From Last Class

- Homogeneous Transformations
  - Combines Rotation + Translation into one single matrix multiplication
  - Composition of Homogeneous Transformations

Homogeneous Representation

- Representation of points & vectors

\[
\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ 1 \end{bmatrix}, \quad \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}
\]

- Properties
  1. Sum & differences of vectors are vectors
  2. Sum of a vector and a point is a point
  3. Difference between two points is a vector
  4. Sum of two points == meaningless

Homogeneous Transformations

\[
q^D = \begin{bmatrix} R_B^A & p_B^A \\ 0 & 1 \end{bmatrix} q^A
\]

\[
q^C = \begin{bmatrix} R_C^B & p_C^B \\ 0 & 1 \end{bmatrix} q^D
\]

\[
q^C = \begin{bmatrix} R_C^B & p_C^B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_B^A & p_B^A \\ 0 & 1 \end{bmatrix} q^A
\]

From Last Class

- Robot Velocity Kinematics
  - Forward vs. Inverse Kinematics
  - Holonomic vs. Non-holonomic Constraints
  - Differential Drive
  - Tricycle
  - Ackerman Steering
Robot Velocity Kinematics

- **Forward Kinematics:**
  \[ \mathbf{q} = f(\phi_0, \phi_1, \ldots, \phi_n) \]

- **Inverse Kinematics:**
  \[ \begin{bmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_n \end{bmatrix} = f(x, y, \theta) \]

Non-Holonomic Constraints

The robot can reach everywhere in the configuration space, BUT, it is under-actuated, and thus the velocity is constrained.

Testing for Integrability of Constraints

- **Given a constraint in the form of**
  \[ Pdx + Qdy + Rdz = 0 \]
- **The test:**
  \[ \frac{P}{Q} \]
  \[ \text{then, if } \nabla \times \mathbf{v} = \mathbf{c} \rightarrow \text{constraint can be integrated} \]

Mobile Robot Drives

- **Differential Steering**
  > Co-axial wheels
  > Independently driven
  > Two-dimensional
  > Non-holonomically constrained

Differential Steering: Forward Kinematics

Given the robot geometry and wheel speeds, what is the robot's velocity?

Let:
- \( r \) - wheel radius
- \( l \) - axle length
- \( \dot{\phi}_R \) - right wheel speed
- \( \dot{\phi}_L \) - left wheel speed

We want

\[ \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(r, l, \dot{\phi}_R, \dot{\phi}_L) \]
Differential Steering: Forward Kinematics (cont.)

Given the robot geometry and wheel speeds, what is the robot’s velocity?

Let:
- $r$ – wheel radius
- $l$ – axle length
- $\dot{\phi}_R$ – right wheel speed
- $\dot{\phi}_L$ – left wheel speed

Goal

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} = f(r, l, \dot{\phi}_R, \dot{\phi}_L)
\]

Forward Velocity

\[
v_f = r (\dot{\phi}_R - \dot{\phi}_L)
\]

Angular Velocity

\[
\dot{\theta} = \frac{r}{l} (\dot{\phi}_R - \dot{\phi}_L)
\]

Differential Steering: Inverse Kinematics

- Instantaneous Center of Curvature

\[
R_{ICC} = \frac{1}{2} \dot{\phi}_R + \dot{\phi}_L
\]

World Coordinates

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} = \begin{bmatrix}
\frac{r}{2} \cos \theta & \frac{r}{2} \cos \theta & \frac{r}{2} \\
\frac{r}{2} \sin \theta & -\frac{r}{2} \sin \theta & \frac{r}{2} \\
-\frac{l}{r} & \frac{l}{r} & \frac{l}{r}
\end{bmatrix} \begin{bmatrix}
\dot{\phi}_L \\
\dot{\phi}_R
\end{bmatrix}
\]

Differential Steering: Inverse Kinematics

\[
\begin{align*}
\dot{\phi}_L &= \frac{1}{2r} (v_f - \frac{l}{2} \dot{\theta}) \\
\dot{\phi}_R &= \frac{1}{2r} (v_f + \frac{l}{2} \dot{\theta})
\end{align*}
\]
Differential Steering

• Benefits
  ➢ Simple construction
  ➢ Zero minimum turning radius

• Drawbacks
  ➢ Small error in wheel speeds translates to large position errors
  ➢ Requires two drive motors
  ➢ Wheels-first is dynamically unstable

Tricycle: Forward Kinematics
Steerable powered front wheel
Free-spinning rear wheels

\[
\begin{align*}
  & r \quad \text{front wheel radius} \\
  & d \quad \text{wheelbase} \\
  & \dot{\phi}_f \quad \text{front wheel speed} \\
\end{align*}
\]

Forward Velocity
\[
\psi_f = r \dot{\phi}_f \cos \alpha
\]

Angular Velocity
\[
\dot{\theta} = \frac{r}{d} \dot{\phi}_f \sin \alpha
\]

Tricycle: Instantaneous Center of Curvature

\[
R_{ICC} = \frac{d}{\tan \alpha} = \frac{v_f}{\dot{\theta}}
\]

Tricycle: Inverse Kinematics
What wheel speed and angle are necessary to produce a desired robot velocity?

\[
\begin{align*}
  & \alpha = \tan^{-1} \left( \frac{\theta_d}{\dot{v}_f} \right) \\
  & \dot{\phi}_f = \frac{1}{r} \sqrt{\frac{\partial^2}{v_f^2} + \dot{\theta}^2 \dot{d}^2}
\end{align*}
\]
**Tricycle**

- **Benefits**
  - Does not require accurate speed matching
- **Drawbacks**
  - Non-zero minimum turning radius
  - More complicated powertrain

---

**Sensing Light**

- How do we see the world?
- Place a piece of film in front of an object, can we get a reasonable image?

---

**Pinhole Camera**

- Let's add a barrier, w/ a "VERY" small opening.
- Purpose: To block off most of the light rays
  - Reduces blurring
  - The opening is known as aperture
  - What happens to the image?
Pinhole Camera

- **Pinhole Model:**
  - Captures pencil of rays – all rays through a single point
  - This point is called – Center of Projection (COP)
  - The image is formed on the Image Plane
  - **Effective focal length**, \( f \), is the distance from COP to the Image Plane

“Home-made” Pinhole Camera

- [www.debevec.org/Pinhole/](http://www.debevec.org/Pinhole/)

Camera w/ Lens

- **Purpose for the lens:**
  - Ideal Pinhole Model is unattainable; light gathering mechanism
  - Keep image in sharp focus

Anatomy of a modern camera

- **Sensor Arrays**
**Image Sensors**

- Images are formed by the interaction of the incident image irradiance with light sensitive elements on the image plane.
- Light sensitive elements:
  - Film
  - Charge Coupled Device (CCD)
  - CMOS imaging element

**Resolution**

- Refers to the precision of the sensor in making measurements (different formal definitions exist):
  - Normal Resolution of a CCD sensor
    - Size of the scene element that images to a single pixel on the image plane.
  - Resolution of a digital image
    - The dimension (in pixels) of the digital image.

**Digital Imaging Systems**

- CCD or CMOS imaging array
  - When light falls on the cells in these arrays a charge accumulates which is proportional to the incident light energy.
- A/D conversion unit
- Host Computer

**Digital Snapshots**

- A digital image is an array of numbers indicating the image irradiance at various points on the image.
- Image intensities are spatially sampled.
- Intensity values are quantized (8-bits, 10-bits, 12-bits, etc).
- Video Imagery
  - For a video camera, Images are taken sequentially by opening and close the shutter 30x/sec (i.e. 30 frames/sec).
Types of Images

- Analog Image – \( A(x,y) \)
- Digital Image – \( I(r,c) \)
  - Grayscale Image – monochrome digital image \( G(r,c) \)
    - Picture Function \( f : \mathbb{R}^2 \rightarrow \mathbb{R} \)
  - Multi-spectral Image – color digital image \( M(r,c) \)
    - Picture Function \( f : \mathbb{R}^2 \rightarrow \mathbb{R}^n \)
  - Binary Image – \( B(r,c) \)
    - Picture Function \( f : \mathbb{R}^2 \rightarrow [0, 1] \)
  - Labeled Image – \( L(r,c) \)
    - Picture Function \( f : \mathbb{R}^2 \rightarrow \mathcal{L} \)

Sensing Color

- In a 3-CCD video camera the light path is split 3 ways which are passed through colored light filters and then imaged
- As a result – a color image contains three channels of information: red, green, and blue image intensities
- In a 1-CCD color camera color information is obtained by converting the individual elements with a spatially varying pattern of filters, RGB

Practical Color Sensing: Bayer Grid

Images in a Computer

- An image is a 2-D table of numbers or 2D Matrix

http://en.wikipedia.org/wiki/Bayer_filter

http://en.wikipedia.org/wiki/Bayer_filter
Example

\[(r,g,b) = (255,255,251), (222,15,7), (0,0,0), (89,120,1)\]

\[(19,37,87), (255,255,115)\]

Brightness = 0.5*(R+G)

```
>> I = imread(image_file);
>> figure(1); image(I);
>> pixval on;
```

\[I = double(I);
Ig = 0.5*(I(:,:,1) + I(:,:,2));
figure(2); imagesc(Ig);
Colormap(gray);
```
A word of caution

- For this class, image processing in Matlab
- Matlab extremely powerful in terms of matrix manipulation and signal processing
- Implementation in MatLab vs. in C/C++/Java NOT EQUAL!!

Dimensionality Reduction Machine (3D => 2D)

- What have we lost?
  - Angles
  - Distances (lengths)

Funny things happen ...

- Parallel lines are not ....
- Distances can't be trusted ...
• Size of field of view governed by the size of the image plane:

\[ \varphi = \tan^{-1} \frac{d}{2f} \]

• Smaller FOV = larger Focal Length

Conversion from mm to pixels

• In the digital camera, measurements are made in image pixels. Need to convert the focal length (in mm) to pixels.

• Typical digital sensors are smaller than (35 mm) film, this effectively increases the focal length
FOV Depends on Focal Length

- Size of field of view governed by the size of the image plane:
  \[ \varphi = \tan^{-1} \left( \frac{d}{2f} \right) \]

- Smaller FOV = larger Focal Length

Field of View (Zoom)

From London and Upton

Large FOV, small f
Camera closer to car

Small FOV, large f
Camera far from car
Image Projection

- Modeling (pinhole) projection

- The coordinate system
  - Use the pinhole model as an approximation
  - Put the optical center (COP) at origin
  - Put image plane (Projection Plane, PP) in front of the COP
  - Why?
  - The camera looks down the negative z-axis
    - Required for right-handed coordinates

Projection Equations

- Note: \( \vec{OP} \) and \( \vec{OP}' \) are colinear

- Given
  \[
  \begin{bmatrix}
  x \\
  y \\
  z
  \end{bmatrix}
  \text{then}
  \begin{bmatrix}
  x' \\
  y' \\
  z
  \end{bmatrix}
  =
  f \begin{bmatrix}
  x/z \\
  y/z \\
  z
  \end{bmatrix}
  \]
Images in a Computer

• An image is a 2-D table of numbers or 2D Matrix

Any 2D matrix can be seen as an image

Binary Image Analysis

• Binary Image
  Image w/ only 0 and 1 as entries

• Notation:
  ➢ B – denotes the binary image
  ➢ B(r,c) – denotes a pixel in the image
  ➢ B[0,0] – upper leftmost pixel

• Neighborhoods
  ➢ 4 – Neighborhood
  ➢ 8 – Neighborhood

• How would you convert any image into a Binary Image?

Applying Masks to Images

• Convolution of the Image w/ another “Signal”

• Masks have origins
  ➢ Symmetric masks – origins are the center pixels
**Recursive Labeling Algorithm**

- Let \( I \) denote the image.
- Step 1: Convert \( I \) to \( B \).
- Step 2: Let \( B' = -B \), label = 0

\[
\text{for } i = 1: \text{rows} \\
\text{for } j = 1: \text{columns} \\
\text{if } B'(i,j) == -1 \\
\quad \text{label} = \text{label} + 1; \\
\quad \text{Add } (i,j) \text{ onto Stack and set } B'(i,j) = \text{label}; \\
\text{end} \\
\text{end} \\
\text{while } (\text{Stack is not empty}) \\
\quad \text{Remove pixel } P \text{ from Stack} \\
\quad \text{Let } M = \text{neighbors of } P \text{ whose pixel value } = -1; \\
\quad \text{Add } M \text{ onto Stack; } \text{set } B'(M) = \text{label}; \\
\text{end} \\
\text{end} \\
\]
**Morphological Operators**

- **Structuring Elements** ($S$)
  - BOX(3,5)
  - DISK(5)
  - RING(5)

- **Basic Operators**
- Translation $X_t$ of a set of pixels by a position vector $t$ is given by
  
  $X_t = \{ x + t \mid x \in X \}$

- **Dilation**: Minkowski addition
  
  $B \ominus S = \bigcup_{b \in B} S_b$

- **Erosion**

- **Opening**: $B \bullet S = (B \ominus S) \ominus S$
### Morphological Operators

- **Closing:** $B \circ S = (B \ominus S) \oplus S$

### Region Properties

- Area – Total # of pixels in the region of interest
- Centroid
- Perimeter
- Circularity
- Mean Radial Distance
- Standard Deviation/Variance of Radial Distance
- Bounding Box & Extremal Points
- Spatial Moments
- Ellipse Properties

### Thresholding & Histograms

- **Thresholding:**
  - Manual – For grayscale images $G$, in Matlab $G > \epsilon$
  - Automatic – Otsu Method (See Shapiro & Stockman)

- **Histograms** – use `hist(values, binNumber)` in Matlab

---