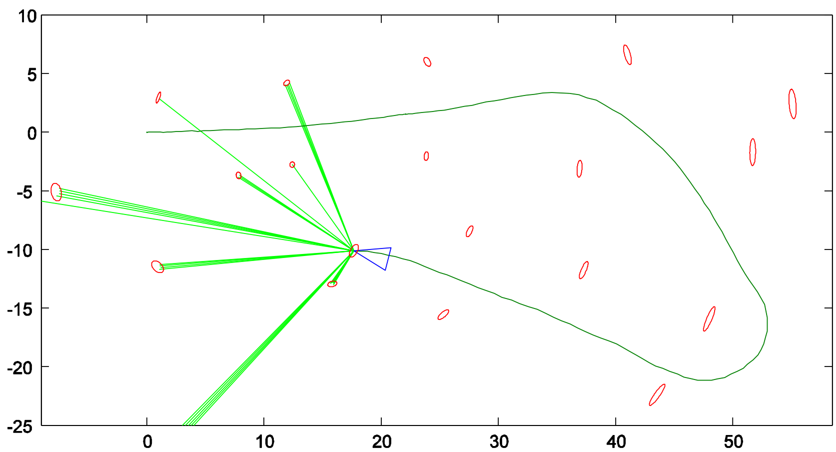


Basic SLAM Operation



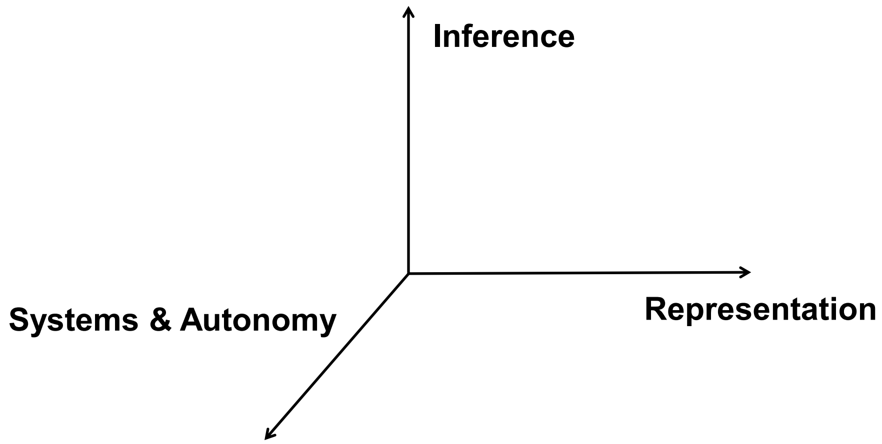
METR 4202: **Robotics** October 12, 2016 -32

Basic SLAM Operation



METR 4202: **Robotics** October 12, 2016 -33

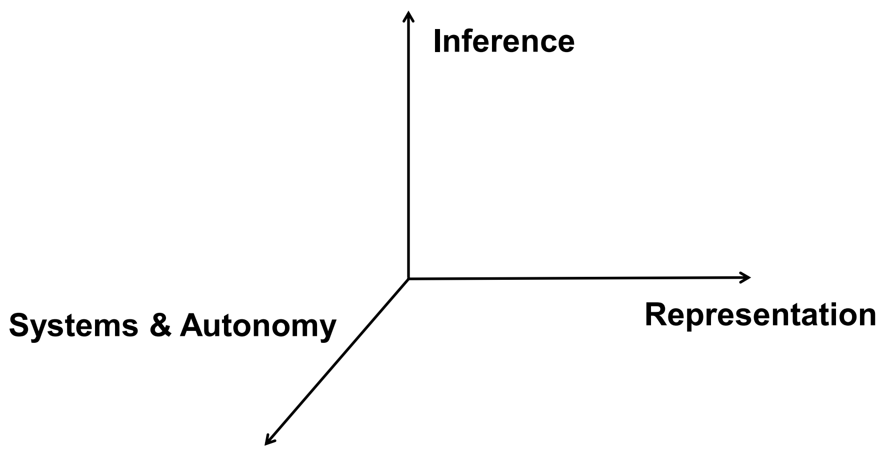
Why is SLAM Difficult?



Source: Leonard (MIT)

METR 4202: **Robotics** October 12, 2016 -34

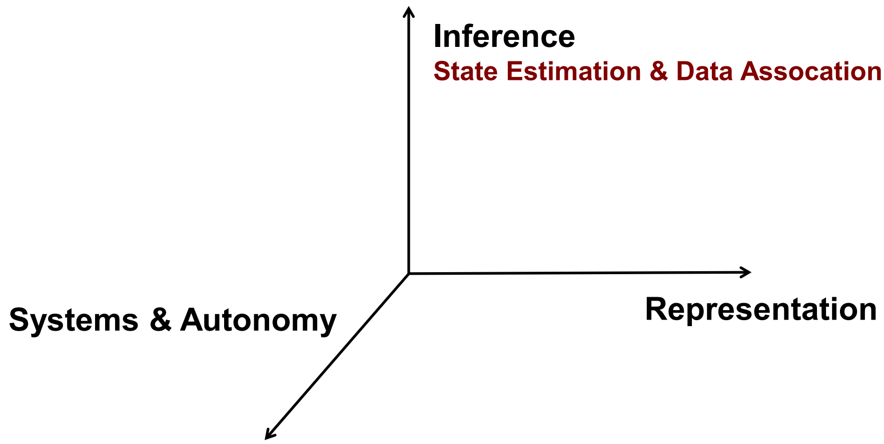
Why is SLAM Difficult?



Source: Leonard (MIT)

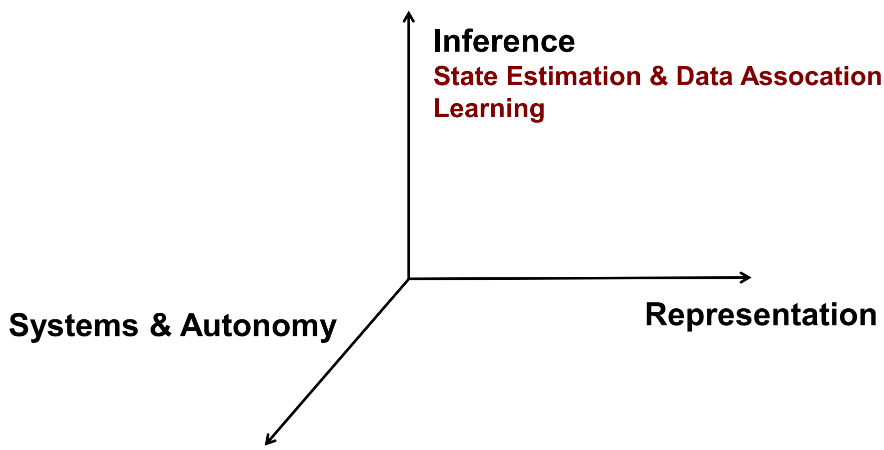
METR 4202: **Robotics** October 12, 2016 -35

Why is SLAM Difficult?



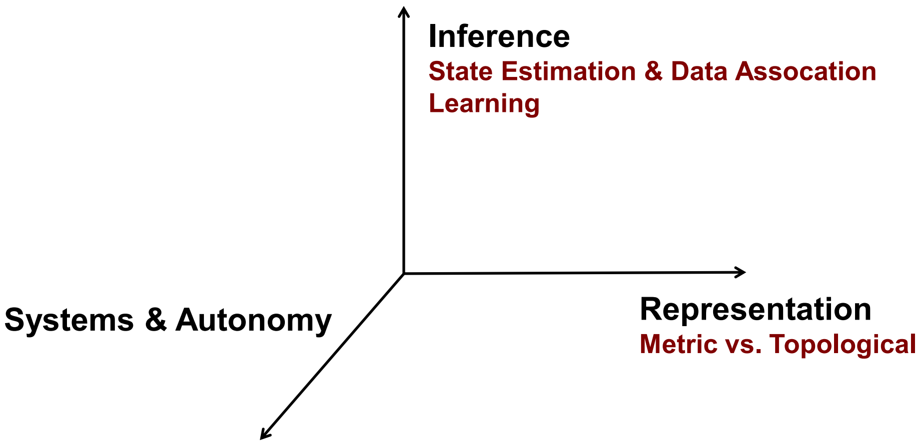
Source: Leonard (MIT)

Why is SLAM Difficult?



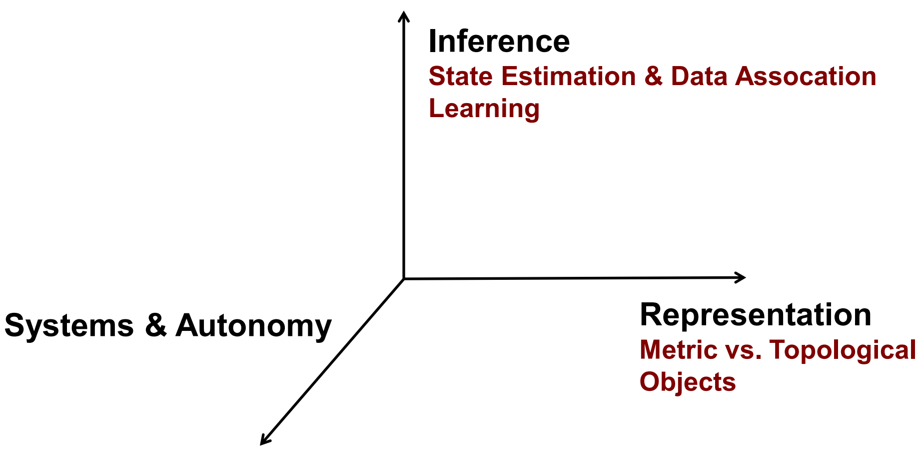
Source: Leonard (MIT)

Why is SLAM Difficult?



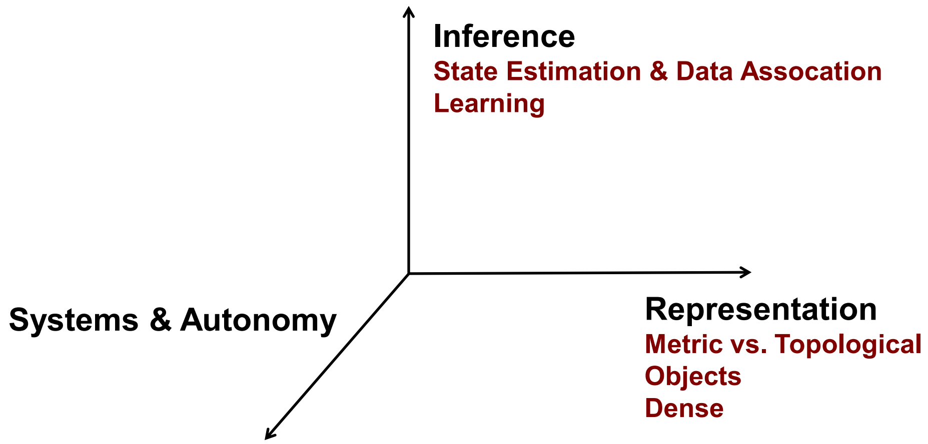
Source: Leonard (MIT)

Why is SLAM Difficult?



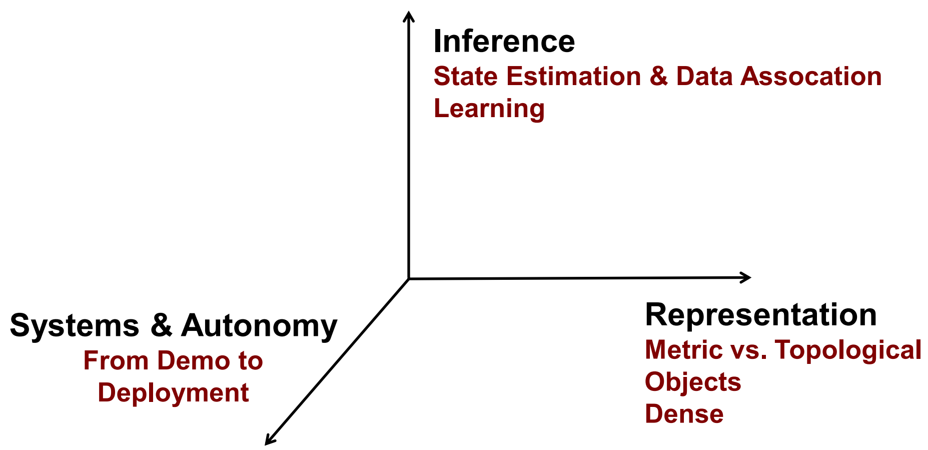
Source: Leonard (MIT)

Why is SLAM Difficult?



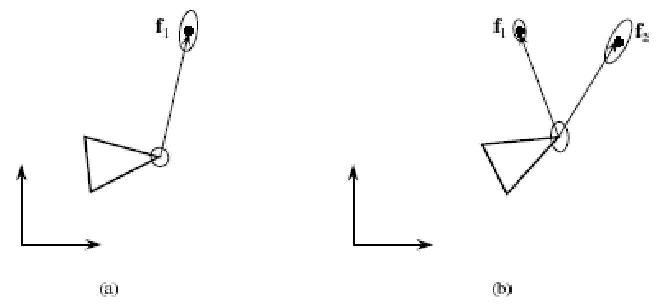
Source: Leonard (MIT)

Why is SLAM Difficult?

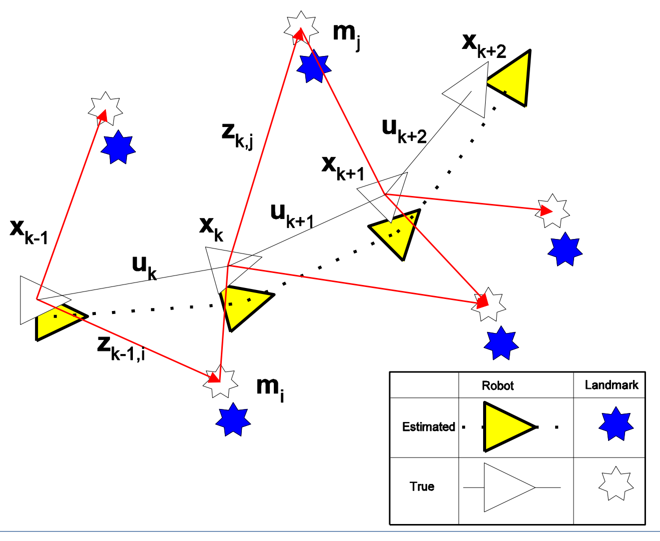


Source: Leonard (MIT)

Dependent Errors



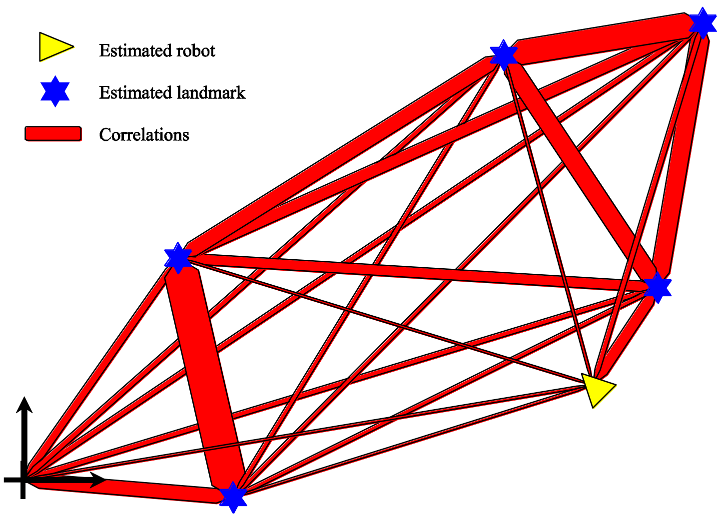
Correlated Estimates



SLAM Convergence

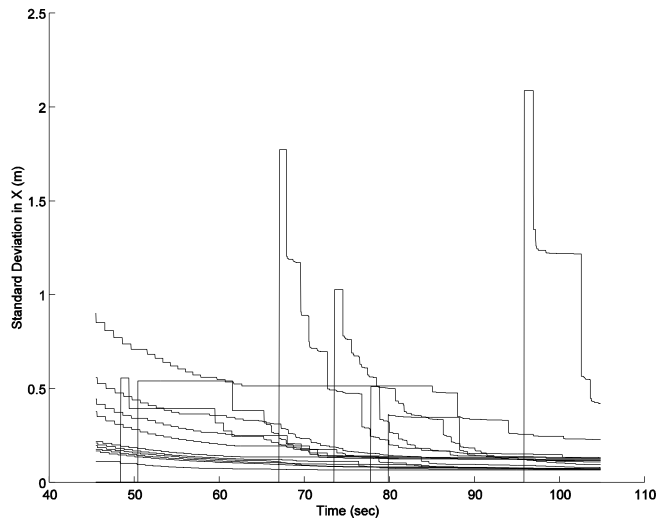
- An observation acts like a displacement to a spring system
  - Effect is greatest in a close neighbourhood
  - Effect on other landmarks diminishes with distance
  - Propagation depends on local stiffness (correlation) properties
- With each new observation the springs become increasingly (and monotonically) stiffer.
- In the limit, a rigid map of landmarks is obtained.
  - A perfect *relative* map of the environment
- The location accuracy of the robot is bounded by
  - The current quality of the map
  - The relative sensor measurement

Spring Analogy



Monotonic Convergence

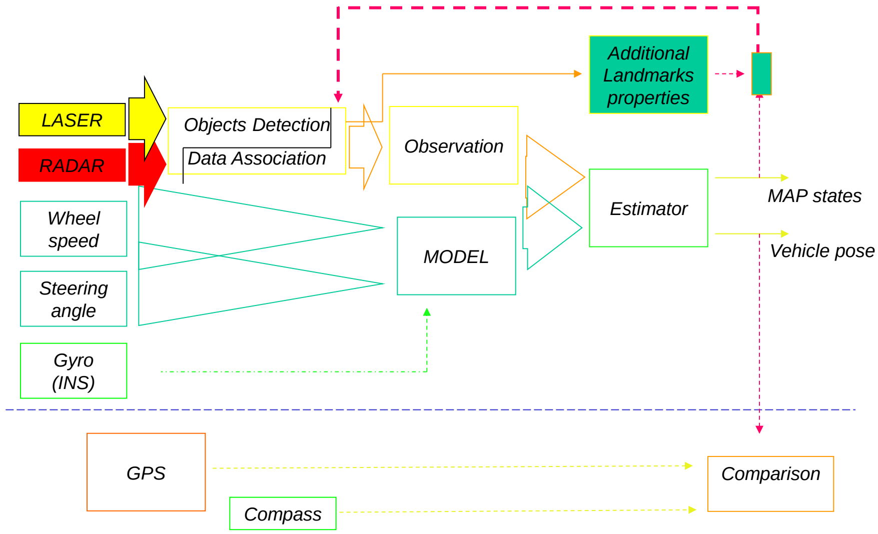
- With each new observation, the determinant decreases over the map and for any submatrix in the map.



Models

- Models are central to creating a representation of the world.
- Must have a mapping between sensed data (eg, laser, cameras, odometry) and the states of interest (eg, vehicle pose, stationary landmarks)
- Two essential model types:
  - Vehicle motion
  - Sensing of external objects

An Example System



States, Controls, Observations

Joint state with momentary pose

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{v_k} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Joint state with pose history

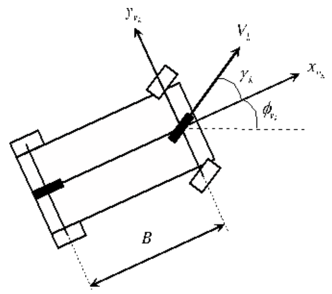
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{v_k} \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Control inputs: $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\} = \{\mathbf{U}_{0:k-1}, \mathbf{u}_k\}$

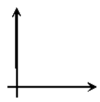
Observations: $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$

Vehicle Motion Model

- Ackerman steered vehicles: Bicycle model



- Discrete time model:



$$\mathbf{x}_{v_k} = \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right) = \begin{bmatrix} x_{v_{k-1}} + V_k \Delta T \cos(\phi_{v_{k-1}} + \gamma_k) \\ y_{v_{k-1}} + V_k \Delta T \sin(\phi_{v_{k-1}} + \gamma_k) \\ \phi_{v_{k-1}} + \frac{V_k \Delta T}{B} \sin(\gamma_k) \end{bmatrix}$$

SLAM Motion Model

$$\mathbf{x}_{v_k} = \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right) = \begin{bmatrix} x_{v_{k-1}} + V_k \Delta T \cos(\phi_{v_{k-1}} + \gamma_k) \\ y_{v_{k-1}} + V_k \Delta T \sin(\phi_{v_{k-1}} + \gamma_k) \\ \phi_{v_{k-1}} + \frac{V_k \Delta T}{B} \sin(\gamma_k) \end{bmatrix}$$

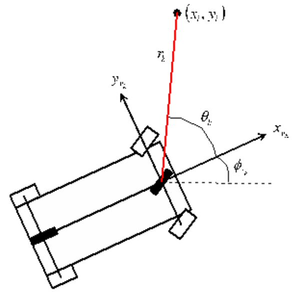
- Joint state: Landmarks are assumed stationary

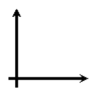
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix} \qquad \mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

Observation Model

- Range-bearing measurement

$$\mathbf{z}_{i_k} = \mathbf{h}_i(\mathbf{x}_k) = \begin{bmatrix} \sqrt{(x_i - x_{v_k})^2 + (y_i - y_{v_k})^2} \\ \arctan \frac{y_i - y_{v_k}}{x_i - x_{v_k}} - \phi_{v_k} \end{bmatrix}$$





Applying Bayes to SLAM: Available Information

- States $\mathbf{x}_k$ (Hidden or inferred values)
  - Vehicle poses
  - Map; typically composed of discrete parts called landmarks or features
- Controls $\mathbf{U}_{0:k}$
  - Velocity
  - Steering angle
- Observations $\mathbf{Z}_{0:k}$
  - Range-bearing measurements

Augmentation: Adding new poses and landmarks

- Add new pose

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{f}_v(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k) \\ \mathbf{x}_{v_{k-1}} \\ \vdots \\ \mathbf{x}_{v_0} \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

- Conditional probability is a Markov Model

$$\begin{aligned} p\left(\mathbf{x}_{v_k} | \mathbf{x}_{k-1}\right) &= \int p\left(\mathbf{x}_{v_k} | \mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= \int \delta\left(\mathbf{x}_{v_k} - \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right)\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= p\left(\mathbf{x}_{v_k} | \mathbf{x}_{v_{k-1}}\right) \end{aligned}$$

Augmentation

$$\begin{aligned} p\left(\mathbf{x}_{v_k} | \mathbf{x}_{k-1}\right) &= \int p\left(\mathbf{x}_{v_k} | \mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= \int \delta\left(\mathbf{x}_{v_k} - \mathbf{f}_v\left(\mathbf{x}_{v_{k-1}}, \mathbf{u}_k\right)\right) p\left(\mathbf{u}_k\right) d\mathbf{u}_k \\ &= p\left(\mathbf{x}_{v_k} | \mathbf{x}_{v_{k-1}}\right) \end{aligned}$$

- Product rule to create joint PDF $p(x_k)$

$$p\left(\mathbf{x}_{v_k}, \mathbf{x}_{k-1}\right) = p\left(\mathbf{x}_{v_k} | \mathbf{x}_{v_{k-1}}\right) p\left(\mathbf{x}_{v_{k-1}}, \ldots, \mathbf{x}_{v_0}, \mathbf{m}_1, \ldots, \mathbf{m}_N\right)$$

- Same method applies to adding new landmark states

Marginalisation: Removing past poses and obsolete landmarks

- Augmenting with new pose and marginalising the old pose gives the classical SLAM prediction step

$$p\left(\mathbf{x}_{v_k}, \mathbf{m}_1, \ldots, \mathbf{m}_N\right) = \int p\left(\mathbf{x}_{v_k}, \mathbf{x}_{v_{k-1}}, \mathbf{m}_1, \ldots, \mathbf{m}_N\right) d\mathbf{x}_{v_{k-1}}$$

METR 4202: **Robotics** October 12, 2016 -58

Fusion: Incorporating observation information

- Conditional PDF according to observation model

$$\begin{aligned} p\left(\mathbf{z}_{i_k} | \mathbf{x}_k\right) &= \int p\left(\mathbf{z}_{i_k} | \mathbf{x}_{v_k}, \mathbf{m}_i, \mathbf{r}_k\right) p\left(\mathbf{r}_k\right) d\mathbf{r}_k \\ &= \int \delta\left(\mathbf{z}_{i_k} - \mathbf{h}\left(\mathbf{x}_{v_k}, \mathbf{m}_i, \mathbf{r}_k\right)\right) p\left(\mathbf{r}_k\right) d\mathbf{r}_k \end{aligned}$$

- Bayes update:
  proportional to product of likelihood and prior

$$p\left(\mathbf{x}_k | \mathbf{Z}_{0:k}\right) = \frac{p\left(\mathbf{z}_{i_k} = \mathbf{z}_0 | \mathbf{x}_k\right) p\left(\mathbf{x}_k | \mathbf{Z}_{0:k-1}\right)}{p\left(\mathbf{z}_{i_k} = \mathbf{z}_0\right)}$$

METR 4202: **Robotics** October 12, 2016 -59

Implementing Probabilistic SLAM

- The problem is that Bayesian operations are intractable in general.
  - General equations are good for analytical derivations, not good for implementation
- We need approximations
  - Linearised Gaussian systems (EKF, UKF, EIF, SAM)
  - Monte Carlo sampling methods (Rao-Blackwellised particle filters)

EKF SLAM

- The complicated Bayesian equations for augmentation, marginalisation, and fusion have simple and efficient closed form solutions for linear Gaussian systems
- For non-linear systems, just linearise
  - EKF, EIF: Jacobians
  - UKF: use deterministic samples

Kalman Implementation

- So can we just plug the process and observation models into the standard EKF equations and turn the crank?

- Several additional issues:
  - Structure of the SLAM problem permits more efficient implementation than naïve EKF.
  - Data association.
  - Feature initialisation.

Structure of SLAM

- Key property of stochastic SLAM
  - Largely a *parameter* estimation problem

- Since the map is stationary
  - No process model, no process noise

- For Gaussian SLAM
  - Uncertainty in each landmark reduces monotonically after landmark initialisation
  - Map converges

- Examine computational consequences of this structure in next session.

Data Association

- Before the Update Stage we need to determine if the feature we are observing is:
  - An old feature
  - A new feature

- If there is a match with only one known feature, the Update stage is run with this feature information.

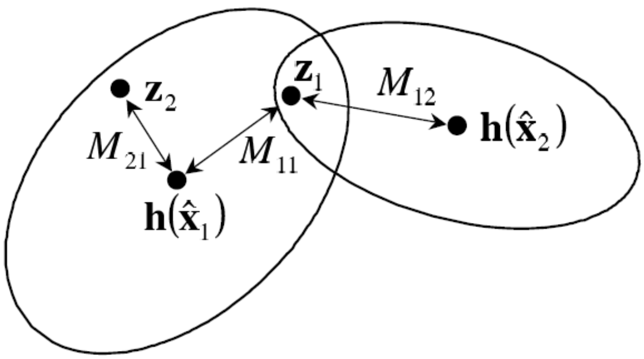
$$\mu(k) = z(k) - h(\hat{x}(k/k-1))$$

$$S(k) = \nabla h_x(k) P(k/k-1) \nabla h_x^T(k) + R$$

$$\alpha = \mu^T(k) S^{-1}(k)\ \mu(k) < \chi^2_{0.95}$$

METR 4202: **Robotics** October 12, 2016 -64

Validation Gating



METR 4202: **Robotics** October 12, 2016 -65

New Features

- If there is no match then a potential new feature has been detected

- We do not want to incorporate a spurious observation as a new feature
  - It will not be observed again and will consume computational time and memory
  - It will add clutter, increasing risk of future mis-associations
  - The features are assumed to be static. We don't not want to accept dynamic objects as features: cars, people etc.

Acceptance of New Features

- **APPROACH 1**

  - Get the feature in a list of potential features
  - Incorporate the feature once it has been observed for a number of times

  - Advantages:
    - Simple to implement
    - Appropriate for High Frequency external sensor

  - Disadvantages:
    - Loss of information
    - Potentially a problem with sensor with small field of view: a feature may only be seen very few times

Acceptance of New Features

- **APPROACH 2**
  - The state vector is extended with past vehicle positions and the estimation of the cross-correlation between current and previous vehicle states is maintained. With this approach improved data association is possible by combining data form various points
    - J. J. Leonard and R. J. Rikoski. Incorporation of delayed decision making into stochastic mapping
    - Stephan Williams, PhD Thesis, 2001, University of Sydney
  - Advantages:
    - No Loss of Information
    - Well suited to low frequency external sensors (ratio between vehicle velocity and feature rate information)
    - Absolutely necessary for some sensor modalities (eg, range-only, bearing-only)
  - Disadvantages:
    - Cost of augmenting state with past poses
    - The implementation is more complicated

Incorporation of New Features

- **We have the vehicle states and previous map**

$$P_0 = \begin{bmatrix} P^0_{v,v} & P^0_{v,m} \\ P^0_{m,v} & P^0_{m,m} \end{bmatrix}$$

We observed a new feature and the covariance and cross-covariance terms need to be evaluated

$$P_1 = \begin{bmatrix} P^0_{v,v} & P^0_{v,m} & ? \\ P^0_{m,v} & P^0_{m,m} & ? \\ ? & ? & ? \end{bmatrix}$$

Incorporation of New Features

- Approach 1

$$P_0 = \begin{bmatrix} P_{vv}^0 & P_{vm}^0 & 0 \\ P_{mv}^0 & P_{mm}^0 & 0 \\ 0 & 0 & A \end{bmatrix}$$

With A very large

$$W(k) = P(k/k-1)\nabla h_x^T(k) S^{-1}(k)$$
$$S(k) = \nabla h_x(k) P(k/k-1)\nabla h_x^T(k) + R$$
$$P(k/k) = P(k/k-1) - W(k)S(k)W^T(k)$$

$$P_1 = \begin{bmatrix} P_{vv}^1 & P_{vm}^1 & P_{vn}^1 \\ P_{mv}^1 & P_{mm}^1 & P_{mn}^1 \\ P_{nv}^1 & P_{nm}^1 & P_{nn}^1 \end{bmatrix}$$

- **Easy to understand and implement**
- **Very large values of A may introduce numerical problems**

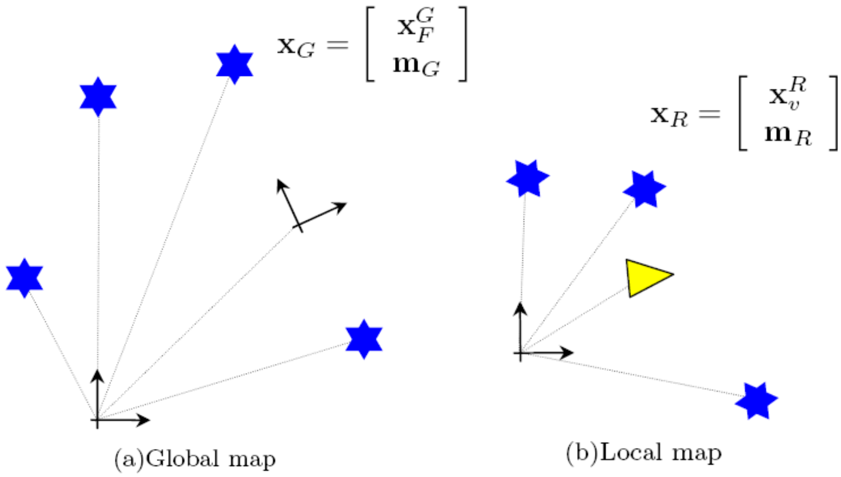
Analytical Approach

$$P_0 = \begin{bmatrix} P_{v,v}^0 & P_{v,m}^0 \\ P_{m,v}^0 & P_{m,m}^0 \end{bmatrix}$$

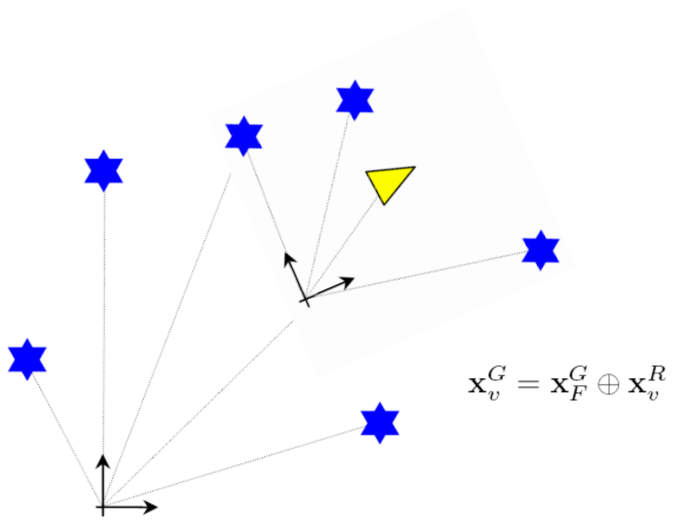
- **We can also evaluate the analytical expressions of the new terms**

$$P_1 = \begin{bmatrix} P_{v,v}^0 & P_{v,m}^0 & ? \\ P_{m,v}^0 & P_{m,m}^0 & ? \\ ? & ? & ? \end{bmatrix}$$

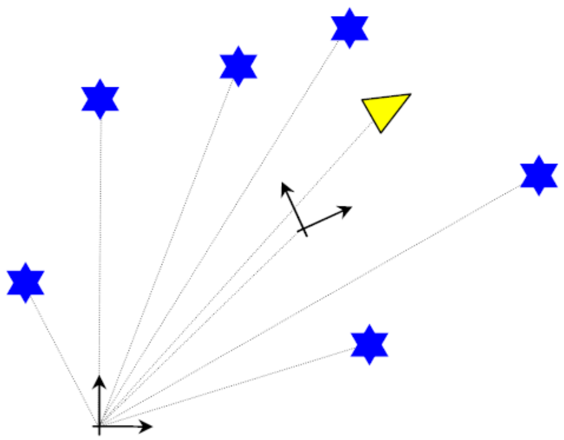
Constrained Local Submap Filter



CLSF Registration



CLSF Global Estimate



Probabilistic Robotics: Planning & Control

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 12

October 19, 2016

metr4202@itee.uq.edu.au

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

Schedule of Events

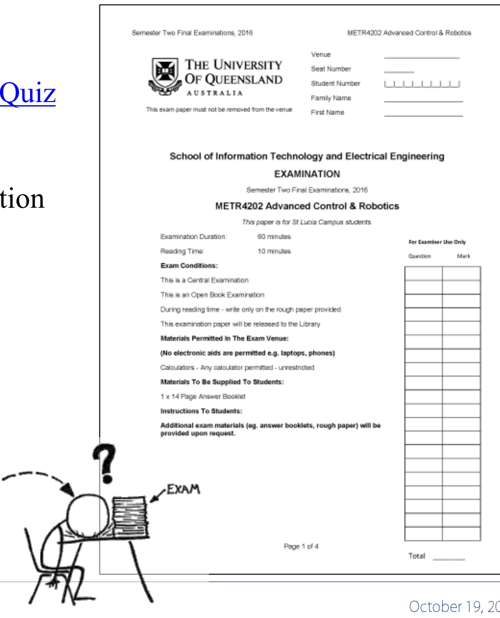
Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	14-Sep	Robot Sensing: Feature Detection
9	21-Sep	Robot Sensing: Multiple View Geometry
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	Probabilistic Robotics: Localization & SLAM
12	**19-Oct**	**Probabilistic Robotics: Planning & Control (State-Space/Shaping the Dynamic Response/LQR)**
13	26-Oct	The Future of Robotics/Automation + Challenges + Course Review

Final Exam!

- 4 Questions | 60 Minutes
- **Open Book**
- Similar in nature to the [2015 Quiz]

Topics:

- Position, orientation and location in space
- Robot analysis (forward/Inverse kinematics, recursive Newton-Euler formulations, etc.)
- Sensing geometry (including camera calibration)
- Multiple-view geometry
- Motion planning and control



Semester Two Final Examinations, 2016 METR4202 Advanced Control & Robotics

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

This exam paper must not be removed from the venue

Venue ____

Seat Number ____

Student Number ____

Family Name ____

First Name ____

School of Information Technology and Electrical Engineering

EXAMINATION

Semester Two Final Examinations, 2016

METR4202 Advanced Control & Robotics

This paper is for St Lucia Campus students.

Examination Duration: 60 minutes

Reading Time: 10 minutes

Exam Conditions:

This is a Central Examination

This is an Open Book Examination

During reading time - write only on the rough paper provided

This examination paper will be released to the Library

Materials Permitted In The Exam Venue:

(No electronic aids are permitted e.g. laptops, phones)

Calculators - Any calculator permitted - unrestricted

Materials To Be Supplied To Students:

1 x 14 Page Answer Booklet

Instructions To Students:

Additional exam materials (eg. answer booklets, rough paper) will be provided upon request.

For Examiner Use Only

Question	Mark

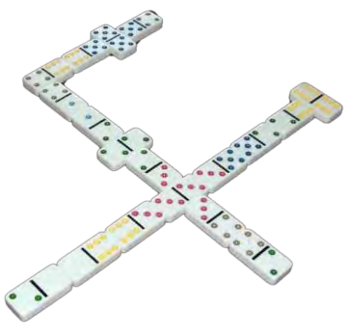
Total ____

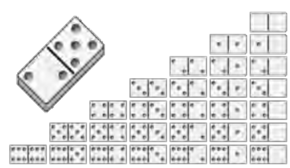
Page 1 of 4

Lab 3: Robotics of a Domino Sort

I. **Playing Dominoes** of a Sort

- Option 2: Play Dominos





Source https://en.wikipedia.org/wiki/Muggins#/media/File:Muggins.jpg

Cool Robotics Share

White House AI Symposium

- Report on the opportunities, considerations, challenges of AI

(Oct 12, 2016)

- Related Event (similar thread): **ETHZ Cybathlon**

Source: https://www.whitehouse.gov/blog/2016/10/12/administrations-report-future-artificial-intelligence | http://www.nbcnews.com/health/health-news/brain-chip-helps-paralyzed-man-feel-his-fingers-n665881

Planning & Control Robotics

- **Obstacles** + **Dynamics**

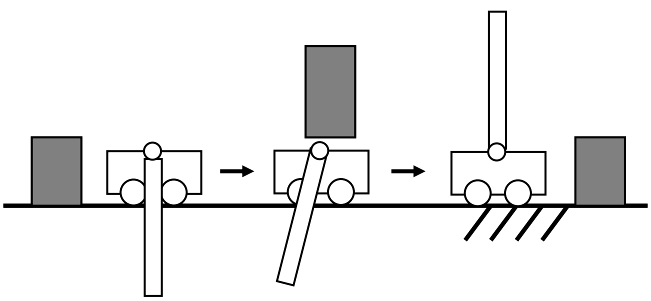
- Partial collisions allowed!

(Oct 14, 2016)

Source: (top) https://youtu.be/voN9CCmzxYk | (bottom) https://youtu.be/LS8sa42xevA

Integrated Planning and Control Methods…

- A motivating problem (for agility)
  - Cart and pole in a cluttered workspace …



Outline

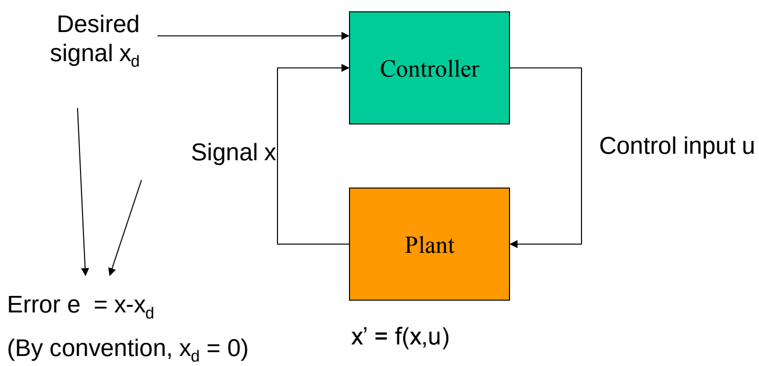
1. Guest Presentation:
   Josh Song, CHARM Planning
2. Probabilistic Robotics
3. Sample-Based Planning
4. Control (State-Space | Shaping Response | LQR)
5. Integrated Planning & Control

Control

(Feedback is our friend!)

Control Theory

- The use of feedback to regulate a signal

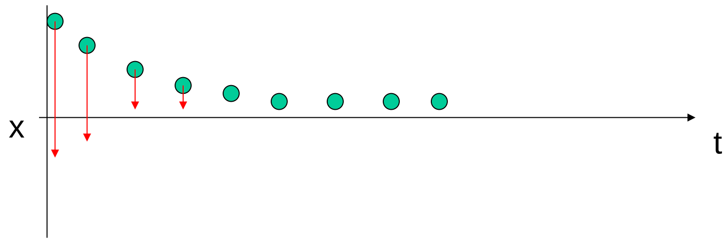


Model-free vs model-based

- Two general philosophies:
  - Model-free: do not require a dynamics model to be provided
  - Model-based: do use a dynamics model during computation
- Model-free methods:
  - Simpler (eg. **PID**)
  - Tend to require much more manual tuning to perform well
- Model-based methods:
  - Can achieve good performance (optimal w.r.t. some cost function)
  - Are more complicated to implement
  - Require reasonably good models (system-specific knowledge)
  - Calibration: build a model using measurements before behaving
  - Adaptive control: “learn” parameters of the model online from sensors

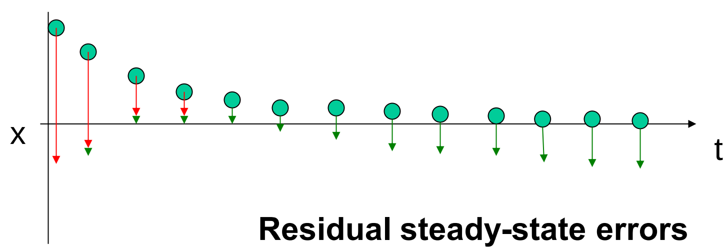
PID control

- Proportional-Integral-Derivative controller
  - A workhorse of 1D control systems
  - Model-free
- Proportional Case:
  - $u(t) \overbrace{=}^{\text{Gain}} -K_p\, x(t)$
  - Negative sign assumes control acts in the same direction as x



PID control: Integral term

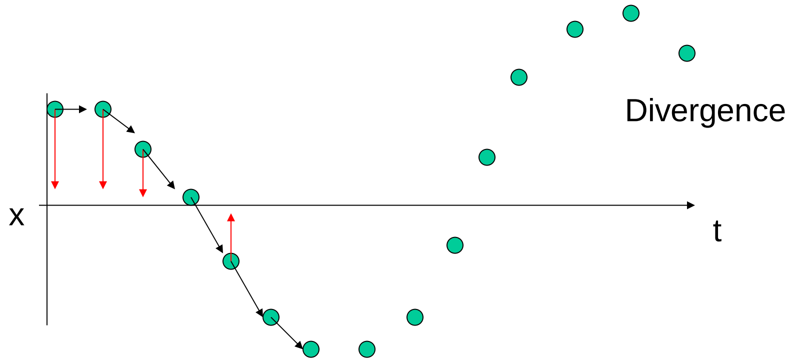
- $u(t) = -K_p\, x(t) - \overbrace{K_i}^{\text{Integral gain}} I(t)$
- $I(t) = \int_0^t x(t)dt$ (accumulation of errors)



Residual steady-state errors driven asymptotically to 0

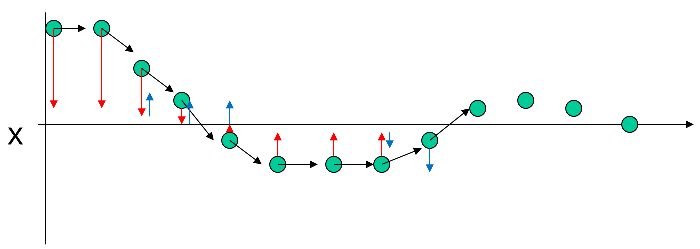
PID control: Integral term: Instability

- I adds a pole
- If not tuned correctly → this adds instability
- Ex: For a 2nd order system (momentum), P control



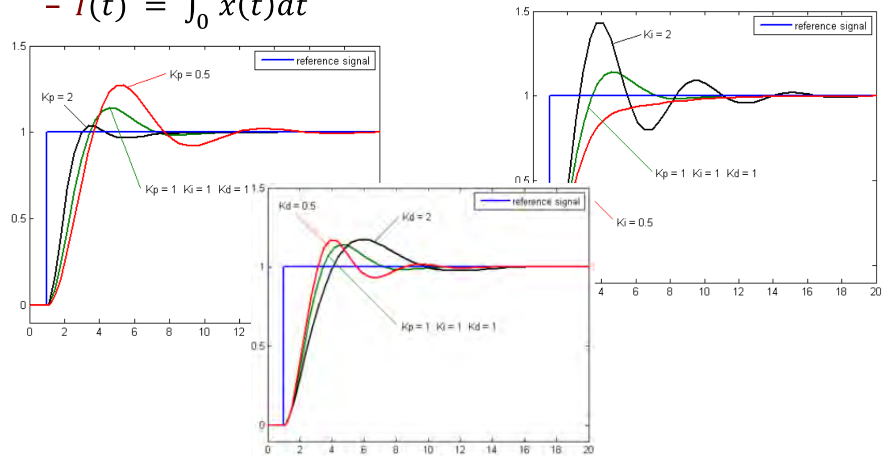
PID control: Derivative term

- $u(t) = -K_p\, x(t) - \overbrace{K_d}^{\text{Derivative gain}}\, x'(t)$



PID control: Together

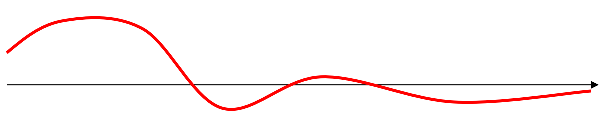
- P+I+D:
  - $u(t) = -K_p\, x(t) - K_i\, I(t) - K_d\, x'(t)$
  - $I(t) = \int_0^t x(t)dt$



Stability and Convergence

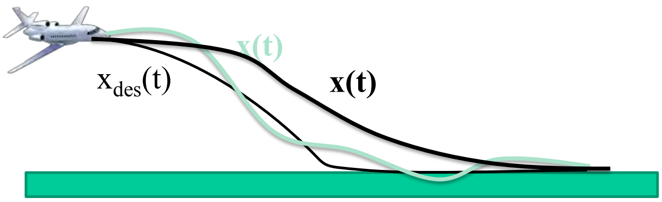
- System is *stable* if errors stay bounded

- System is *convergent* if errors -> 0



Example: Trajectory following

- Say a trajectory $x_{des}(t)$ has been designed
  - E.g., a rocket's ascent, a steering path for a car, a plane's landing
- Apply PID control
  - $u(t) = K_p (x_{des}(t) - x(t)) - K_i I(t) + K_d (x'_{des}(t) - x'(t))$
  - $I(t) = \int_0^t x_{des}(t) - x(t) dt$
- The designer of x_{des} needs to be knowledgeable about the controller's behavior!



Controller Tuning Workflow

- Hypothesize a control policy
- Analysis:
  - Assume a model
  - Assume disturbances to be handled
  - Test performance either through **mathematical analysis**, or through **simulation**
- Go back and redesign control policy
  - Mathematical techniques give you more insight to improve redesign, but require more work

Multivariate Systems

What about more than more interacting aspect?

$$x' = f(x, u)$$
$$x \in \mathbf{X} \subseteq \mathbb{R}^n$$
$$u \in \mathbf{U} \subseteq \mathbb{R}^m$$

- Note: $m \neq n$ *and* variables are **coupled**

→ This is not as easy as setting n PID controllers

➔ Derive a "space" of controllers??

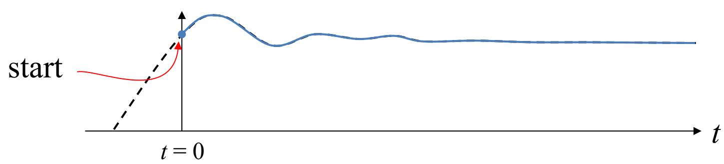
State-Space Modelling (ELEC3004 Super-Summary!)

("Hear Ye! It be stated")

Affairs of state

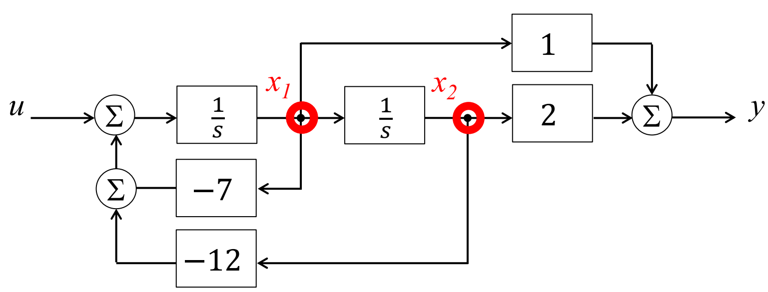
- Introductory brain-teaser:
  - If you have a dynamic system model with history (ie. integration) how do you represent the instantaneous state of the plant?

Eg. how would you setup a simulation of a step response, mid-step?



Introduction to state-space

- We can identify the nodes in the system
  - These nodes 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 variable transformation

- Important note!
  - The states of a control canonical form system are not the same as the modal states
  - They represent the same dynamics, and give the same output, but the vector values are different!
- However we can convert between them:
  - Consider state representations, $\boldsymbol{x}$ and $\boldsymbol{q}$ where

$$\boldsymbol{x} = \mathbf{T}\boldsymbol{q}$$

T is a "transformation matrix"

State variable transformation

- Two homologous representations:

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}u \qquad \dot{\boldsymbol{q}} = \mathbf{F}\boldsymbol{q} + \mathbf{G}u$$
$$y = \mathbf{C}\boldsymbol{x} + Du \quad \text{and} \quad y = \mathbf{H}\boldsymbol{q} + Ju$$

We can write:

$$\dot{\boldsymbol{x}} = \mathbf{T}\dot{\boldsymbol{q}} = \mathbf{AT}\boldsymbol{z} + \mathbf{B}u$$
$$\dot{\boldsymbol{q}} = \mathbf{T}^{-1}\mathbf{AT}\boldsymbol{z} + \mathbf{T}^{-1}\mathbf{B}u$$

Therefore, $\mathbf{F} = \mathbf{T}^{-1}\mathbf{AT}$ and $\mathbf{G} = \mathbf{T}^{-1}\mathbf{B}$

Similarly, $\mathbf{C} = \mathbf{HT}$ and $D = J$

METR 4202: **Robotics** October 19, 2016 - 25

Example: (Back To) Robot Arms

Slides 17-27 Source: R. Lindeke, ME 4135, "Introduction to Control"

METR 4202: **Robotics** October 19, 2016 - 26

Remembering the Motion Models:

- Recall from Dynamics, the Required Joint Torque is:

$$\tau_i = D_i\left(q\right)\ddot{q}_i + C_i\left(q, \dot{q}_i\right) + h\left(q\right) + b\left(\dot{q}_i\right)$$

Dynamical Manipulator Inertial Tensor – a function of position and acceleration

Coupled joint effects (centrifugal and coriolis) issues due to multiple moving joints

Gravitational Effects

Frictional Effect due to Joint/Link movement

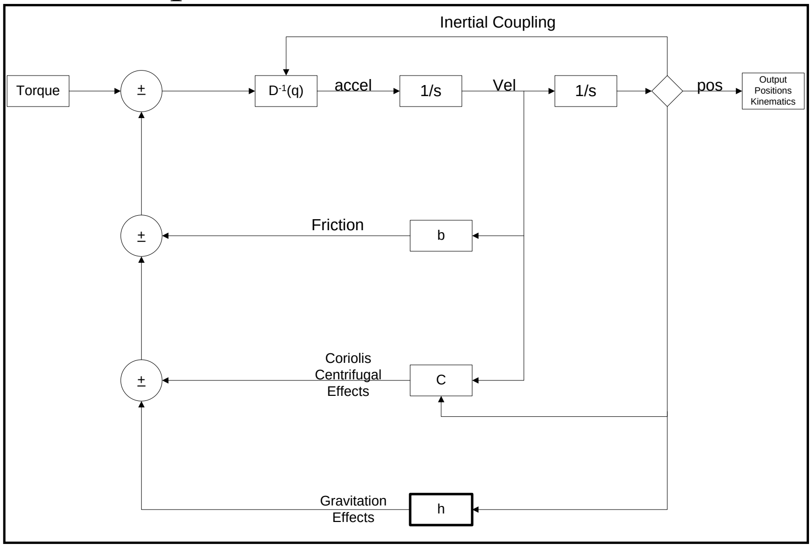
Lets simplify the model

- This torque model is a 2nd order one (in position) lets look at it as a velocity model rather than positional one then it becomes a system of highly coupled 1st order differential equations

- We will then isolate Acceleration terms (acceleration is the 1st derivative of velocity)

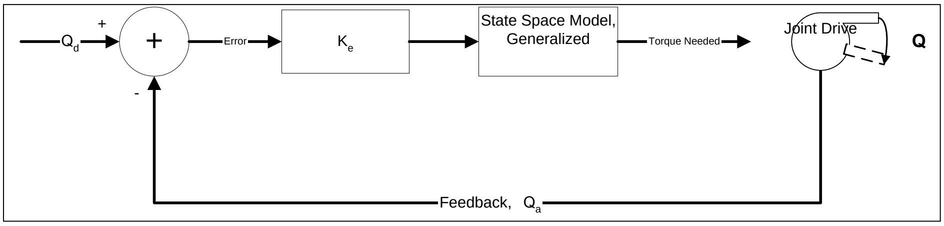
$$a = \dot{v} = \ddot{q} = D_i^{-1}(q)\left(\tau_i - C_i(q, \dot{q}_i) - h(q) - b\left(\dot{q}_1\right)\right)$$

The State-Space Control Model:



Setting up a Real Control

- We will (start) by using positional error to drive our torque devices



- This simple model is called a PE (proportional error) controller

PE Controller:

- To a 1st approximation, $\tau = K_m * I$
  - Torque is proportional to motor current

- And the Torque required is a function of 'Inertial' (Acceleration) and 'Friction' (velocity) effects as suggested by our L-E models

$$\tau_m \simeq J_{eq}\ddot{q} + F_{eq}\dot{q}$$

→ Which can be approximated as:

$$K_m I_m = J_{eq}\ddot{q} + F_{eq}\dot{q}$$

Setting up a "Control Law"

- We will use the positional error (as drawn in the state model) to develop our torque control
- We say then for PE control:

$$\tau \propto k_{pe}(\theta_d - \theta_a)$$

- Here, k_{pe} is a "gain" term that guarantees sufficient current will be generated to develop appropriate torque based on observed positional error

Using this Control Type:

- It is a representation of the physical system of a mass on a spring!
- We say after setting our target as a ‘zero goal’ that:

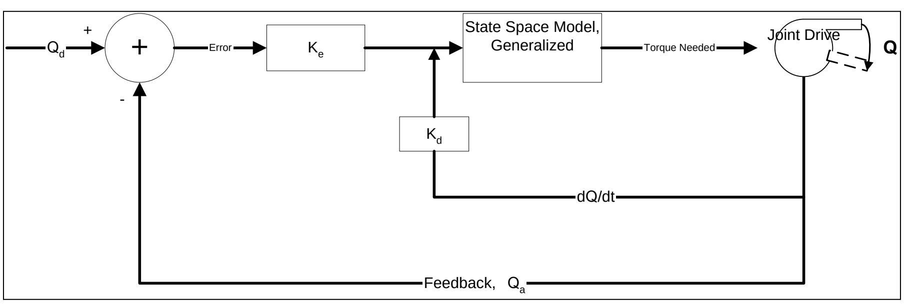
$$-k_{pe} * \theta_a = J\ddot{\theta} + F\dot{\theta}$$

the solution of which is:

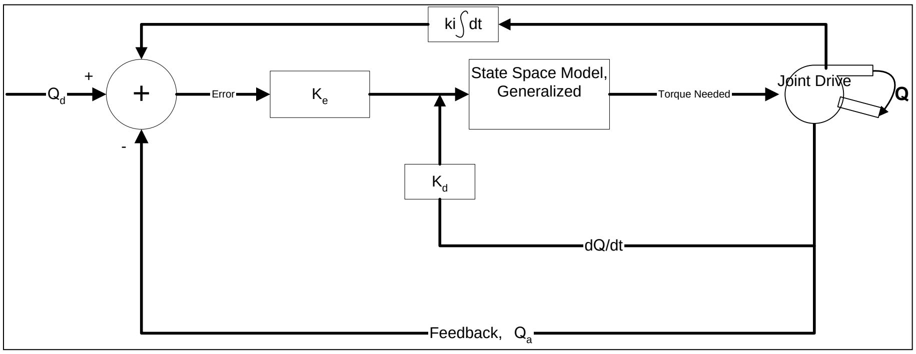
θ_a is a function of the servo feedback as a function of time!

$$\left(\theta_a = e^{-\left(F/2J\right)t}\left[C_1 e^{\left(1/2\right)\omega t} + C_2 e^{-\left(1/2\right)\omega t}\right]\right.$$

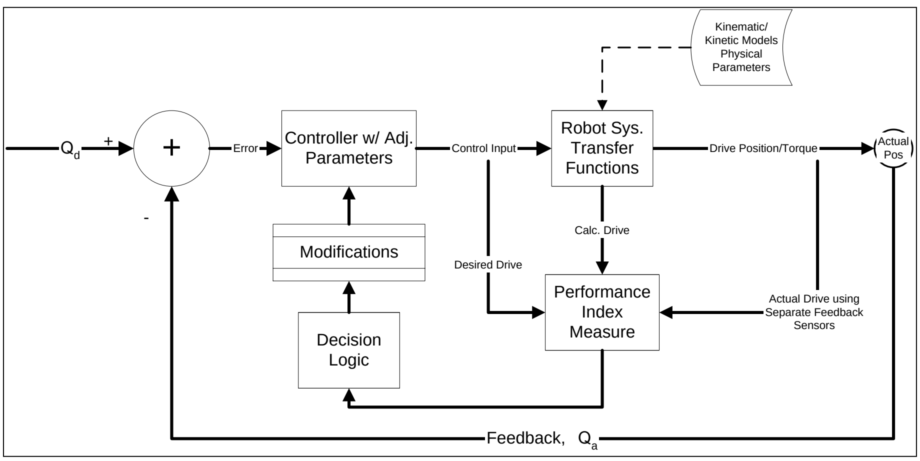
State Space Model of PD:



PID State Space Model:



State Model of Adjustable Controller



State-Space Control Design

AKA, it's all about $\vec{\mathbf{u}}$ ☺

Punchline: it's an Optimization
*(Algebraic Riccati equation → Nonlinear optimization → **Search! …***
Just like SLAM | Motion Planning!!)

State-space control design

- Design for discrete state-space systems is just like the continuous case.
  - Apply linear state-variable feedback:

$$u = -\mathbf{K}\boldsymbol{x}$$

such that $\det(z\mathbf{I} - \boldsymbol{\Phi} + \boldsymbol{\Gamma}\mathbf{K}) = \alpha_c(z)$

where $\alpha_c(z)$ is the desired control characteristic equation

Predictably, this requires the system controllability matrix $\boldsymbol{\mathcal{C}} = [\boldsymbol{\Gamma} \quad \boldsymbol{\Phi}\boldsymbol{\Gamma} \quad \boldsymbol{\Phi}^2\boldsymbol{\Gamma} \quad \cdots \quad \boldsymbol{\Phi}^{n-1}\boldsymbol{\Gamma}]$ to be full-rank.

Great, so how about control?

- Given $\dot{\boldsymbol{x}} = \mathbf{F}\boldsymbol{x} + \mathbf{G}u$, if we know $\mathbf{F}$ and $\mathbf{G}$, we can design a controller $u = -\mathbf{K}\boldsymbol{x}$ such that

$$\mathrm{eig}(\mathbf{F} - \mathbf{GK}) < 0$$

- In fact, if we have full measurement and control of the states of $\boldsymbol{x}$, we can position the poles of the system in arbitrary locations!

(Of course, that never happens in reality.)

Example: PID control

- Consider a system parameterised by three states:
  - x_1, x_2, x_3
  - where $x_2 = \dot{x}_1$ and $x_3 = \dot{x}_2$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -2 \end{bmatrix} \boldsymbol{x} - \mathbf{K}u$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \boldsymbol{x} + 0u$$

x_2 is the output state of the system;

x_1 is the value of the integral;

x_3 is the velocity.

- We can choose **K** to move the eigenvalues of the system as desired:

$$\det\begin{bmatrix} 1-K_1 & & \\ & 1-K_2 & \\ & & -2-K_3 \end{bmatrix} = \mathbf{0}$$

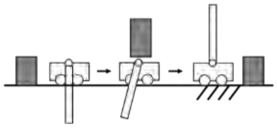
All of these eigenvalues must be positive.

It's straightforward to see how adding derivative gain K_3 can stabilise the system.

Example: Inverted Pendulum

Digital Control

Wikipedia,
Cart and pole



$$L = \frac{1}{2}Mv_1^2 + \frac{1}{2}mv_2^2 - mg\ell\cos\theta$$

where v_1 is the velocity of the cart and v_2 is the velocity of the point mass m. v_1 and v_2 can be expressed in terms of x and θ by writing the velocity as the first derivative of the position;

$$v_1^2 = \dot{x}^2$$
$$v_2^2 = \left(\frac{d}{dt}(x - \ell\sin\theta)\right)^2 + \left(\frac{d}{dt}(\ell\cos\theta)\right)^2$$

Simplifying the expression for v_2 leads to:

$$v_2^2 = \dot{x}^2 - 2\ell\dot{x}\dot{\theta}\cos\theta + \ell^2\dot{\theta}^2$$

The Lagrangian is now given by:

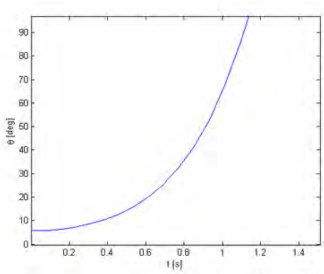
$$L = \frac{1}{2}(M+m)\dot{x}^2 - m\ell\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}m\ell^2\dot{\theta}^2 - mg\ell\cos\theta$$

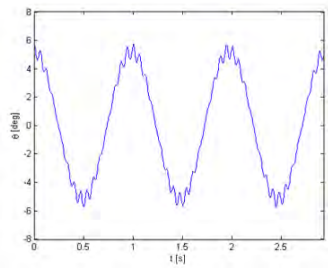
and the equations of motion are:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0$$

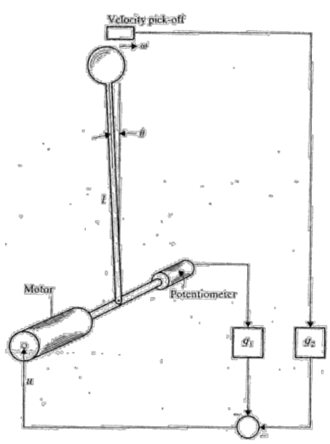
substituting L in these equations and simplifying leads to the equations that describe the motion o

$$(M+m)\ddot{x} - m\ell\ddot{\theta}\cos\theta + m\ell\dot{\theta}^2\sin\theta = F$$
$$\ell\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$





Inverted Pendulum



$$L = \frac{1}{2}Mv_1^2 + \frac{1}{2}mv_2^2 - mg\ell\cos\theta$$

where v_1 is the velocity of the cart and v_2 is the velocity of the point mass m. v_1 and v_2 can be expressed in terms of x and θ by writing the velocity as the first derivative of the position;

$$v_1^2 = \dot{x}^2$$
$$v_2^2 = \left(\frac{d}{dt}(x - \ell\sin\theta)\right)^2 + \left(\frac{d}{dt}(\ell\cos\theta)\right)^2$$

Simplifying the expression for v_2 leads to:

$$v_2^2 = \dot{x}^2 - 2\ell\dot{x}\dot{\theta}\cos\theta + \ell^2\dot{\theta}^2$$

The Lagrangian is now given by:

$$L = \frac{1}{2}(M+m)\dot{x}^2 - m\ell\dot{x}\dot{\theta}\cos\theta + \frac{1}{2}m\ell^2\dot{\theta}^2 - mg\ell\cos\theta$$

and the equations of motion are:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0$$

substituting L in these equations and simplifying leads to the equations that describe the motion o

$$(M+m)\ddot{x} - m\ell\ddot{\theta}\cos\theta + m\ell\dot{\theta}^2\sin\theta = F$$
$$\ell\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$

Inverted Pendulum – Equations of Motion

- The equations of motion of an inverted pendulum (under a small angle approximation) may be linearized as:

$$\dot{\theta} = \omega$$
$$\dot{\omega} = \ddot{\theta} = Q^2\theta + Pu$$

Where:

$$Q^2 = \left(\frac{M+m}{Ml}\right)g$$
$$P = \frac{1}{Ml}.$$

If we further assume unity $\boldsymbol{Ml}$ ($Ml \approx 1$), then $P \approx 1$

Inverted Pendulum –State Space

- We then select a state-vector as:

$$\boldsymbol{x} = \begin{bmatrix}\theta \\ \omega\end{bmatrix}, \text{ hence } \dot{\boldsymbol{x}} = \begin{bmatrix}\dot{\theta} \\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}\omega \\ \dot{\omega}\end{bmatrix}$$

- Hence giving a state-space model as:

$$A = \begin{bmatrix}0 & 1 \\ Q^2 & 0\end{bmatrix}, B = \begin{bmatrix}0 \\ 1\end{bmatrix}$$

- The resolvent of which is:

$$\Phi(s) = (sI - A)^{-1} = \begin{bmatrix}s & -1 \\ -Q^2 & s\end{bmatrix}^{-1} = \frac{1}{s^2 - Q^2}\begin{bmatrix}s & 1 \\ Q^2 & s\end{bmatrix}$$

- And a state-transition matrix as:

$$\Phi(t) = \begin{bmatrix}\cosh Qt & \dfrac{\sinh Qt}{Q} \\ Q\ sinh\ Qt & \cosh Qt\end{bmatrix}$$

Shaping of Dynamic Responses

ELEC3004 Flashback: Another way to see P I|D

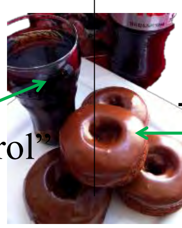
- Derivative

  D provides:
  - High sensitivity
  - Responds to change
  - Adds "damping" & ∴ permits larger K_P
  - Noise sensitive
  - Not used alone (∵ its on rate change of error – by itself it wouldn't get there)

→ "**D**iet Coke of control"

- Integral
  - Eliminates offsets (makes regulation ☺)
  - Leads to Oscillatory behaviour
  - Adds an "order" but instability (Makes a 2nd order system 3rd order)

→ "**I**nteresting cake of control"



PID control

- Consider a system parameterised by three states:
  - x_1, x_2, x_3
  - where $x_2 = \dot{x}_1$ and $x_3 = \dot{x}_2$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -2 \end{bmatrix} \boldsymbol{x} - \mathbf{K}u$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \boldsymbol{x} + 0u$$

x_2 is the output state of the system;

x_1 is the value of the integral;

x_3 is the velocity.

PID control [2]

- We can choose **K** to move the eigenvalues of the system as desired:

$$\det \begin{bmatrix} 1 - K_1 & & \\ & 1 - K_2 & \\ & & -2 - K_3 \end{bmatrix} = \mathbf{0}$$

All of these eigenvalues must be positive.

It's straightforward to see how adding derivative gain K_3 can stabilise the system.

Implementation of Digital PID Controllers

We will consider the PID controller with an s-domain transfer function

$$\frac{U(s)}{X(s)} = G_c(s) = K_P + \frac{K_I}{s} + K_D s. \qquad (13.54)$$

We can determine a digital implementation of this controller by using a discrete approximation for the derivative and integration. For the time derivative, we use the **backward difference rule**

$$u(kT) = \left.\frac{dx}{dt}\right|_{t=kT} = \frac{1}{T}(x(kT) - x[(k-1)T]). \qquad (13.55)$$

The z-transform of Equation (13.55) is then

$$U(z) = \frac{1 - z^{-1}}{T} X(z) = \frac{z-1}{Tz} X(z).$$

The integration of $x(t)$ can be represented by the **forward-rectangular integration** at $t = kT$ as

$$u(kT) = u[(k-1)T] + Tx(kT), \qquad (13.56)$$

Source: Dorf & Bishop, Modern Control Systems, §13.9, pp. 1030-1

Implementation of Digital PID Controllers (2)

where $u(kT)$ is the output of the integrator at $t = kT$. The z-transform of Equation (13.56) is

$$U(z) = z^{-1}U(z) + TX(z),$$

and the transfer function is then

$$\frac{U(z)}{X(z)} = \frac{Tz}{z-1}.$$

Hence, the z-domain transfer function of the **PID controller** is

$$\boxed{G_c(z) = K_P + \frac{K_I T z}{z-1} + K_D \frac{z-1}{Tz}.} \qquad (13.57)$$

The complete difference equation algorithm that provides the PID controller is obtained by adding the three terms to obtain [we use $x(kT) = x(k)$]

$$\begin{aligned} u(k) &= K_P x(k) + K_I[u(k-1) + Tx(k)] + (K_D/T)[x(k) - x(k-1)] \\ &= [K_P + K_I T + (K_D/T)]x(k) - K_D T x(k-1) + K_I u(k-1). \end{aligned} \qquad (13.58)$$

Equation (13.58) can be implemented using a digital computer or microprocessor. Of course, we can obtain a PI or PD controller by setting an appropriate gain equal to zero.

Source: Dorf & Bishop, Modern Control Systems, §13.9, pp. 1030-1

Let's Generalize This: Shaping the Dynamic Response

- A method of designing a control system for a process in which all the state variables are accessible for Measurement

➔ This method is also known as ***pole-placement***

- Theory:
  - We will find that in a controllable system, with all the state variables accessible for measurement, it is possible to place the closed-loop poles anywhere we wish in the complex s plane!
- Practice:
  - Unfortunately, however, what can be attained in principle may not be attainable in practice. Speeding the response of a sluggish system requires the use of large control signals which the actuator (or power supply) may not be capable of delivering. And, control system gains are **very sensitive** to the location of the open-loop poles

Pole Placement

Pole Placement (Following FPW – Chapter 6)

- FPW has a slightly different notation:

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}u,$$
$$y = \mathbf{H}\mathbf{x}.$$

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k),$$
$$y(k) = \mathbf{H}\mathbf{x}(k),$$

$$\mathbf{\Phi} = e^{\mathbf{F}T},$$
$$\mathbf{\Gamma} = \int_0^T e^{\mathbf{F}\eta} d\eta \mathbf{G},$$

Pole Placement

- Start with a simple feedback control law ("controller")

$$u = -\mathbf{K}\mathbf{x} = -[K_1 K_2 \ldots] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix}$$

- It's actually a regulator
  ∵ it does not allow for a reference input to the system.
  (there is no "reference" **r** (r = 0))

- Substitute in the difference equation

$$x(k+1) = \Phi x(k) - \Gamma K x(k)$$

- $\mathcal{Z}$ Transform:

$$(zI - \Phi + \Gamma K)X(z) = 0$$

➔Characteristic Eqn:

$$\det|zI - \Phi + \Gamma K| = 0$$

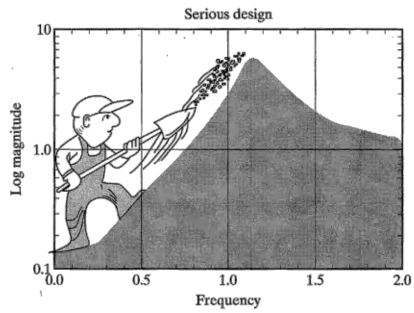
Pole Placement

Pole placement: Big idea:

- Arbitrarily select the desired root locations of the closed-loop system and see if the approach will work.
- AKA: full state feedback
  - ∵ enough parameters to influence all the closed-loop poles
- Finding the elements of K so that the roots are in the desired locations. Unlike classical design, where we iterated on parameters in the compensator (hoping) to find acceptable root locations, the full state feedback, pole-placement approach guarantees success and allows us to arbitrarily pick any root locations, providing that n roots are specified for an n^{th}-order system.

Meaning – No Free Lunch

- The energy (and sensitivity) moves around (in this case in "frequency")



- Sensitivity reduction at low frequency unavoidably leads to sensitivity increase at higher frequencies.

Source: Gunter Stein's interpretation of the water bed effect – G. Stein, *IEEE Control Systems Magazine,* 2003.

Back to Pole Placement

- Given:

$$z_i = \beta_1, \beta_2, \beta_3, \ldots$$

- This gives the desired control-characteristic equation as:

$$a_c(z) = (z - \beta_1)(z - \beta_2)(z - \beta_3) \ldots =$$

- Now we "just solve" for **K** and "bingo"

Pole Placement Example (FPW p. 241)

Example 6.1: Suppose we want to design a control law for the satellite attitude-control system described by (2.45) with $\boldsymbol{x} = [\boldsymbol{x}_1 \;\; \boldsymbol{x}_2]$. Example 2.13 showed that the discrete model for this system is

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Gamma} = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}.$$

We want to pick z-plane roots of the closed-loop characteristic equation so that the equivalent s-plane roots have a damping ratio of $\zeta = 0.5$ and real part of $s = -1.8$ rad/sec (i.e., $s = -1.8 \pm j3.12$ rad/sec). Using $z = e^{sT}$ with a sample period of $T = 0.1$ sec, we find that $z = 0.8 \pm j0.25$, as shown in Fig. 6.1. The desired characteristic equation is then

$$z^2 - 1.6z + 0.70 = 0, \qquad (6.9)$$

and the evaluation of (6.7) for any control law $\mathbf{K}$ leads to

$$\det \left| z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} [K_1 \quad K_2] \right| = 0$$

or

$$z^2 + (TK_2 + (T^2/2)K_1 - 2)z + (T^2/2)K_1 - TK_2 + 1 = 0. \qquad (6.10)$$

Pole Placement Example (FPW p. 241)

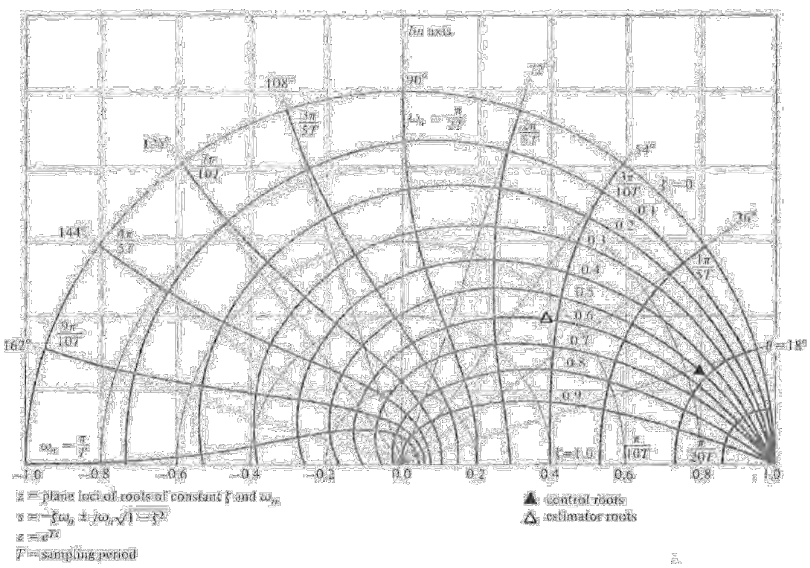
Equating coefficients in (6.9) and (6.10) with like powers of z, we obtain two simultaneous equations in the two unknown elements of $\mathbf{K}$:

$$TK_2 + (T^2/2)K_1 - 2 = -1.6,$$
$$(T^2/2)K_1 - TK_2 + 1 = 0.70,$$

which are easily solved for the coefficients and evaluated for $T = 0.1$ sec:

$$K_1 = \frac{0.10}{T^2} = 10, \qquad K_2 = \frac{0.35}{T} = 3.5.$$

Pole Placement Example (FPW p. 241)



Ackermann's Formula (FPW p. 245)

- Gains maybe approximated with:

$$\boxed{\mathbf{K} = [0 \ldots 0 \quad 1][\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma} \quad \mathbf{\Phi}^2\mathbf{\Gamma} \ldots \mathbf{\Phi}^{n-1}\mathbf{\Gamma}]^{-1}\alpha_c(\mathbf{\Phi}),}$$

- Where: C = controllability matrix, *n* is the order of the system (or number of state elements) and α_c:

$$\mathcal{C} = [\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma} \ldots]$$

$$\alpha_c(\mathbf{\Phi}) = \mathbf{\Phi}^n + \alpha_1\mathbf{\Phi}^{n-1} + \alpha_2\mathbf{\Phi}^{n-2} + \cdots + \alpha_n\mathbf{I},$$

  - α_i: coefficients of the desired characteristic equation

Ackermann's Formula Example (FPW p.246)

Example 6.2: Applying Ackermann's formula to the satellite attitude-control system of Example 6.1, we find from (6.9) that

$$\alpha_1 = -1.6, \qquad \alpha_2 = +0.70,$$

and therefore

$$\alpha_c(\mathbf{\Phi}) = \begin{bmatrix} 1 & 2T \\ 0 & 1 \end{bmatrix} - 1.6\begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} + 0.70\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.4T \\ 0 & 0.1 \end{bmatrix}.$$

Furthermore, we find that

$$[\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma}] = \begin{bmatrix} T^2/2 & 3T^2/2 \\ T & T \end{bmatrix}$$

and

$$[\mathbf{\Gamma} \quad \mathbf{\Phi\Gamma}]^{-1} = 1/T^2\begin{bmatrix} -1 & +3T/2 \\ 1 & -T/2 \end{bmatrix},$$

and finally

$$\mathbf{K} = [K_1 \; K_2] = (1/T^2)[0 \; 1]\begin{bmatrix} -1 & 3T/2 \\ 1 & -T/2 \end{bmatrix}\begin{bmatrix} 0.1 & 0.4T \\ 0 & 0.1 \end{bmatrix};$$

therefore

$$[K_1 \; K_2] = \frac{1}{T^2}[0.1 \; 0.35T]$$
$$= [10 \quad 3.5],$$

which is the same result as that obtained earlier.

Viewing State-Space as a Tool for Solving ODEs Simultaneously

METR 4202: **Robotics** October 19, 2016 -65

State Space as an ODE

- The basic mathematical model for an LTI system consists of the state differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \qquad \mathbf{x}(t_0) = \mathbf{x}_0$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

- The solution is can be expressed as a sum of terms owing to the initial state and to the input respectively:

$$x(t) = \underbrace{e^{at}x_0}_{\text{zero-input response}} + \underbrace{\int_0^t e^{a(t-\tau)}bu(\tau)d\tau}_{\text{zero-state response}} \qquad y(t) = ce^{at}x_0 + \int_0^t ce^{a(t-\tau)}bu(\tau)d\tau + du(t)$$

- This is a first-order solution similar to what we expect

METR 4202: **Robotics** October 19, 2016 -66

State Equation Solution: Matrix Exponential

$$x(t) = \boxed{e^{at}x_0} + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau \qquad y(t) = ce^{at}x_0 + \int_0^t ce^{a(t-\tau)}bu(\tau)d\tau + du(t)$$

- The first term can be handled via a Taylor Series

$$e^{\mathbf{A}(t-t_0)} = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{A}^k (t-t_0)^k = \mathbf{I} + \mathbf{A}(t-t_0) + \frac{1}{2}\mathbf{A}^2(t-t_0)^2 + \frac{1}{6}\mathbf{A}^3(t-t_0)^3 + \ldots$$

→ This case is known as the matrix exponential function

→ Also referred to as the state-transition matrix, denoted by $\Phi(t, t_0)$:

$$\mathbf{x}(t) = \mathbf{\Phi}(t)\mathbf{x}_0 + \int_{t_0}^t \mathbf{\Phi}(t-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau$$

- The state-transition matrix satisfies the homogeneous state equation, thus, it represents the free response of the system. That is, it governs the response that is excited by the initial conditions only

Output Equation Solution

- Having the solution for the complete state response, a solution for the complete output equation can be obtained as:

$$\mathbf{y}(t) = \underbrace{\mathbf{C}e^{\mathbf{A}t}\mathbf{x}_0}_{\text{zero-input response: } y_{zi}(t)} + \underbrace{\int_{t_0}^t \mathbf{C}e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u}(t)}_{y_{zs}(t)\text{: zero-state response}}$$

State Equation Solution

- Thus, the solution to the unforced system (u=0):

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} \phi_{11}(t) & \cdots & \phi_{1n}(t) \\ \phi_{21}(t) & \cdots & \phi_{n2}(t) \\ \vdots & & \vdots \\ \phi_{n1}(t) & \cdots & \phi_{nn}(t) \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_2(0) \\ \vdots \\ x_n(0) \end{bmatrix}$$

Note: the term $\phi_{ij}(t)$ can be interpreted as the response of the i^{th} state variable due to an initial condition on the j^{th} state variable when there are zero initial conditions on all other states.

- The solution of the state differential equation can also be obtained using the Laplace transform:

$$L[\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)]$$

$$L[\dot{\mathbf{x}}(t)] = L[\mathbf{A}\mathbf{x}(t)] + L[\mathbf{B}\mathbf{u}(t)]$$

$$s\mathbf{X}(s) - \mathbf{x}_0 = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s)$$

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)$$

$$\mathbf{X}(s) = \mathbf{\Phi}(s)\mathbf{x}_0 + \mathbf{\Phi}(s)\mathbf{B}\mathbf{U}(s)$$

$\Rightarrow$ $L[\mathbf{\Phi}(t)] = \mathbf{\Phi}(s) = [s\mathbf{I} - \mathbf{A}]^{-1} \longrightarrow \boxed{\mathbf{\Phi}(t) = L^{-1}[s\mathbf{I} - \mathbf{A}]^{-1}}$

Properties of the Matrix Exponential

- Note that $e^{\mathbf{A}t}$ is just a notation used to represent a power series.

$$e^{\mathbf{A}t} \neq [e^{a_{ij}t}]$$

- Example 1: Consider the following 4x4 matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Let's obtain the first terms of the power series:

$$A^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad A^3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \quad A^4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad A^k = 0 \quad \forall k \geq 4$$

The power series contains only a finite number of nonzero terms:

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{1}{2}\mathbf{A}^2t^2 + \frac{1}{6}\mathbf{A}^3t^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -t & 1 & 0 & 0 \\ \frac{1}{2}t^2 & -t & 1 & 0 \\ -\frac{1}{6}t^3 & \frac{1}{2}t^2 & -t & 1 \end{bmatrix} \neq [e^{a_{ij}t}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ e^{-t} & 0 & 0 & 0 \\ 0 & e^{-t} & 0 & 0 \\ 0 & 0 & e^{-t} & 0 \end{bmatrix}$$

Properties of the matrix exponential

- For any real nxn matrix **A**, the matrix exponential $e^{\mathbf{A}t}$ satisfies:
  1. $e^{\mathbf{A}t}$ is the <u>unique</u> matrix for which: $\frac{d}{dt}e^{\mathbf{A}t} = \mathbf{A}e^{\mathbf{A}t} \quad e^{\mathbf{A}t}\big|_{t=0} = \mathbf{I}(nxn)$
  2. For any t_1 and t_2: $e^{\mathbf{A}(t_1+t_2)} = e^{\mathbf{A}t_1}e^{\mathbf{A}t_2}$

     As a consequence: $e^{\mathbf{A}(0)} = e^{\mathbf{A}(t-t)} = e^{\mathbf{A}t}e^{-\mathbf{A}t} = \mathbf{I}$

     Thus, $e^{\mathbf{A}t}$ is invertible for all t, being the inverse: $\left[e^{\mathbf{A}t}\right]^{-1} = e^{-\mathbf{A}t}$
  3. For all t, A and $e^{\mathbf{A}t}$ commute with respect to matrix product: $\mathbf{A}e^{\mathbf{A}t} = e^{\mathbf{A}t}\mathbf{A}$
  4. For all t: $\left[e^{\mathbf{A}t}\right]^T = e^{\mathbf{A}^T t}$
  5. For any real nxn matrix B, $e^{(\mathbf{A}+\mathbf{B})t} = e^{\mathbf{A}t}e^{\mathbf{B}t}$ for all t if and only if **AB=BA**
  6. Finally, a useful property of the matrix exponential is that it can be reduced to a finite power series involving n scalar analytic functions $\alpha_j(t)$

$$e^{\mathbf{A}t} = \sum_{k=0}^{n-1} \alpha_k(t)\mathbf{A}^k$$

METR 4202: **Robotics** October 19, 2016 - 71

Using this to Solve State Space Problems

- Example:
  - Solve the following linear second-order ordinary differential

    $\ddot{y}(t) + 7\dot{y}(t) + 12y(t) = u(t)$
  - Consider the input u(t) is a step of magnitude 3 and the initial conditions $\dot{y}(0) = 0.05 \quad y(0) = 0.10$

METR 4202: **Robotics** October 19, 2016 - 72

State-Space Exercise

- Solve the following linear second-order ordinary differential eq:
  a. Using standard solution techniques
  b. Using S-S solution techniques

$$\ddot{y}(t)+7\dot{y}(t)+12y(t)=u(t)$$

Consider the input u(t) is a step of magnitude 3

and the initial conditions: $\dot{y}(0)=0.05 \quad y(0)=0.10$

The first question can be solved by the students in order to review the techniques exposed in previous courses.

To solve the second question, we first choose state variables using phase-variable choice.

$$x_1=y$$
$$x_2=\dot{y}=\dot{x}_1$$
$$\dot{x}_2=\ddot{y}=u_s-12x_1-7x_2$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ -12 & -7\end{bmatrix}\cdot\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\underbrace{\begin{bmatrix}0\\ 1\end{bmatrix}}_{B}u(t) \qquad \mathbf{A}=\begin{bmatrix}0 & 1\\ -12 & -7\end{bmatrix}$$

$$\begin{bmatrix}x_1(0)\\ x_2(0)\end{bmatrix}=\begin{bmatrix}0.10\\ 0.05\end{bmatrix}$$

$$y(t)=\underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{C}\cdot\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\underbrace{[0]}_{D}\cdot u(t)$$

Powers of A are not nulls, thus, obtaining the state transition matrix as a power series is not practical

G.Oliver, UIB

State-Space Exercise

- The expression $\mathbf{\Phi}(t)=L^{-1}[s\mathbf{I}-\mathbf{A}]^{-1}$ is recommended:

$$s\mathbf{I}-\mathbf{A}=\begin{bmatrix}s & -1\\ 12 & s+7\end{bmatrix}$$
$$\det(s\mathbf{I}-\mathbf{A})=|s\mathbf{I}-\mathbf{A}|=s^2+7s+12$$

$$\mathbf{\Phi}(s)=(s\mathbf{I}-\mathbf{A})^{-1}=\frac{1}{s^2+7s+12}\begin{bmatrix}s+7 & 1\\ -12 & s\end{bmatrix}$$

- Thus, from $\mathbf{X}(s)=(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}_0+(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}U(s)$

$$\mathbf{X}(s)=\frac{1}{s^2+7s+12}\begin{bmatrix}s+7 & 1\\ -12 & s\end{bmatrix}\begin{bmatrix}0.10\\ 0.05\end{bmatrix}+\frac{1}{s^2+7s+12}\begin{bmatrix}s+7 & 1\\ -12 & s\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix}\frac{3}{s}=$$

$$=\frac{1}{s^2+7s+12}\begin{bmatrix}0.1s+0.75+\frac{3}{s}\\ 0.05s+1.8\end{bmatrix}=\begin{bmatrix}\dfrac{0.1s^2+0.75s+3}{s(s+3)(s+4)}\\ \dfrac{0.05s+1.8}{(s+3)(s+4)}\end{bmatrix}=\begin{bmatrix}X_1(s)\\ X_2(s)\end{bmatrix}$$

$$X_1(s)=\frac{0.1s^2+0.75s+3}{s(s+3)(s+4)}=\frac{0.25}{s}-\frac{0.55}{s+3}+\frac{0.4}{s+4} \qquad X_2(s)=\frac{0.05s+1.8}{(s+3)(s+4)}=\frac{1.65}{s+3}-\frac{1.60}{s+4}$$

LQR

Linear Quadratic Regulator

- $x' = Ax + Bu$
- Objective: minimize quadratic cost

$$\int \underbrace{\boxed{x^T Q\, x}}_{\text{Error term}} + \underbrace{\boxed{u^T R\, u\, dt}}_{\text{“Effort” penalization}}$$

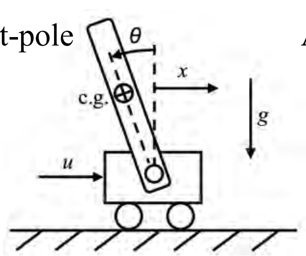
- Over an infinite horizon

Closed form LQR solution

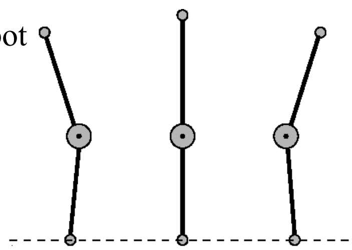
- Closed form solution
  $u = -K x$, with $K = R^{-1}BP$
- Where P is a symmetric matrix that solves the *Riccati equation*
  - $A^TP + PA - PBR^{-1}B^TP + Q = 0$
  - Derivation: calculus of variations
- Packages available for finding solution

Toy Nonlinear Systems

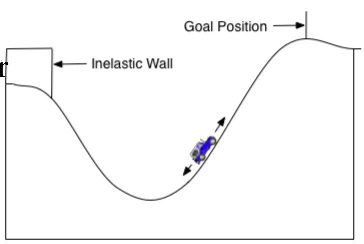
Cart-pole



Acrobot



Mountain car



From Linear to Nonlinear

- We know how to solve (assuming g_t, U_t, X_t convex):

$$\begin{aligned} \min_{u,x} \quad & \sum_{t=0}^{H} g_t(x_t, u_t) \\ \text{subject to} \quad & x_{t+1} = A_t x_t + B_t u_t + c_t \quad \forall t \\ & u_t \in \mathcal{U}_t, x_t \in \mathcal{X}_t \quad \forall t \end{aligned} \qquad (1)$$

- How about nonlinear dy

$$x_{t+1} = f(x_t, u_t) \quad \forall t$$

Shooting Methods (feasible)

Iterate for i=1, 2, 3, …

Execute $u_0^{(i)}, u_1^{(i)}, \ldots, u_T^{(i)}$ (from solving (1))

Linearize around resulting trajectory

Solve (1) for current linearization

Collocation Methods (infeasible)

Iterate for i=1, 2, 3, …

--- (no execution)---

Linearize around current solution of (1)

Solve (1) for current linearization

Sequential Quadratic Programming (SQP) = either of the above methods, but instead of using linearization, linearize equality constraints, convex-quadratic approximate objective function

Model Predictive Control

- Given: $\bar{x}_0$
- For k=0, 1, …, T
  - Solve

$$\begin{aligned} \min_{x,u} \quad & \sum_{t=k}^{T} g_t(x_t, u_t) \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t) \quad \forall t \in \{k, k+1, \ldots, T-1\} \\ & x_k = \bar{x}_k \end{aligned}$$

  - Execute u_k
  - Observe resulting state, $\bar{x}_{k+1}$

Iterative LQR versus Sequential Convex Programming

- Both can solve

$$\begin{aligned} \min_{u,x} \quad & \sum_{t=0}^{H} g_t(x_t, u_t) \\ \text{subject to} \quad & x_{t+1} = f_t(x_t, u_t) \quad \forall t \\ & u_t \in \mathcal{U}_t, x_t \in \mathcal{X}_t \quad \forall t \end{aligned}$$

- Can run iterative LQR both as a shooting method or as a collocation method, it's just a different way of executing "Solve (1) for current linearization." In case of shooting, the sequence of linear feedback controllers found can be used for (closed-loop) execution.

- Iterative LQR might need some outer iterations, adjusting "t" of the log barrier

Shooting Methods	**Collocation Methods**
Iterate for i=1, 2, 3, …	Iterate for i=1, 2, 3, …
Execute feedback controller (from solving (1))	--- (no execution)---
Linearize around resulting **trajectory**	Linearize around current **solution** of (1)
Solve (1) for current linearization	Solve (1) for current linearization

Sequential Quadratic Programming (SQP) = either of the above methods, but instead of using linearization, linearize equality constraints, convex-quadratic approximate objective function

Example Shooting

```
%% a nonlinear control problem: cartpole
clear; clc; close all;

% shooting:
T = 100;
u = randn(1,T)*0.1;
max_iters = 10;
x_init = [-10; 0; 0; 0];
nX = 4; nU = 1;
x_eps = 0.1;
u_eps = 0.1;
dt = 0.1;
Q = eye(nX); R = eye(nU); Q_final = 100*eye(nX);
clear A B c
for iter = 1:max_iters
    % simulate and linearize
    x(:,1) = x_init;
    for t=1:T-1
        x(:,t+1) = sim_cartpole(x(:,t), u(:,t), dt);
        [A{t} B{t} c{t}] = compute_jacobian(@sim_cartpole, x(:,t), u(:,t), dt);
        %cartpole_draw(t*dt, x(:,t));
    end
    figure(1); subplot(3,1,1); hold on; plot(u); ylabel('u');
    subplot(3,1,2); hold on; plot(x(1,:)); ylabel('x'); subplot(3,1,3); hold on; plot(x(2,:));ylabel('\theta');
    cost(iter) = 0;
    for t=1:T-1
        cost(iter) = cost(iter) + x(:,t)'*Q*x(:,t) + u(:,t)'*R*u(:,t) + norm(u(:,t+1)-u(:,t),2);
    end
    cost(iter) = cost(iter) + x(:,T)'*Q_final*x(:,T);
    cost
    % solve convex problem
    cvx_begin
    variables x_cvx(nX,T)  u_cvx(nU,T) s_cvx(1,T);
    minimize( sum(s_cvx(1:T)) )
    subject to
    for t=1:T-1
        x_cvx(:,t+1) == A{t}*(x_cvx(:,t)-x(:,t)) + B{t}*(u_cvx(:,t)-u(:,t)) + c{t};
    end
    for t=1:T-1
        s_cvx(1,t) >= x_cvx(:,t)'*Q*x_cvx(:,t) + u_cvx(:,t)'*R*u_cvx(:,t) + norm(u_cvx(:,t+1)-u_cvx(:,t),2);
    end
    s_cvx(1,T) >= x_cvx(:,T)'*Q_final*x_cvx(:,T);
    for t=1:T
        norm(x_cvx(:,t) - x(:,t),2) <= x_eps;
        norm(u_cvx(:,t) - u(:,t),2) <= u_eps;
    end
    x_cvx(:,1) == x_init;
    cvx_end
    u = u_cvx;
end
```

Example Collocation

```
clear; clc; close all;

T = 100;
u = randn(1,T)*0.1;
max_iters = 10;
x_init = [-10; pi/10; -0.1; 0.1];
nX = 4; nU = 1;
x_eps = 100;
u_eps = 1;
dt = 0.1;
Q = eye(nX); R = eye(nU); Q_final = 100*eye(nX);
clear A B c
x_target = [0;0;0;0];
for i=1:nX
    x_iter1(i,:) = x_init(i):(x_target(i)-x_init(i))/(T-1):x_target(i);
end
u_iter1=zeros(nU,T);

for iter = 1:max_iters
    % linearize (but no simulation)
    if(iter==1)
        x = x_iter1; u = u_iter1;
    else
        x = x_cvx; u = u_cvx;
    end
    for t=1:T-1
        %   x(:,t+1) = sim_cartpole(x(:,t), u(:,t), dt);
        [A{t} B{t} c{t}] = compute_jacobian(@sim_cartpole, x(:,t), u(:,t), dt);
        %cartpole_draw(t*dt, x(:,t));
    end
    figure(2); subplot(3,1,1); hold on; plot(u); ylabel('u');
    subplot(3,1,2); hold on; plot(x(1,:)); ylabel('x'); subplot(3,1,3); hold on; plot(x(2,:));ylabel('\theta');
    cost(iter) = 0;
    for t=1:T-1
        cost(iter) = cost(iter) + x(:,t)'*Q*x(:,t) + u(:,t)'*R*u(:,t) + norm(u(:,t+1)-u(:,t),2);
    end
    cost(iter) = cost(iter) + x(:,T)'*Q_final*x(:,T);
    cost
    % solve convex problem
    cvx_begin
    variables x_cvx(nX,T)  u_cvx(nU,T) s_cvx(1,T);
    minimize( sum(s_cvx(1:T)) )
    subject to
    for t=1:T-1
        x_cvx(:,t+1) == A{t}*(x_cvx(:,t)-x(:,t)) + B{t}*(u_cvx(:,t)-u(:,t)) + c{t};
    end
    for t=1:T-1
        s_cvx(1,t) >= x_cvx(:,t)'*Q*x_cvx(:,t) + u_cvx(:,t)'*R*u_cvx(:,t) + norm(u_cvx(:,t+1)-u_cvx(:,t),2);
    end
    s_cvx(1,T) >= x_cvx(:,T)'*Q_final*x_cvx(:,T);
    for t=1:T
        norm(x_cvx(:,t) - x(:,t),2) <= x_eps;
        norm(u_cvx(:,t) - u(:,t),2) <= u_eps;
    end
    x_cvx(:,1) == x_init;
    cvx_end
    u = u_cvx;
end

% let's evaluate the resulting open-loop sequence:
x(:,1) = x_init;
for t=1:T-1
    x(:,t+1) = sim_cartpole(x(:,t), u(:,t), dt);
end
figure(3); subplot(3,1,1); hold on; plot(u); ylabel('u');
subplot(3,1,2); hold on; plot(x(1,:)); ylabel('x'); subplot(3,1,3); hold on; plot(x(2,:));ylabel('\theta');
final_cost = 0;
for t=1:T-1
  final_cost = final_cost + x(:,t)'*Q*x(:,t) + u(:,t)'*R*u(:,t) + norm(u(:,t+1)-u(:,t),2);
end
final_cost = final_cost + x(:,T)'*Q_final*x(:,T);
final_cost
```


Practical Benefits and Issues with Shooting

+ :

At all times the sequence of controls is meaningful, and the objective function optimized directly corresponds to the current control sequence

-- :

For unstable systems, need to run feedback controller during forward simulation

- Why? Open loop sequence of control inputs computed for the linearized system will not be perfect for the nonlinear system. If the nonlinear system is unstable, open loop execution would give poor performance.
- Fixes:
  - Run Model Predictive Control for forward simulation
  - Compute a linear feedback controller from the 2nd order Taylor expansion at the optimum

Practical Benefits and Issues with Collocation

+ :

Can initialize with infeasible trajectory. Hence if you have a rough idea of a sequence of states that would form a reasonable solution, you can initialize with this sequence of states without needing to know a control sequence that would lead through them, and without needing to make them consistent with the dynamics

-- :

Sequence of control inputs and states might never converge onto a feasible sequence

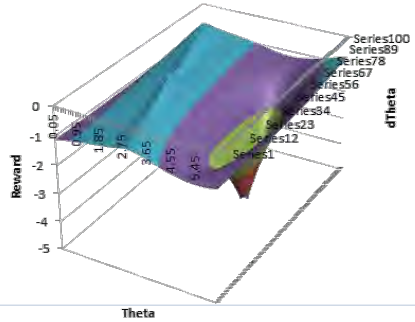
Direct policy synthesis: Optimal control

- *Input*: cost function J(x), estimated dynamics f(x,u), finite state/control spaces X, U
- Two basic classes:
  – **Trajectory optimization**: Hypothesize control sequence u(t), simulate to get x(t), perform optimization to improve u(t), repeat.
  – *Output*: optimal trajectory u(t) (in practice, only a locally optimal solution is found)
  – **Dynamic programming:** Discretize state and control spaces, form a discrete search problem, and solve it.
  – *Output*: Optimal policy u(x) across all of X

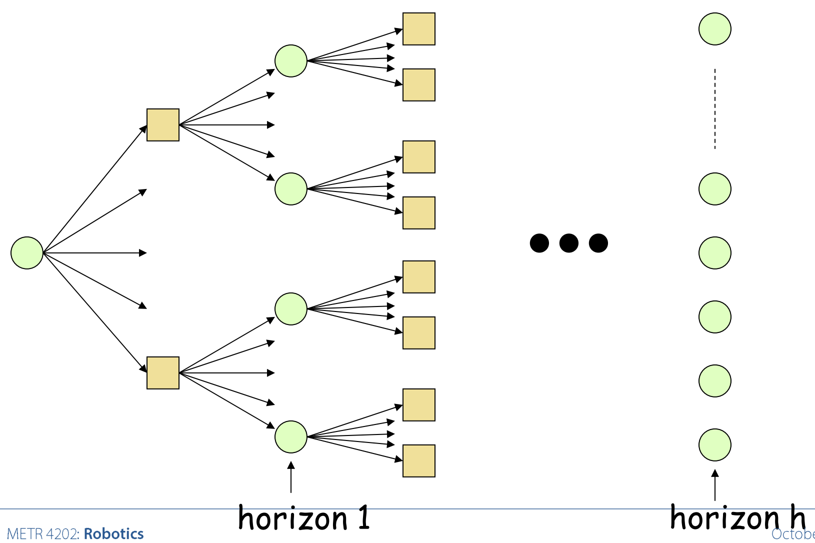
Discrete Search example

- Split X, U into cells $x_1,\dots,x_n$, $u_1,\dots,u_m$
- Build transition function $x_j = f(x_i,u_k)dt$ for all i,k
- State machine with costs dt $J(x_i)$ for staying in state I
- Find $u(x_i)$ that minimizes sum of total costs.
- **Value iteration**: repeated dynamic programming over $V(x_i)$ = sum of total future costs

Value function for 1-joint acrobot



Receding Horizon Control (aka model predictive control)

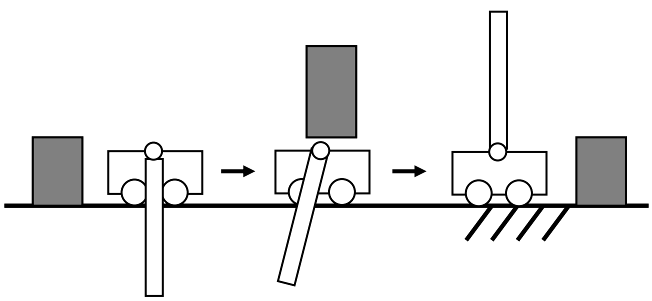


Integrated Planning & Control

(Hooray!!!)

Integrated Planning and Control Methods…

- A motivating problem (for agility)
  - Cart and pole in a cluttered workspace …

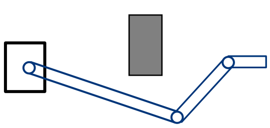


$$\dot{x} = F\left(x, u\right)$$

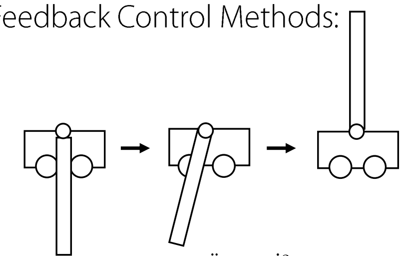
Trajectory Generation with Constraints: Solutions from the Robotics Domain

$$\dot{x} = F\left(x, u\right)$$

Motion Planning Methods:



Feedback Control Methods:

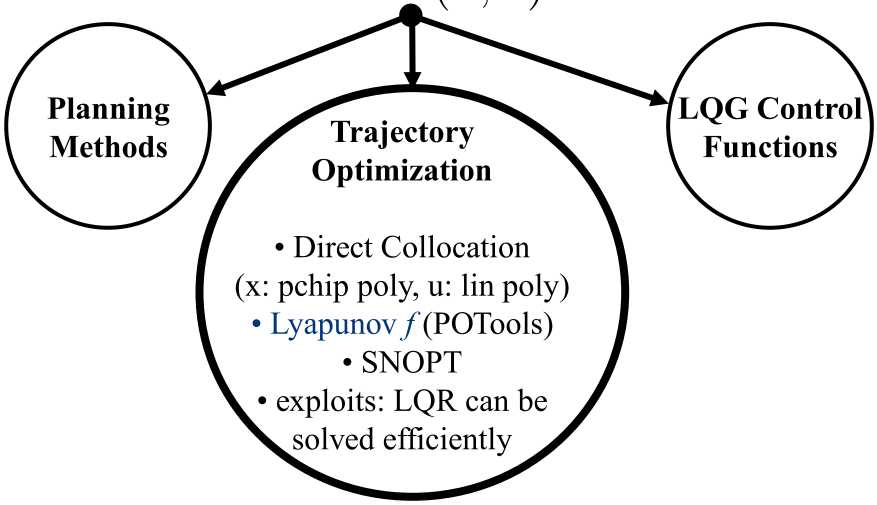


$$(M+m)\,\ddot{y} + ml\cos\theta\ddot{\theta} - ml\dot{\theta}^2\sin\theta = f$$

$$m\left(l\cos\theta\ddot{y} + l^2\ddot{\theta} - gl\sin\theta\right) = 0$$

Trajectory Optimization:
→ Integrated Planning & Feedback:

$$\dot{x} = F(x, u)$$



Few Questions Before Starting…

- *Can it possibly be this hard?*

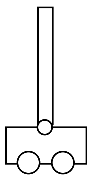
  Yes!
  (1) Dynamic systems are nonlinear $m\left(l^2\ddot{\theta} + gl\sin\theta\right) = 0$
  (2) Decision-theoretic planning problems are combinatorial

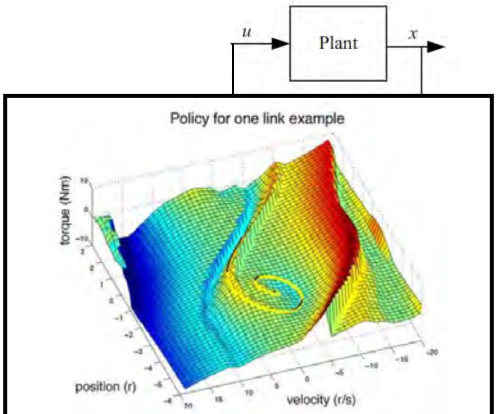
- ***Underactuated system****?*

  [≜ control input cannot drive the state to any arbitrary direction]
  → DOF>actuators: car-like robot, airplane, cart and pole, etc
  ➔ Actuator saturation!

- *Why Now?*
  1. State-of-the-art (LQR-Trees 2009 RSS best paper, kNitro, SDPARA).
  2. Convex Optimization (c/o relaxation) is ~ "online-able"

Viewing This From a Controls & Policy Perspective





Atkeson, C. & Stephens, B.
"Random sampling of states in dynamic programming" IEEE T. Syst. Man Cybern, 2008

Core Idea:
Feedback motion planning (Assistance function) built from a prior model and updated online

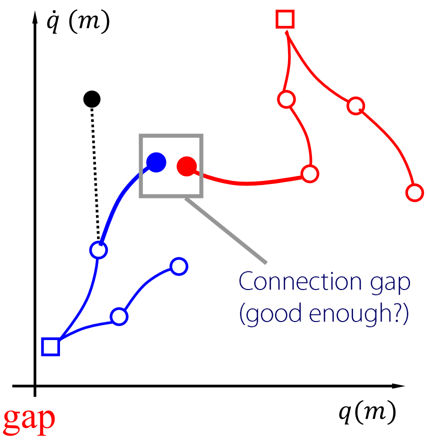
Gain-Scheduled RRT

- Rapidly Exploring Random Trees (RRT) (Background):

Features (+):
1. Solve a control problem
2. Scalable
3. Constrained environments

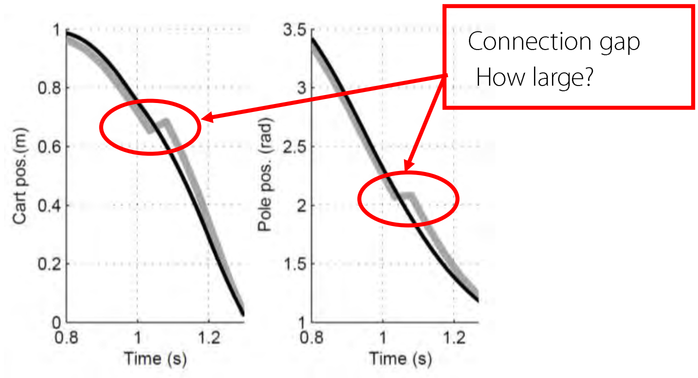
Concerns (--):

4. Works poorly under differential constraints
5. Hard to avoid the connection gap



Gain-Scheduled RRT: RRT Connection Gap

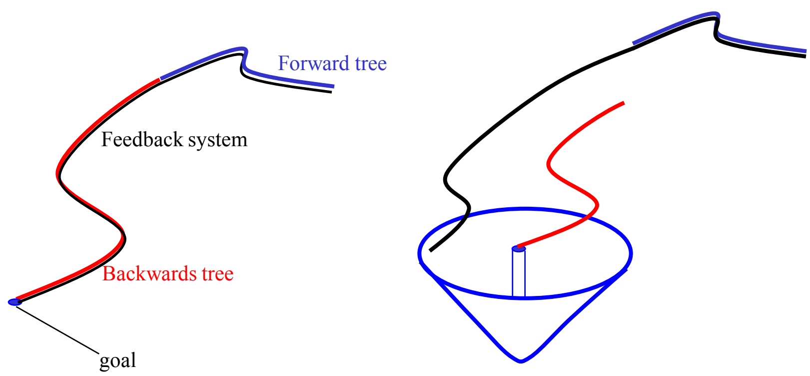
- A RRT solution rarely reaches the goal (or connect the two trees) with zero error



Gain-Scheduled RRT: Relaxing the Search

Single state search:
Extensive exploration

Region search
RRT connection is relaxed



Gain-Scheduled RRT: RoA & Verification

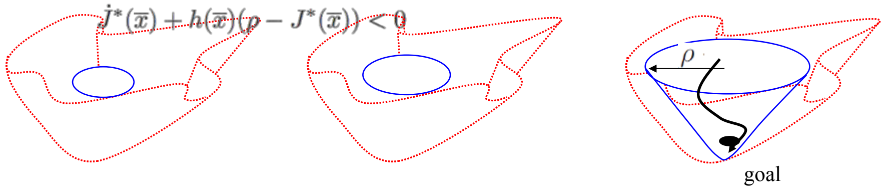
- Find a candidate

In the LQR case: J: optimal cost-to-go S: Algebraic Ricatti Eq.

$$V(x) = J^*(x_{goal} - x) = (x_{goal} - x)^T S(x_{goal} - x)$$

- Maximize candidate (ρ)

$$\dot{J}^*(\bar{x}) + h(\bar{x})(\rho - J^*(\bar{x})) < 0$$



- Verify candidate

Sum of squares relaxation

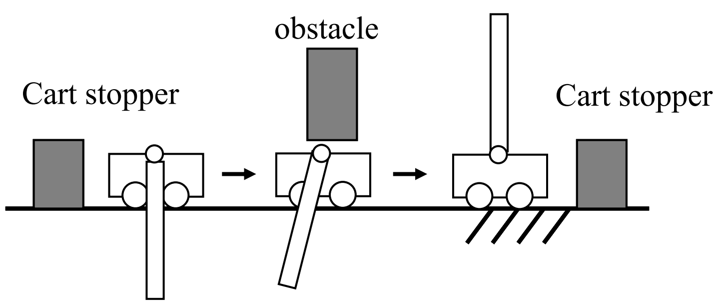
$V(x)$ locally $\boxed{\text{positive definite}}$ in B_ρ

$\dot{V}(x)$ locally $\boxed{\text{negative definite}}$ in B_ρ

**R. Tedrake, *"LQR-Trees: Feedback Motion Planning on Sparse Randomized Trees"*, RSS 2009

Gain-Scheduled RRT: Result

- Cart and pole in a cluttered workspace …



Same initial and final conditions.

Every solution is different due to the random sampling

Gain-Scheduled RRT: Result

Conclusion
No one answer…
Much left to do!

(it's not really magic ☺)

Autonomous Coin Sorting Robot Arm

Authors:
Rory Charlton
Mitch Fullelove
Kianoosh Soltani Naveh
Joshua Song
Tyson Zastrow

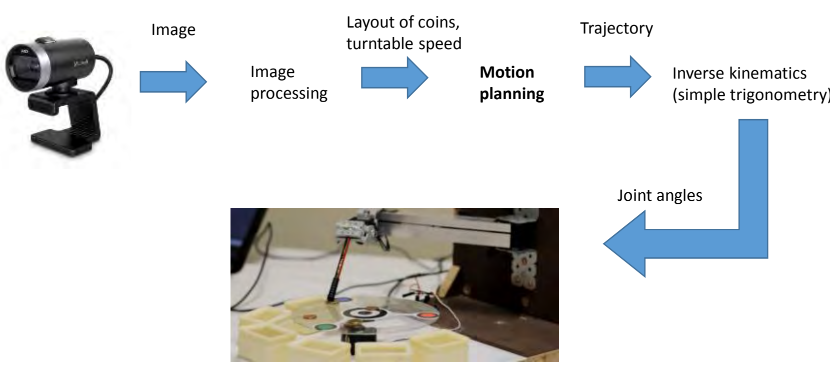
Supervisors:
Dr Surya Singh
Dr Hanna Kurniawati

The University of Queensland
Australia

https://youtu.be/z4qVI7Ybxac

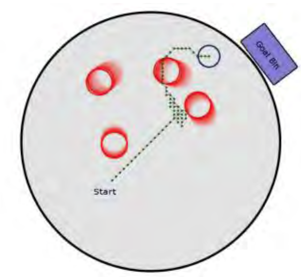
Overview

- Task: sort coins on a moving turntable into bins
- Solution: difficult to pick coins up, so slide them off



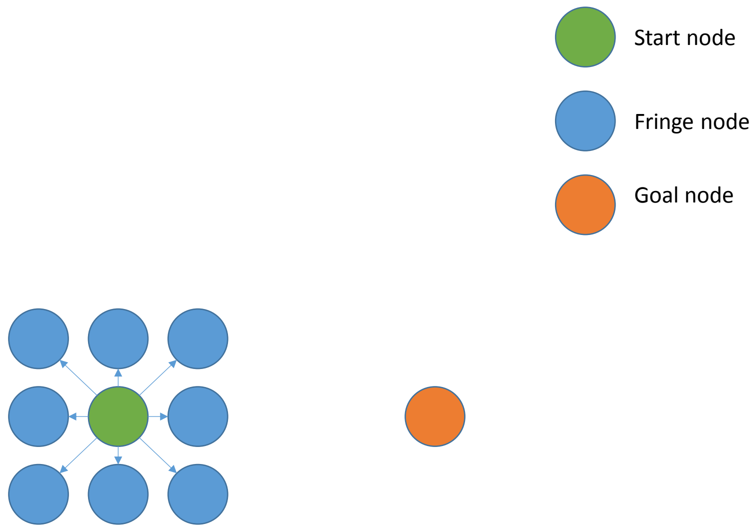
Motion planning problem

- Given a coin and its goal bin, what path can be taken
- Reference frame fixed on table → other coins are moving obstacles
- We planned in a 3 dimensional state space: x, y, time (alternative is to plan in configuration space: joint angle 1, joint angle 2, time)



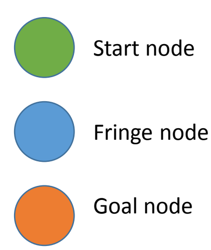
Our simplified A* path finding

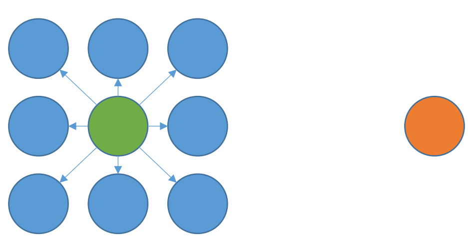
- We slide the coin at a constant speed
- We discretised the state space into a grid (we can get away with this due to sparse obstacles)
- We check for collisions while exploring nodes instead of generating the entire map of valid nodes at the start

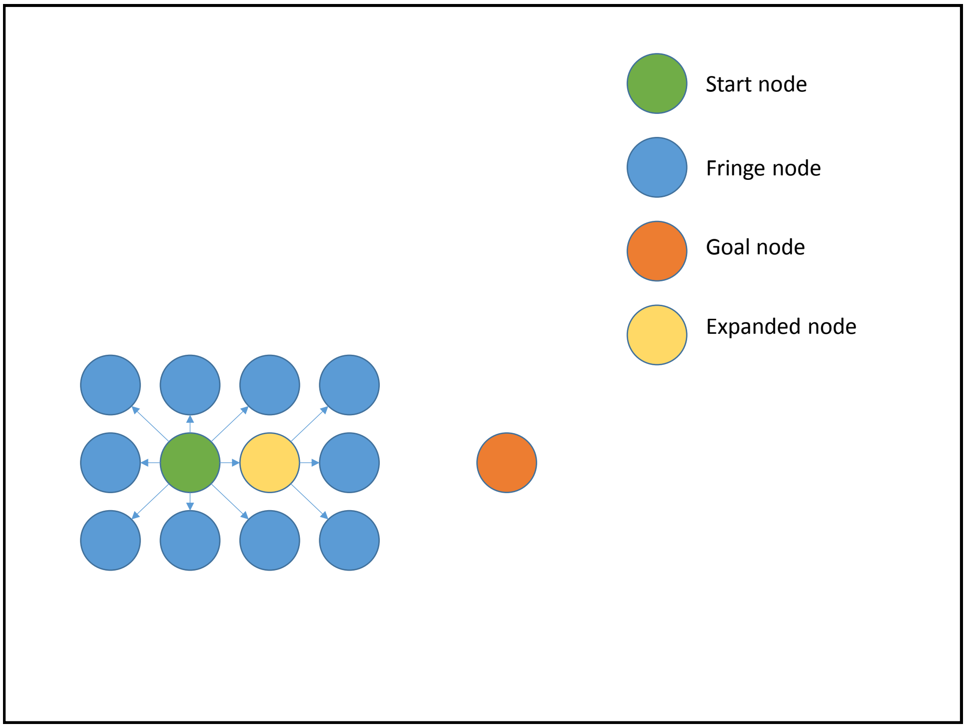


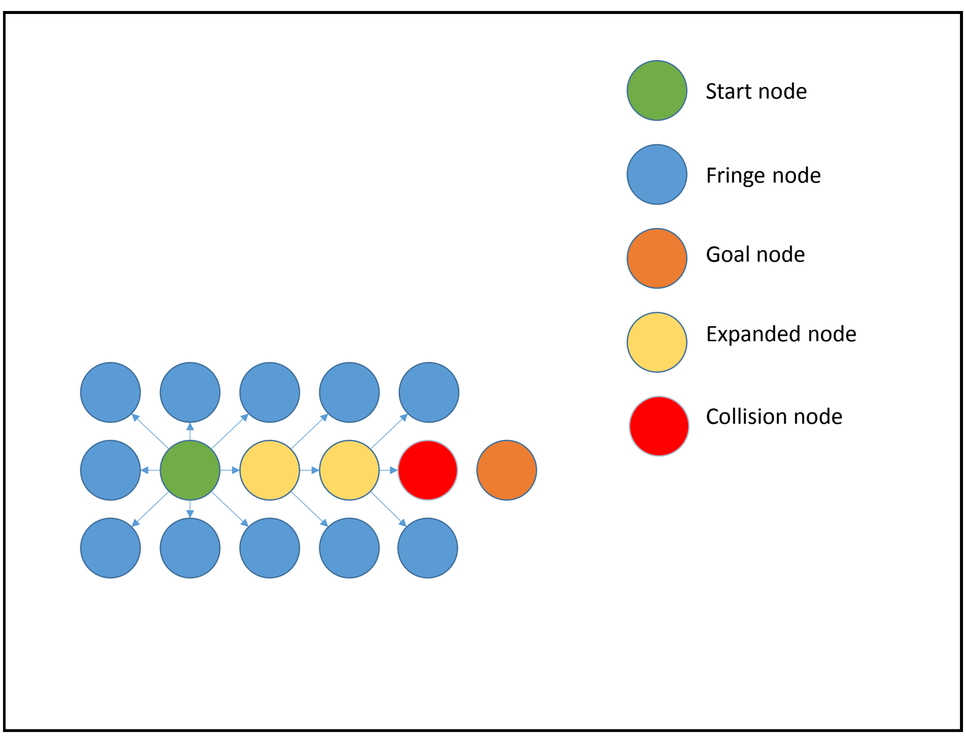
Keep unexplored fringe nodes in a priority queue
Sort score = $g(n) + h(n)$
$g(n)$ is cost from start to n
$h(n)$ is heuristic (estimated cost from n to goal)

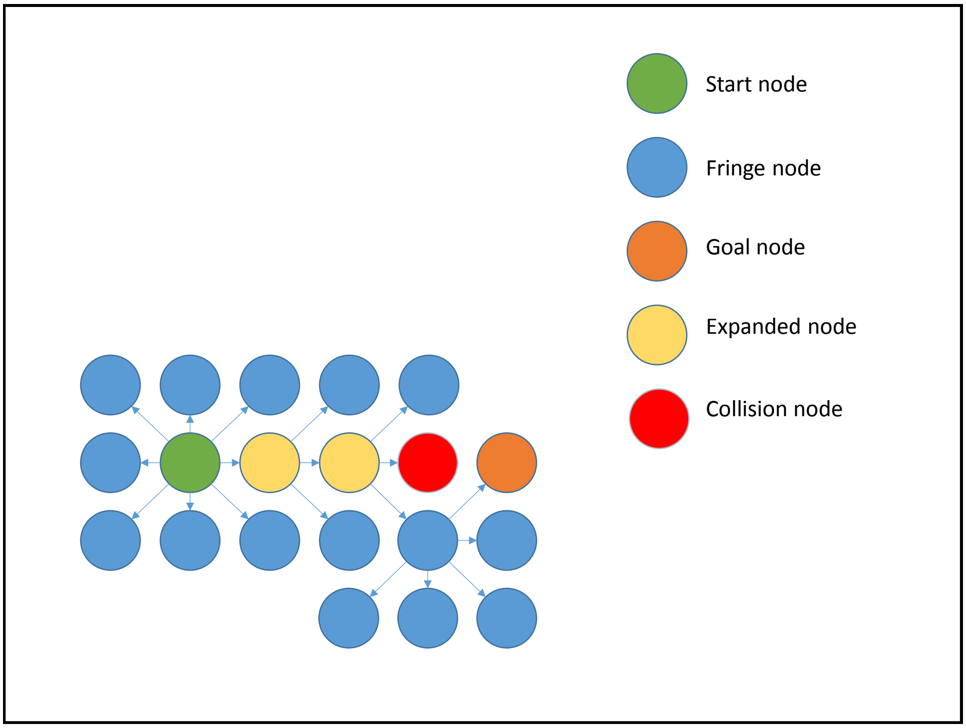
In this case we start with 8 fringe nodes in the queue. Obviously the one on the right is first in the queue.

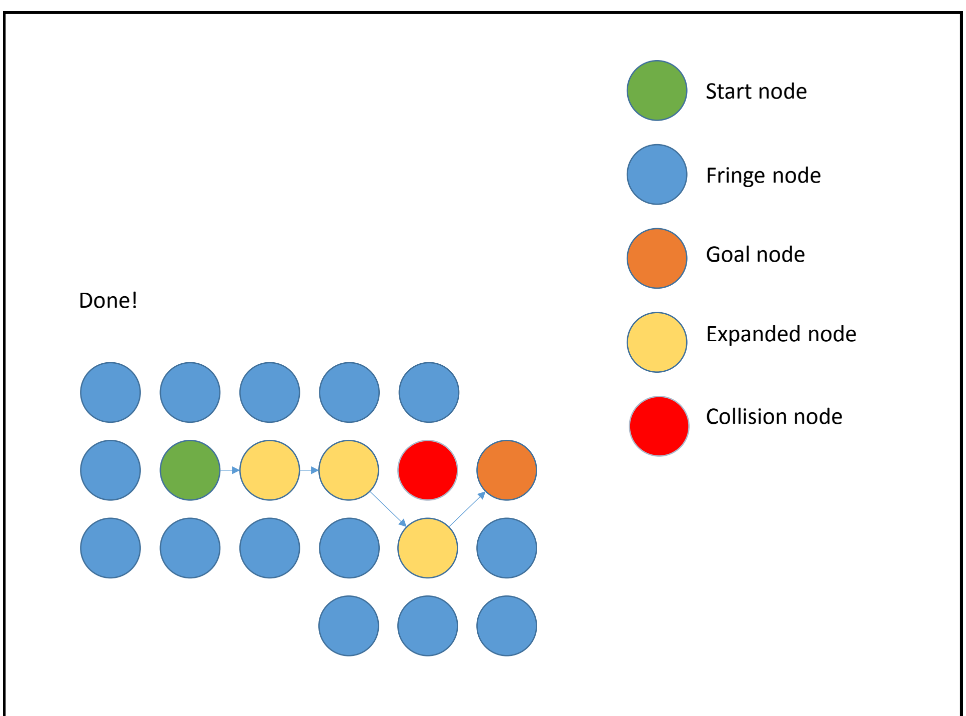





Start node
Fringe node
Goal node
Expanded node


Start node
Fringe node
Goal node
Expanded node
Collision node


Start node
Fringe node
Goal node
Expanded node
Collision node


Done!
Start node
Fringe node
Goal node
Expanded node
Collision node

If grid discretisation isn't good enough:

Instead of grid, randomly sample a bunch of valid states e.g.

Probabilistic Roadmap (PRM)

Rapidly-exploring Random Tree (RRT)

Deciding on which coin to move first

- We used a greedy approach whereby we plan trajectories for every coin and pick the shortest trajectory.
- Estimate new positions of remaining coins (easy as we know turntable rotation speed and time required for the trajectory taken
- Repeat

The Future of Robotics/Automation & Course Review

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 13

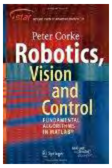
October 26, 2016

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

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	14-Sep	Robot Sensing: Feature Detection
9	21-Sep	Robot Sensing: Multiple View Geometry
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	Probabilistic Robotics: Localization & SLAM
12	19-Oct	Probabilistic Robotics: Planning & Control (State-Space/Shaping the Dynamic Response/LQR)
13	**26-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
- Vision-Based Control
  - §15.1-15.3 (pp. 456-473)

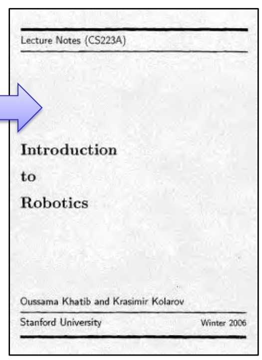
➔ **Review**🡐

- SLAM
  - pp. 123-4 (§6.4-6.5)

- Everything? ☺
  - Many references…

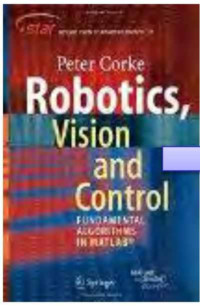
Next Time

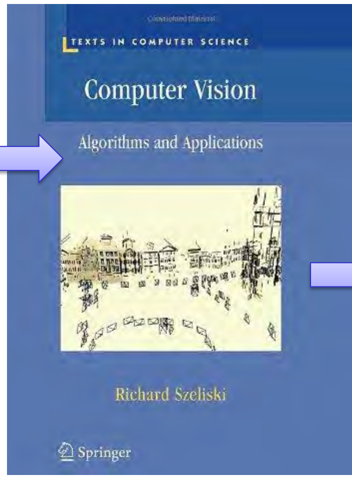
Reference Material

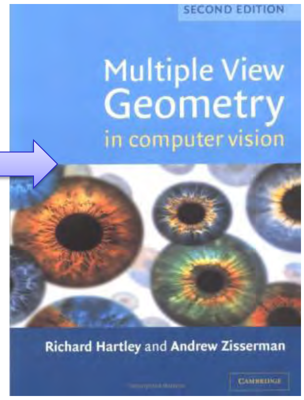



```
On class webpage
Password: metr4202
```

Reference Material



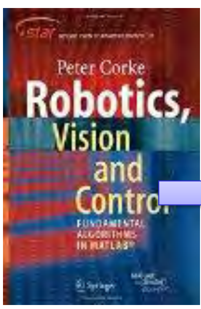


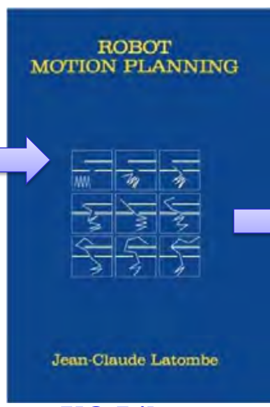


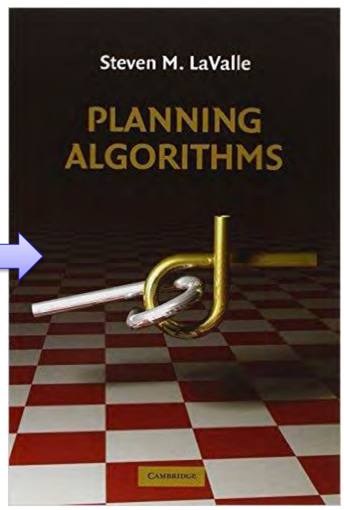
[UQ Library/ SpringerLink]

[UQ Library] (ePDF)

Reference Material







[UQ Library]
(TJ211.4 .L38 1991)

[UQ Library / Online (PDF)]

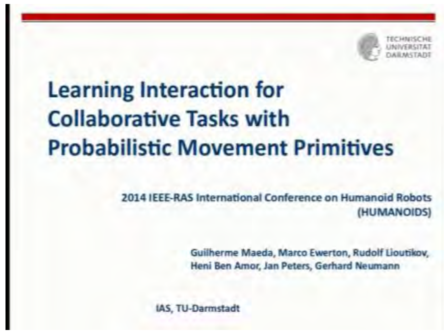
Cool Robotics Share

- Interactive Mobile Robots

- New transmissions

Source: (top) https://youtu.be/hX6pFcfr29c | (bottom) http://spectrum.ieee.org/automaton/robotics/robotics-hardware/sri-demonstrates-abacus-rotary-transmission

HRI: Semi-Autonomous Robots Learning Interaction for Collaborative Tasks



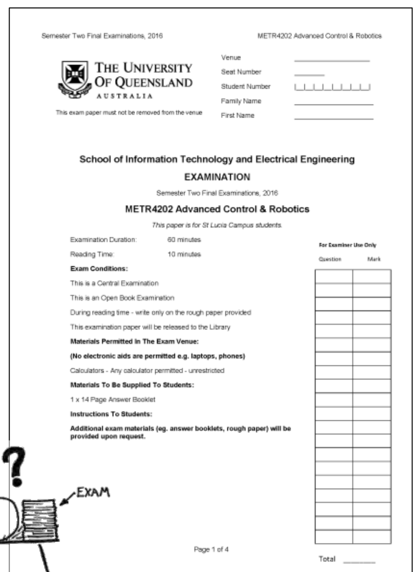
Maeda, G., *et al.* "Learning interaction for collaborative tasks with probabilistic movement primitives." 2014 *IEEE-RAS International Conference on Humanoid Robotics*. IEEE, 2014. https://youtu.be/2Ok6KQQQDNQ

Final Exam!

- 4 Questions | 60 Minutes
- **Open Book**
- Similar in nature to the 2015 Quiz

Topics:

- Position, orientation and location in space
- Robot analysis (forward/Inverse kinematics, recursive Newton-Euler formulations, etc.)
- Sensing geometry (including camera calibration)
- Multiple-view geometry
- Motion planning and control



Semester Two Final Examinations, 2016 METR4202 Advanced Control & Robotics

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

This exam paper must not be removed from the venue

Venue ____
Seat Number ____
Student Number ____
Family Name ____
First Name ____

School of Information Technology and Electrical Engineering

EXAMINATION

Semester Two Final Examinations, 2016

METR4202 Advanced Control & Robotics

This paper is for St Lucia Campus students.

Examination Duration: 60 minutes

Reading Time: 10 minutes

Exam Conditions:

This is a Central Examination

This is an Open Book Examination

During reading time - write only on the rough paper provided

This examination paper will be released to the Library

Materials Permitted In The Exam Venue:

(No electronic aids are permitted e.g. laptops, phones)

Calculators - Any calculator permitted - unrestricted

Materials To Be Supplied To Students:

1 x 14 Page Answer Booklet

Instructions To Students:

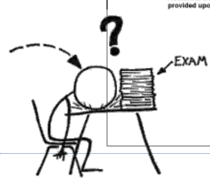
Additional exam materials (eg. answer booklets, rough paper) will be provided upon request.

For Examiner Use Only

Question	Mark

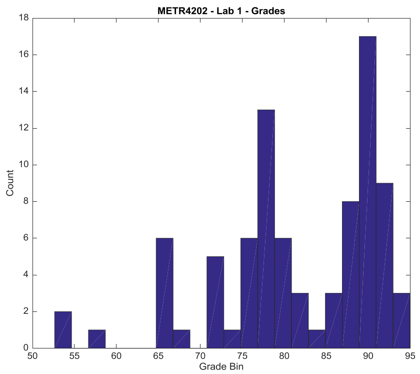
Total ____

Page 1 of 4



Grades!

- Lab 1



- MEAN: 82 | STD: 10
- MIN: 52 | MAX: 95
- MEDIAN: 83.5

- Lab 2

Coming Soon!

PS: For those who have not LAB 2 “PAF-ed” please do so by tonight! ☺

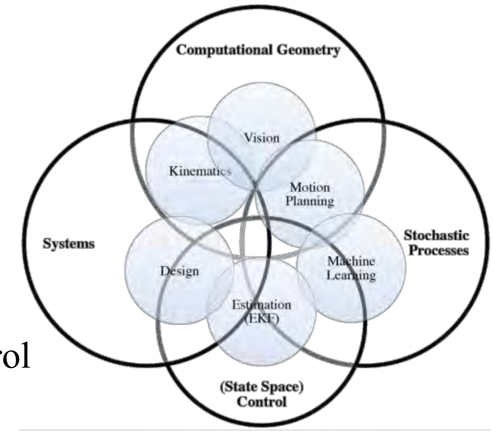
Some 2017 Robotics Thesis Topics

Projects (2017 | RDL | S. Singh)	
ID	**Title**
1	Light Fields in Motion
2	Image Sensing and Control
3	One Sweet Robot
4	Remote Access CT imaging Laboratory for clinical skills education and training
5	Semi-Automatic Tracking of Athletes Diving using Pre-selected Keypoints
7	(RDL*) Dermatology Outback
8	Interactive Ball / Beeper Ball - Smart Tones
9	Affine Breathing: Tracking
10	Underactuated Robotics: Katita Walks The Line
11	Assistive Ultrasound Support
13	SuperResolve 3D [NEW]
14	**Privacy Preserving Roadmap Planning [NEW]**
15	Color My World (Art Meets Robotics) [NEW]
16	**Robots: In Play (Probabilistically) [NEW]**
17	Project with Sound and Hearing and Mechatronics [NEW]
18	**Biomedical Engineering Meets Robotics [NEW] [ARC DP co-funding]**
19	(Virtual) Robotics and Experimental Platform [NEW]
20	BYO Robot Project [NEW]

Learning Objectives

Robotics:
Facets of overarching principles

- Scene Geometry
- Structure | Unstructured
- Adaptive models for control
- Interactions:
  Deterministic | Probabilistic



Estimation

Along multiple dimensions

State Space

- We collect our set of uncertain variables into a vector …
  $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$
- The set of values that **x** might take on is termed the *state space*
- There is a *single* true value for **x**,
  but it is unknown

State Space Dynamics

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

$$H(s) = C(sI - A)^{-1}B$$

Measured versus True

- Measurement errors are inevitable
- So, add Noise to State...
  - State Dynamics be $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + w$
    $\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} + v$
- Can represent this as a "Normal" Distribution

$$\mathcal{N}(x;\mu,\sigma) = \frac{1}{\left(\sqrt{2\pi}\right)\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Recovering The Truth

- Numerous methods
- Termed "Estimation" because we are trying to estimate the truth from the signal
- A strategy discovered by Gauss
- Least Squares in Matrix Representation

$$\begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = \begin{bmatrix} n & \sum_1^n t_i \\ \sum_1^n t_i & \sum_1^n t_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_1^n z_i \\ \sum_1^n t_i z_i \end{bmatrix}$$

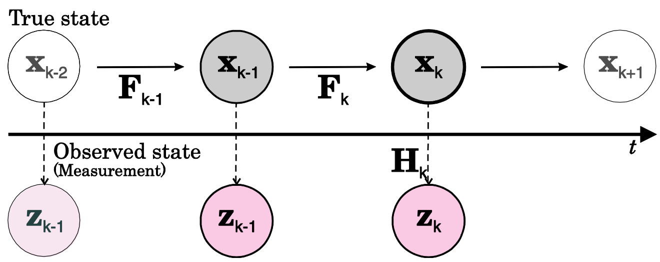
Recovering the Truth: Terminology

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

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v}$$

$\mathbf{x}$: the state vector
$\mathbf{x}_{A|B}$: the state of $\mathbf{x}$ at time A based on data taken up to time B
$\hat{\mathbf{x}}$: estimate of the true state vector
$\mathbf{F}$: system dynamics matrix in continuous time (equivalent to $\mathbf{A}$ in Eq. 1)
$\mathbf{G}$: system control matrix relating deterministic input, $\mathbf{u}$, to the state (equivalent to $\mathbf{B}$ in Eq. 1)
$\mathbf{H}$: measurement matrix in continuous time (equivalent to $\mathbf{C}$ in Eq. 2)
$\mathbf{F}_i$: system model in **discrete** time at $t = t_i$
$\mathbf{H}_i$: measurement model in **discrete** time at $t = t_i$
$\mathbf{P}_i$: estimate covariance in **discrete** time at $t = t_i$
$\mathbf{w}$: process uncertainty (noise) vector (of type $\mathcal{N}(0, s)$)
$\mathbf{Q}$: process noise matrix, $\mathbf{Q} = E\left[ww^{\intercal}\right]$
$\mathbf{Q}_i$: $\mathbf{Q}$ in discrete time at $t = t_i$
$\mathbf{v}$: measurement noise vectors (of type $\mathcal{N}(0, \sigma)$)
$\mathbf{R}_i$: the measurement variance matrix, $\mathbf{R} = E\left[vv^{\intercal}\right]$, in discrete time at $t = t_i$

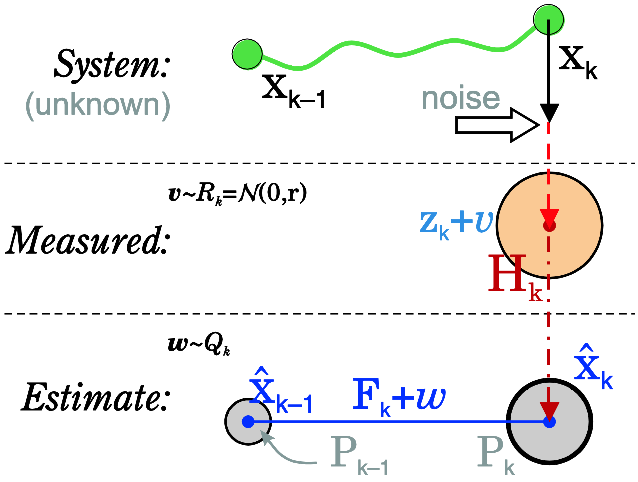
General Problem…



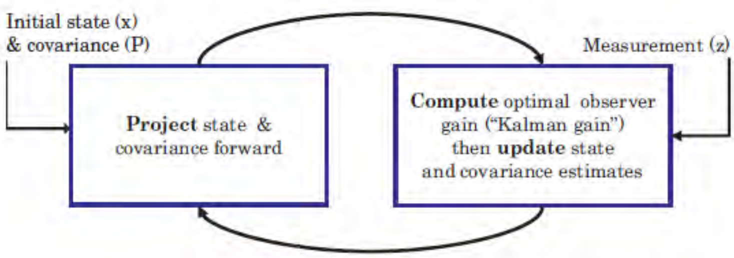
Duals and Dual Terminology

	Estimation		**Control**
Model:	$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x}$ (discrete: $\mathbf{x} = \mathbf{F}_k\mathbf{x}$)	$\leftrightarrow$	$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, $\mathbf{A} = \mathbf{F}^\intercal$
Regulates:	P (covariance)	$\leftrightarrow$	M (performance matrix)
Minimized function:	Q (or $GQG^\intercal$)	$\leftrightarrow$	V
Optimal Gain:	K	$\leftrightarrow$	G
Completeness law:	Observability	$\leftrightarrow$	Controllability

Estimation Process in Pictures



Kalman Filter Process



KF Process in Equations

Prediction:	$\hat{\mathbf{x}}_{k\|k-1} = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1\|k-1},$	(state prediction)
	$\mathbf{P}_{k\|k-1} = \mathbf{Q}_{k-1} + \mathbf{F}_{k-1}\mathbf{P}_{k-1\|k-1}\mathbf{F}^{\mathsf{T}}{}_{k-1},$	(covariance prediction)
Kalman Gain:	$\mathbf{K}_k = \mathbf{P}_{k\|k-1}\mathbf{H}^{\mathsf{T}}[\mathbf{H}\mathbf{P}_{k\|k-1}\mathbf{H}^{\mathsf{T}} + \mathbf{R}_k]^{-1},$	
Update:	$\mathbf{P}_{k\|k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}]\mathbf{P}_{k\|k-1},$	(covariance update)
	$\hat{\mathbf{x}}_{k\|k} = \hat{\mathbf{x}}_{k\|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k\|k-1})$	(state update)

METR 4202: **Robotics** October 26, 2016 - 23

KF Considerations

$$\underbrace{\hat{\mathbf{x}}_{k|k-1}}_{n\times 1} = \underbrace{\mathbf{F}_{k-1}}_{n\times n}\hat{\mathbf{x}}_{k-1|k-1} + \underbrace{\mathbf{G}_{k-1}}_{n\times j}\underbrace{\mathbf{u}_{k-1}}_{j\times 1}$$

$$\underbrace{\mathbf{P}_{k|k-1}}_{n\times n} = \underbrace{\mathbf{Q}_{k-1}}_{n\times n} + \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}^{\mathsf{T}}{}_{k-1}$$

$$\underbrace{\mathbf{K}_k}_{n\times m} = \mathbf{P}_{k|k-1}\underbrace{\mathbf{H}^{\mathsf{T}}}_{n\times m}\underbrace{[\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^{\mathsf{T}} + \mathbf{R}_k]^{-1}}_{m\times m}$$

$$\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}]\mathbf{P}_{k|k-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\underbrace{\mathbf{z}_k}_{m\times 1} - \underbrace{\mathbf{H}}_{m\times n}\hat{\mathbf{x}}_{k|k-1} - \mathbf{H}\mathbf{G}_k\mathbf{u}_{k-1})$$

METR 4202: **Robotics** October 26, 2016 - 24

Ex: Kinematic KF: Tracking

- Consider a System with Constant Acceleration

$$\ddot{y} = -g$$
$$\dot{y} = gt + p_1$$
$$y = p_0 + p_1 t + \frac{gt^2}{2}$$

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ g \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{F_k} = \begin{bmatrix} 0 & t_s & \frac{t_s{}^2}{2} \\ 0 & 0 & t_s \\ 0 & 0 & 0 \end{bmatrix}$$

$$\hat{\mathbf{x}}_k = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1})$$

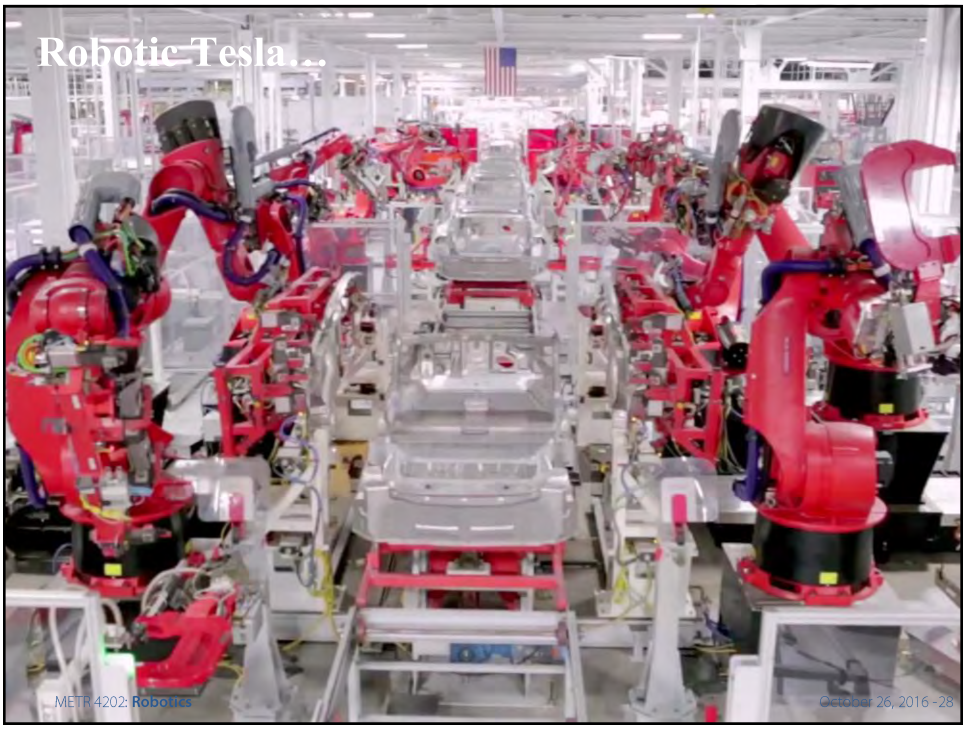
In Summary

- KF:
  - The true state (*x*) is separate from the measured (*z*)
  - Lets you **combine** prior controls knowledge with measurements to filter signals and find the <u>truth</u>
  - It **regulates** the covariance (P)
    - As P is the scatter between *z* and *x*
    - So, if P → 0, then *z* → *x* (measurements → truth)
- EKF:
  - Takes a Taylor series approximation to get a local “F” (and “G” and “H”)

Future of Robotics

(Self-Driving Vehicles)

(Notes from Prof. John Leonard, MIT)



Other Robotic Tesla…

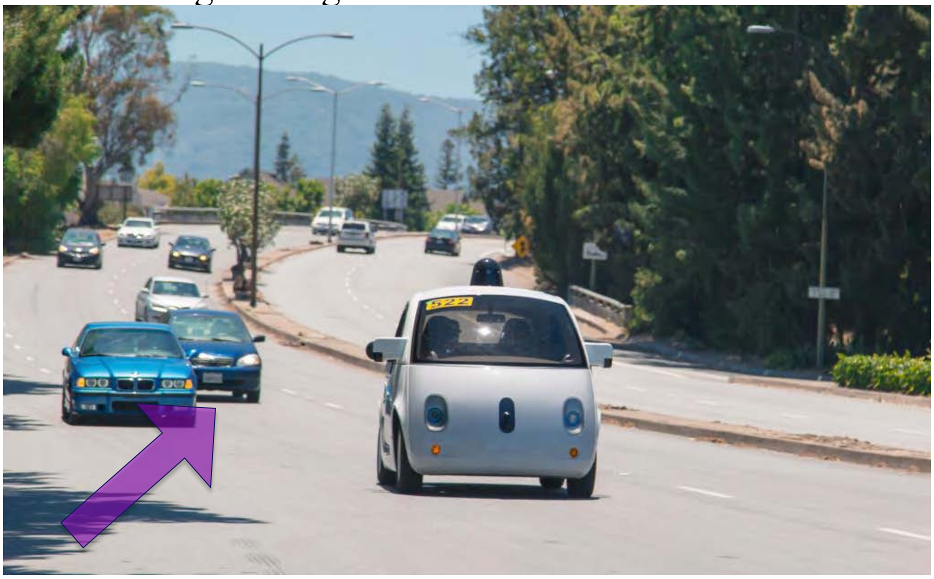
METR 4202: **Robotics**   October 26, 2016 -29

Cars: Software/Robots With 4 Wheels

METR 4202: **Robotics**   October 26, 2016 -30

Robotics & Automation Has Limits Too

More to Dynamic "Obstacles" than one's own Control… Ethics in Engineering



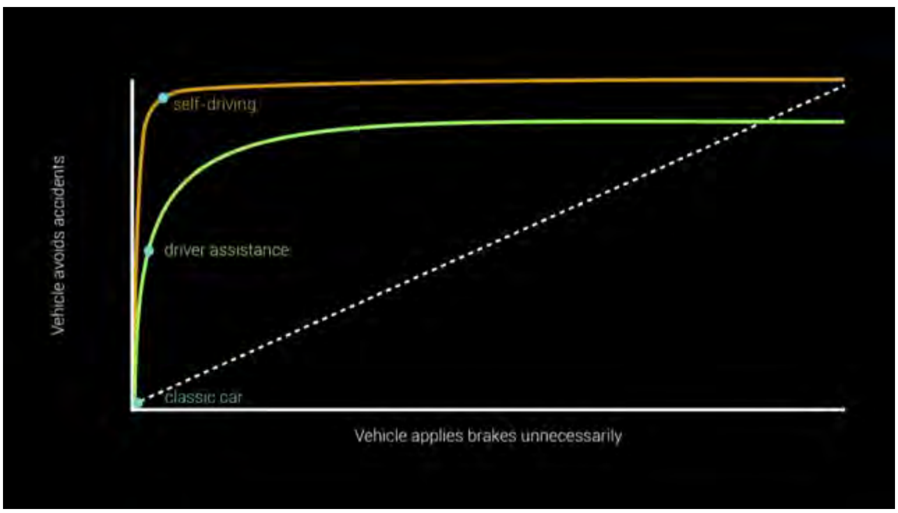
Q: Why has Google has chosen to exclusively pursue level 4?

A: They don't trust people to pay attention

t=41.56: "people do really stupid stuff when they are driving...it isn't pretty. The assumption that humans can be a reliable backup for the system was a total fallacy. Once people trust the system, they trust it"

Astro Teller, Head of GoogleX, March 2015

"Vehicle Avoids Accidents" vs. "Vehicles Applies Breaks Unnecessarily"



Chris Urmson Keynote Address at the Intelligent Transportation Systems 25th Annual Meeting & Expo, Pittsburgh, May 2015

THE WALL STREET JOURNAL.

Google Brings in Chief for Self-Driving Cars

Hiring of Detroit veteran John Krafcik indicates keenness to commercialize the technology

Google has hired Detroit veteran John Krafcik to run its self-driving car division, sending a message that it is serious about the commercial viability of the autonomous vehicles business. Photo: Zuma Press

By ALISTAIR BARR And MIKE RAMSEY

49 COMMENTS

Sept. 13, 2015 11:04 p.m. ET

METR 4202: Robotics

September 13, 2015

"When the company wanted a team of roboticists, it raided a University Lab to get them. Can high-tech academia survive today's Silicon Valley talent binge?"

METR 4202: Robotics

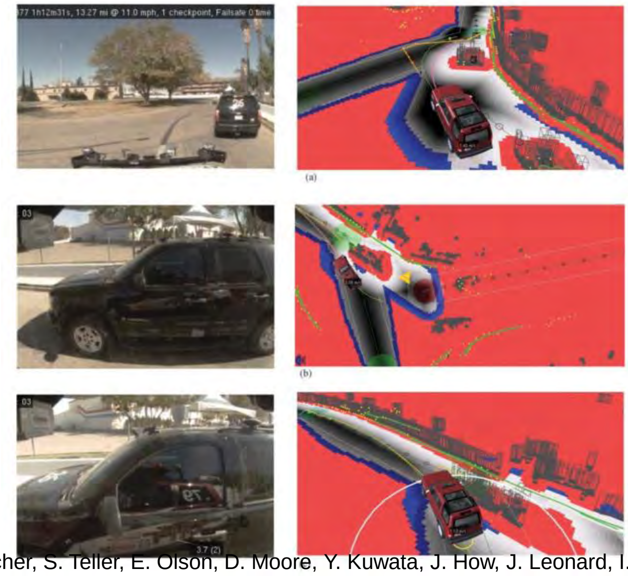
by Clive Thompson, New York Times, September 11, 2015

MIT DARPA Urban Challenge Team (2006-2007)

Leonard et al., JFR 2008 ; Karaman and Frazzoli, IJRR 2011; Huang et al., AR 2009

2007 DARPA Urban Challenge – Collision between MIT and Cornell

2007 DARPA Urban Challenge – Collision between MIT and Cornell



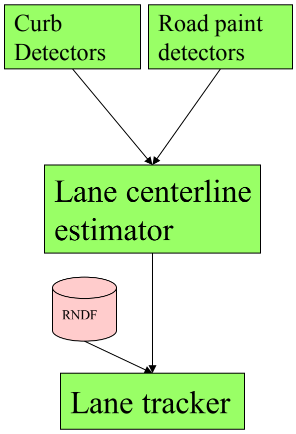
L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, and others, "The MIT–Cornell collision and why it happened." In Journal of Field Robotics, 25(10), pages 775-807. 2008.

From Prof. Ed Olson (Umich): The logic of whether to represent an "obstacle" as a track (i.e., something with velocity) or as a blob, was this (relevant part is highlighted):

```
int use_track = 0, use_rects = 1;
//        if (t->vmag > 4)
//            use_rects = 0;

    if (t->vmag > 3.0 && t->maturity > 8)
            use_track = 1;
    double MAX_DIM = 10;
    if (t->box.size[0] > MAX_DIM || t->box.size[1] > MAX_DIM)
            use_track = 0;
```

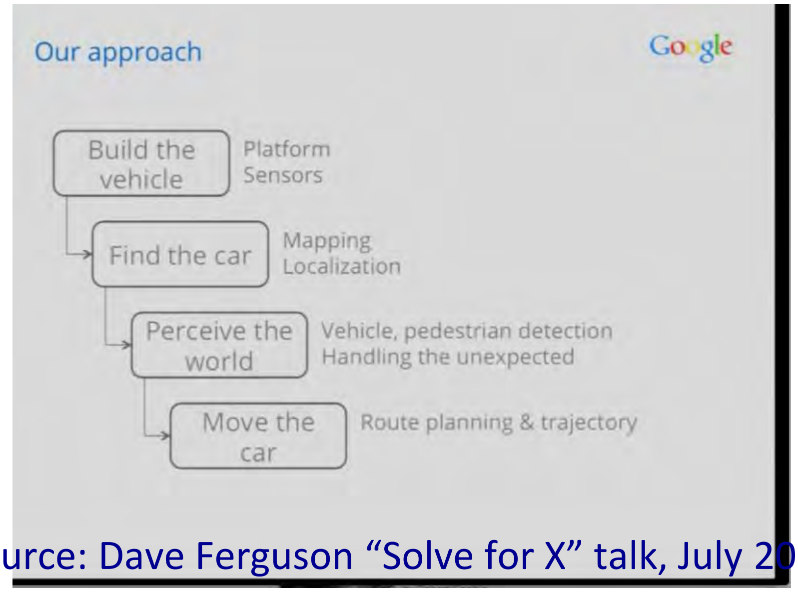
Lane Estimation (PhD Thesis of Albert Huang, supervised by Prof. Seth Teller)



2015: Self-Driving Vehicles Have a Perception Problem

- The Google Car is an amazing research project that might one day transform mobility
- The technology of the Google Car, however, has been over-hyped and is poorly misunderstood
- This has led many people to say that self-driving is a "solved" problem
- "Just because it works for Google", doesn't mean it will work for everyone else

How Does Google's Self Driving Car Work?



Source: Dave Ferguson "Solve for X" talk, July 2013

http://www.youtube.com/watch?v=KA_C6OpL_Ao

Google: Lidar Localization with an a priori map

https://plus.google.com/+GoogleSelfDrivingCars/videos

SDVs: The Big Questions Going Forward

- Technical Challenges:
- Maintaining Maps
- Adverse Weather
- Interacting with People
- Robust Computer Vision (towards PD=1.0, PFA = 0.0)?

SDVs: The Big Questions Going Forward

- Technical Challenges:
- Maintaining Maps
- Adverse Weather
- Interacting with People
- Robust Computer Vision (towards PD=1.0, PFA = 0.0)?
- The big question for Level 3 approaches? (i.e., Musk)
- Can humans be trusted to take control when necessary?

SDVs: The Big Questions Going Forward

- Technical Challenges:
- Maintaining Maps
- Adverse Weather
- Interacting with People
- Robust Computer Vision (towards PD=1.0, PFA = 0.0)?
- The big question for Level 3 approaches? (i.e., Musk)
- Can humans be trusted to take control when necessary?
- The big question for Level 4 approaches? (i.e., Urmson)
- Can near-perfect ROC curves be obtained in a wide variety of demanding settings?

SDVs: The Big Questions Going Forward

- Technical Challenges:
- Maintaining Maps
- Adverse Weather
- Interacting with People
- Robust Computer Vision (towards PD=1.0, PFA = 0.0)?
- The big question for Level 3 approaches? (i.e., Musk)
- Can humans be trusted to take control when necessary?
- The big question for Level 4 approaches? (i.e., Urmson)
- Can near-perfect ROC curves be obtained in a wide variety of demanding settings?
- Level 2.99 – Hidden Autonomy (Human must pay attention, but autonomy will jump in to prevent accidents)

Summary – Self-Driving Vehicles

- Transformative technology that can/will change the world, but many open questions
- Hope for reducing accidents and saving lives
- Admiration for Google's audacious vision and amazing progress
- Impressed by recent efforts by auto manufacturers
- Pride for the robotics community's contributions
- Fear that the technology is being over-hyped
- Uncertainty about open technological challenges, such as:
  – left-turn across high-speed traffic onto busy roads
  – Interpretation of gestures by traffic cops, crossing guards etc
  – Effect of changes in road surface appearance on map-based localization
  – Capability to "predict what will happen next" in demanding situations
  – Operations in adverse weather

Future of Robotics

Move Heaven & Earth

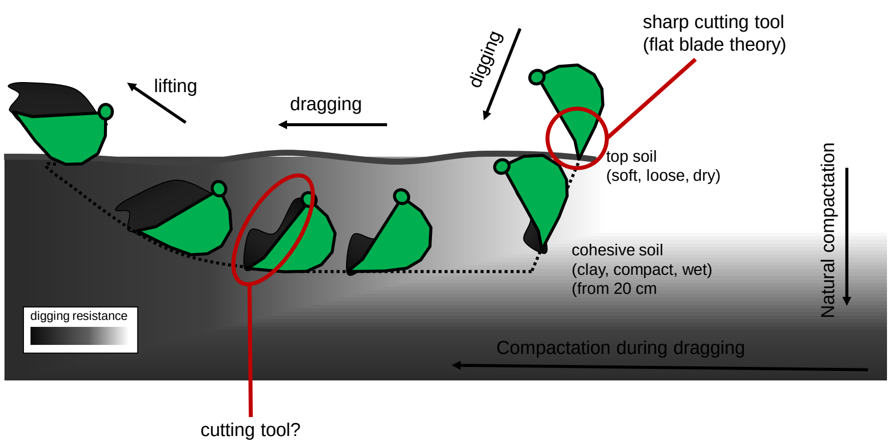
"Field Arm" Motion Generation

METR 4202: **Robotics** October 26, 2016 -51

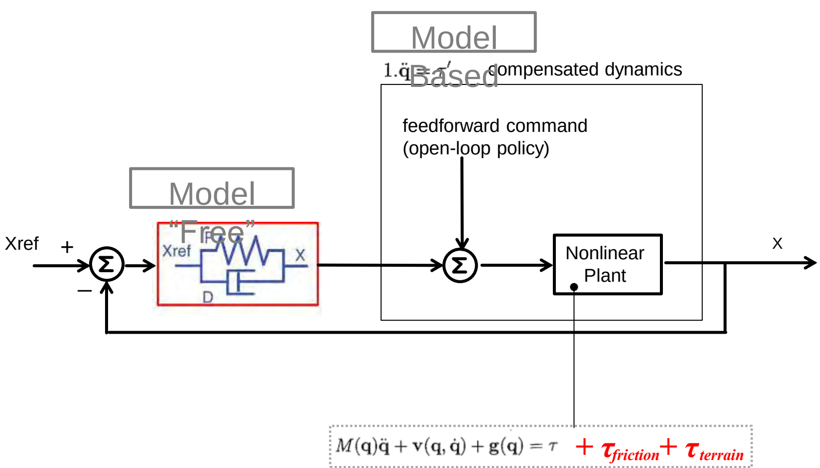
Terrain is Not "Structured," But It's Not Random…

METR 4202: **Robotics** October 26, 2016 -52

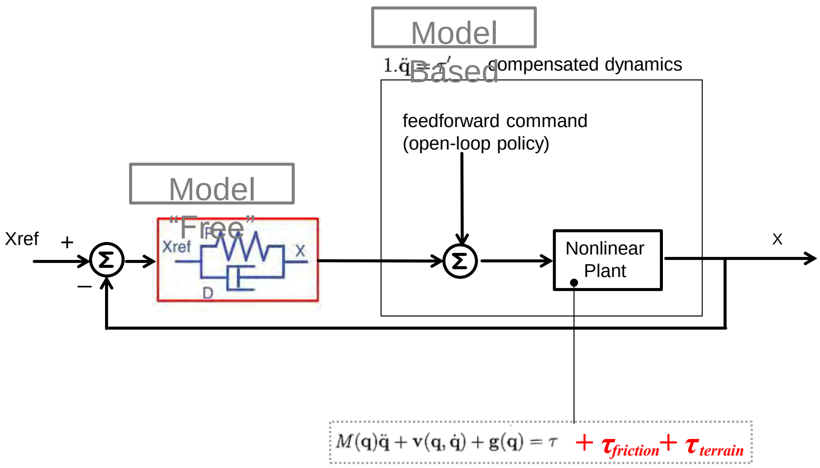
Excavation as Terrain Manipulation



Operation Space (Computed Torque) (2 DOF Example)

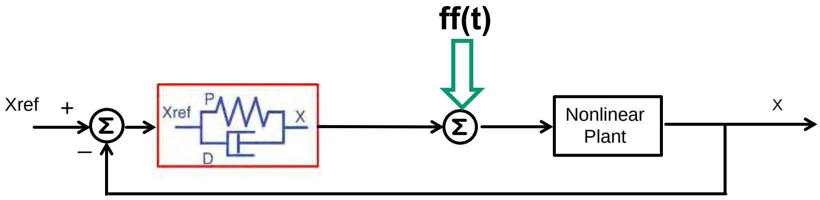


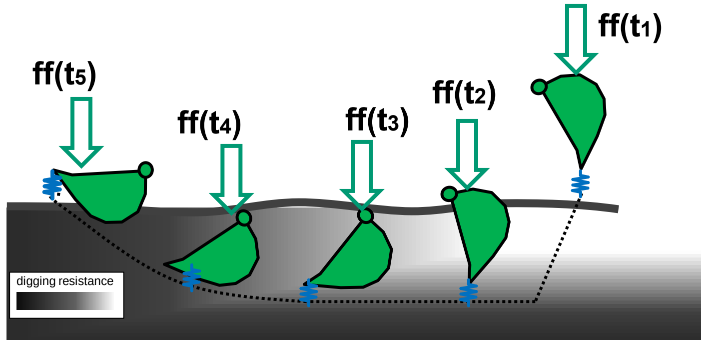
Operation Space (Computed Torque) (2 DOF Example)



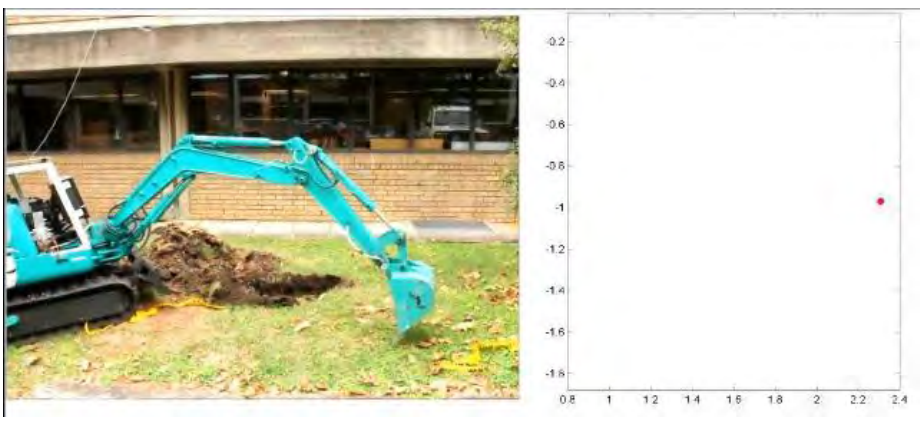
Reminder: Compensated Manipulation

Thus for Excavation …





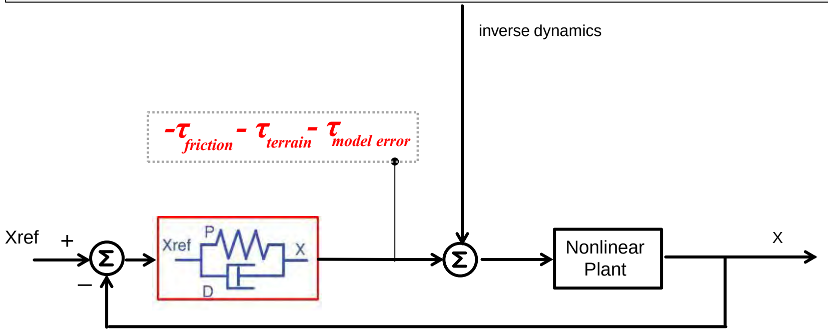
Manipulation under Large Disturbances



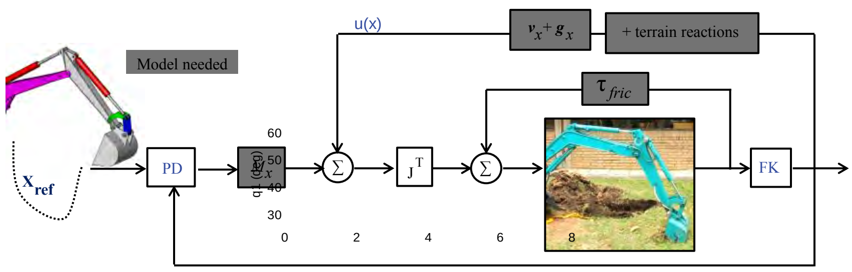
Inverse Dynamics is Not Trivial

Inverse dynamics helps, but performance is dependent on model structure and parameter identification

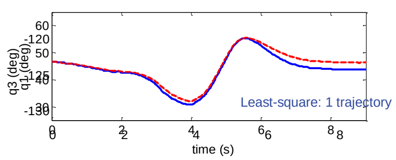
```
z = [...
    d11*q1dot - u1 - q3dot*(L2*L3c*m3*q3dot*sin(q3 + qcm3) + L1*L3c*m3*q3dot*sin(q2 + q3 + qcm3)) - q1dot*(2*L1*L2c*m2*q2dot*sin(q2 + qcm2) + 2*L2*L3c*m3*q3dot*sin(q3 + qcm3) + 2*L1*L2*m3*q2dot*sin(q2) +
2*L1*L3c*m3*q2dot*sin(q2 + q3 + qcm3) + 2*L1*L3c*m3*q3dot*sin(q2 + q3 + qcm3)) - q2dot*(L1*L2c*m2*q2dot*sin(q2 + qcm2) + 2*L2*L3c*m3*q3dot*sin(q3 + qcm3) + L1*L2*m3*q2dot*sin(q2) + L1*L3c*m3*q2dot*sin(q2 + q3 +
qcm3) + 2*L1*L3c*m3*q3dot*sin(q2 + q3 + qcm3)) - ((q1dot - q1dot_k)*(m1*L1c^2 + Izz1 + Izz2 + Izz3 + m2*(L1^2 + 2*cos(q2 + qcm2)*L1*L2c + L2c^2) + m3*(L1^2 + 2*cos(q2)*L1*L2 + 2*cos(q2 + q3 + qcm3)*L1*L3c + L2^2 +
2*cos(q3 + qcm3)*L2*L3c + L3c^2)))/dt - ((Izz3 + L3c*m3*(L3c + L2*cos(q3 + qcm3) + L1*cos(q2 + q3 + qcm3)))*(q3dot - q3dot_k))/dt + g*m3*(L3c*cos(q1 + q2 + q3 + qcm3) + L2*cos(q1 + q2) + L1*cos(q1)) - ((q2dot - q2dot_k)*(Izz2 +
Izz3 + m3*(L2^2 + 2*cos(q3 + qcm3)*L2*L3c + L1*cos(q2)*L2 + L3c^2 + L1*cos(q2 + q3 + qcm3)*L3c) + L2c*m2*(L2c + L1*cos(q2 + qcm2))))/dt + g*m2*(L1*cos(q1) + L2c*cos(q1 + q2 + qcm2)) + L1c*g*m1*cos(q1 + qcm1)
    d22*q2dot - u2 + q1dot*(L1*L2c*m2*q1dot*sin(q2 + qcm2) - 2*L2*L3c*m3*q3dot*sin(q3 + qcm3) + L1*L2*m3*q1dot*sin(q2) + L1*L3c*m3*q1dot*sin(q2 + q3 + qcm3)) + g*m3*(L3c*cos(q1 + q2 + q3 + qcm3) + L2*cos(q1 + q2)) -
((q2dot - q2dot_k)*(m2*L2c^2 + Izz2 + Izz3 + m3*(L2^2 + 2*cos(q3 + qcm3)*L2*L3c + L3c^2)))/dt - ((q3dot - q3dot_k)*(Izz3 + L3c*m3*(L3c + L2*cos(q3 + qcm3))))/dt - ((q1dot - q1dot_k)*(Izz2 + Izz3 + m3*(L2^2 + 2*cos(q3 +
qcm3)*L2*L3c + L1*cos(q2)*L2 + L3c^2 + L1*cos(q2 + q3 + qcm3)*L3c) + L2c*m2*(L2c + L1*cos(q2 + qcm2))))/dt + L2c*g*m2*cos(q1 + q2 + qcm2) - L2*L3c*m3*q3dot^2*sin(q3 + qcm3) - 2*L2*L3c*m3*q2dot*q3dot*sin(q3 + qcm3)
    q1dot*(L2*L3c*m3*q1dot*cos(q3)*sin(qcm3) + L2*L3c*m3*q1dot*cos(qcm3)*sin(q3) + 2*L2*L3c*m3*q2dot*cos(q3)*sin(qcm3) + 2*L2*L3c*m3*q2dot*cos(qcm3)*sin(q3) + L1*L3c*m3*q1dot*cos(q2)*cos(q3)*sin(qcm3) +
L1*L3c*m3*q1dot*cos(q2)*cos(qcm3)*sin(q3) + L1*L3c*m3*q1dot*cos(q3)*cos(qcm3)*sin(q2) - L1*L3c*m3*q1dot*sin(q2)*sin(q3)*sin(qcm3)) - u3 + d33*q3dot + q2dot*(L2*L3c*m3*q2dot*cos(q3)*sin(qcm3) +
L2*L3c*m3*q2dot*cos(qcm3)*sin(q3)) - ((Izz3 + L3c*m3*(L3c + L2*cos(q3 + qcm3) + L1*cos(q2 + q3 + qcm3)))*(q1dot - q1dot_k))/dt - ((q2dot - q2dot_k)*(Izz3 + L3c*m3*(L3c + L2*cos(q3 + qcm3))))/dt - ((m3*L3c^2 + Izz3)*(q3dot -
q3dot_k))/dt + L3c*g*m3*cos(q1 + q2 + q3 + qcm3)
    ];
```



Local Compensation

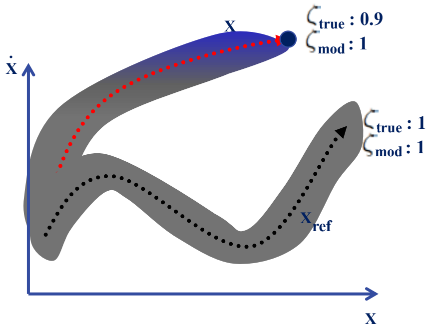


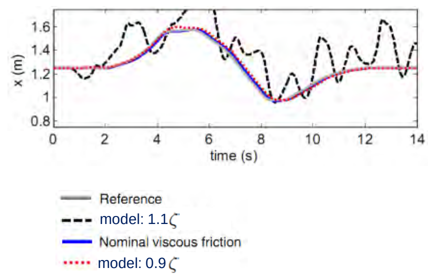
- Modelling globally is hard
- Local models are easy, but can destabalize



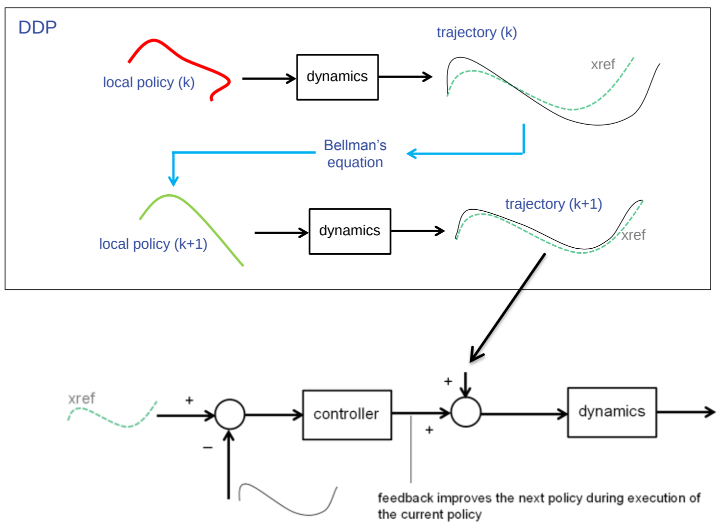
Local model + Disturbance = Wrong Compensation

- Unless tracking is perfect, a state that is far from $\mathbf{x}_{\mathbf{ref}}$ will require a different model





Model Updating & Iterative Tracking…



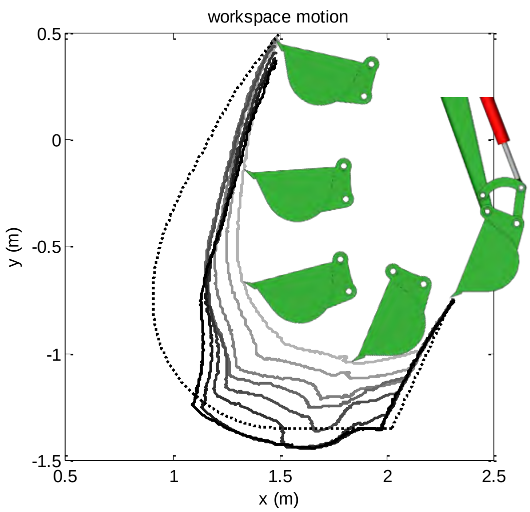
Pay Dirt!

Clay friction

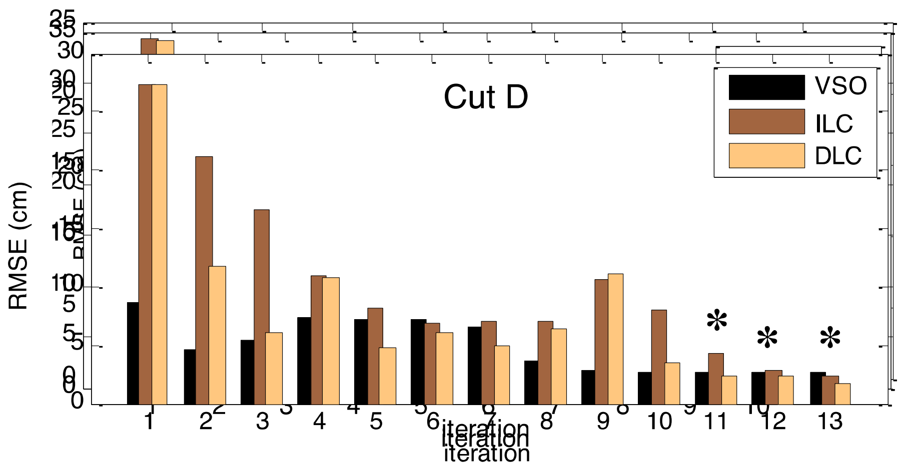
Broken red bricks

Clay friction

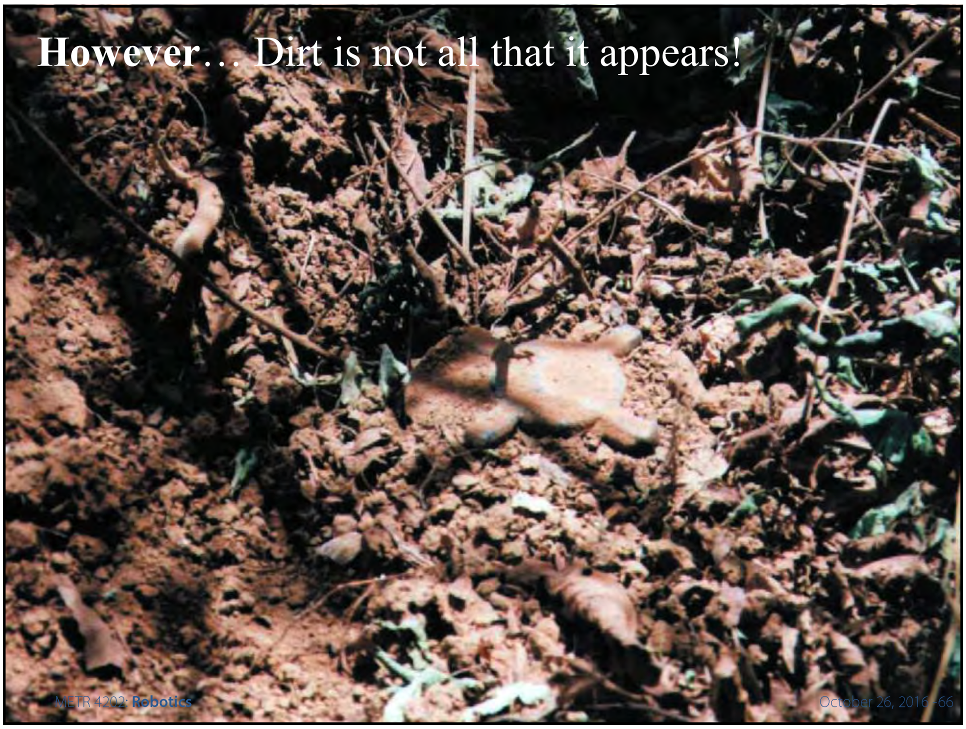
Pay Dirt:
Looking at Trajectories



Pay Dirt:
Looking at Trajectories



However… Dirt is not all that it appears!



Future of Robotics

Medical Robotics

Conclusion and Future Research Challenges

"Soft" robots yield "hard" problems

Goals:

- My dream is to achieve dynamic motion,
  particularly of compliant systems under feedback.
- To *adapt* & *learn* in highly dynamic environments
- Can we robustly integrate continuous planning/control with continuum mechanics to extend our reach

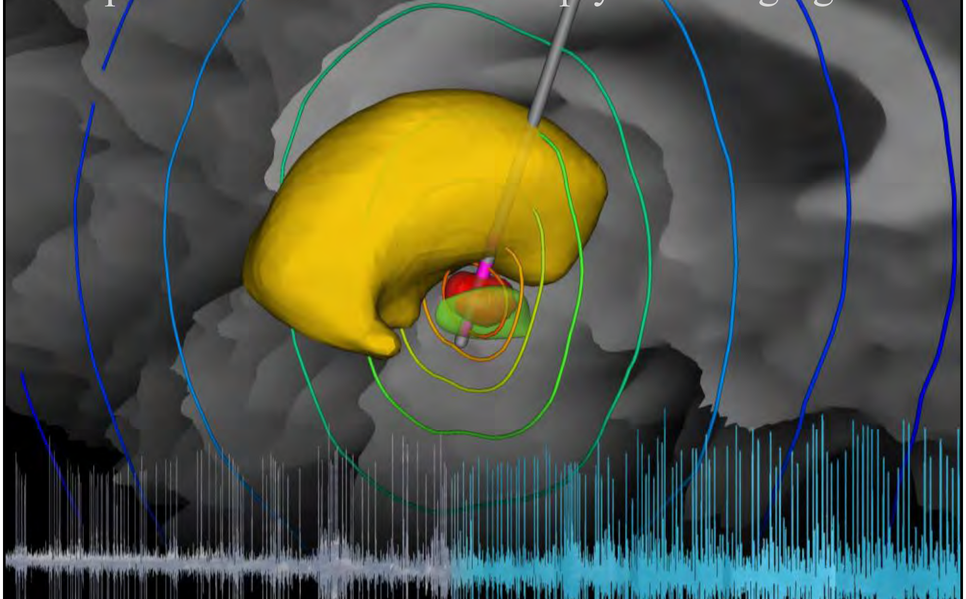
Open Questions:

- Robustness – we would love to have guarantees of performance, but we do not have them for most approaches
- Representation – how can we integrate many different types?
- We need dynamic understanding and robust control
  (recent work in computer vision/machine learning is exciting,
  but current precision-recall curves indicate we have a long way to go)

Clinically-motivated applications:

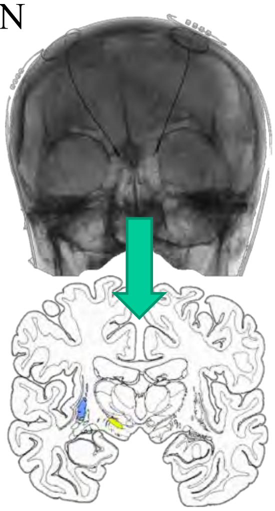
- **Surgical robotics** and **guided therapeutic techniques**

Deep Brian Stimulation is Deeply Challenging

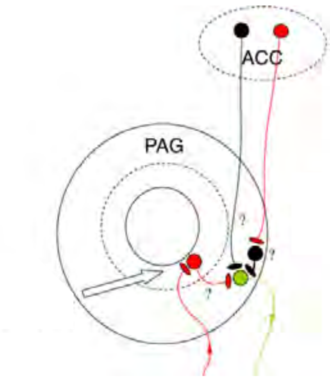


Accuracy is *sine qua non*

- STN



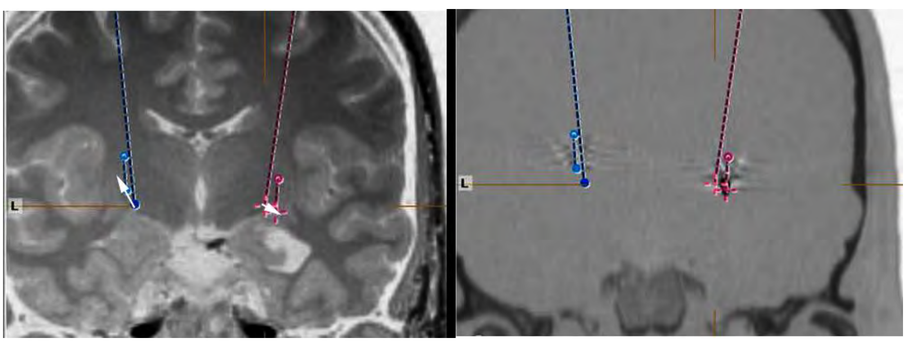
- PAG



Source: Nauta, Feirtag, and Donner, *Fundamental Neuroanatomy*, Freeman, 1986

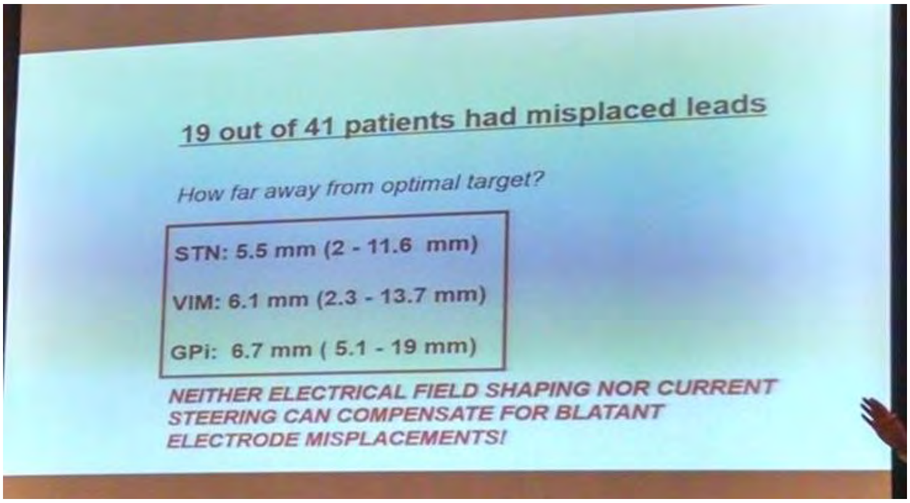
Accuracy is *sine qua non*

- Accuracy of Frame Based Stereotactic Placement via CT/MRI Comparison



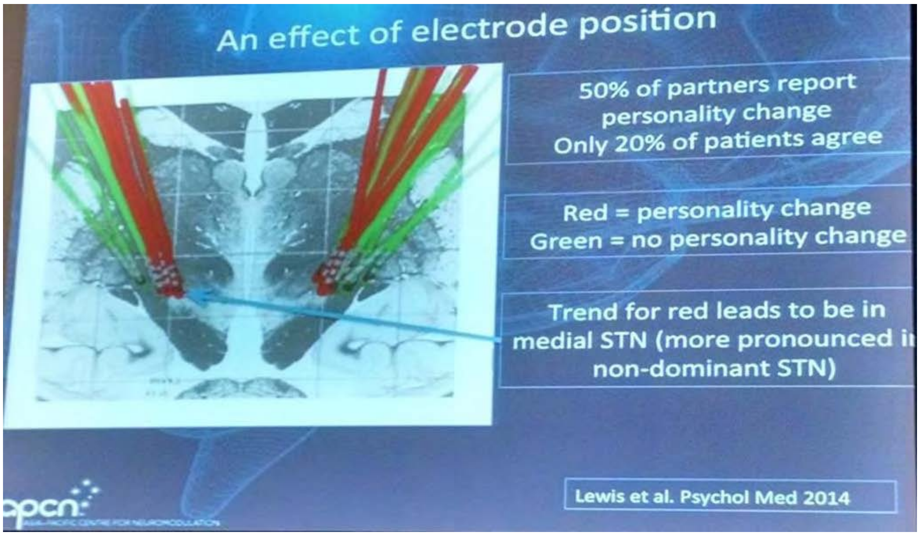
Source: Holloway, Docef, *Neurosurgery* 72[ONS Suppl 1]:ons47–ons57, 2013

DBS Targeting is Hard



METR 4202: **Robotics** October 26, 2016 -73

It has consequences…

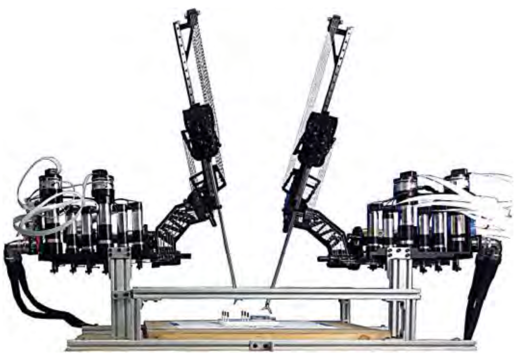


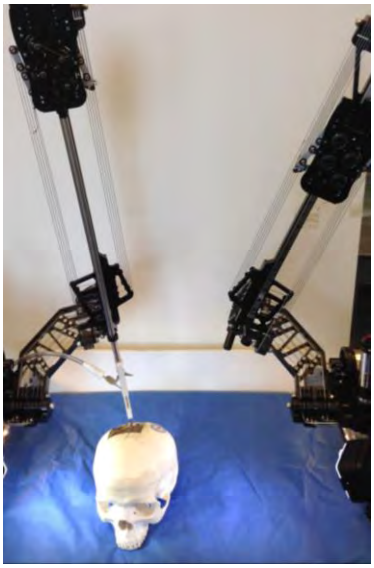
METR 4202: **Robotics** October 26, 2016 -74

Computer Aided Surgery: Lots of Potential

- Unstructured environment (patient tissue) makes this harder

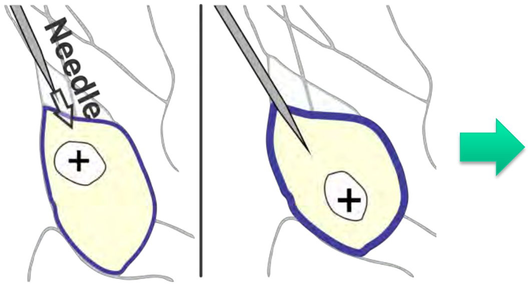
Neurosurgical Robotics

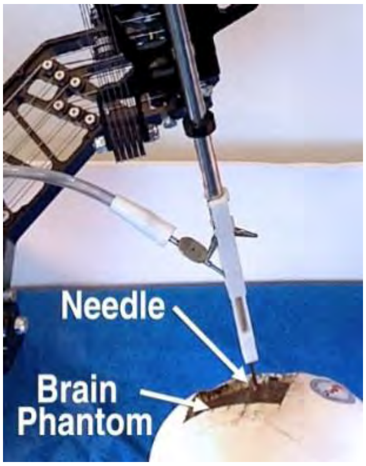




Neurosurgical Robotics:

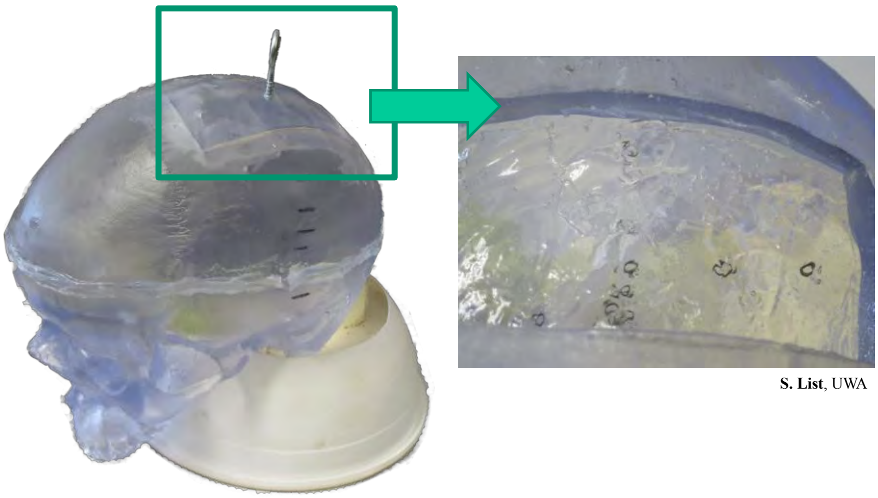
- **Biomechanics approach**:
  Predict expected tissue trajectories





ARC DP160100714

"**Soft**" is "**Hard**" (but not impossible)

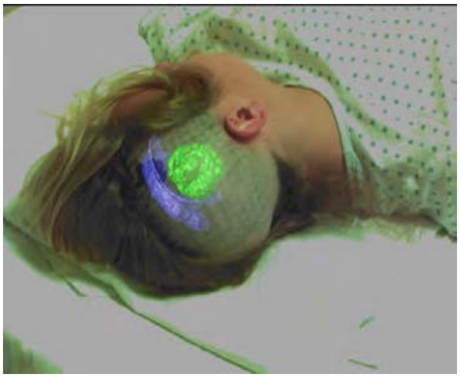


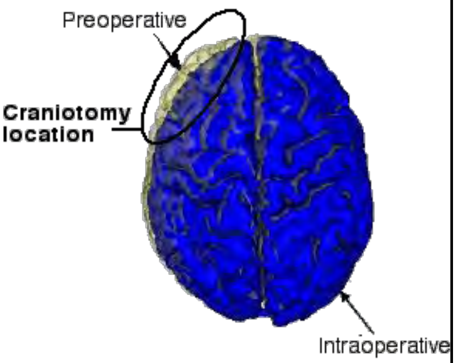
S. **List**, UWA

➔ Many Issues: Including Craniotomy Induced Brain Shift

Soft Tissue Mechanics: Brain Shift/Brain Sag

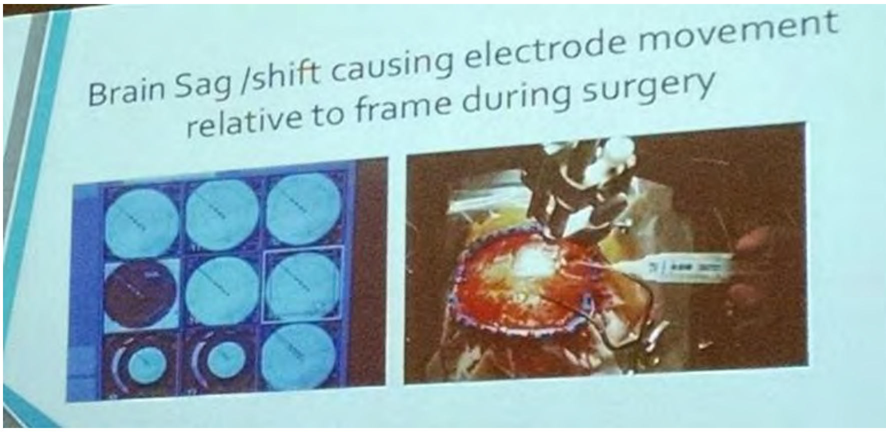
Ex: Image-guided neurosurgery



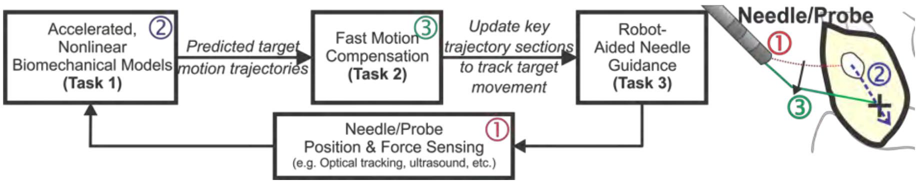


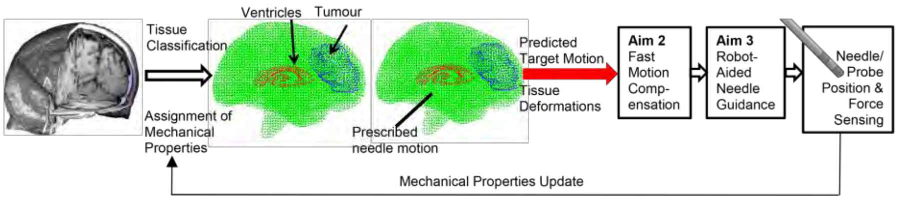
Courtesy: SPL, Harvard

Brain Shift Identified in Neurosurgery Community



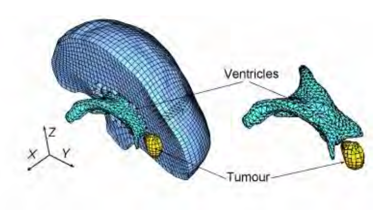
A Robotic "Plan": Handling Brian Shift

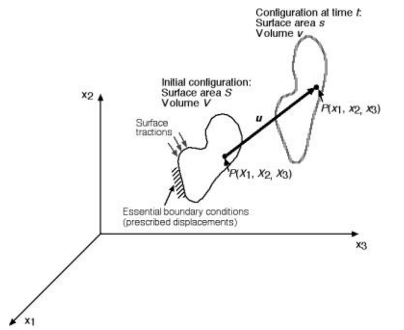




Treating Brain Shift Mechanically

- Post this as a Biomechanics Problem
- Non-linear Continuum Mechanics Problem





$$\int_V \tau_{ij}\delta\varepsilon_{ij}dV = \int_V f_i^B \delta u_i dV + \int_S f_i^S \delta u_i dS$$

Qualitative Evaluation – Canny Edges

Biomechanics **BSpline**

Red contours – Intra-operative

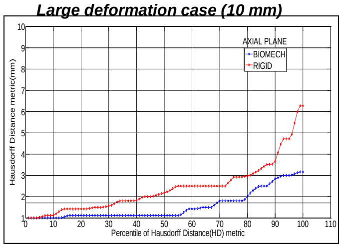
Blue contours – Warped pre-operative

Green contours – Overlap

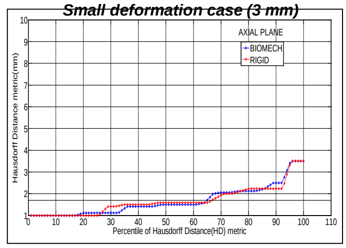
Mostayed et al. (2013) *Annals of Biomed. Eng.* 41(11), 2409-2425

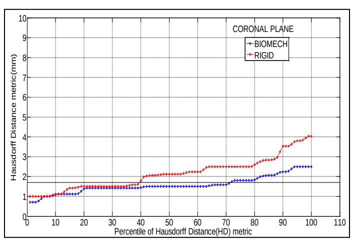
Comparison:Hausdorff Distance Metric

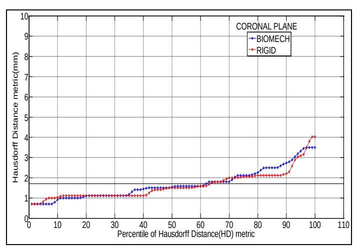
Large deformation case (10 mm)



Small deformation case (3 mm)

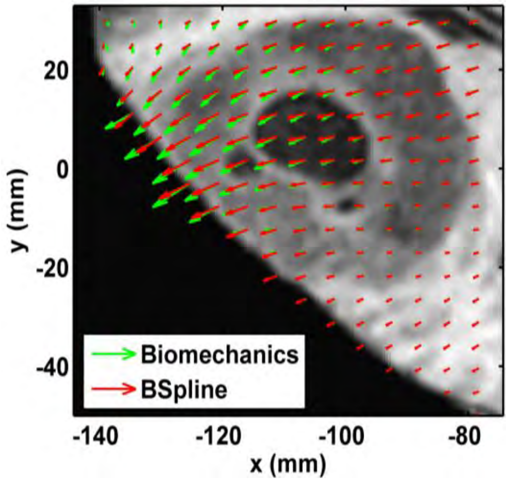






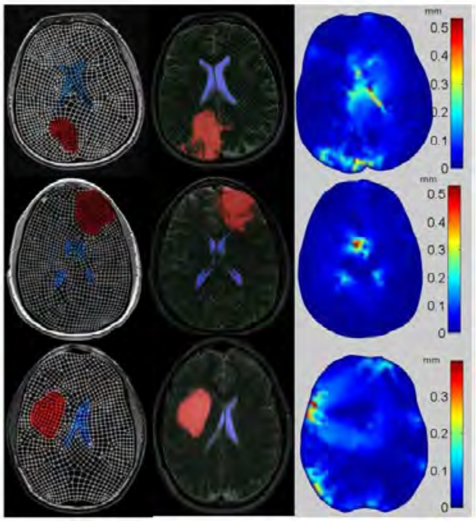
- Comparison of Biomechanics-based & rigid registrations

Qualitative Evaluation: Deformation Field



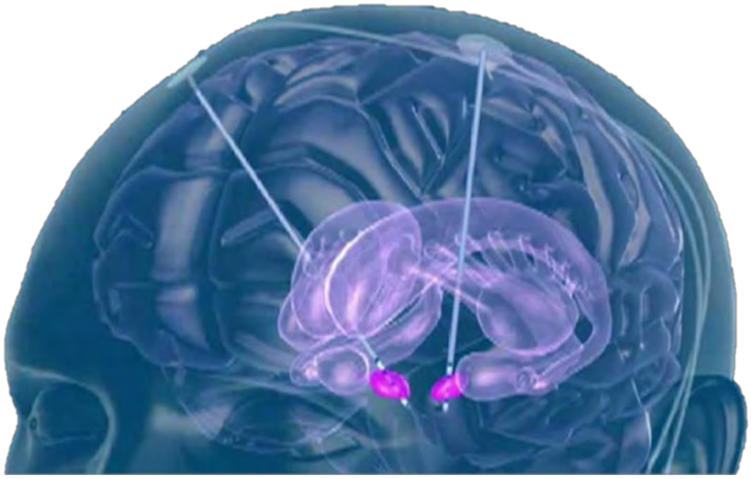
Mostayed et al. (2013) Biomechanical Model as a Registration Tool for Image-Guided Neurosurgery: Evaluation Against BSpline Registration. *Annals of Biomedical Engineering. 41(11), 2409-2425*

Accuracy for Mesh-free Models



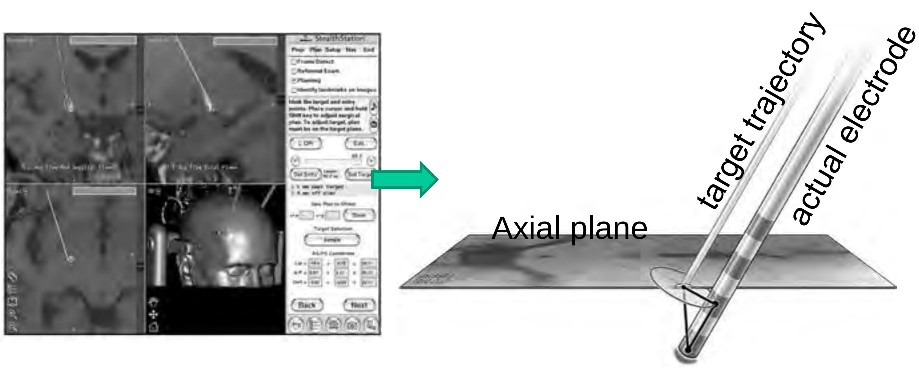
***Left*: Finite Element Models** ***Middle*: Fuzzy Mesh-free Model** ***Right* : Difference of the simulation results**

Targeting:
We Already Do Careful Preoperative Planning



METR 4202: **Robotics** October 26, 2016 -89

The Best Laid Plans …

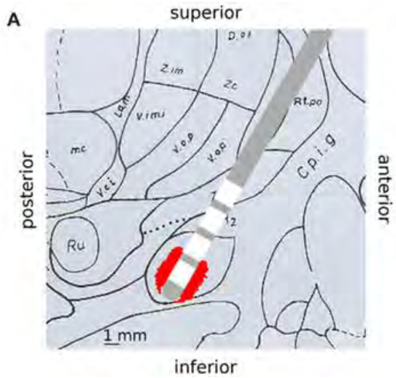


Source: Burchiel, McCartney, *et al.*, *J Neurosurg* **119**:301–306, 2013

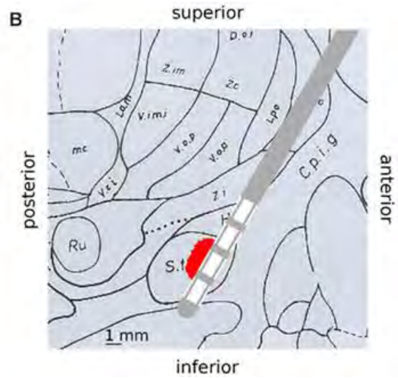
METR 4202: **Robotics** October 26, 2016 -90

Does This Suggest Steering/Path Correction?

- Plan:

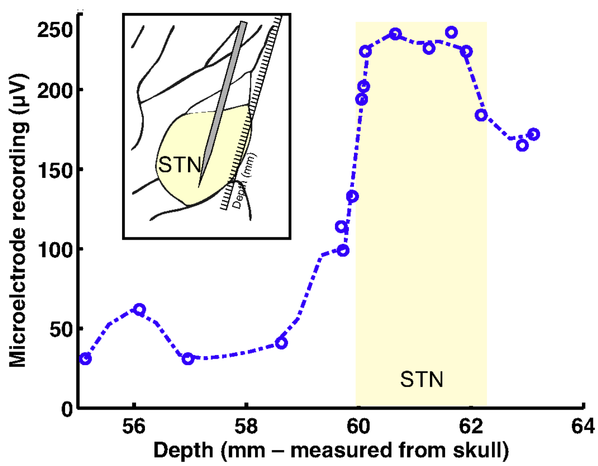


- Result:

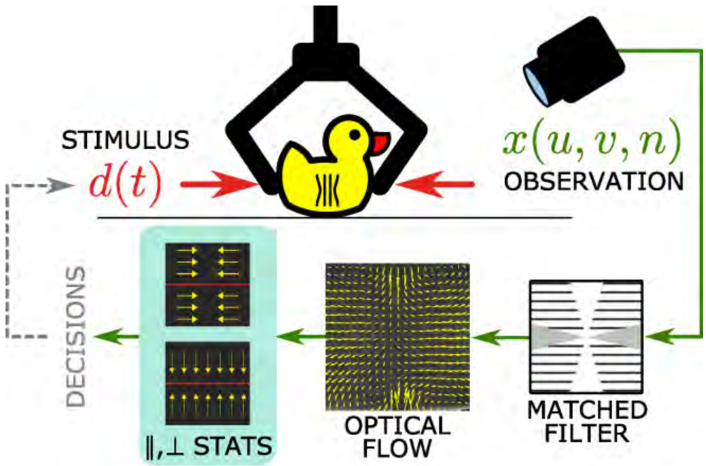


- Stereotactic (Leksell) frames alone are not enough…
- Brain Shift, Compliance, Drift, etc.

In-Vivo Feedback: Incorporating MER
Incorporating tissue signal signatures

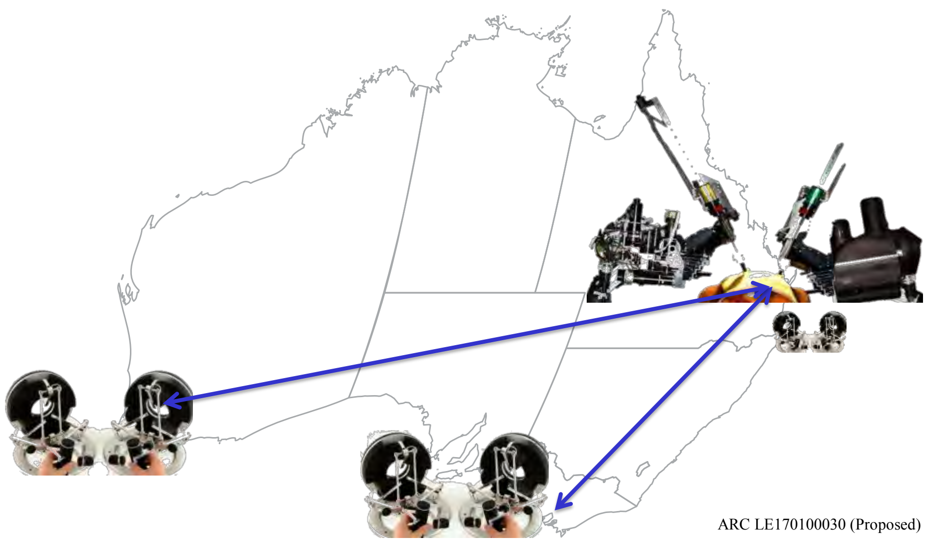


What's Next? Incorporating Stiffness: Visual Deformable Object Analysis



Dansereau, Singh, Leitner, *ICRA 2016*

What's Next: Open Access Robotics Infrastructure for High-Fidelity Telesurgical Research



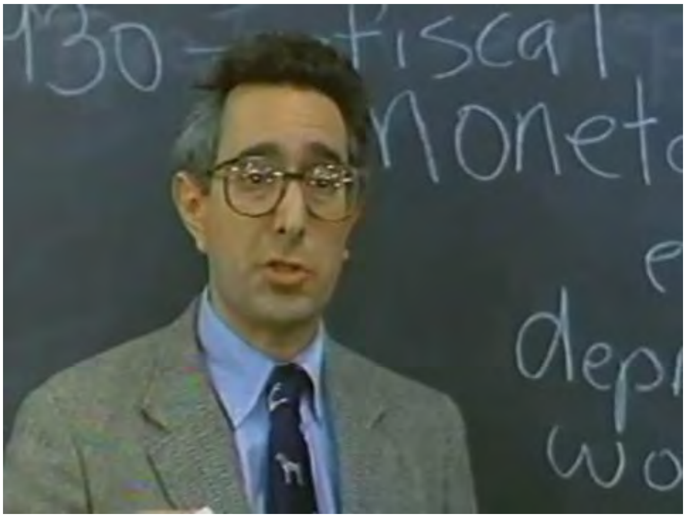
ARC LE170100030 (Proposed)

SECaT Time! … Brought To You By the Number 5

METR 4202: **Robotics** October 26, 2016 -96

“4” Is Average

- What is a 3?



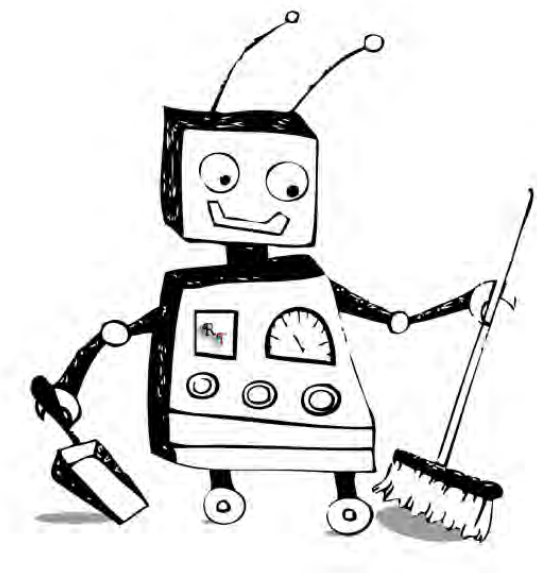
METR 4202: **Robotics** October 26, 2016 -97

SECaTs: Some Lessons in the Works for Next Year

- I shall only use my own slides
- Less is more!
  - Smaller assignments
  - More time for Examples
- Better organization
  - Better tutorials
  - More examples!!
  - I get that. But, we've come a long way

➔ To make this happen I need your support!

METR 4202: **Robotics** October 26, 2016 -98

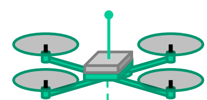


UQ Robotics: Dynamic Systems in Motion

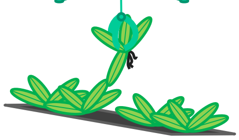
Diverse international research group

Hanna Kurniwati
(NUS/MIT)
Paul Pounds
(ANU/Yale)
Surya Singh
(Stanford/Syd)

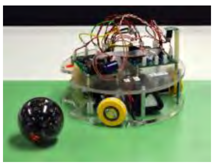
Aerial Systems



Bio-inspired Systems



Planning



Mechanics of motion