METR4202: Robotics & Automation
Lecture Compendium

METR 4202: **Robotics** & Automation

Dr Surya Singh

July 27-October 30, 2016

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

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

PART I:
Kinematics & Dynamics of Robotic Motion & Sensing

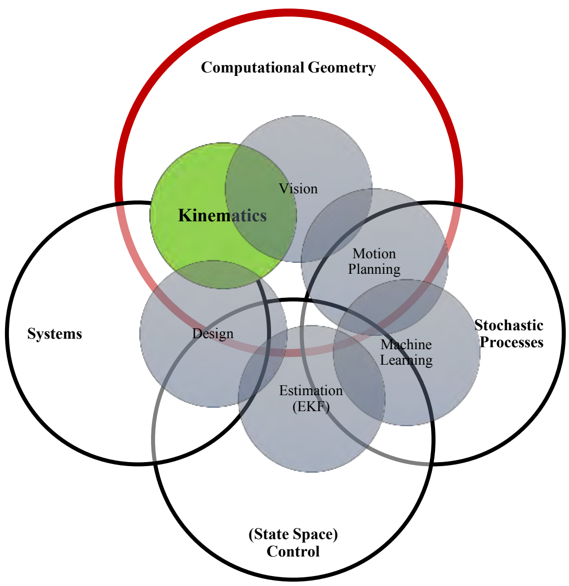
DRAFT

Lectures 1-7

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



Introduction to Robotics

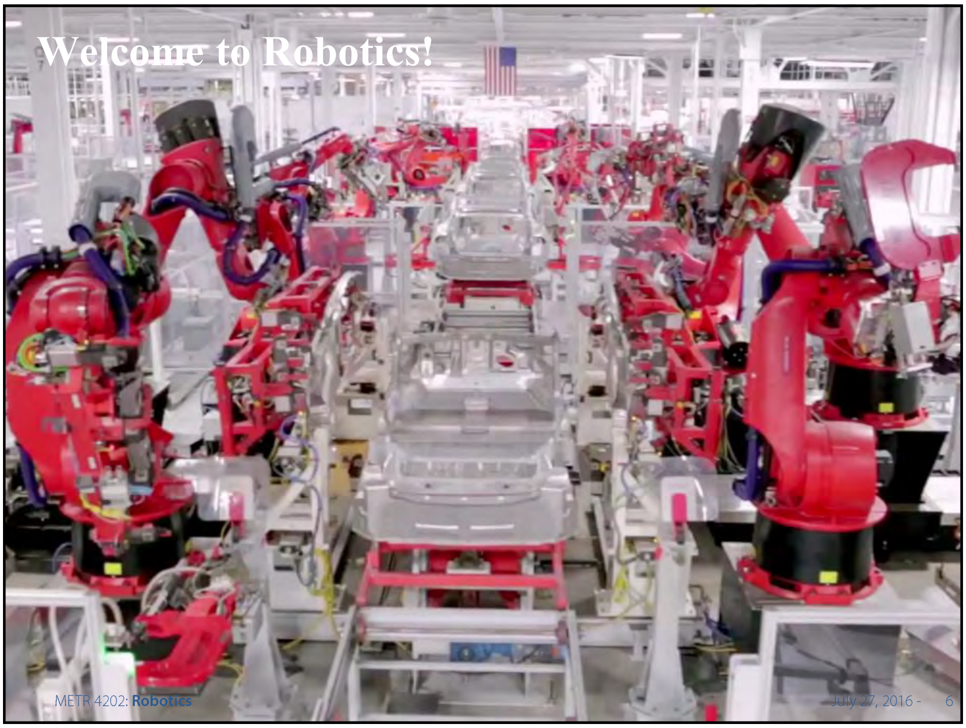
METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 1

July 27, 2016

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

Welcome to Robotics!




ASIMO
The Honda Humanoid Robot ASIMO
History of Humanoids
Man's dream takes first step forward

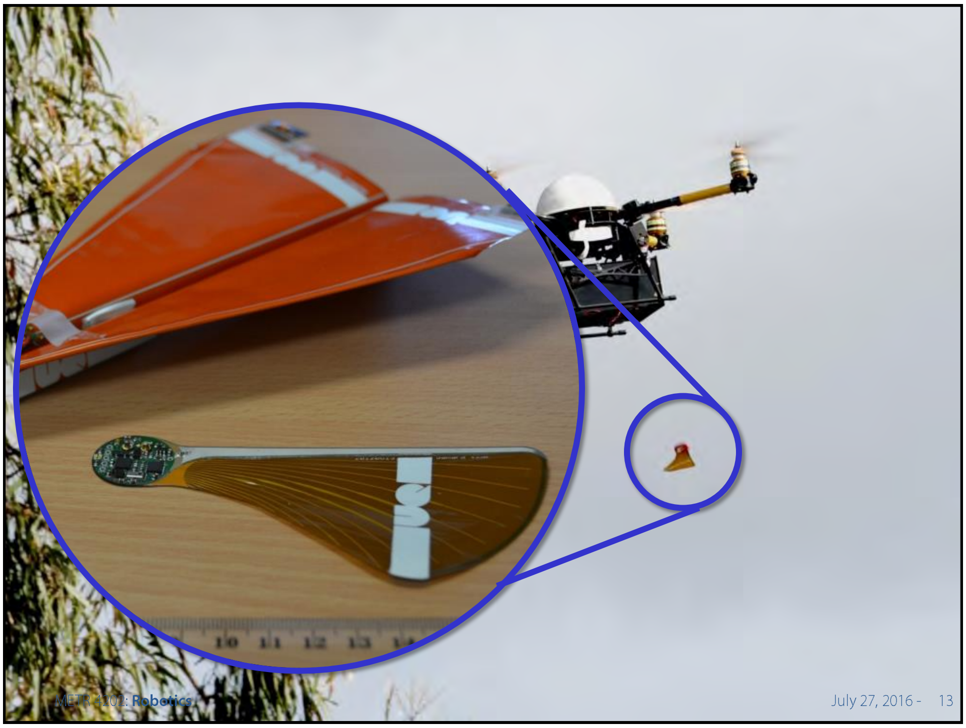

Powered by Diesel
Red Bull
MDV


METR 4202: Robotics
July 27, 2016 - 9


METR 4202: Robotics
July 27, 2016 - 10


METR 4202: Robotics
July 27, 2016 - 11


METR 4202: Robotics
July 27, 2016 - 12



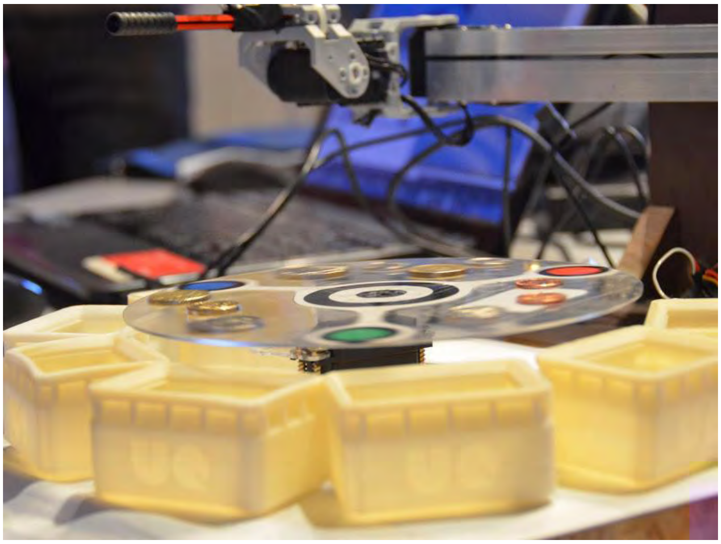
METR 4202: Robotics
July 27, 2016 - 13

METR 4202: Robotics
7, 2016 - 14


Magicarms


METR 4202: Robotics

Change. The Future!



Win. The (DARPA Robotics) Challenge!

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]

Assessment

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

Lectures

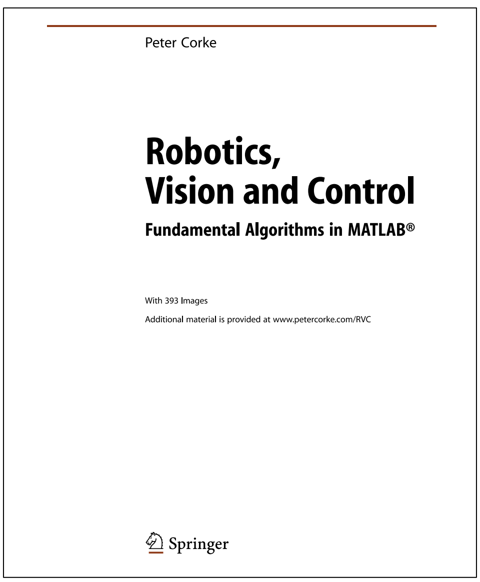
- Wednesdays from 12:05 – 1:50 pm
- Lectures will be posted to the course website **after** the lecture (so please attend)
  - Slides are like dessert – enjoy afterwards!
- Please ask questions (preferably about the material ☺)

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!)

Textbook



Robotics, Vision and Control Fundamental Algorithms in MATLAB

By:
Peter Corke

Available online (on campus) via SpingerLink

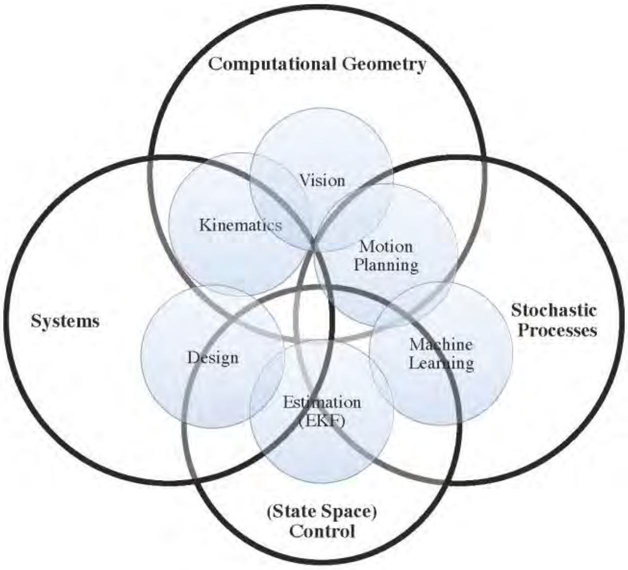
E-mail & website

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)

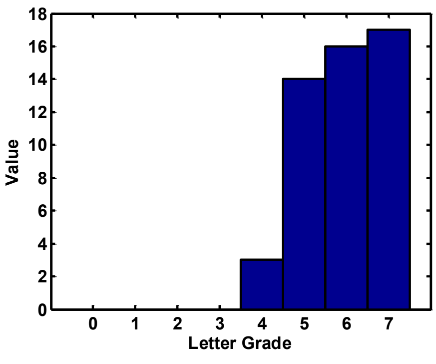
Course Objectives

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



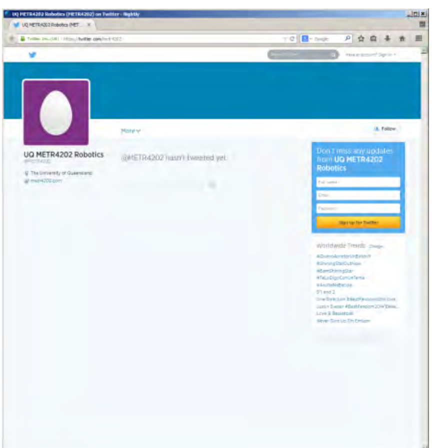
- ~ 67 % received D or HD
- Worry about **learning**, not about marks [Seriously!]
- Though a "7" might be bit more exclusive this year!

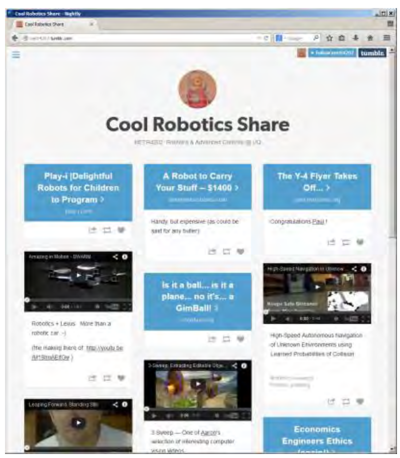
What I expect from you

- 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** …

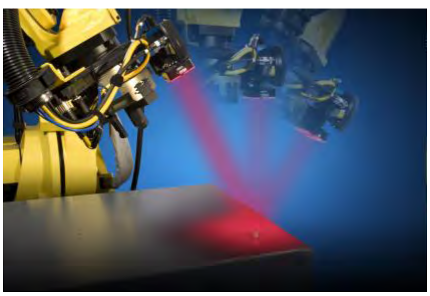
Either by:

- 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!

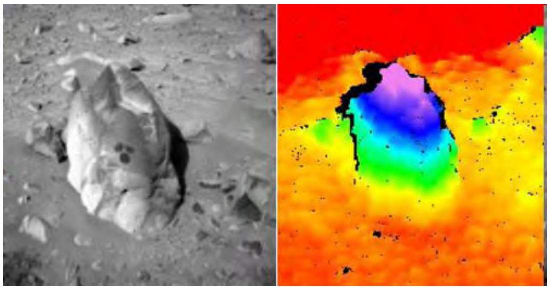
Sensing

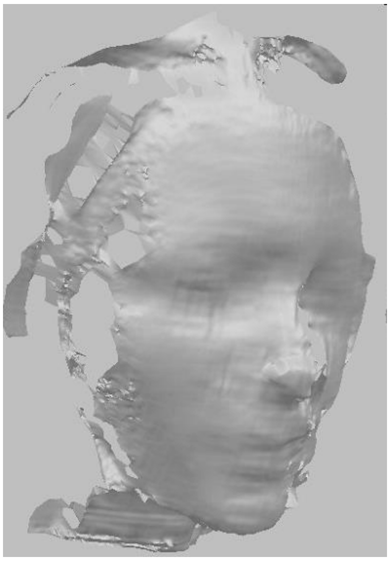
Perception: Vision



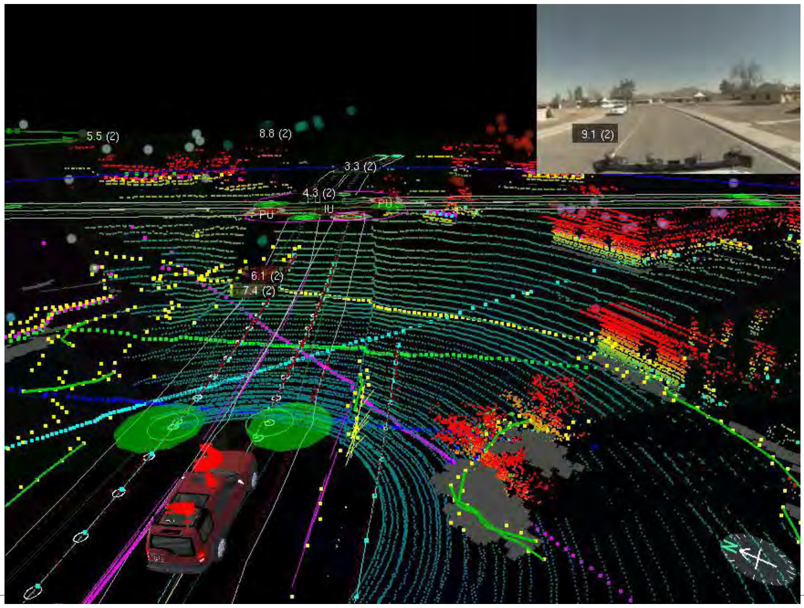
Edges, Segments, Colour, Texture

3D Stereo Vision



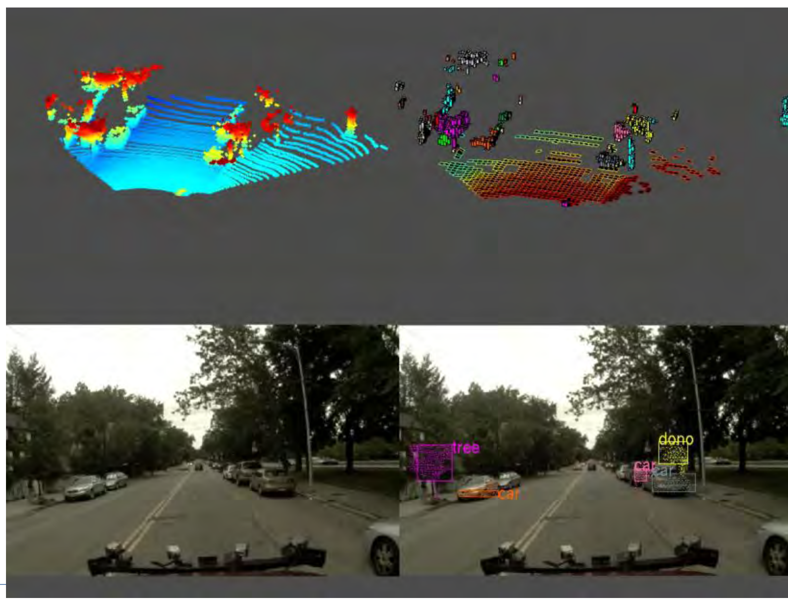


Laser Sensors

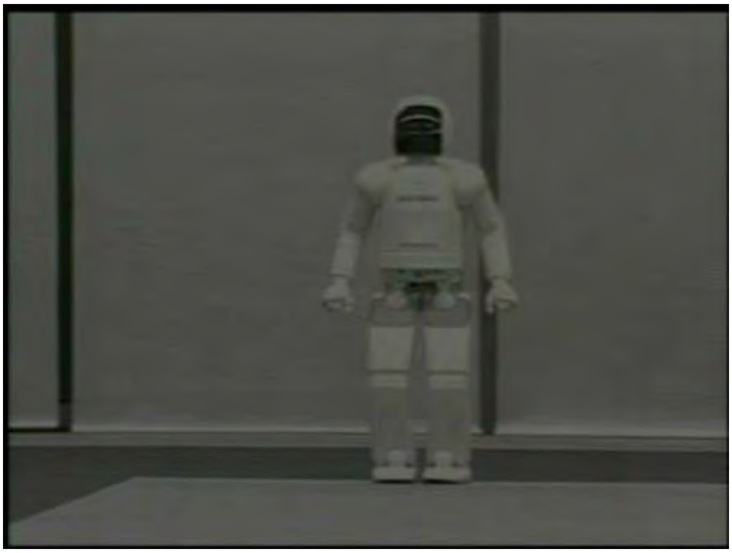


Control (Processing) …

Environment Understanding



Honda Asimov Humanoid



Act(ion)

Robot Sniper Training Robots

Extending Our Reach…

(what's hard is not what you expect…)

Throwing and Catching

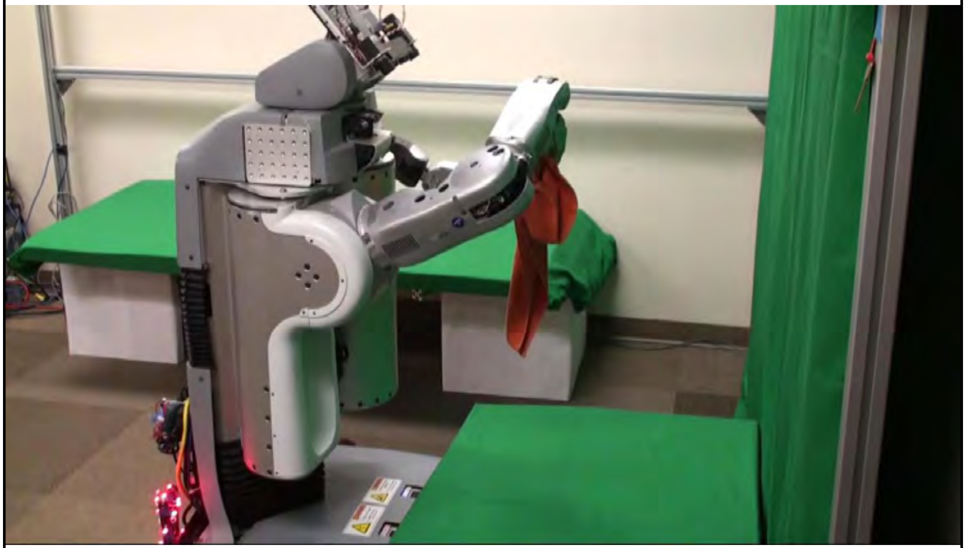
Making Iced Tea

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)…

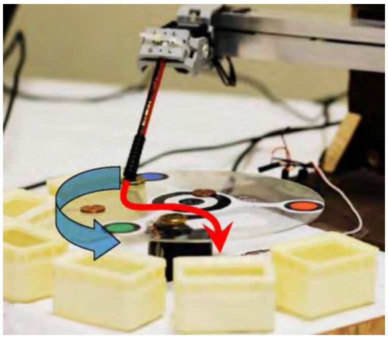


Shirt-Folding (1/3 Speed!)

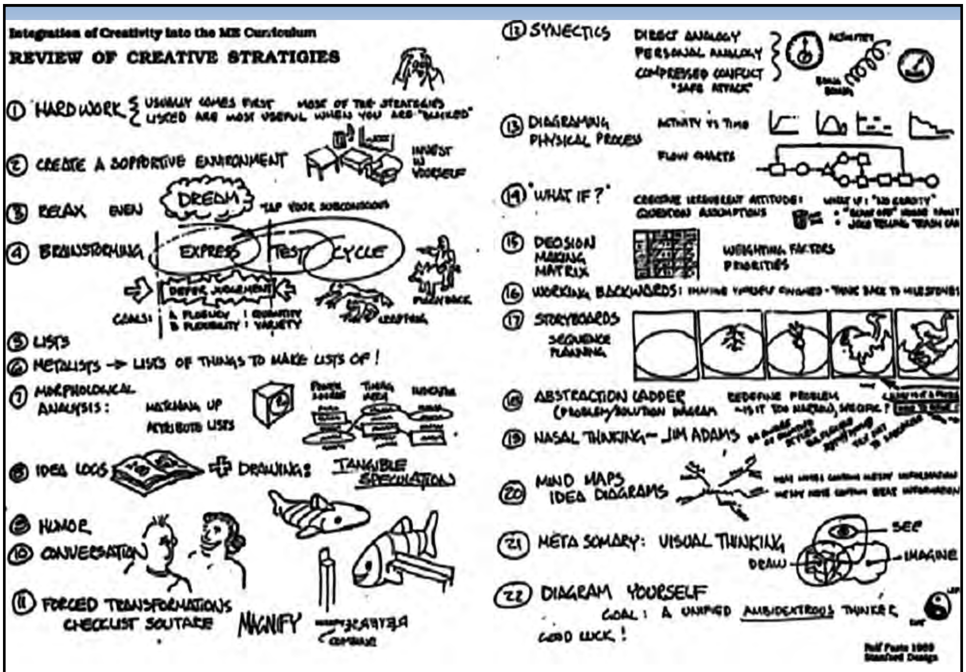
Parallel-Parking…

Parallel Parking…

The Project!
"Robotics: Domino Effect"



Next Week ☺



First thing about structure

→ **Space**

Representing Position & Orientation & State

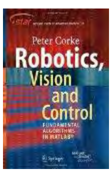
METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 2

August 3, 2016

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

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

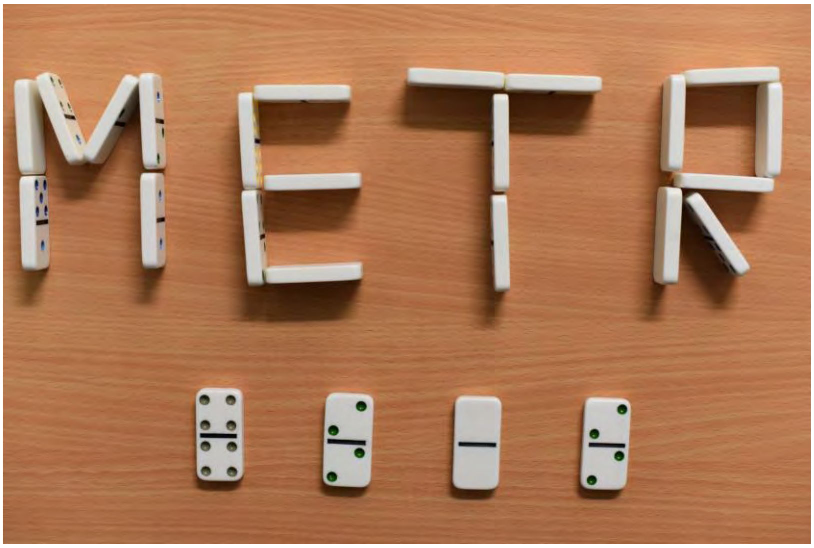
➔ Representing Position ←

- RVC
  - Ch. 2: Representing Position & Orientation

- Kinematics
  - RVC
  - Chapter 7: Robot Arm Kinematics

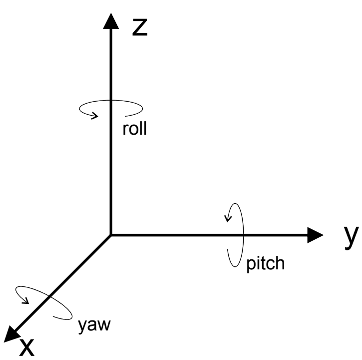
Next Time

The Project!
"Robotics: Domino Effect"

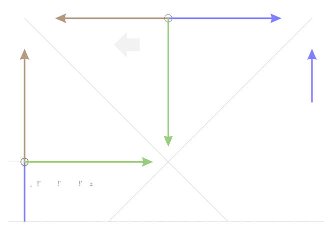


METR 4202: **Robotics** August 3, 2016 68

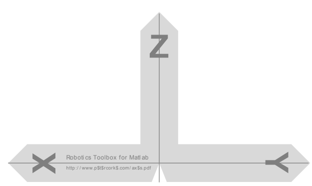
Today's Lecture is about:
Frames & Their Mathematics



- Make one (online):
  - SpnS Template



  - Peter Corke's template

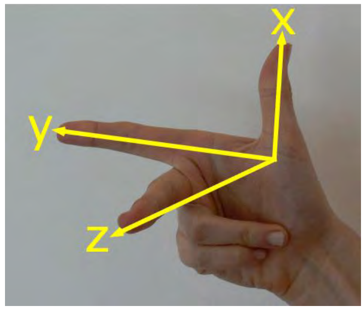


METR 4202: **Robotics** August 3, 2016 - 69

Don’t Confuse a Frame with a Point

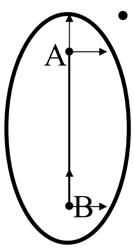
- Points
  - Position Only –
    Doesn’t Encode Orientation

- Frame
  - Encodes both position and orientation
  - Has a “handedness”

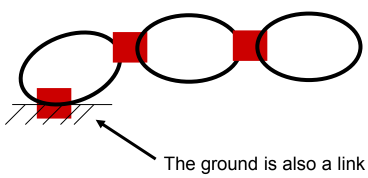


Kinematics Definition

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



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

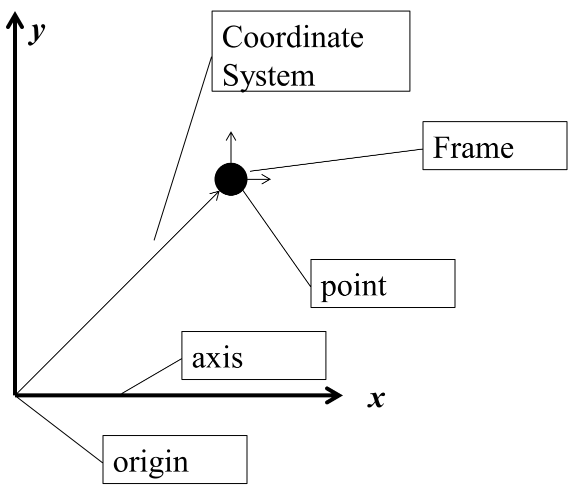


N links
M joints
→DOF = 6N-5M
→ If N=M, then DOF=N.

Kinematics

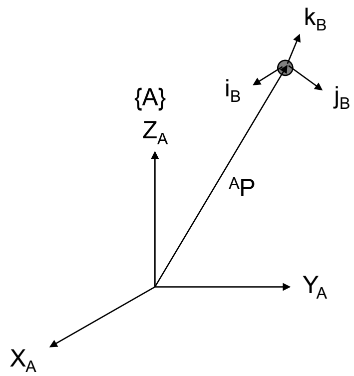
- Kinematic modelling is one of the most important analytical tools of robotics.
- Used for modelling mechanisms, actuators and sensors
- Used for on-line control and off-line programming and simulation
- In mobile robots kinematic models are used for:
  - steering (control, simulation)
  - perception (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



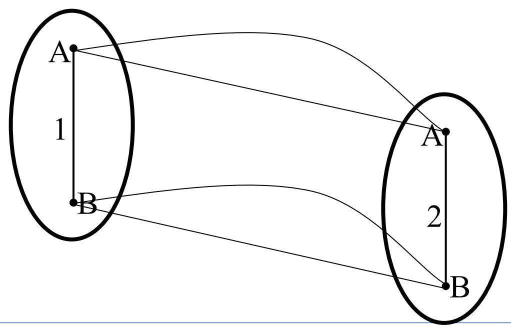
Frames of Reference

- 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

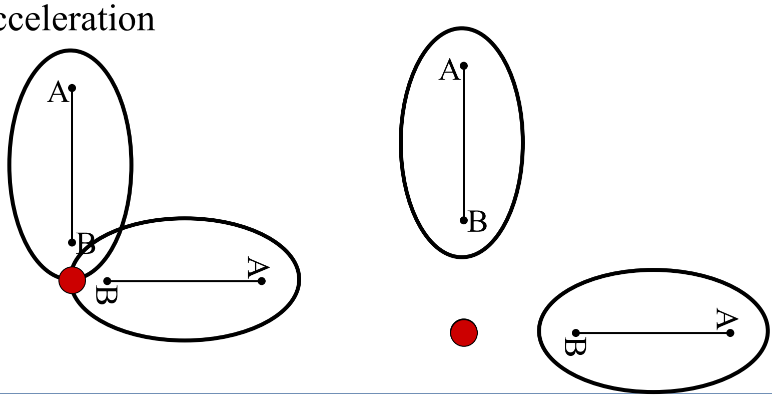
Translation

- 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

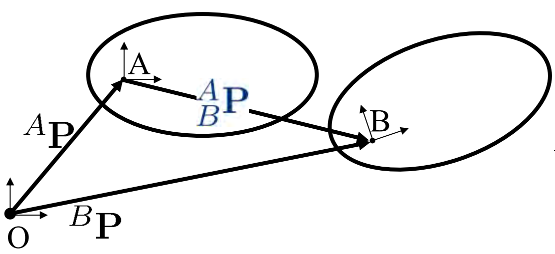
- Orientation are not “Cartesian”
  - Non-commutative
  - Multiple representations

- 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

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



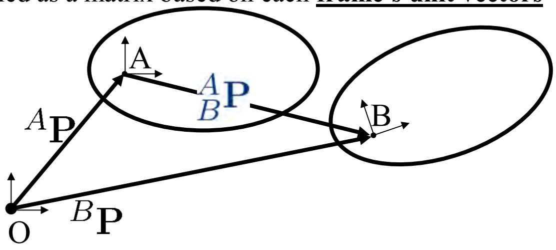
$${}^{A}\mathbf{P} + {}^{A}\mathbf{P}^{B} - {}^{B}\mathbf{P} = 0$$

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

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

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}"

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

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

Position and Orientation [3]

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

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

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

Position and Orientation [4]

- Rotation is orthonormal

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

- The of a rotation matrix inverse = the transpose

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

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

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

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

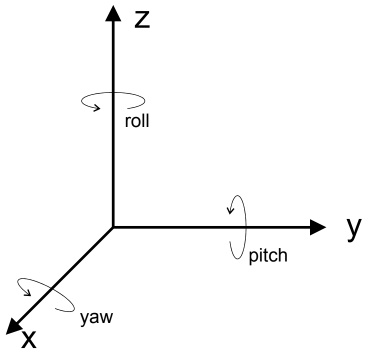
- Minimal representation of orientation (α,β,γ)
- Represent a rotation about an axis of a **moving** coordinate frame
  → ${}^{A}_{B}\mathbf{R}$: Moving frame **B** w/r/t fixed A
- The location of the axis of each successive rotation depends on the previous one! …
- So, Order Matters (12 combinations,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

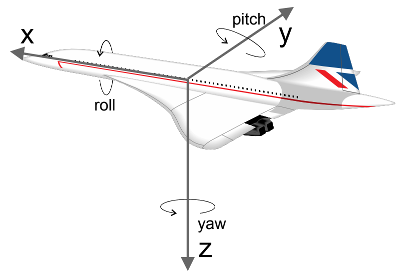
- Represent a rotation about an axis of a **fixed** coordinate frame.
- Again 12 different orders
- Interestingly:
  3 rotations about 3 axes of a **fixed** frame define the same orientation as the same 3 rotations taken in the **opposite order** of the **moving** frame
- For X-Y-Z:
  - ψ: rotation about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

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

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

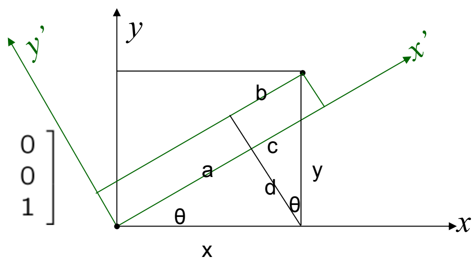
- Another method of describing the orientation of {B} is:
  - Start with the frame coincident with a known reference {A}. Rotate {B} first about Z_B by an angle α, then about Y_B by an angle β and finally about X_B by an angle γ.

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

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 $(\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3)$ [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:

$$\begin{aligned}\mathbf{ab} \;=\; & (a_0b_0 - a_1b_1 - a_2b_2 + a_3b_3) \\ & + (a_0b_1 + a_1b_0 + a_2b_3 - a_3b_2)\,\hat{i} \\ & + (a_0b_2 + a_2b_0 + a_3b_1 + a_1b_3)\,\hat{j} \\ & + (a_0b_3 + a_3b_0 + a_1b_2 - a_2b_1)\,\hat{k}\end{aligned}$$

- 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

$$^{A}\mathbf{u} = u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k}$$

$$^{A}\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$$

$$^{A}\mathbf{w} = w_x\mathbf{i} + w_y\mathbf{j} + w_z\mathbf{k}$$

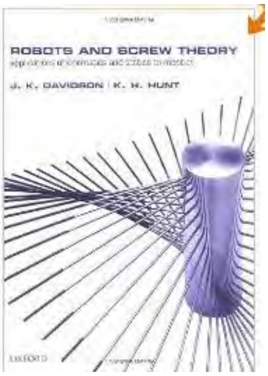
- It forms a rotation matrix!

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

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

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

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

- 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

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

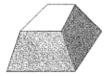
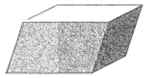
- ρ is a scaling value

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	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		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.

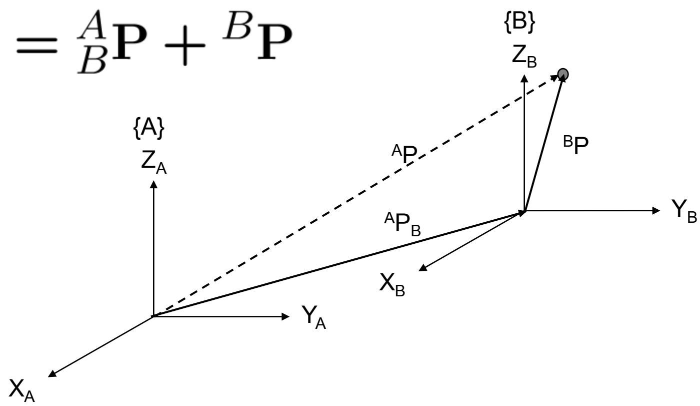
p.78, R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*

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

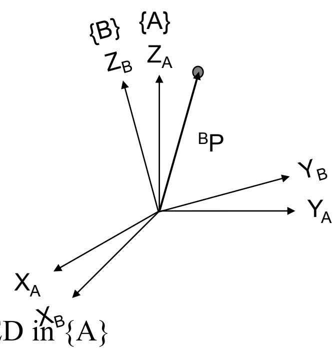
- NOTE:

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

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

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

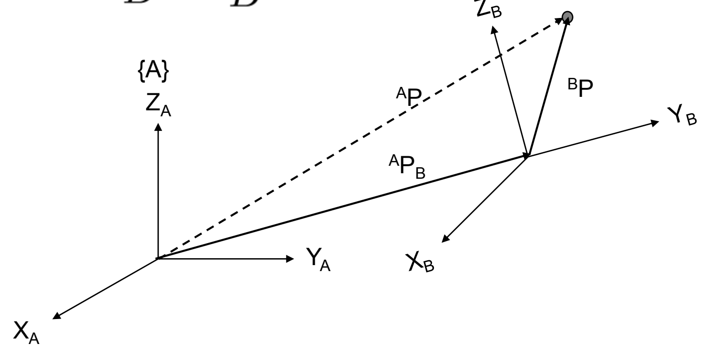
  – This gives Point Positions of {B} ORIENTED in {A}



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**

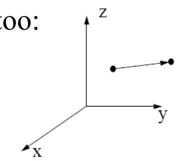
$$ {}^{A}_{B}\mathbf{T} = \begin{bmatrix} & {}^{A}_{B}\mathbf{R} & & {}^{A}\mathbf{P}_{B} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

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

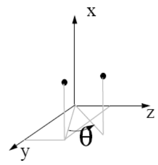
$$ \begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = {}^{A}_{B}\mathbf{T} \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix} $$

General Coordinate Transformations [2]

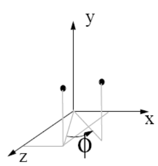
- Similarly, fundamental orthonormal transformations can be represented in this form too:



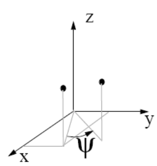
$$ Trans(u, v, w) = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



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



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

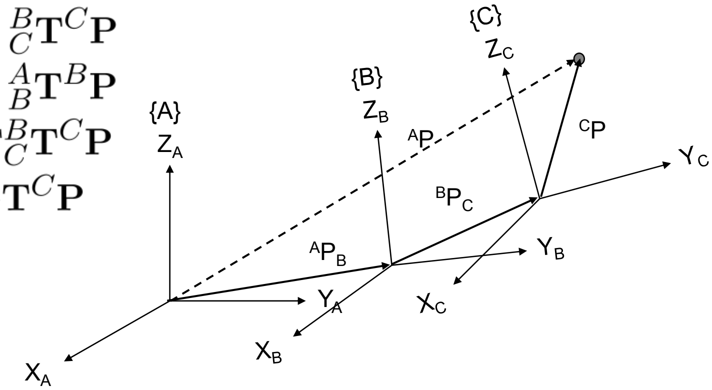


$$ Rotz(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 & 0 \\ s\psi & c\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

General Coordinate Transformations [3]

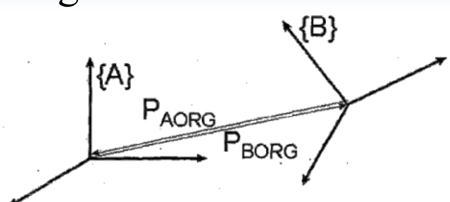
- Multiple transformations compounded as a chain

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



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

Inverse of a Homogeneous Transformation Matrix



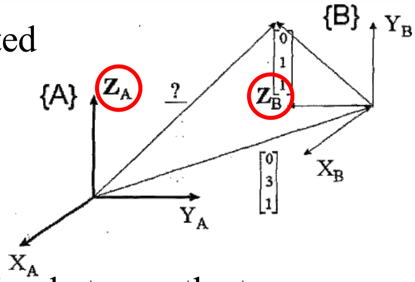
- The inverse of the transform is **not** equal to its transpose because this 4×4 matrix is not orthonormal ($T^{-1} \neq T^{T}$)
- Invert by parts to give:

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

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

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 $_BT^A$ 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} $

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**]

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]

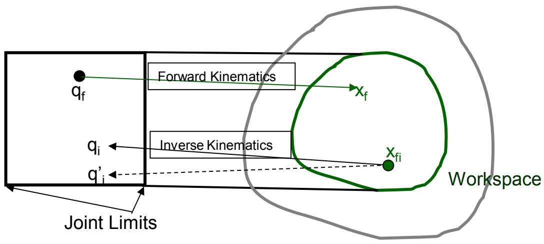
- Forward kinematics is the process of chaining homogeneous transforms together. For example to:
  - Find the articulations of a mechanism, or
  - the fixed transformation between two frames which is known in terms of linear and rotary parameters.
- Calculates the final position from the **machine** (**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,α,β,γ):
    The robot's position & orientation
  - Joint space (θ_1 … θ_n):
    A state-space vector of joint variables

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

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

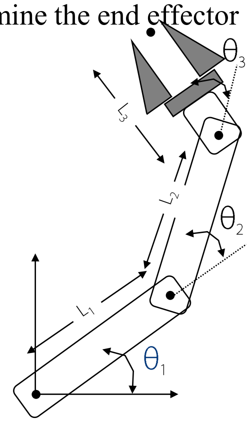


Forward Kinematics [3]

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

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

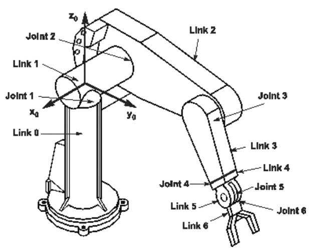
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1+\theta_2) + \ldots$$
$$L_3 \sin(\theta_1+\theta_2+\theta_3)$$

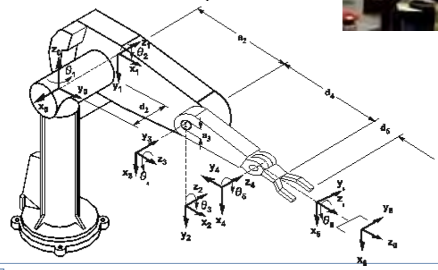


- This isn't too difficult to determine for a simple, planar manipulator. BUT …

Forward Kinematics [4]: The PUMA 560!

- What about a more complicated mechanism?





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

$) - s_1(s_4c_5c_6 + c_4s_6)$

$) + c_1(s_4c_5c_6 + c_4s_6)$

$s_6) - s_1(-s_4c_5s_6 + c_4c_6)$

$s_6) + c_1(-s_4c_5s_6 + c_4c_6)$

$a_x = c_1(c_{23}c_4s$

$a_y = s_1(c_{23}c_4s$

$a_z = -s_{23}c_4s_5$

$p_x = c_1(d_6(c_{23}$

$p_y = s_1(d_6(c_{23}$

$p_z = d_6(c_{23}c_5$

Denavit Hartenberg [DH] Notation

Denavit Hartenberg [DH] Notation

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

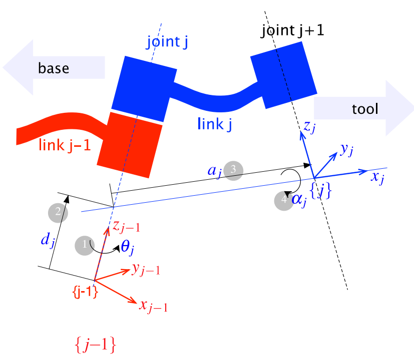
  (But B. Roth, introduced it to robotics)

- 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. 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*)

Denavit-Hartenberg → Rotation Matrix

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

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

DH Example [1]: RRR Link Manipulator

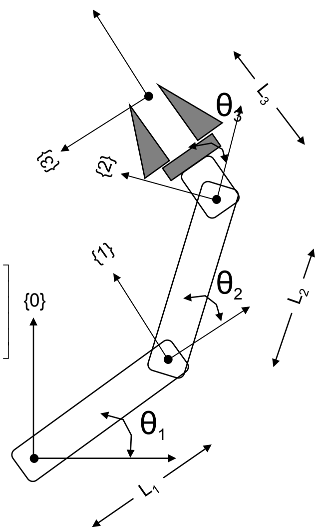
1. Assign the frames at the joints …
2. Fill DH Table …

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

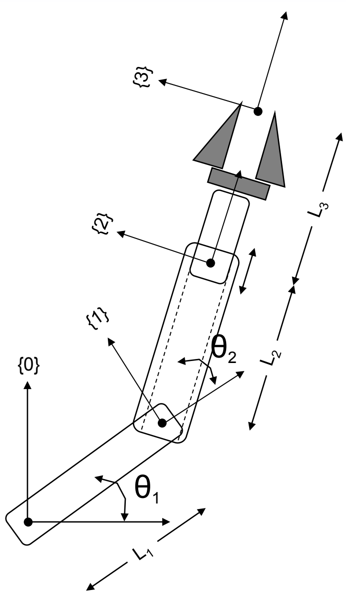
1. Assign the frames at the joints …
2. Fill DH Table …

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	$\boldsymbol{L_3}$	0	0	0

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

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

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



DH Example [3]: Puma 560

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

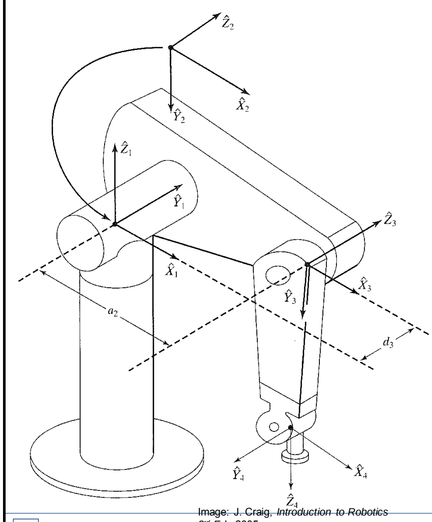
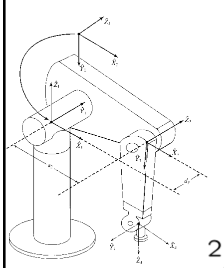


Image: J. Craig, *Introduction to Robotics* 3rd Ed., 2005

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]



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

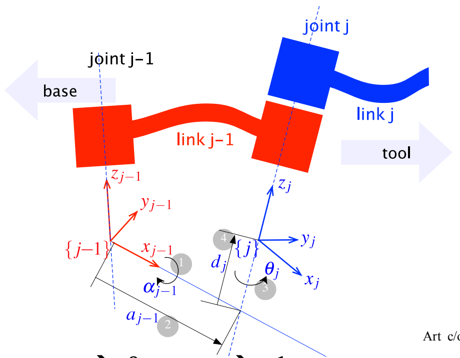
$$^2A_3 = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ^3A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & L_3 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

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

Modified DH

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

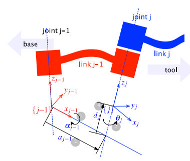


Art c/o P. Corke

- a (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)



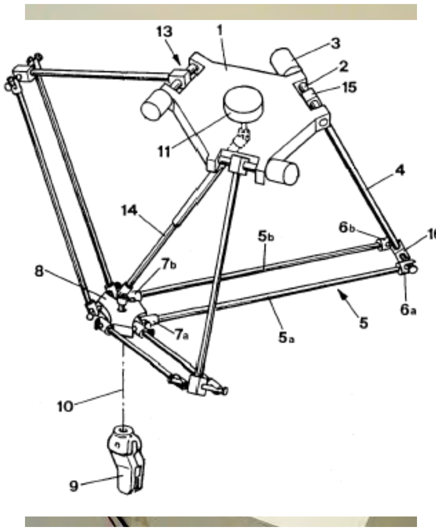
$$\Rightarrow {}^{i-1}A_i = R_x(\alpha_{i-1})\, T_x(a_{i-1})\, R_z(\theta_i)\, T_x(d_i)$$

- Refactoring Standard → to Modified

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

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

Parallel Manipulators



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

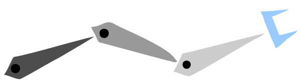
- The "central" Kinematic structure is made up of closed-loop chain(s)
- Compared to Serial Mechanisms:
  - \+ Higher Stiffness
  - \+ Higher Payload
  - \+ Less Inertia
  - – Smaller Workspace
  - – Coordinated Drive System
  - – More Complex & $$$

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

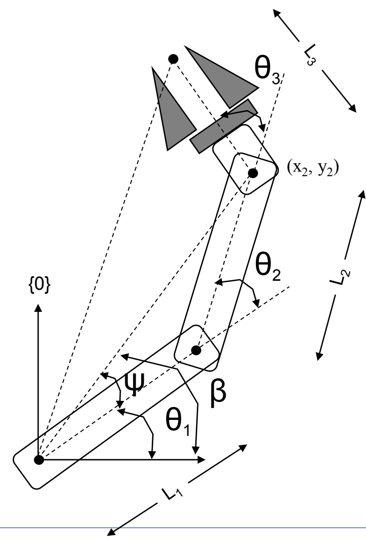
- **Closed-form** and **numerical** methods exist

- 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**

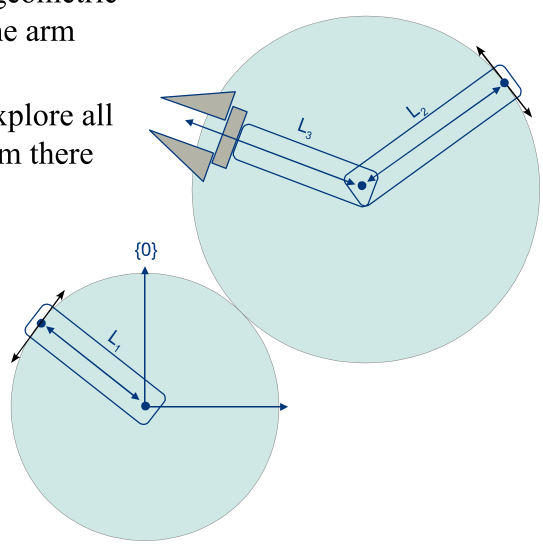
Inverse Kinematics: Geometrical Approach

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



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

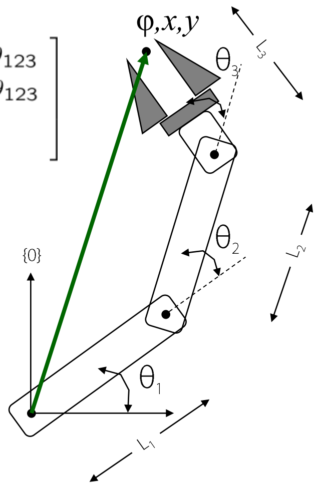
- We have a series of equations which define this system
- Recall, from Forward Kinematics:

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

- The end-effector pose is given by

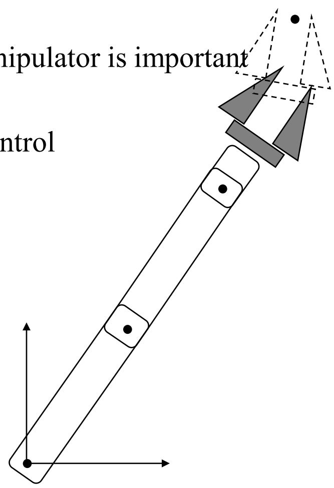
$$^0T_3 = \begin{bmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Equating terms gives us a set of algebraic relationships



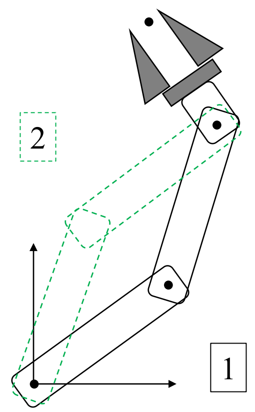
No Solution - Singularity

- 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**



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 & Kinetics

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 4

August 17, 2016

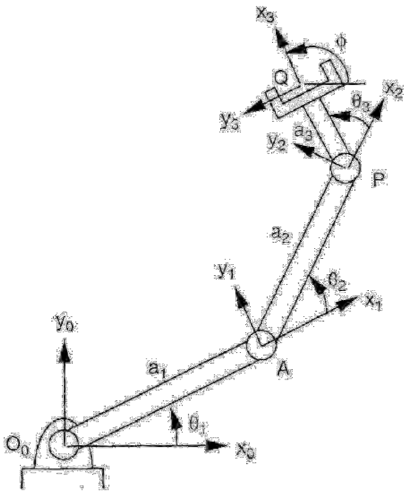
metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

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)

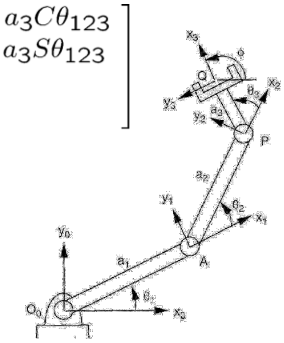
Example: 3R Planar Arm [2]

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

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

Substituting gives:

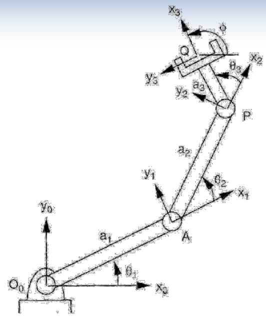
$$ {}^0A_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 & a_1C\theta_1 + a_2C\theta_{12} + a_3C\theta_{123} \\ S\theta_{123} & C\theta_{123} & 0 & a_1S\theta_1 + a_2S\theta_{12} + a_3S\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



Example: 3R Planar Arm [2]

Forward Kinematics

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



Fairly straight forward:

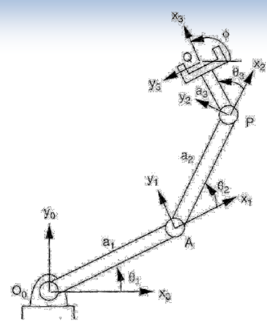
$$ {}^{0}R_3 = \begin{bmatrix} C\theta_{123} & -S\theta_{123} & 0 \\ S\theta_{123} & C\theta_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

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

Example: 3R Planar Arm [3]

Inverse Kinematics

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



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

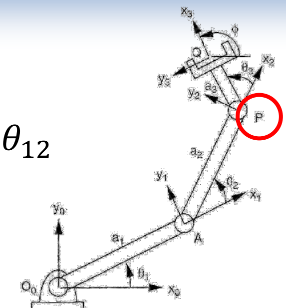
$$ q_x - a_3 C\phi = a_1 C\theta_1 + a_2 C\theta_{12} $$

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

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 $\mathbb{R}$ roots if $|\kappa|<1$)

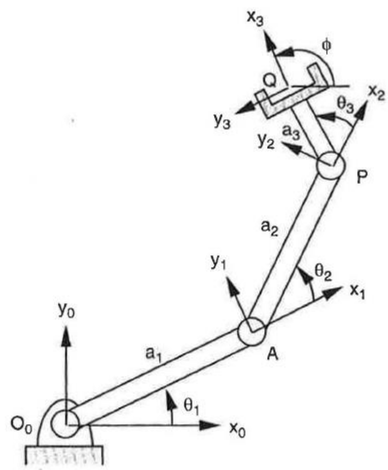
- Solve for θ_1:

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

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

Inverse Kinematics: Example I

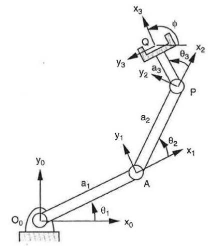
Planar Manipulator:



Inverse Kinematics: Example I

- Forward Kinematics:

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



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

$\because$

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

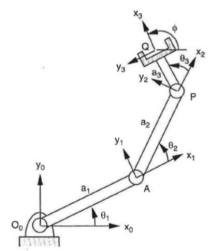
- For an arbitrary point **G** in the end effector: ${}^3\mathbf{g} = [g_u, g_v, 0, 1]^T$

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

Inverse Kinematics: Example I

- Forward Kinematics:

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



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

$\because$

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

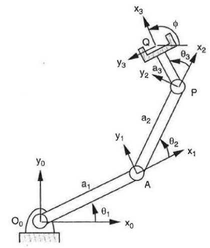
- For an arbitrary point **G** in the end effector: ${}^3\mathbf{g} = [g_u, g_v, 0, 1]^T$

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

Inverse Kinematics: Example I

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

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

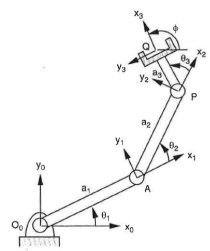


- The solution strategy is to equate the elements of $^0\mathbf{A}_3$ to that of the given position (q_x, q_y) and orientation ϕ

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:

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

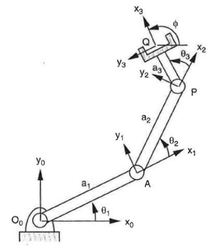
Inverse Kinematics: Example I

- 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}:

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

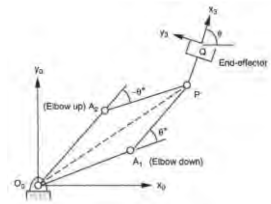
  

  – This gives 2 real ($\mathbb{R}$) roots if $|\kappa| < 1$

  – One double root if $|\kappa| = 1$

  – No real roots if $|\kappa| > 1$

- Elbow up/down:

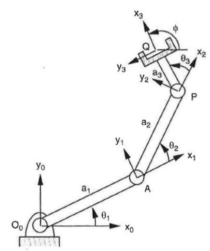
  

  – In general, **if** θ_2 is a solution **then** -θ_2 is a solution

Inverse Kinematics: Example I

- Solving for θ_1…

  – Corresponding to each θ_2, we can solve θ_1

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

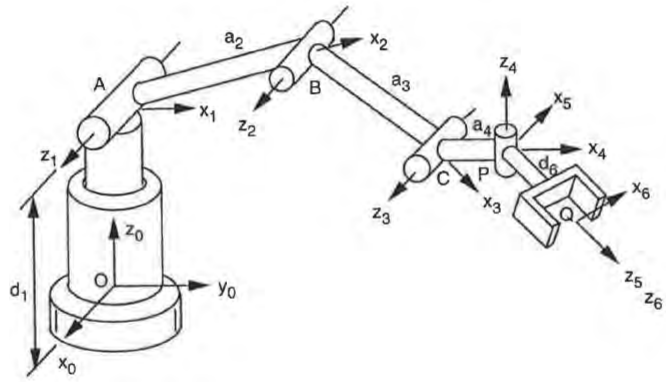
  $$c\theta_1 = \frac{p_x(a_1 + a_2c\theta_2) + p_ya_2s\theta_2}{\Delta},$$
  $$s\theta_1 = \frac{-p_xa_2s\theta_2 + p_y(a_1 + a_2c\theta_2)}{\Delta}$$

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

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

Inverse Kinematics: Example II

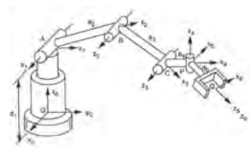
Elbow Manipulator:



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:

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

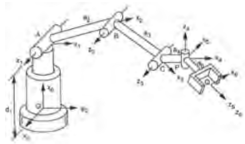
$$A_2 = \begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ s\theta_2 & 0 & c\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

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

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

Inverse Kinematics: Example II

- Key Matrix Products:

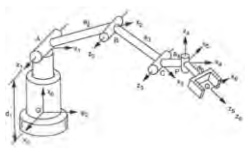


$$A_2A_3A_4 = \begin{bmatrix} c\theta_{234} & 0 & -s\theta_{234} & a_2c\theta_2 + a_3c\theta_{23} - (a_2+a_3)c\theta_{234} \\ 0 & 1 & 0 & 0 \\ s\theta_{234} & 0 & c\theta_{234} & a_2s\theta_2 + a_3s\theta_{23} - (a_2+a_3)s\theta_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Inverse Kinematics: Example II

- Inverse Kinematics:



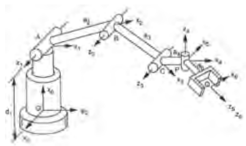
$$\mathbf{p} = A_1A_2A_3A_4\mathbf{p}_0.$$

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

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

Inverse Kinematics: Example II

- Solving the System:



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

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

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

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

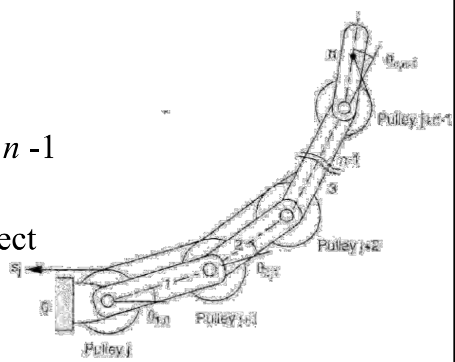
$$a_2^2 + a_3^2 + 2a_2 a_3 c\theta_3 = k_1^2 + k_2^2.$$

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

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

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:



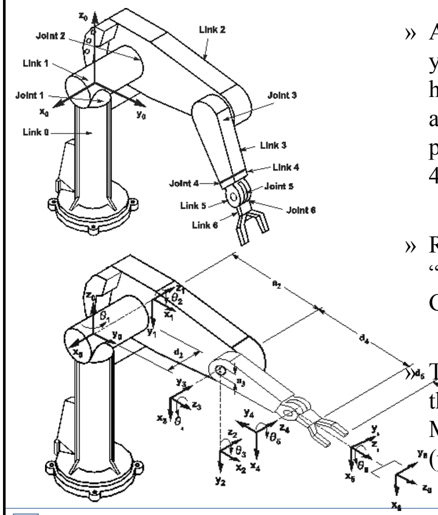
$$r_{j+i-1}\theta_{j+i-1,i} = \pm r_{j+i}\theta_{j+i,i} \qquad \text{for } i = 1, 2, \ldots, n-1.$$

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

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

Inverse Kinematics

- What about a more complicated mechanism?



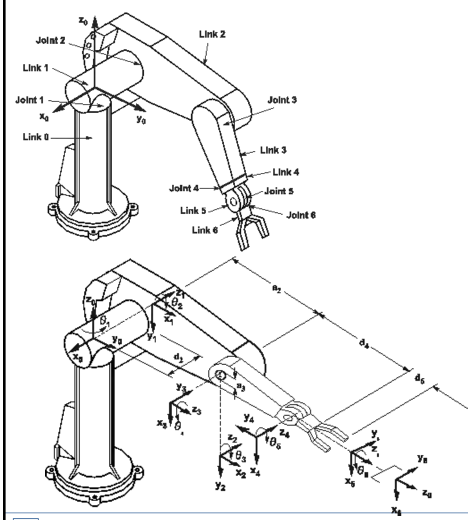
» A sufficient condition for a serial manipulator to yield a closed-form inverse kinematics solution is to have any three consecutive joint axes intersecting at a common point or any three consecutive joint axes parallel to each other. (Pieper and Roth (1969) via 4×4 matrix method)

» 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:

- # 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)

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

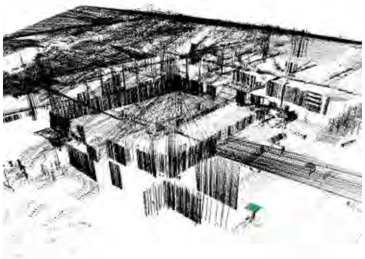
Where: λ: The DOF of the space that the system is in (e.g., λ=6 for 3D space).

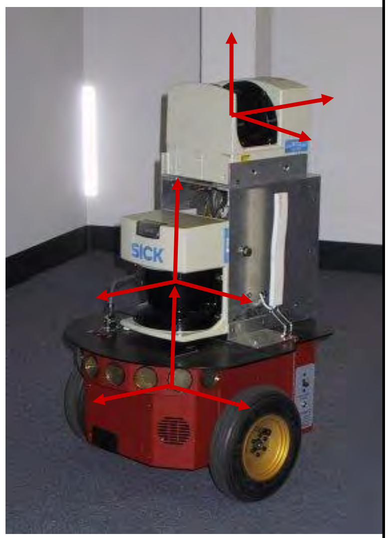
Mobile Platforms

- 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

Reference Material

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

Robot Dynamics
(& Jacobeans)

METR 4202: **Robotics & Automation**

Dr Surya Singh -- Lecture # 5 **August 24, 2016**

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

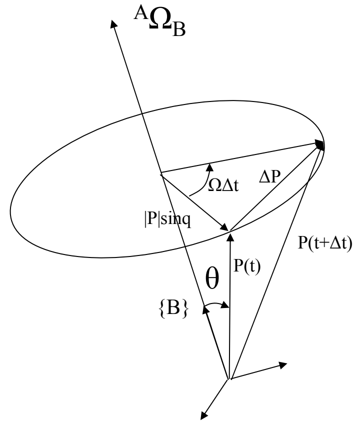
Robot Dynamics

Robot Dynamics

Angular Velocity

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

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



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

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

$${}^{A}\mathbf{V}_P = \frac{d}{dt}{}^{A}\mathbf{P} = \lim_{\Delta t \to 0} \frac{{}^{A}\mathbf{P}(t+\Delta t) - {}^{A}\mathbf{P}(t)}{\Delta t}$$
$$= {}^{A}\dot{\mathbf{P}}_B + {}^{A}_{B}\mathbf{R}{}^{B}\dot{\mathbf{P}} + {}^{A}_{B}\dot{\mathbf{R}}{}^{B}\mathbf{P}$$

- This can be rewritten as

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

Skew – Symmetric Matrix

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

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

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

Velocity Representations

- Euler Angles
  - For Z-Y-X (α,β,γ):

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

- Quaternions

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

Manipulator Velocities

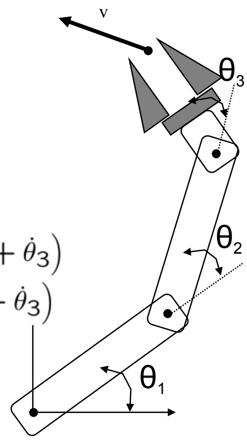
- Consider again the schematic of the planar manipulator shown. We found that the end effector position is given by

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

- Differentiating w/r/t to t

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

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



Manipulator Velocities [2]

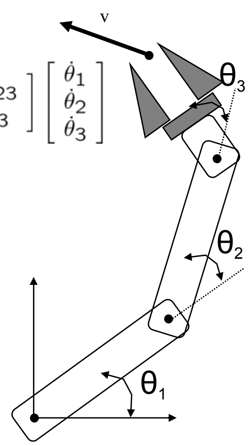
- Rearranging, we can recast this relation in matrix form

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

- Or

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

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



Moving On…Differential Motion

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

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

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

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

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

Differential Motion [2]

- Viewing this as a matrix → Jacobian

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

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

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

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

Infinitesimal Rotations

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

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

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

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

- Note that:

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

→ Therefore … they **commute**

The Jacobian

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

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

- Or more succinctly

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

Jacobian [2]

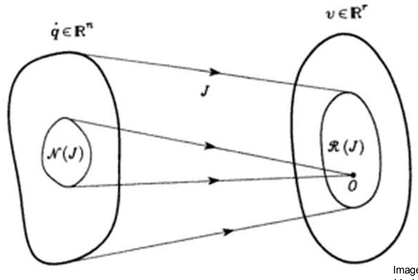


Image:. Sciavicco and Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed, 2000

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

Revisiting The Jacobian

- I told you:

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

- True, but we can be more “explicit”

Jacobian: **Explicit Form**

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

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

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

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

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

Jacobian: **Explicit Form [2]**

- The overall Jacobian takes the form

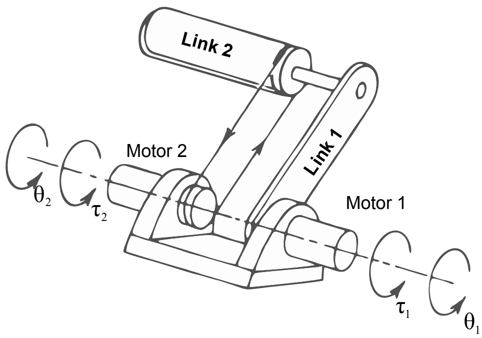
$$J = \begin{bmatrix} \frac{\partial x_P}{\partial q_1} & \cdots & \frac{\partial x_P}{\partial q_n} \\ \bar{\varepsilon}_1 z_1 & \cdots & \bar{\varepsilon}_1 z_n \end{bmatrix}$$

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

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

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

Motivating Example: Remotely Driven 2DOF Manipulator



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

Graphic based on Asada & Slotine p. 112

Dynamics

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

Dynamics of Serial Manipulators

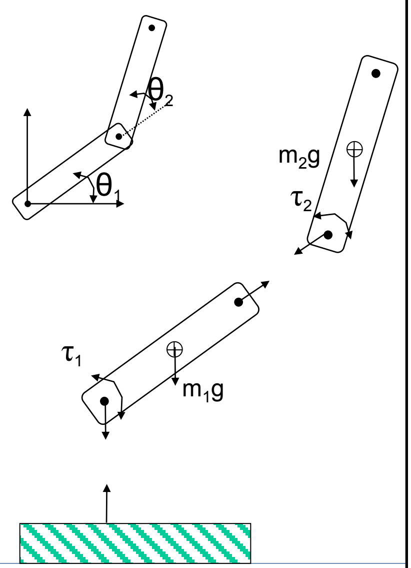
- Systems that keep on manipulating (the system)
- Direct Dynamics:
  - Find the response of a robot arm with torques/forces applied
- Inverse Dynamics:
  - Find the (actuator) torques/forces required to generate a desired trajectory of the manipulator

Dynamics – Newton-Euler

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

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

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



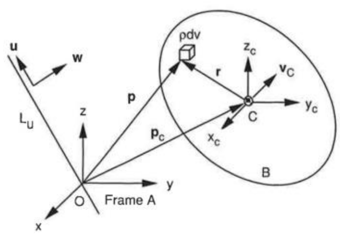
Dynamics

- For Manipulators, the general form is

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

  where
  - τ is a vector of joint torques
  - Θ is the nx1 vector of joint angles
  - $M(\Theta)$ is the nxn mass matrix
  - $V(\Theta, \dot{\Theta})$is the nx1 vector of centrifugal and Coriolis terms
  - $G(\Theta)$ is an nx1 vector of gravity terms
- Notice that all of these terms depend on Θ so the dynamics varies as the manipulator move

Dynamics: Inertia



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

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

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

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

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

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

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

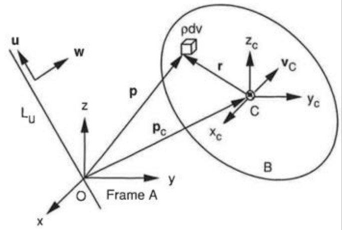
$$r_g = p^2 - \left(p^T u\right)^2 = (u \times p)^2$$

Dynamics: Mass Matrix & Inertia Matrix

- This can be written in a Matrix form as:

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

- Where I^O_B is the inertial matrix or inertial tensor of the body B about a reference point O



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

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

If CM is the center of mass, then:

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

Dynamics: Mass Matrix

- The Mass Matrix: Determining via the Jacobian!

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

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

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

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

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

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

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

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

→Thus: the kinetic energy and gravity terms become

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

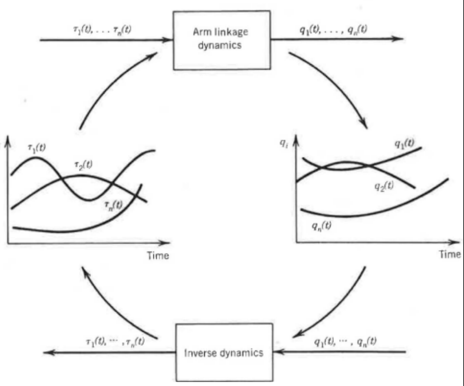
where: $\mathbf{H}^* = \left(\mathbf{J}^{-1}\right)^T \mathbf{H}\mathbf{J}^{-1}$

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

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

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

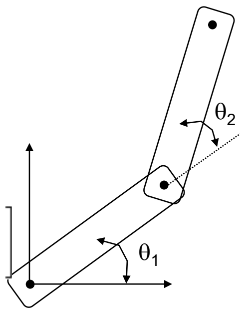
Inverse Jacobian Example

- 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

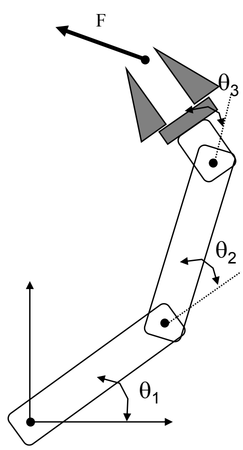
$$\delta\mathbf{X} = \mathbf{J}\delta\theta$$

- So that

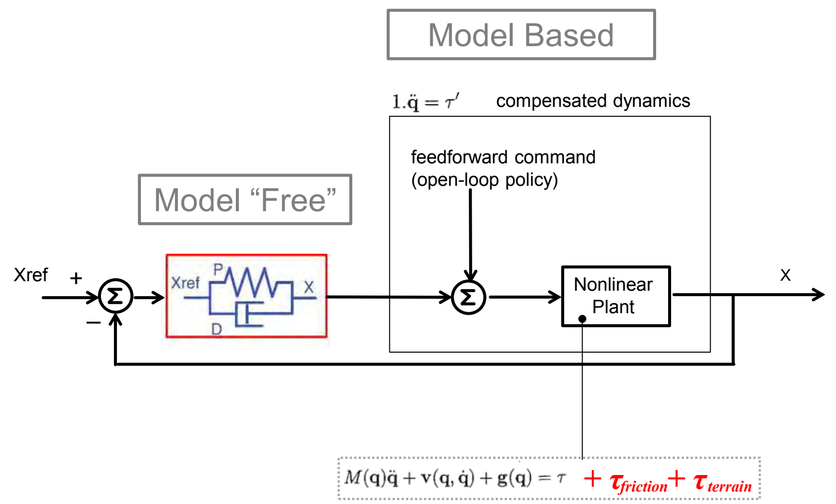
$$F^T \mathbf{J} = \tau^T$$

- Or

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

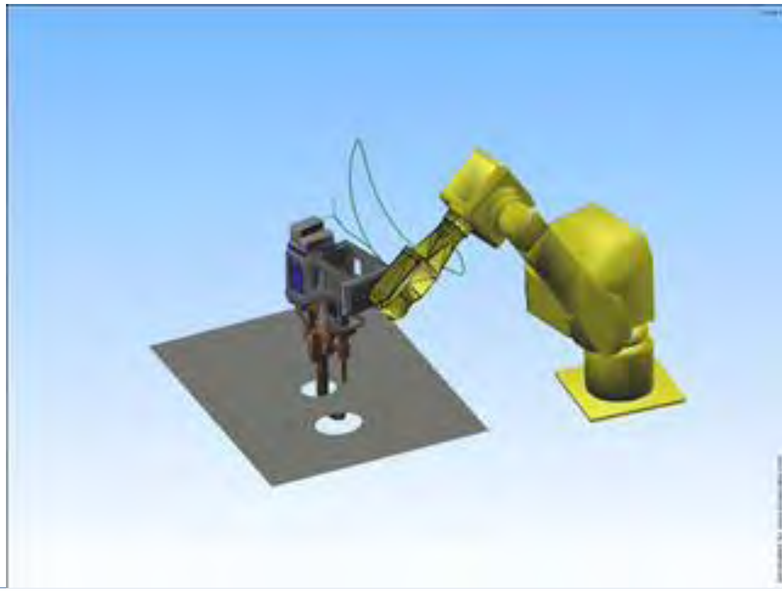


Operation Space (Computed Torque)



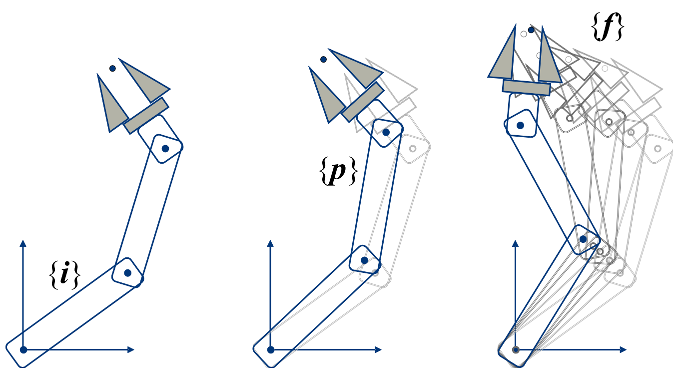
Compensated Manipulation

Trajectory Generation & Planning



Trajectory Generation

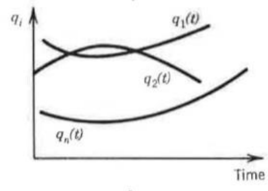
- The goal is to get from an initial position $\{\boldsymbol{i}\}$ to a final position $\{\boldsymbol{f}\}$ via a path points $\{\boldsymbol{p}\}$



Joint Space

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

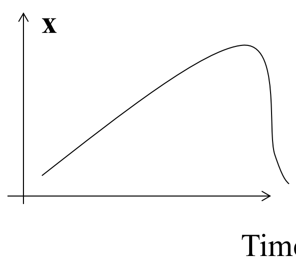
- + Since we control the joints, this is more direct
- -- If we want to follow a particular trajectory, not easy
  - at best lots of intermediate points
  - No guarantee that you can solve the Inverse Kinematics for all path points



Cartesian Workspace

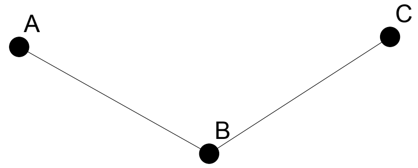
Consider the **Cartesian positions** as a function of time

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



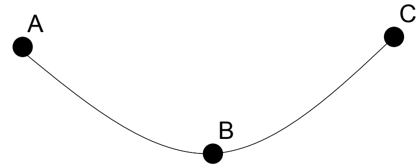
Polynomial Trajectories

- 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/

Robot Sensing: Perception & Linear Observers

METR 4202: **Robotics** & Automation

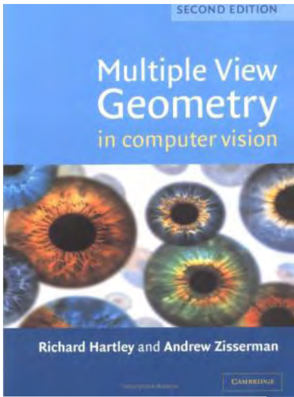
Dr Surya Singh -- Lecture # 6

August 31, 2016

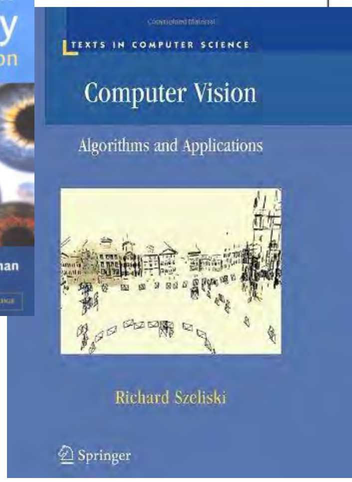
metr4202@itee.uq.edu.au

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

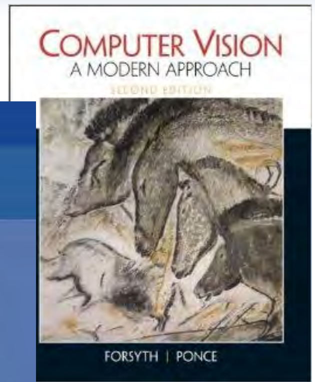
Reference Material



[UQ Library](#)
(ePDF)



[UQ Library/SpringerLink](#)



[UQ Library](#)
(Hardcopy)

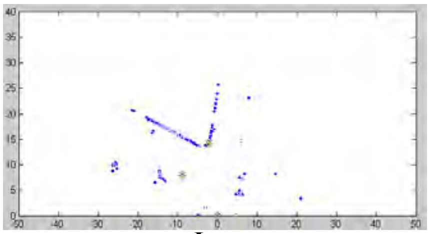
Quick Outline

- Frames
- Kinematics

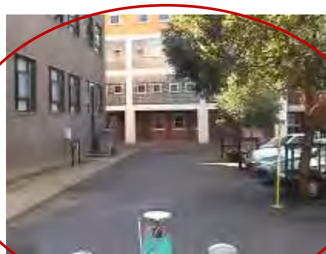
➔ “Sensing Frames” (in space) ➔ Geometry in Vision

1. **Perception ➔ Camera Sensors**
   1. Image Formation
      ➔ “Computational Photography”
   2. Calibration
   3. Features
   4. Stereopsis and depth
   5. Optical flow

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



Laser

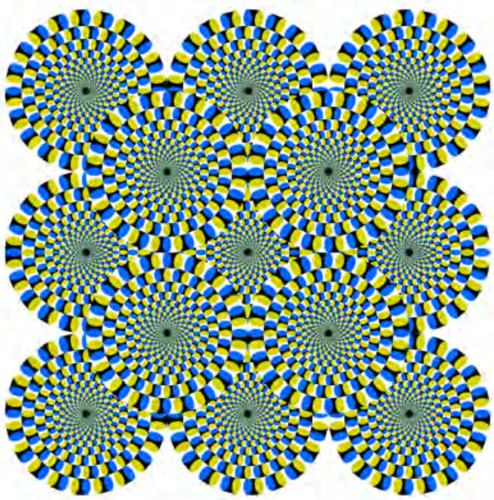


Vision/Cameras

GPS

Perception

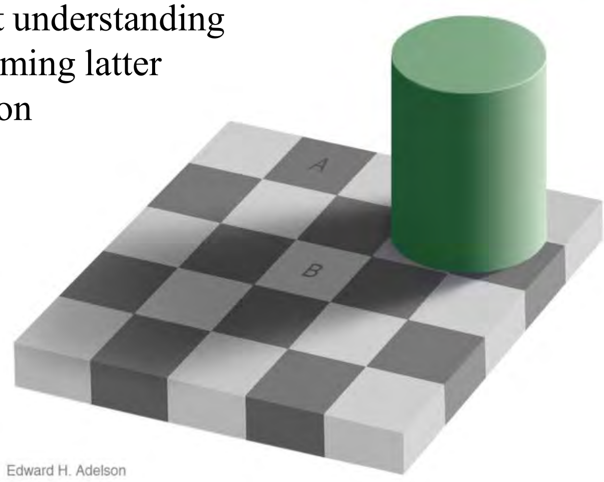
- Making Sense from Sensors



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

Perception

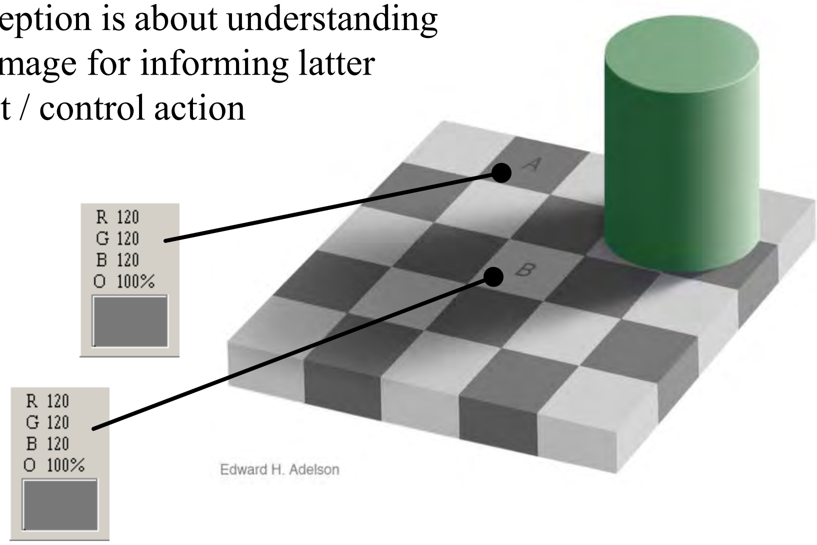
- Perception is about understanding the image for informing latter robot / control action



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

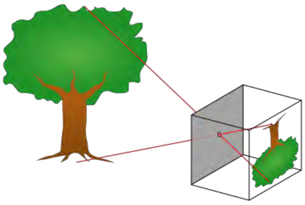
Perception

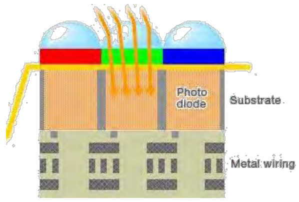
- Perception is about understanding the image for informing latter robot / control action



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

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





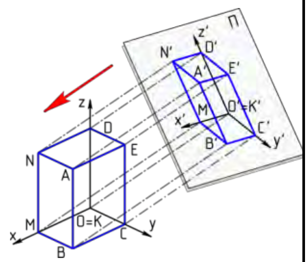
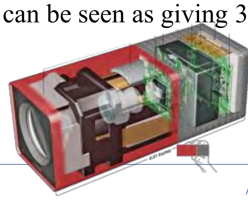


Image Formation → Image Sensing → (Re)Projection

Sources: Wikipedia, Pinhole Camera, Sony sensor, Wikipedia, Graphical projection,

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



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

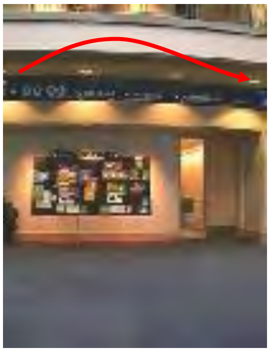
Camera Image Formation "Aberrations"[I]: Lens Optics (Aperture / Depth of Field)

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

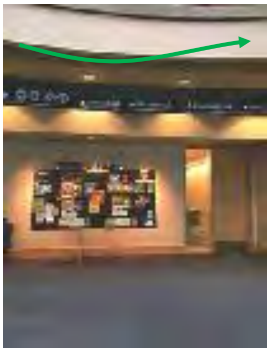
http://en.wikipedia.org/wiki/File:Aperture_in_Canon_50mm_f1.8_II_lens.jpg

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

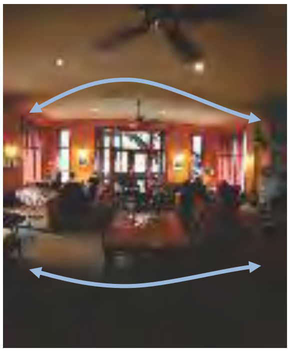
Barrel



Pincushion



Fisheye

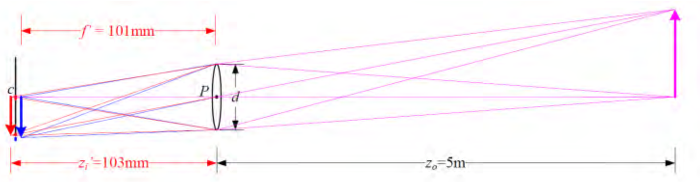


➔ Explore these with `visualize_distortions` in the [Camera Calibration Toolbox]

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

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

- Chromatic Aberration:

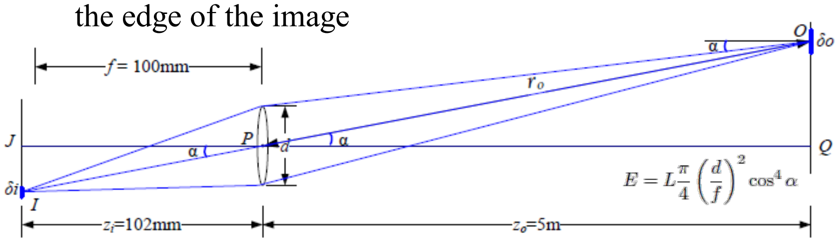


  - In a lens subject to chromatic aberration, light at different wavelengths (e.g., the red and blur arrows) is focused with a different focal length f' and hence a different depth z_i, resulting in both a geometric (in-plane) displacement and a loss of focus

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

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

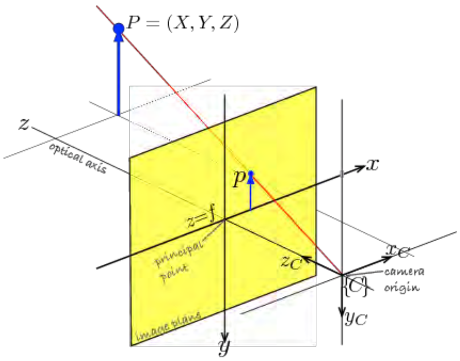
- Vignetting:
  - The tendency for the brightness of the image to fall off towards the edge of the image



  - 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]

Image Formation – Single View Geometry



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

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

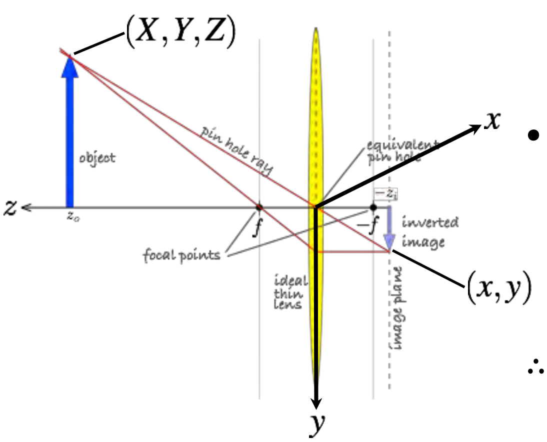
$$(X,Y,Z) \mapsto (x,y)$$

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

Image and Slide from: Corke, Ch. 11

Image Formation: (Thin-Lens) Projection model

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

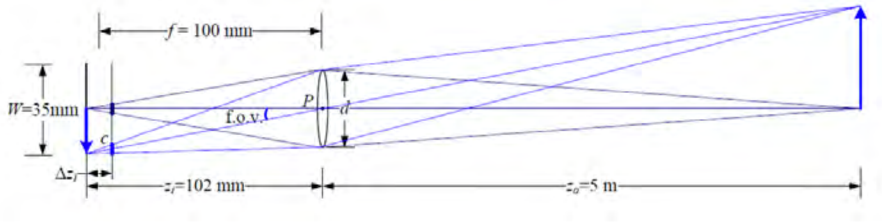


- $\frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f}$

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

Image and Slide from: Corke, Ch. 11

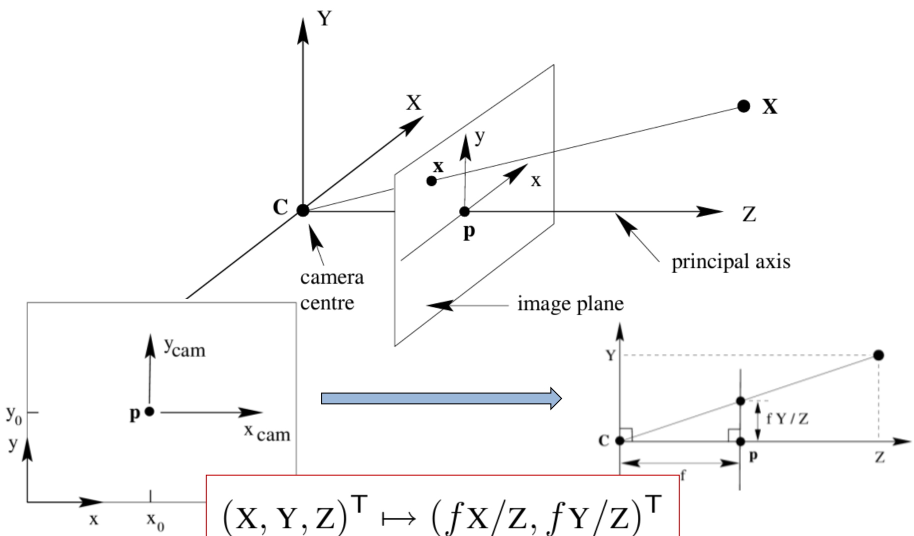
Image Formation: Simple Lens Optics ≅ Thin-Lens



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

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

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

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

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

➔Perspective Camera as:

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

where: P is **3**×4 and of **rank 3**

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

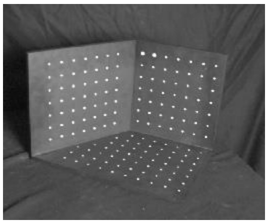
See: *Camera Calibration Toolbox for Matlab*
(http://www.vision.caltech.edu/bouguetj/calib_doc/)

- 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

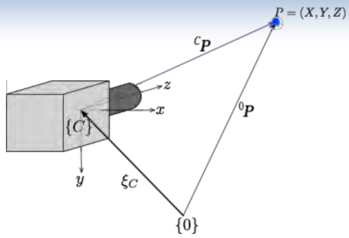
Camera calibration

- 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

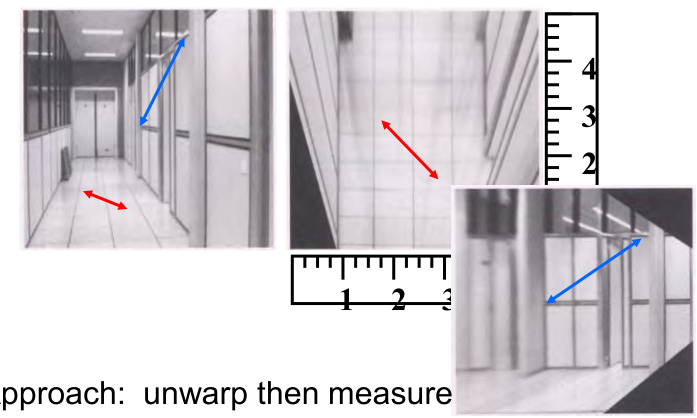
$$\begin{pmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{pmatrix} = \underbrace{\underbrace{\begin{pmatrix}\frac{1}{\rho_u} & 0 & u_0\\ 0 & \frac{1}{\rho_v} & v_0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix} f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}}_{\mathbf{K}}\begin{pmatrix}\mathbf{R} &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

intrinsic parameters

camera matrix

Image and Slide from: Corke, Ch. 11

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



Approach: unwarp then measure

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

Features -- Colour Features

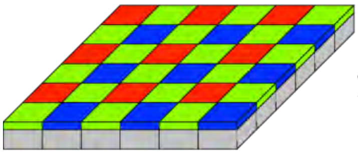
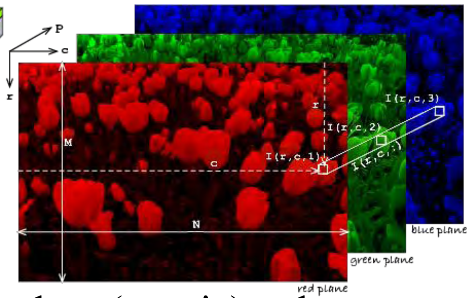


Fig: Ch. 10, *Robotics Vision and Control*

Bayer Patterns



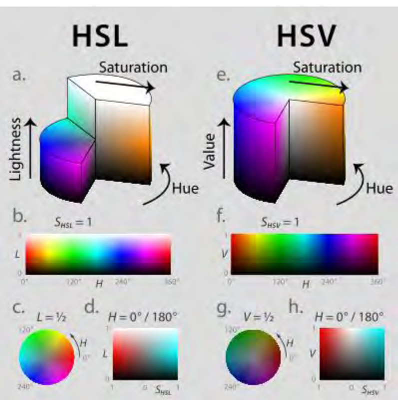
- RGB is **NOT** an absolute (metric) colour space

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)

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

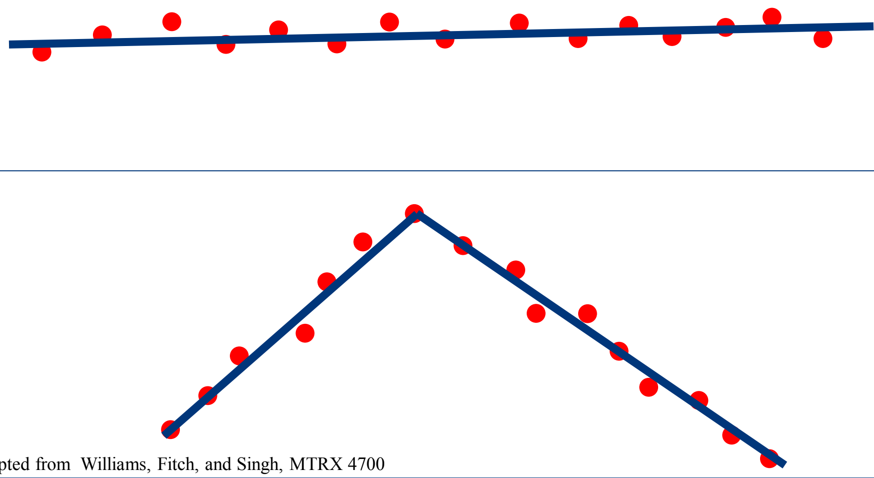
Lines

Edge Detection

- Canny edge detector:
  - Pepsi Sequence:

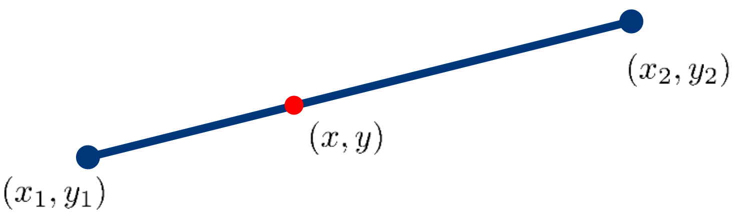
Image Data: http://www.cs.brown.edu/~black/mixtureOF.html and Szeliski, CS223B-L9
See also: Use of Temporal information to aid segmentation:
http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html

Line Extraction and Segmentation



Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Formula

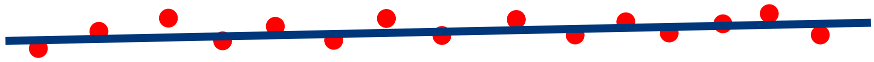


$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

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

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Estimation

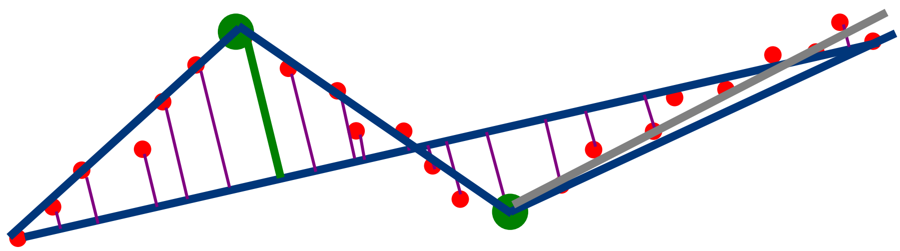


Least squares minimization of the line:

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

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Splitting / Segmentation



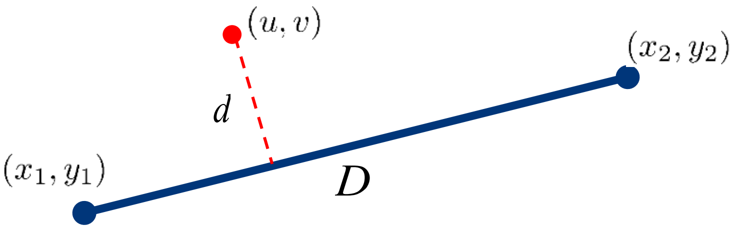
- What about corners?

➔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

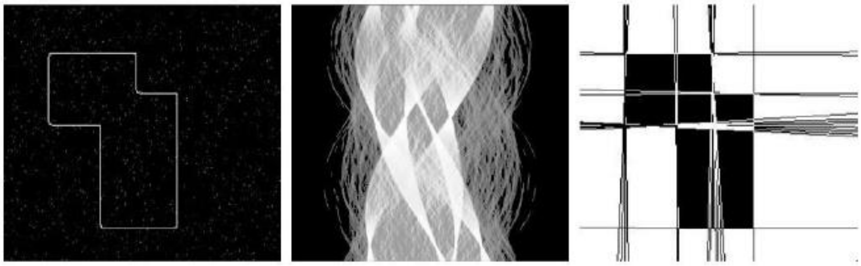


$$r = u(y_1 - y_2) + v(x_2 - x_1) + y_2x_1 - y_1x_2$$

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

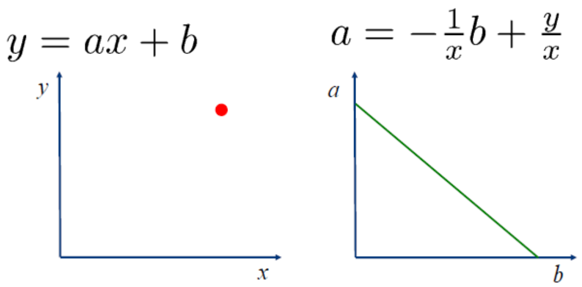
Adopted from Williams, Fitch, and Singh, MTRX 4700

Hough Transform



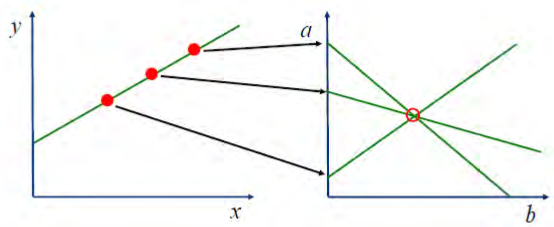
- Uses a voting mechanism
- Can be used for other lines and shapes (not just straight lines)

Hough Transform: Voting Space



- 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.

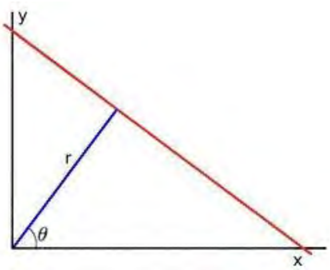
2. Assume that each cell in the parameter space is an accumulator. Initialize all cells to zero.

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:

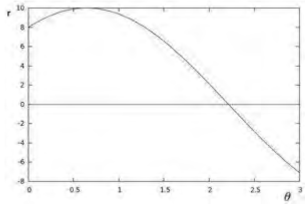
$$r = x_1 \cos\theta + y_1 \sin\theta$$

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

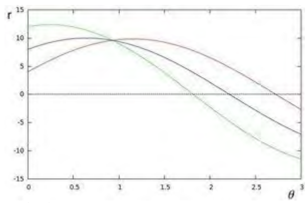
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [2]

- Thus a given point gives a sinusoid



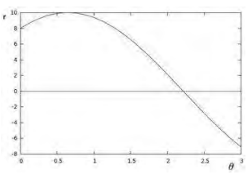
- Repeating for all points on the image



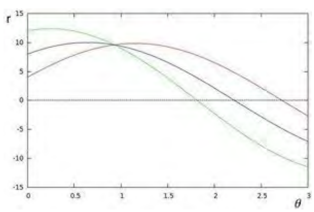
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [3]

- Thus a given point gives a sinusoid



- Repeating for all points on the image

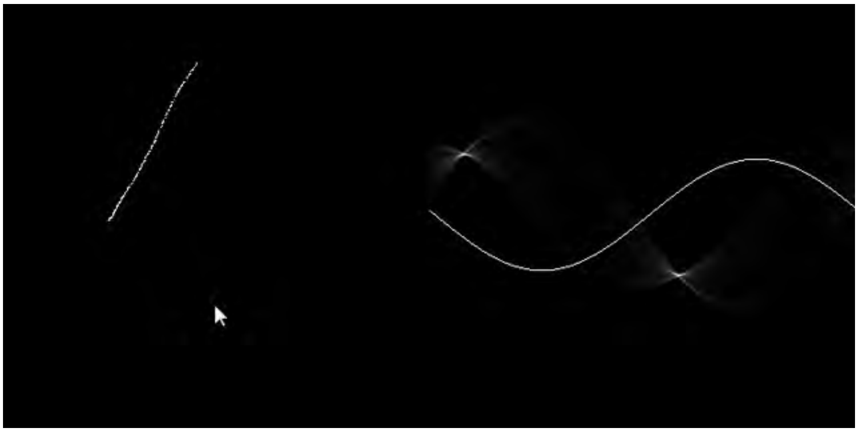


- NOTE that an intersection of sinusoids represents (**a point**) represents **a line** in which pixel points lay.

➔ 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

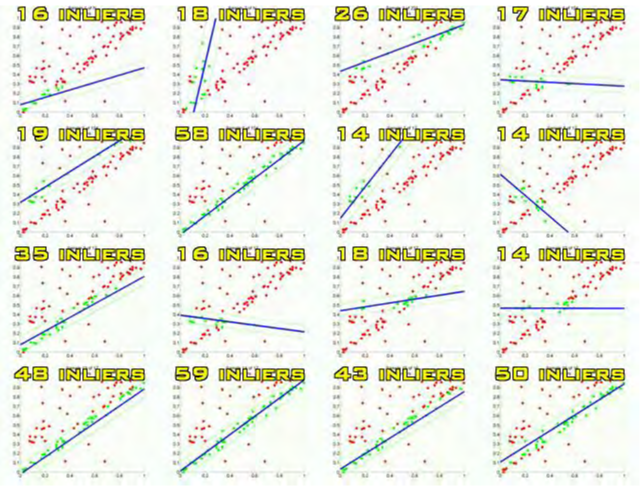
RANdom SAmple Consensus

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*

Robot Sensing: Multiple View Geometry & Feature Detection

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 7

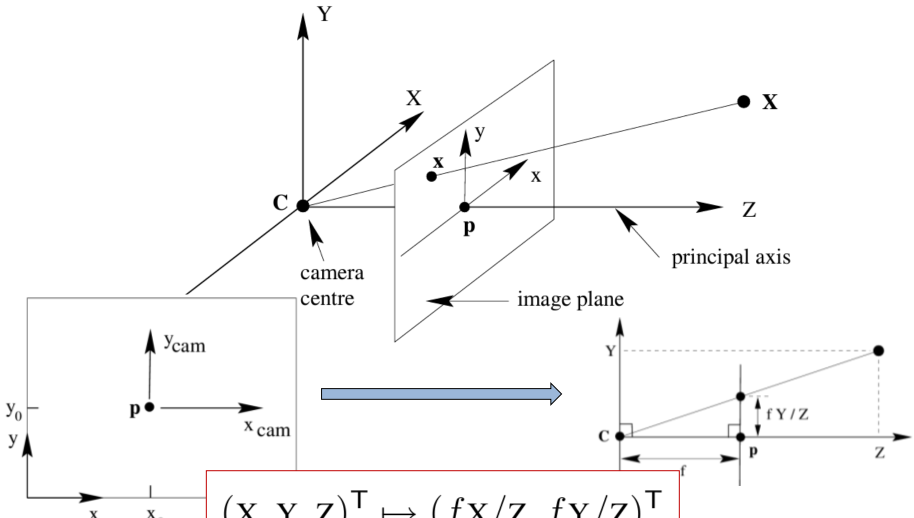
September 7, 2016

metr4202@itee.uq.edu.au

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

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

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

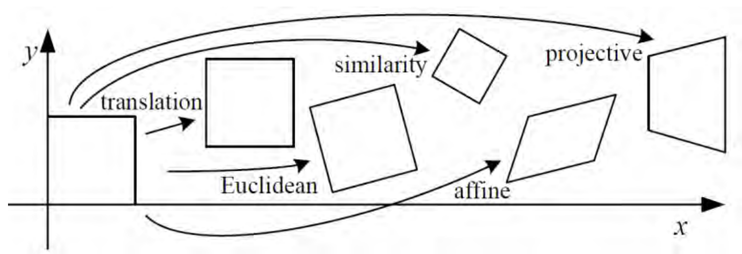
$$\mathbf{x} = K[I \mid \mathbf{0}]\mathbf{X}_{\mathrm{cam}}.$$

➔Perspective Camera as:

where: P is **3**×4 and of **rank 3**

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

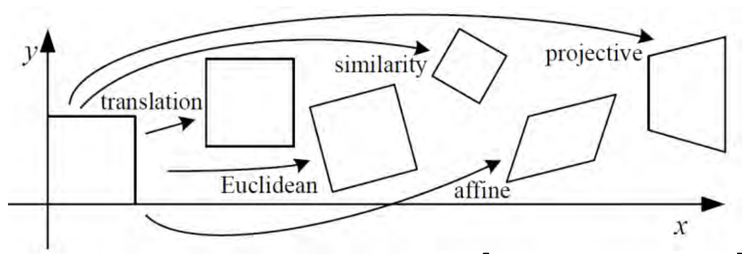
Transformations ☆



- ***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

2-D Transformations

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

3D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} I & t \end{array}\right]_{3\times 4}$	3	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} R & t \end{array}\right]_{3\times 4}$	6	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} sR & t \end{array}\right]_{3\times 4}$	7	angles $+\cdots$	
affine	$\left[\begin{array}{c} A \end{array}\right]_{3\times 4}$	12	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{H} \end{array}\right]_{4\times 4}$	15	straight lines	

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

Projection Models

- Orthographic

- Weak Perspective

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

- Affine

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

- Perspective

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

- Projective

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

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

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

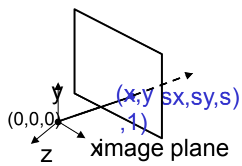
Properties of Projection

- 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

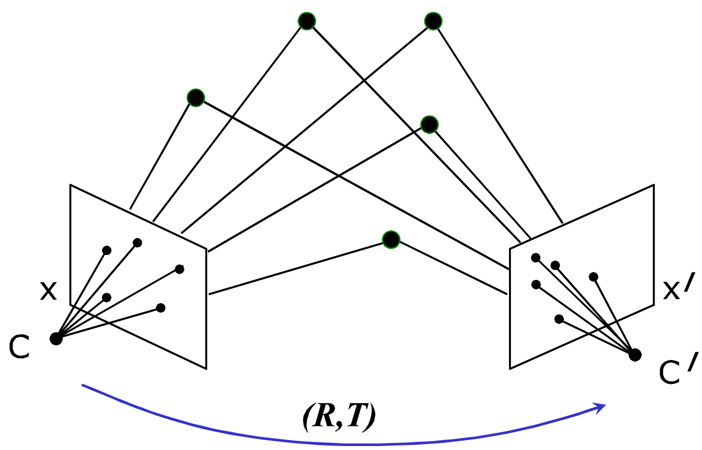
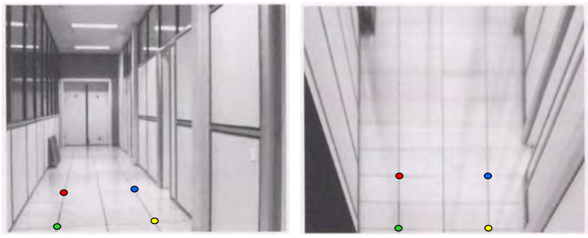


Image Rectification



To unwarp (rectify) an image

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

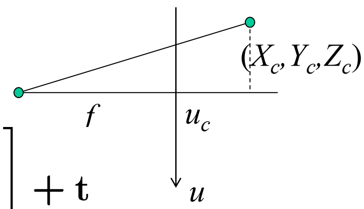
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

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):

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

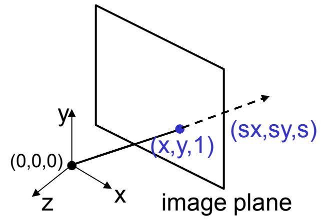
It's useful to decompose Π into **T** → **R** → project → **A**

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

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

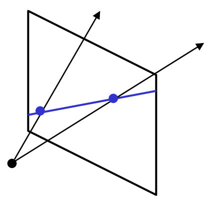


- Each *point* (x,y) on the plane is represented by a *ray* (sx,sy,s)
  - all points on the ray are equivalent: (x, y, 1) ≅ (sx, sy, s)

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

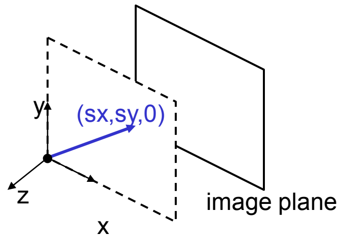
$$\text{in vector notation:} \quad 0 = \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{\mathbf{l}^T} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{p}}$$

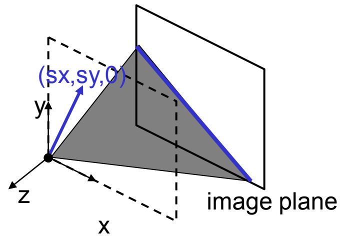
- A line is represented as a homogeneous 3-vector **l**

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

Ideal points and lines

- Ideal point ("point at infinity")
  - $p \cong (x, y, 0)$ – parallel to image plane
  - It has infinite image coordinates



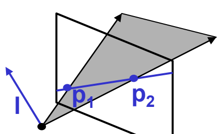


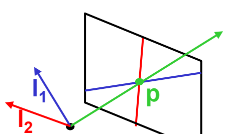
Line at infinity

- $l_\infty \cong (0, 0, 1)$ – parallel to image plane
- Contains all ideal points

Point and Line Duality

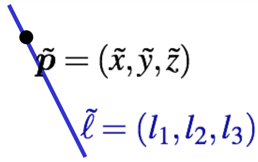
- A line l is a homogeneous 3-vector (a ray)
- It is ⊥ to every point (ray) p on the line: lT p=0





- 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:

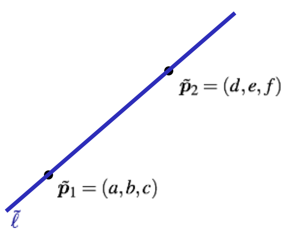
  $\tilde{\boldsymbol{P}} = (\tilde{x}, \tilde{y}, \tilde{z}) \quad | \quad \boldsymbol{P} = (x, y) \qquad x = \frac{\tilde{x}}{\tilde{z}},\ y = \frac{\tilde{y}}{\tilde{z}}$

- 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

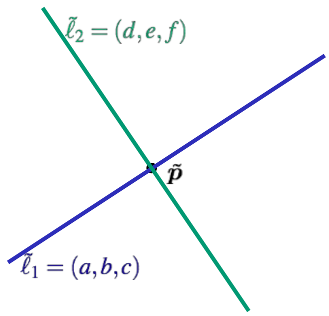
Point and Line Duality [III]

- 2 Points Make a Line



$$\tilde{\ell} = \tilde{\boldsymbol{p}}_1 \times \tilde{\boldsymbol{p}}_2$$

- 2 Lines Make Point!

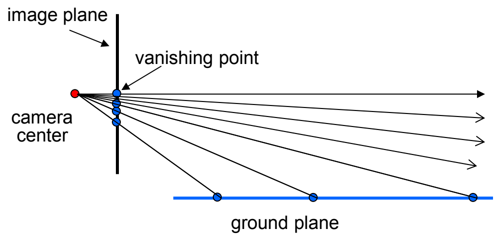


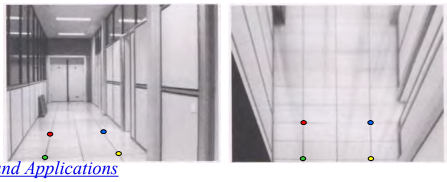
$$\tilde{\boldsymbol{p}} = \tilde{\ell}_1 \times \tilde{\ell}_2$$

Image/Notation from: Corke, Ch. 11

Vanishing Points

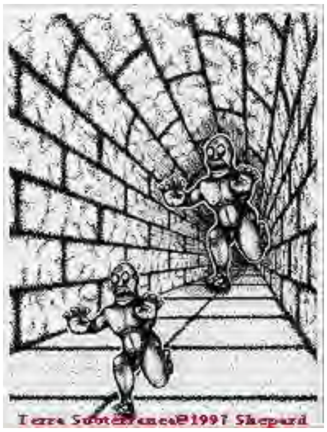
- Vanishing point
  - projection of a point at infinity
  - whiteboard capture, architecture,…





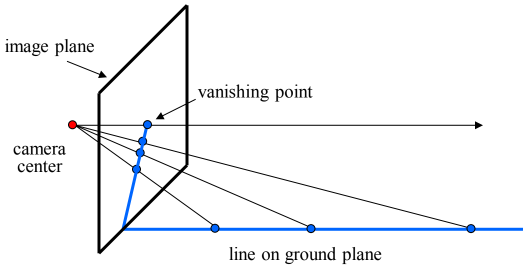
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Fun With Vanishing Points

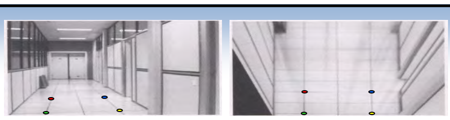


Slide from Szeliski, *Computer Vision: Algorithms and Applications*

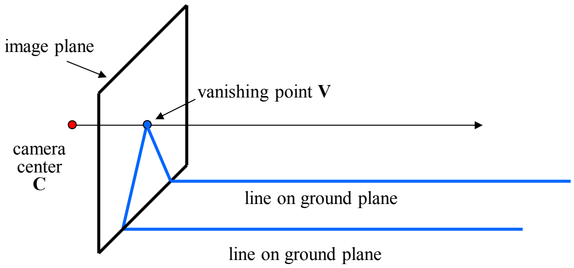
Vanishing Points (2D)



Vanishing Points

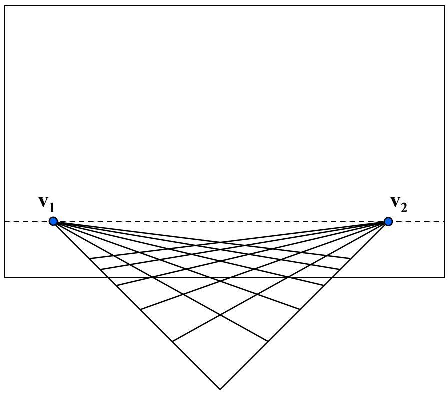


- Properties
  - Any two parallel lines have the same vanishing point
  - The ray from C through v point is parallel to the lines
  - An image may have more than one vanishing point

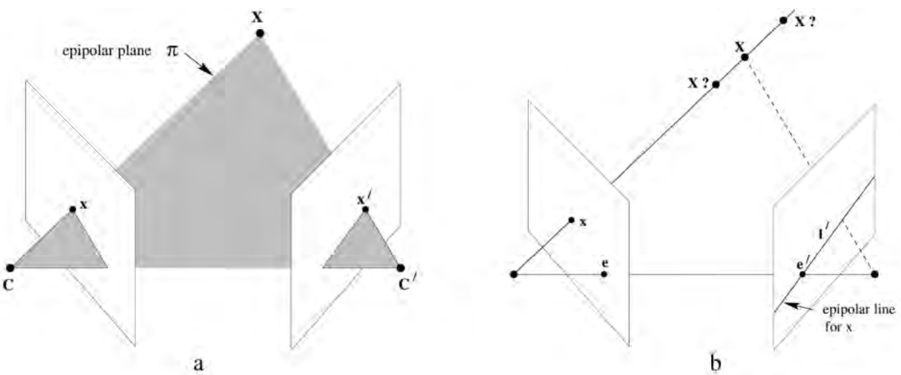


Vanishing Lines

- 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]()

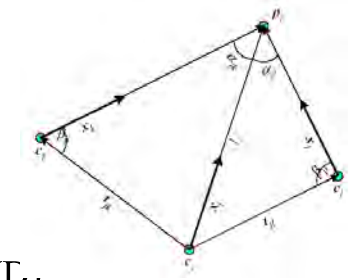
Fundamental matrix

- Camera calibrations are unknown
- $x' F x = 0$ with $F = [e]_\times H = K'[t]_\times R K^{-1}$
- Solve for F using least squares (SVD)
  - re-scale (xi, xi') so that $|xi| \approx 1/2$ [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

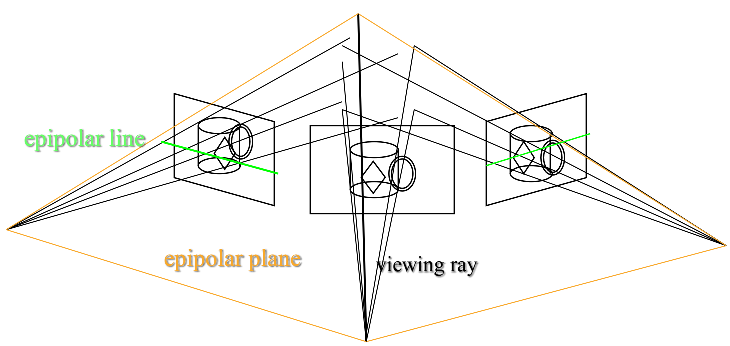
- Co-planarity constraint:
- $x' \approx R\,x + t$
- $[t]_\times x' \approx [t]_\times R\,x$
- $x'^T [t]_\times x' \approx x'^T [t]_\times R\,x$
- $x'^T E\,x = 0$ with $E = [t]_\times R$
- Solve for E using least squares (SVD)
- t is the least singular vector of E
- R obtained from the other two s.v.s



From Szeliski, *Computer Vision: Algorithms and Applications*

Stereo: Epipolar geometry

- Match features along epipolar lines



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

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.

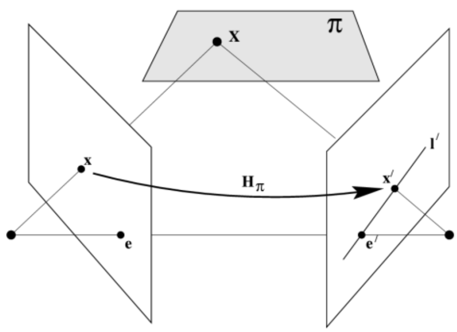


Fig. 9.5. *A point* $\mathbf{x}$ *in one image is transferred via the plane* π *to a matching point* $\mathbf{x}'$ *in the second image. The epipolar line through* $\mathbf{x}'$ *is obtained by joining* $\mathbf{x}'$ *to the epipole* $\mathbf{e}'$*. In symbols one may write* $\mathbf{x}' = \mathtt{H}_\pi \mathbf{x}$ *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 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:

$$\mathbf{e} = P\begin{pmatrix} -R^\mathsf{T}\mathbf{t} \\ 1 \end{pmatrix} = KR^\mathsf{T}\mathbf{t} \qquad \mathbf{e}' = P'\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = K'\mathbf{t}.$$

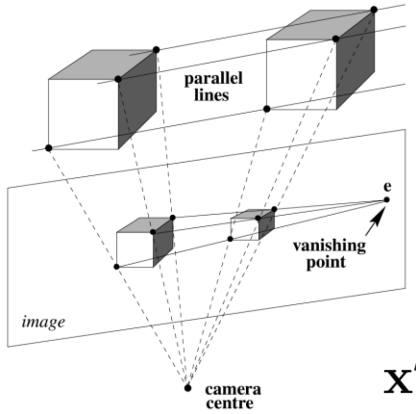
$\therefore$

$$F = [\mathbf{e}']_\times K'RK^{-1} = K'^{-\mathsf{T}}[\mathbf{t}]_\times RK^{-1} = K'^{-\mathsf{T}}R[R^\mathsf{T}\mathbf{t}]_\times K^{-1} = K'^{-\mathsf{T}}RK^\mathsf{T}[\mathbf{e}]_\times$$

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 $\mathbf{x}'^\mathsf{T}F\mathbf{x} = 0$.
- **Epipolar lines**:
  - ◇ $\mathbf{l}' = F\mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$.
  - ◇ $\mathbf{l} = F^\mathsf{T}\mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$.
- **Epipoles**:
  - ◇ $F\mathbf{e} = \mathbf{0}$.
  - ◇ $F^\mathsf{T}\mathbf{e}' = \mathbf{0}$.
- **Computation from camera matrices** P, P':
  - ◇ General cameras, $F = [\mathbf{e}']_\times P'P^+$, where P^+ is the pseudo-inverse of P, and $\mathbf{e}' = P'\mathbf{C}$, with $P\mathbf{C} = \mathbf{0}$.
  - ◇ Canonical cameras, $P = [I \mid \mathbf{0}]$, $P' = [M \mid \mathbf{m}]$, $F = [\mathbf{e}']_\times M = M^{-\mathsf{T}}[\mathbf{e}]_\times$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = M^{-1}\mathbf{m}$.
  - ◇ Cameras not at infinity $P = K[I \mid \mathbf{0}]$, $P' = K'[R \mid \mathbf{t}]$, $F = K'^{-\mathsf{T}}[\mathbf{t}]_\times RK^{-1} = [K'\mathbf{t}]_\times K'RK^{-1} = K'^{-\mathsf{T}}RK^\mathsf{T}[KR^\mathsf{T}\mathbf{t}]_\times$.

Fundamental Matrix & Motion

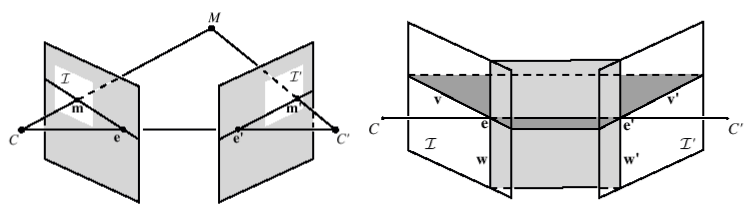


$$\mathbf{x}'^{\mathsf{T}}\mathbf{F}\mathbf{x} = 0.$$

- 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.

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**

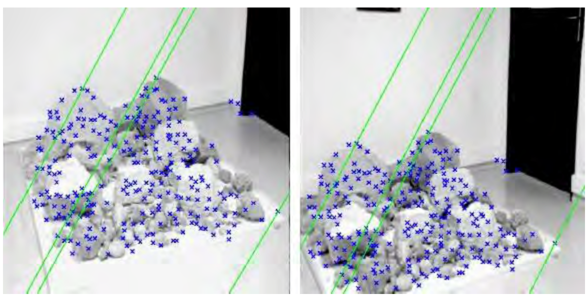
- ~~Colour~~
- Corners
- Edges
- Lines
- Statistics on Edges: SIFT, SURF, ORB…

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))

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

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

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

$$W = R_{2F\times 3}\, S_{3\times P}$$

SVD

$$W = U \Lambda V \quad \Lambda \text{ must be rank 3}$$

$$W' = (U \Lambda^{1/2})(\Lambda^{1/2} V) \quad = U' V'$$

Make R orthogonal

$$R = QU', \;\; S = Q^{-1}V'$$

$$i_f^T Q^T Q i_f = 1 \;\ldots$$

From Szeliski, Computer Vision: Algorithms and Applications

Results

- Look at paper figures…

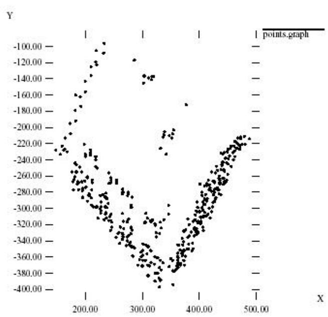


Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

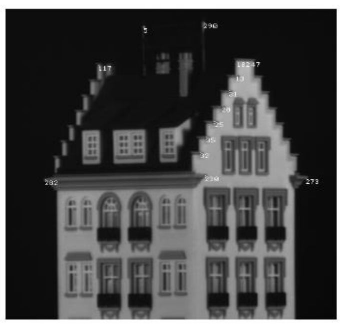


Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

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!