# Lecture Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture (M: 2:05p-3:50, 50-N202)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23-Jul</td>
<td>Introduction + Representing Position &amp; Orientation &amp; State</td>
</tr>
<tr>
<td>2</td>
<td>30-Jul</td>
<td>Robot Kinematics: Frames, Transformation Matrices &amp; Affine Transformations</td>
</tr>
<tr>
<td>3</td>
<td>6-Aug</td>
<td>Robot Kinematics: Forward, Inverse &amp; Differential Kinematics</td>
</tr>
<tr>
<td>4</td>
<td>13-Aug</td>
<td>Robot Dynamics: Jacobians &amp; Joint Torque</td>
</tr>
<tr>
<td>5</td>
<td>20-Aug</td>
<td>Robot Sensing: Linear Observers → Single View Geometry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS 1 [15%, Friday! — 24/Aug]</td>
</tr>
<tr>
<td>6</td>
<td>27-Aug</td>
<td>Robot Sensing: Multiple View Geometry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quiz [15%, 27/Aug]</td>
</tr>
<tr>
<td>7</td>
<td>3-Sep</td>
<td>Robot Sensing: Feature Detection &amp; Direct Methods</td>
</tr>
<tr>
<td>8</td>
<td>10-Sep</td>
<td>Motion Planning (Deterministic)</td>
</tr>
<tr>
<td>9</td>
<td>17-Sep</td>
<td>Motion Planning (Probabilistic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PS 2 [20%, 21/Sept]</td>
</tr>
<tr>
<td></td>
<td>24-Sep</td>
<td>(Spring) Study Break</td>
</tr>
<tr>
<td>10</td>
<td>1-Oct</td>
<td>Queen’s Birthday!</td>
</tr>
<tr>
<td>11</td>
<td>8-Oct</td>
<td>Probabilistic Robotics: Planning &amp; Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LQR, Value Functions, Q-Learning, etc.)</td>
</tr>
<tr>
<td>12</td>
<td>15-Oct</td>
<td>Probabilistic Robotics: Localization &amp; SLAM</td>
</tr>
<tr>
<td>13</td>
<td>22-Oct</td>
<td>The Future of Robotics/Automation + Open Challenges</td>
</tr>
</tbody>
</table>
Robot Dynamics (continued!)

Reference Material

- On class webpage
  - Password: metr4202
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
    \[ q = [q_1 \ \ldots \ \ q_n] \rightarrow [\theta_1(t) \ \theta_2(t) \ \theta_3(t)] \]
  - Outputs are joint torques to be applied at each instance
    \[ \tau = [\tau_1 \ \ldots \ \tau_n] \]

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

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

\[ \dot{\theta} = J(\theta)^{-1} \dot{X} \]

- 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 \( \theta_2 \) approaches 0 or \( \pi \) 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
  \[ F \cdot \delta X = \tau \cdot \delta \theta \]
- Or
  \[ F^T \delta X = \tau^T \delta \theta \]
- Earlier we saw that
  \[ \delta X = J \delta \theta \]
- So that
  \[ F^T J = \tau^T \]
- Or
  \[ \tau = J^T F \]
Operation Space (Computed Torque)

Model Based

\[ \dot{q} = \tau' \]

compensated dynamics

feedforward command
(open-loop policy)

Model “Free”

\[ M(q)\ddot{q} + \nabla(q, \dot{q}) + g(q) = \tau + \tau_{\text{friction}} + \tau_{\text{terrain}} \]

Cool Robotics Share: Compensated Manipulation
Dynamics of Parallel Manipulators

• Traditional Newton-Euler formulation:
  – Equations of motion to be written once for each body of a manipulator
  – Large number of equations

• Lagrangian formulation
  – eliminates all of the unwanted reaction forces and moments at the outset.
  – It is more efficient than the Newton- Euler formulation
  – Numerous constraints imposed by closed loops of a parallel manipulator

• To simplify the problem
  – Lagrangian Multipliers are often introduced
  – Principle of virtual work
Trajectory Generation

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

Polynomial Trajectories

- Straight line Trajectories
- Polynomial Trajectories

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

\[ u(t) = a_0 + a_1t + a_2t^2 + a_3t^3 \]
Trajectory Control in 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

---

Trajectory Control in Cartesian Workspace

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

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

---
Dynamic Simulation Software

- v-rep
  virtual robot experimentation platform

- Reflexxes

http://www.coppeliarobotics.com/  
http://www.reflexxes.com/

CRS²: Trajectory Generation & Planning (VREP Example)
Summary

• Kinematics is the study of motion without regard to the forces that create it

• Kinematics is important in many instances in Robotics

• The study of dynamics allows us to understand the forces and torques which act on a system and result in motion

• Understanding these motions, and the required forces, is essential for designing these systems

✯ – Quiz 1 Assessable Demarkation – ✯
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

Reference Material

UQ Library/ SpringerLink

UQ Library (ePDF)
Sensor Information: (not only) Cameras!

Okay, Cameras Are Common! [1]
Okay, (Smartphone) Cameras Are Common! [2]


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

Image Formation  Image Sensing  (Re)Projection

* Well Almost… RGB-D and Light-Field cameras can be seen as giving 3D ⇒ 3D
Camera Image Formation: $3D \mapsto 2D \Rightarrow \text{Loose Scale!}$

Source: [https://www.flickr.com/groups/nasa-eclipse2017/](https://www.flickr.com/groups/nasa-eclipse2017/)  Eclipses, perhaps mythical, also show that 2D Vision is “up to scale” 😊

Why? ∴ Image formation is a $3D \mapsto 2D$ Mapping

Source: Wikipedia, Camera obscura, Emoticon inspired by a slide from Peter Corke
More Deeply: Part of a **Computational Imaging** Pipeline

- Intensity $\ell(.)$
- Position (3)
- Direction (2)
- Time (1)
- Wavelength (1)
- Polarization (1)

Source: Donald Dansereau

---

Seeing it Generally as “Computational Imaging” …

**Example 1:** Flutter Shutter

[raskar2006]
Seeing it Generally as “Computational Imaging” …

- **Example 2:** Coded Aperture  

  [gottesman89, levin2007, zhou2009 and others]

Camera **MODELS!**

(not those found in a shop 😃)
Cameras

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

\[ N = \frac{f}{\#} = \frac{f}{d} \]

Large Aperture $\rightarrow$ “Fast” (why?)

- A pinhole lets little light through & the original chemistry wasn’t very sensitive
  - one has to wait to collect photons
  - UNTIL a large aperture/multifocal lens was invented for this & sold as “fast Daguerreotypes”
- But this introduces depth of field (DOF)

Fig. Daguerreotype, Wikipedia  
DOF Applet at: http://graphics.stanford.edu/courses/cs178/applets/dof.html

Image Formation: Simple Lens Optics $\approx$ 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: (Thin-Lens) Projection model

\[ x = \frac{fX}{z}, \ y = \frac{fY}{z} \]

\[ \frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f} \]

\[ \therefore \text{ as } z_0 \to \infty, z_i \to f \]

Image and Slide from: Corke, Ch. 11

Image Formation – Single View Geometry [I]

\[ (X, Y, Z)^T \mapsto \left( \frac{fx}{Z}, \frac{fY}{Z} \right)^T \]

Hartley & Zisserman, Ch. 6
2-D Transformations

\[ x' = \text{point in the new (or 2^{nd}) image} \]
\[ x = \text{point in the old image} \]

- 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

Transformations

\[ x': \text{New Image} \quad \& \quad x: \text{Old Image} \]

- Euclidean: \[ x' = \begin{bmatrix} R & t \end{bmatrix} x \] (Distances preserved)
- Similarity (Scaled Rotation): \[ x' = \begin{bmatrix} sR & t \end{bmatrix} x \] (Angles preserved)
Transformations [2]

- **Affine**: (|| lines remain ||)
  \[ x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} x \]

- **Projective**: (straight lines preserved)
  \[ x' = H x \]
  \[ x' = \begin{bmatrix} h_{00}x + h_{01}y + h_{02} \\ h_{10}x + h_{11}y + h_{12} \\ h_{20}x + h_{21}y + h_{22} \end{bmatrix} \]
  \[ y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}} \]

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

Image Formation – Single View Geometry [II]

**Camera Projection Matrix**

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

- \( x = \text{Image point} \)
- \( X = \text{World point} \)
- \( K = \text{Camera Calibration Matrix} \)

**Perspective Camera as:**

\[ P = K[R \ | \ t] \]

where: \( P \) is 3×4 and of **rank 3**
Camera Calibration!

Calibration matrix

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

\[
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix}
\sim
\begin{bmatrix}
  f & 0 & u_c \\
  0 & f & v_c \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X_c \\
  Y_c \\
  Z_c
\end{bmatrix}
= K \begin{bmatrix}
  X_c
\end{bmatrix}
\]

\[
\begin{bmatrix}
  f a & s & u_c \\
  0 & f & v_c \\
  0 & 0 & 1
\end{bmatrix}
= K
\]

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

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

\[
\begin{pmatrix}
\tilde{u} \\
\tilde{v} \\
\tilde{w}
\end{pmatrix} =
\begin{pmatrix}
\frac{1}{\rho_u} & 0 & u_0 \\
0 & \frac{1}{\rho_v} & v_0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
R \\
0_{1x3} \\
t \\
1
\end{pmatrix}^{-1}
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\]

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

- Barrel
- Pincushion
- Fisheye

鹄 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 \( \delta i \) 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 \( \alpha \), \( \cos^4 \alpha \).

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

Image Features & Perception: A Picture is a $10^3$ Words

- 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
Features -- Colour Features

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**
  
  ➔ Gamma Corrected Luma (Y) + Chrominance
  
  ➔ BW ➔ Colour TVs : Just add the Chrominance
  
  ➔ γ Correction: CRTs γ=2.2-2.5

  Y' = 16 + 0.481 · R' + 128.5 · G' + 24.966 · B'
  
  C_y = 128 + (-37.797 · R' - 74.203 · G' + 112.0 · B')
  
  C_b = 128 + (112.0 · R' - 93.786 · G' - 18.214 · B')

- **YCrCb**

- **L*ab**

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

- \( C = W - R \)
- \( M = W - G \)
- \( Y = W - B \)
- \( K = -W \)

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

- Start with RGB \( \rightarrow \) CIE XYZ
  (Under \textbf{Illuminant D65})

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

Colour: Illumination Variant

- Toy Image
- Toy Image \textbf{With Flash}

Source: \%MATLABROOT\%\toolbox\images\imdata\toysflash.png
Colour Spaces:
- Red | Green | Blue :
- Hue | Saturation | V (Brightness Value) :

“False-colour”: Show HSV as “RGB”

CRS: Mapping: Indoor robots

ACFR, IROS 2002