Welcome back!

• Project 0 will be out today
• Project 1 will be next week
• Read Szeliski 2.1, especially 2.1.4
• Today
  – Image projection
  – Filtering
Chapter 2

Image formation

2.1 Geometric primitives and transformations
  2.1.1 2D transformations
  2.1.2 3D transformations
  2.1.3 3D rotations
  2.1.4 3D to 2D projections
  2.1.5 Lens distortions

2.2 Photometric image formation
  2.2.1 Lighting
  2.2.2 Reflectance and shading
  2.2.3 Optics

2.3 The digital camera
  2.3.1 Sampling and aliasing
  2.3.2 Color
  2.3.3 Compression

2.4 Additional reading

2.5 Exercises
If $X = 2$, $Y = 3$, $Z = 5$, and $f = 2$

What are $U$ and $V$?

\[
\frac{v'}{-f} = \frac{y}{z} \quad \frac{v'}{-f} = -y \cdot \frac{f}{z} \quad v' = -3 \cdot \frac{2}{5}
\]

\[
-u' = -x \cdot \frac{f}{z} \quad u' = -2 \cdot \frac{2}{5}
\]
Projection: world coordinates \( \rightarrow \) image coordinates

\[
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} = P
\]

Optical Center \((u_0, v_0)\)

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

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

Homogeneous Coordinates \quad Cartesian Coordinates

Point in Cartesian is ray in Homogeneous
Projection matrix

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

- **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

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

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

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

\[
\begin{bmatrix}
    u \\
    v \\
    w \\
    1
\end{bmatrix} = \begin{bmatrix}
    f & 0 & 0 & 0 \\
    0 & f & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}
\]
Projection matrix

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

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

\[
x = K \begin{bmatrix} 1 & 0 \end{bmatrix} X
\]

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

Intrinsic Assumptions
• Unit aspect ratio
• No skew

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

$$\mathbf{x} = \mathbf{K} [\mathbf{I} \quad \mathbf{0}] \mathbf{X}$$

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$
Remove assumption: square pixels

Intrinsic Assumptions
• No skew

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

\[ \mathbf{x} = \mathbf{K}[\mathbf{I} \ 0] \mathbf{X} \]

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

Intrinsic Assumptions
Extrinsic Assumptions

• No rotation
• Camera at (0,0,0)

\[ \mathbf{x} = \mathbf{K}[\mathbf{I} \ 0] \mathbf{X} \]

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

Intrinsic Assumptions

Extrinsic Assumptions

• No rotation

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

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

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

\[ \mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X} \]

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

Computer Vision

James Hays

Many slides by Derek Hoiem
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...

- Feature Detection (e.g., DoG)
- Feature Description (e.g., SIFT)
- Database of local descriptors
- 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

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

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

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

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

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

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

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

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

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

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

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

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

Credit: S. Seitz
Image filtering

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

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

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

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
Box Filter

What does it do?

• Replaces each pixel with an average of its neighborhood

• Achieve smoothing effect (remove sharp features)

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

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

Slide credit: David Lowe (UBC)
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

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

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Other filters

Vertical Edge (absolute value)

Sobel Filter:

```
1  0  -1
2  0  -2
1  0  -1
```
Other filters

Horizontal Edge (absolute value)

Sobel filter:

```
1 2 1
0 0 0
-1 -2 -1
```
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:** \(k a * b = a * k b = k (a * b)\)

• **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

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

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

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

Source: D. Lowe
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