The Geometry of Image Formation

Mapping between image and world coordinates
- Pinhole camera model
- Projective geometry
  - Vanishing points and lines
- Projection matrix
What do you need to make a camera from scratch?
Let’s design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?
Pinhole camera

Idea 2: add a barrier to block off most of the rays

– This reduces blurring
– The opening known as the aperture

Slide source: Seitz
Pinhole camera

\[ f = \text{focal length} \]
\[ c = \text{center of the camera} \]
Camera obscura: the pre-camera

- Known during classical period in China and Greece (e.g. Mo-Ti, China, 470BC to 390BC)

Illustration of Camera Obscura

Freestanding camera obscura at UNC Chapel Hill

Photo by Seth Ilys
Camera Obscuroa used for Tracing

Lens Based Camera Obscuroa, 1568
Accidental Cameras

Accidental Pinhole and Pinspeck Cameras
Revealing the scene outside the picture.
Antonio Torralba, William T. Freeman
Accidental Cameras

a) Input (occluder present)

b) Reference (occluder absent)

c) Difference image (b-a)

d) Crop upside down

e) True view
First Photograph

Oldest surviving photograph
– Took 8 hours on pewter plate

Joseph Niepce, 1826

Photograph of the first photograph

Stored at UT Austin

Niepce later teamed up with Daguerre, who eventually created Daguerrotypes
“Louis Daguerre—the inventor of daguerreotype—shot what is not only the world's oldest photograph of Paris, but also the first photo with humans. The 10-minute long exposure was taken in 1839 in Place de la République and it's just possible to make out two blurry figures in the left-hand corner.”
Camera and World Geometry

How tall is this woman?

How high is the camera?

What is the camera rotation?

What is the focal length of the camera?

Which ball is closer?
Dimensionality Reduction Machine (3D to 2D)

3D world

Point of observation

2D image

Figures © Stephen E. Palmer, 2002
Projection can be tricky...
Projection can be tricky...
Projective Geometry

What is lost?

- Length

Who is taller?

Which is closer?
Length and area are not preserved
Projective Geometry

What is lost?

• Length
• Angles
Projective Geometry

What is preserved?

• Straight lines are still straight
Vanishing points and lines

Parallel lines in the world intersect in the image at a “vanishing point”
Vanishing points and lines
Vanishing points and lines

Vertical vanishing point (at infinity)

Vanishing point

Vanishing point

Slide from Efros, Photo from Criminisi
• Project 1 will be out soon
• Read Szeliski 2.1, especially 2.1.4
• Image projection
• Filtering
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Geometric primitives and transformations</td>
<td></td>
</tr>
<tr>
<td>2.1.1 2D transformations</td>
<td>39</td>
</tr>
<tr>
<td>2.1.2 3D transformations</td>
<td>43</td>
</tr>
<tr>
<td>2.1.3 3D rotations</td>
<td>45</td>
</tr>
<tr>
<td>2.1.4 3D to 2D projections</td>
<td>50</td>
</tr>
<tr>
<td>2.1.5 Lens distortions</td>
<td>62</td>
</tr>
<tr>
<td>2.2 Photometric image formation</td>
<td></td>
</tr>
<tr>
<td>2.2.1 Lighting</td>
<td>65</td>
</tr>
<tr>
<td>2.2.2 Reflectance and shading</td>
<td>66</td>
</tr>
<tr>
<td>2.2.3 Optics</td>
<td>73</td>
</tr>
<tr>
<td>2.3 The digital camera</td>
<td></td>
</tr>
<tr>
<td>2.3.1 Sampling and aliasing</td>
<td>82</td>
</tr>
<tr>
<td>2.3.2 Color</td>
<td>85</td>
</tr>
<tr>
<td>2.3.3 Compression</td>
<td>97</td>
</tr>
<tr>
<td>2.4 Additional reading</td>
<td></td>
</tr>
<tr>
<td>2.5 Exercises</td>
<td>99</td>
</tr>
</tbody>
</table>
If X = 2, Y = 3, Z = 5, and f = 2
What are U and V?

\[ x = \begin{bmatrix} u' \\ v' \end{bmatrix} \]

\[ X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \]

\[ u' = -x \frac{f}{z} \]

\[ v' = -y \frac{f}{z} \]

\[ u' = -2 \frac{2}{5} \]

\[ v' = -3 \frac{2}{5} \]
Projection: world coordinates $\rightarrow$ image coordinates

How do we handle the general case?
Interlude: why does this matter?
Relating multiple views
Photo Tourism
Exploring photo collections in 3D

Noah Snavely  Steven M. Seitz  Richard Szeliski
University of Washington  Microsoft Research

SIGGRAPH 2006
Projection: world coordinates $\rightarrow$ image coordinates

How do we handle the general case?
Homogeneous coordinates

Conversion

Converting to *homogeneous* coordinates

\[(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\]

homogeneous image coordinates

\[(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\]

homogeneous scene coordinates

Converting *from* homogeneous coordinates

\[
\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \\
\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)
\]
Homogeneous coordinates

Invariant to scaling

\[
k \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}
\]

Homogeneous Coordinates \quad Cartesian Coordinates

Point in Cartesian is ray in Homogeneous
Projection matrix

\[ \mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X} \]

- **x**: Image Coordinates: \((u,v,1)\)
- **K**: Intrinsic Matrix (3x3)
- **R**: Rotation (3x3)
- **t**: Translation (3x1)
- **X**: World Coordinates: \((X,Y,Z,1)\)
Projection matrix

\[ \mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \]

**Intrinsic Assumptions**
- Unit aspect ratio
- Optical center at (0,0)
- No skew

**Extrinsic Assumptions**
- No rotation
- Camera at (0,0,0)

Slide Credit: Savarese
Projection matrix

Intrinsic Assumptions
- Unit aspect ratio
- Optical center at (0,0)
- No skew

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

\[ \mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} \mathbf{X} \]

\[
\begin{bmatrix}
\mathbf{u} \\
\mathbf{v} \\
1
\end{bmatrix}
= \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0 \\
1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]
Remove assumption: known optical center

Intrinsic Assumptions
- Unit aspect ratio
- No skew

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

\[
x = K[I \ 0]X
\]
Remove assumption: square pixels

Intrinsic Assumptions
• No skew

Extrinsic Assumptions
• No rotation
• Camera at (0,0,0)

\[
x = K[I \ 0]X
\]

\[
x = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]
Remove assumption: non-skewed pixels

Intrinsic Assumptions

Extrinsic Assumptions
  • No rotation
  • Camera at (0,0,0)

\[ \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \]

Note: different books use different notation for parameters
Oriented and Translated Camera

\[ \begin{align*}
\Pi' & \quad f' \\
C' & \quad k \\
x & \quad i
\end{align*} \]

\[ \begin{align*}
x' & \quad X \\
t & \quad O_w
\end{align*} \]

\[ \begin{align*}
R & \quad j_w \\
k_w
\end{align*} \]
Allow camera translation

Intrinsic Assumptions  Extrinsic Assumptions
• No rotation

\[
x = K[I \ t] X \quad \Rightarrow \quad \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]
3D Rotation of Points

Rotation around the coordinate axes, **counter-clockwise**:

\[
R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}
\]

\[
R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}
\]

\[
R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]
Allow camera rotation

\[ x = K[R \ t]X \]

\[
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix} = \begin{bmatrix}
\alpha & s & u_0 \\
0 & \beta & v_0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
r_{11} & r_{12} & r_{13} & t_x \\
r_{21} & r_{22} & r_{23} & t_y \\
r_{31} & r_{32} & r_{33} & t_z \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]
Degrees of freedom

\[ x = K[R \ t]X \]
Field of View (Zoom, focal length)

From London and Upton
Beyond Pinholes: Radial Distortion

No Distortion

Barrel Distortion

Pincushion Distortion

Corrected Barrel Distortion

Image from Martin Habbecke
Things to remember

- Vanishing points and vanishing lines
- Pinhole camera model and camera projection matrix
- Homogeneous coordinates
Reminder: read your book

• Lectures have assigned readings
• Szeliski 2.1 and especially 2.1.4 cover the geometry of image formation
2 minute break
BBC Clip: https://www.youtube.com/watch?v=OlumoQ05gS8
From the 3D to 2D

- Let’s now focus on 2D
- Extract building blocks
Extract useful building blocks
The big picture...

1. Feature Detection
   - e.g. DoG

2. Feature Description
   - e.g. SIFT

   database of local descriptors

3. Matching/Indexing/Detection
Hybrid Images

Upcoming classes: two views of filtering

• Image filters in spatial domain
  – Filter is a mathematical operation of a grid of numbers
  – Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  – Filtering is a way to modify the frequencies of images
  – Denoising, sampling, image compression
Image filtering

• Image filtering: compute function of local neighborhood at each position

• Really important!
  – Enhance images
    • Denoise, resize, increase contrast, etc.
  – Extract information from images
    • Texture, edges, distinctive points, etc.
  – Detect patterns
    • Template matching
  – Deep Convolutional Networks
Example: box filter

\[ g[\cdot, \cdot] \]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[ \frac{1}{9} \]

Slide credit: David Lowe (UBC)
Image filtering

\[ f[.,.] \]

\[ h[.,.] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]
Image filtering

\[f[\cdot, \cdot] \quad h[\cdot, \cdot]\]

\[
h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]
\]

Credit: S. Seitz
Image filtering

\[ g[\cdot, \cdot] = \frac{1}{9} \]

\[ f[\cdot, \cdot] \]

\[ h[\cdot, \cdot] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

Credit: S. Seitz
Image filtering

\[ f[\ldots] \quad h[\ldots] \]

\[
h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l]
\]

Credit: S. Seitz
Image filtering

\[ f[\cdot,\cdot] \quad h[\cdot,\cdot] \]

\[
h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l]
\]

Credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l] \]

Credit: S. Seitz
Image filtering

\[ f[\cdot, \cdot] \quad \rightarrow \quad h[\cdot, \cdot] \]

\[ h[m,n] = \sum_{k,l} g[k,l] \ f[m+k, n+l] \]

Credit: S. Seitz
Image filtering

\[ f[\ldots] \quad h[\ldots] \]

\[
h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l]
\]

Credit: S. Seitz
What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)
Smoothing with box filter
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Filtered (no change)

Source: D. Lowe
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

0 0 0
0 0 1
0 0 0

Shifted left
By 1 pixel

Source: D. Lowe
Practice with linear filters

Original

(Note that filter sums to 1)

Source: D. Lowe
Practice with linear filters

**Original**

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Sharpening filter**

- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Other filters

Sobel

Vertical Edge (absolute value)
Other filters

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Sobel

Horizontal Edge (absolute value)
Filtering vs. Convolution

• 2d filtering
  \[ h = \text{filter2}(f,I); \quad \text{or} \quad h = \text{imfilter}(I,f); \]

\[ h[m,n] = \sum_{k,l} f[k,l] I[m+k,n+l] \]

• 2d convolution
  \[ h = \text{conv2}(f,I); \]

\[ h[m,n] = \sum_{k,l} f[k,l] I[m-k,n-l] \]
Key properties of linear filters

**Linearity:**
\[
imfilter(I, f_1 + f_2) = imfilter(I, f_1) + imfilter(I, f_2)
\]

**Shift invariance:** same behavior regardless of pixel location
\[
imfilter(I, \text{shift}(f)) = \text{shift}(imfilter(I, f))
\]

Any linear, shift-invariant operator can be represented as a convolution

Source: S. Lazebnik
More properties

• Commutative: $a * b = b * a$
  – Conceptually no difference between filter and signal
  – But particular filtering implementations might break this equality

• Associative: $a * (b * c) = (a * b) * c$
  – Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  – This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

• Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

• Scalars factor out: $ka * b = a * kb = k (a * b)$

• Identity: unit impulse $e = [0, 0, 1, 0, 0]$, $a * e = a$

Source: S. Lazebnik
Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness

\[ G_{\sigma} = \frac{1}{2\pi \sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

5 x 5, \( \sigma = 1 \)

Slide credit: Christopher Rasmussen
Smoothing with Gaussian filter
Smoothing with box filter
Gaussian filters

• Remove “high-frequency” components from the image (low-pass filter)
  – Images become more smooth

• Convolution with self is another Gaussian
  – So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  – Convolving two times with Gaussian kernel of width $\sigma$ is same as convolving once with kernel of width $\sigma\sqrt{2}$

• *Separable* kernel
  – Factors into product of two 1D Gaussians

Source: K. Grauman
Separability of the Gaussian filter

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp \left( \frac{-x^2 + y^2}{2\sigma^2} \right) \]

\[ = \left( \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-x^2}{2\sigma^2} \right) \left( \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-y^2}{2\sigma^2} \right) \]

The 2D Gaussian can be expressed as the product of two functions, one a function of \( x \) and the other a function of \( y \).

In this case, the two functions are the (identical) 1D Gaussian.
Separability example

2D convolution (center location only)

The filter factors into a product of 1D filters:

Perform convolution along rows:

Followed by convolution along the remaining column:

Source: K. Grauman
Separability

- Why is separability useful in practice?
Some practical matters
Practical matters

How big should the filter be?

- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about $3 \sigma$
Practical matters

• What about near the edge?
  – the filter window falls off the edge of the image
  – need to extrapolate
  – methods:
    • clip filter (black)
    • wrap around
    • copy edge
    • reflect across edge

Source: S. Marschner
To be continued...
Next class: Light and Color and Thinking in Frequency