Lecture 12

Geometry and Cameras
Administrative

A3 is out
- Due May 9th - 12th

A4 out this weekend

A5 is half the length of other assignments
Administrative

Recitation
- Vivek Jayaram
- Multi-view geometry
So far: Segmentation and clustering

- Goal: identify groups of pixels that go together
So far: Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
So far: K-means clustering

1. Initialize Cluster Centers
2. Assign Points to Clusters
3. Re-compute Means
Repeat (2) and (3)

Illustration Source: wikipedia
So far: Mean-Shift Clustering

- Initialize multiple window at random locations
- All pixels that end up in the same location belong to the same **cluster**
- **Attraction basin**: the feature region for which all windows end up in the same location
Today’s agenda

- How biological vision understands geometry
- Brief history of geometric vision
- Geometric transformations
- Pinhole camera
- The Pinhole camera transformation
Today’s agenda

● How biological vision understands geometry
● Brief history of geometric vision
● Geometric transformations
● Pinhole camera
● The Pinhole camera transformation
Our goal: Recover the 3D geometry of the world

Let’s Take a Picture!

Photosensitive Material
Single-view Ambiguity

• Given a camera and an image, we only know the ray corresponding to each pixel.
• We don’t know how far away the object the ray was reflected from
  ○ We don’t have enough constraints to solve for $X$ (depth)
Single-view Ambiguity

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

Slide Credit: J. Hays
Single-view Ambiguity
Resolving Single-view Ambiguity

- Shoot light (lasers etc.) out of your eyes!
- Con: not so biologically plausible, dangerous?
Resolving Single-view Ambiguity

- Shoot light (lasers etc.) out of your eyes!
- Con: not so biologically plausible, dangerous?
How do humans estimate depth? **Two eyes!**

- **Stereo:** given 2 calibrated cameras in different views and correspondences, can solve for $X$
Stereo photography and stereo viewers

Take two pictures of the same subject from two slightly different viewpoints and display so that each eye sees only one of the images.
http://www.johnsonshawmuseum.org
Not all animals see stereo:

Prey animals are Stereoblind
(large field of view to spot predators)
Resolving Single-view Ambiguity

- One option: move the camera, find matching correspondences
- If you know how you moved in the physical world and have corresponding points in image space, you can solve for $X$
How do you estimate how much you moved in the physical world?

Can estimate using our eyes!

Can estimate using our ears!

- Our inner ears have 3 ducts
- Can estimate movement via signals sent to muscles
But even without moving, we can estimate depth from a single image. But how?

- You haven’t been here before, yet you probably have a fairly good understanding of this scene.
We use pictorial cues – such as shading
We use pictorial cues – such as *perspective effects*.
We use pictorial cues – such as familiar objects

Monitor: probably not 12 feet wide.

Desk surface: probably flat
Reality of 3D Perception

● 3D perception is absurdly complex and involves integration of many cues:
  ○ Learned cues for 3D
  ○ Stereo between eyes
  ○ Stereo via motion
    ○ Integration of known motion signals to muscles (afferent copy), acceleration sensed via ears
  ○ Past experience of touching objects

● All connect: learned cues from 3D probably come from stereo/motion cues in large part

Really fantastic article on cues for 3D from Cutting and Vishton, 1995: https://pmvish.people.wm.edu/cutting%26vishton1995.pdf
Regardless, illusions can still fool this complex system

Ames illusion persists (in a weaker form) even if you have stereo vision – guessing the texture is rectilinear is usually incredibly reliable.

Today’s agenda

- How biological vision understands geometry
- Brief history of geometric vision
- Geometric transformations
- Pinhole camera
- The Pinhole camera transformation
Simplified Image Formation
Geometric vision is an **ill-posed** inverse problem.

2D Image  

3D Scene

Graphics  

Vision

Pixel Matrix

<table>
<thead>
<tr>
<th>217</th>
<th>191</th>
<th>252</th>
<th>255</th>
<th>239</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>80</td>
<td>200</td>
<td>146</td>
<td>138</td>
</tr>
<tr>
<td>159</td>
<td>94</td>
<td>91</td>
<td>121</td>
<td>138</td>
</tr>
<tr>
<td>179</td>
<td>106</td>
<td>136</td>
<td>85</td>
<td>41</td>
</tr>
<tr>
<td>115</td>
<td>129</td>
<td>83</td>
<td>112</td>
<td>67</td>
</tr>
<tr>
<td>94</td>
<td>114</td>
<td>105</td>
<td>111</td>
<td>89</td>
</tr>
</tbody>
</table>

Objects

Shape/Geometry  

Semantics

Material  

Motion  

3D Pose

Ranjay Krishna, Jieyu Zhang  

Lecture 12 - 33

May 2, 2024

Slide credit: Andreas Geiger
Brief History of Geometric Vision

- 2020-: geometry + learning
- 2010s: deep learning
- 2000s: local detectors and descriptors
- 1990s: digital camera, 3D geometry estimation
- 1980s: epipolar geometry (stereo)
- …
Brief History of Geometric Vision

- 1860s: Willème invented photo-sculptures

Source: P. Sturm
Brief History of Geometric Vision

- 1860s: Willème invented photo-scultures
- 1850s: birth of photogrammetry [Laussedat]
- 1840s: panoramic photography

1864

“Cloud camera”, 190?

Source: P. Sturm
Brief History of Geometric Vision

● 1860s: Willème invented photo-scultures
● 1850s: birth of photogrammetry [Laussedat]
● 1840s: panoramic photography
● 1822-39: birth of photography [Niépce, Daguerre]
● 1773: general 3-point pose estimation [Lagrange]
● 1715: basic intrinsic calibration (pre-photography!) [Taylor]
● 1700’s: topographic mapping from perspective drawings [Beautemps-Beaupré, Kappeler]
Brief History of Geometric Vision

● 15th century: start of mathematical treatment of 3D, first AR app?

Augmented reality invented by Filippo Brunelleschi (1377-1446)?

Tavoletta prospettica di Brunelleschi
Brief History of Geometric Vision

- **5th century BC**: principles of pinhole camera, a.k.a. camera obscura
  - China: 5th century BC
  - Greece: 4th century BC
  - Egypt: 11th century
  - Throughout Europe: from 11th century onwards

First mention …

First camera?

Chinese philosopher Mozi (470 to 390 BC)

Greek philosopher Aristotle (384 to 322 BC)

Source: P. Sturm
Today’s agenda

- How biological vision understands geometry
- Brief history of geometric vision
- Geometric transformations
- Pinhole camera
- The Pinhole camera transformation
Points

2D points: \( \mathbf{x} = (x, y) \in \mathcal{R}^2 \) or column vector \( \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \)

3D points: \( \mathbf{x} = (x, y, z) \in \mathcal{R}^3 \) (often noted \( \mathbf{X} \) or \( \mathbf{P} \))

Homogeneous coordinates: append a 1

Why? \( \bar{\mathbf{x}} = (x, y, 1) \quad \bar{\mathbf{x}} = (x, y, z, 1) \)
Everything is easier in Projective Space

\[2D \text{ Lines:}
\]
- Representation: \( l = (a, b, c) \)
- Equation: \( ax + by + c = 0 \)
- In homogeneous coordinates: \( \bar{x}^T l = 0 \)

General idea: homogeneous coordinates unlock the full power of linear algebra!
Homogeneous coordinates in 2D

2D Projective Space $\mathcal{P}^2 = \mathbb{R}^3 - (0, 0, 0)$  (same story in 3D with $\mathcal{P}^3$)

- heterogeneous $\rightarrow$ homogeneous
  $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- homogeneous $\rightarrow$ heterogeneous
  $$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow \begin{bmatrix} x/w \\ y/w \\ \_ \end{bmatrix}$$

- points differing only by scale are equivalent: $(x, y, w) \sim \lambda (x, y, w)$
  $$\tilde{x} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\tilde{x}.$$
The camera as a coordinate transformation

A camera is a mapping
from: the 3D world
to: a 2D image

Source: K. Kitani
Cameras and objects can move!

\[ p = (X, Y, Z, 1) \]

\[ \tilde{x}_0 = (x_0, y_0, 1, d_0) \]

\[ \tilde{x}_1 = (x_1, y_1, 1, d_1) \]

\[ M_{10} \]
2D Transformations in pixel locations (not pixel values)

Figure: R. Szeliski
Scaling

\[
\begin{bmatrix}
  s_x & 0 \\
  0 & s_y \\
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
\end{bmatrix}
= 
\begin{bmatrix}
  s_x x \\
  s_y y \\
\end{bmatrix}
\]
Rotation

\[ x' = x \cos \theta - y \sin \theta \]
\[ y' = x \sin \theta + y \cos \theta \]

or in matrix form:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  \cos \theta & -\sin \theta \\
  \sin \theta & \quad \cos \theta
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

Rotation matrix:
- Inverse is transpose
- Orthonormal

\[ R \cdot R^T = R^T \cdot R = I \]
\[ \det(R) = 1 \]
2D Translation

\[ x' = x + t_x \]
\[ y' = y + t_y \]

As a matrix?
# Transformation = Matrix Multiplication

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Matrix Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scale</strong></td>
<td>$M = \begin{bmatrix} s_x &amp; 0 \ 0 &amp; s_y \end{bmatrix}$</td>
</tr>
<tr>
<td><strong>Flip across y</strong></td>
<td>$M = \begin{bmatrix} -1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td><strong>Rotate</strong></td>
<td>$M = \begin{bmatrix} \cos \theta &amp; -\sin \theta \ \sin \theta &amp; \cos \theta \end{bmatrix}$</td>
</tr>
<tr>
<td><strong>Flip across origin</strong></td>
<td>$M = \begin{bmatrix} -1 &amp; 0 \ 0 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td><strong>Shear</strong></td>
<td>$M = \begin{bmatrix} 1 &amp; s_x \ s_y &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td><strong>Identity</strong></td>
<td>$M = \begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>
2D Translation with homogeneous coordinates

\[ p = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} \]

\[ t = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \rightarrow \begin{bmatrix} t_x \\ t_y \end{bmatrix} \]

\[ p' = Tp \]

\[ p' \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix} p = Tp \]
Euclidean transformations: rotation + translation

Euclidean (rigid): rotation + translation

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & t_x \\
\sin \theta & \cos \theta & t_y \\
0 & 0 & 1
\end{bmatrix}
\]

SE(2): Special Euclidean group
Important in robotics: describes poses on plane

How many degrees of freedom?
Similarity = Euclidean + scaling equally in x and y

Similarity:
Scaling
+ rotation
+ translation

\[
\begin{pmatrix}
a & -b & t_x \\
b & a & t_y \\
0 & 0 & 1
\end{pmatrix}
\]
2D Transformations with homogeneous coordinates

- **No change**
  
  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- **Translate**
  
  $\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$

- **Scale about origin**
  
  $\begin{bmatrix} W & 0 & 0 \\ 0 & H & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- **Rotate about origin**
  
  $\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- **Shear in x direction**
  
  $\begin{bmatrix} 1 & \tan \phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- **Shear in y direction**
  
  $\begin{bmatrix} 1 & 0 & 0 \\ \tan \psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Figure: Wikipedia
Affine transformation = similarity + no restrictions on scaling

Properties of affine transformations:
- arbitrary 6 Degrees Of Freedom
- lines map to lines
- parallel lines map to parallel lines
- ratios are preserved

\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix} = \begin{bmatrix}
a & b & c \\
d & e & f \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]
Properties of projective transformations:

- 8 degrees of freedom
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix} = \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]
Composing Transformations

Transformations = Matrices => Composition by Multiplication!

\[ p' = R_2R_1Sp \]

In the example above, the result is equivalent to

\[ p' = R_2(R_1(Sp)) \]

Equivalent to multiply the matrices into single transformation matrix:

\[ p' = (R_2R_1S)p \]

Order Matters! Transformations from right to left.
Scaling & Translating $\neq$ Translating & Scaling

\[ p'' = TSp = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_xx + t_x \\ s_yy + t_y \\ 1 \end{bmatrix} \]

\[ p''' = STp = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & s_xt_x \\ 0 & s_y & s_yt_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_xx + s_xt_x \\ s_yy + s_yt_y \\ 1 \end{bmatrix} \]
Scaling + Rotation + Translation

\[ p' = (T R S) p \]

\[ p' = TRSP = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]

\[ = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \\ 0 & 0 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]

\[ = [R \quad t] [S \quad 0] [x \quad 0] = [RS \quad t] [x \quad 0] \]

This is the form of the general-purpose transformation matrix.
3D Transforms = Matrix Multiplication

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Matrix</th>
<th># DoF</th>
<th>Preserves</th>
<th>Icon</th>
</tr>
</thead>
</table>
| translation     | \[
\begin{bmatrix}
1 & 0 & 0 & t_1 \\
0 & 1 & 0 & t_2 \\
0 & 0 & 1 & t_3 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}_{3\times4}
\] | 3     | orientation   |      |
| rigid (Euclidean) | \[
\begin{bmatrix}
R & t \\
0 & 1 \\
\end{bmatrix}_{3\times4}
\] | 6     | lengths       |      |
| similarity      | \[
\begin{bmatrix}
\delta R & t \\
0 & 1 \\
\end{bmatrix}_{3\times4}
\] | 7     | angles        |      |
| affine          | \[
\begin{bmatrix}
A \\
0 & 0 & 0 & 1 \\
\end{bmatrix}_{3\times4}
\] | 12    | parallelism   |      |
| projective      | \[
\begin{bmatrix}
H \\
0 & 0 & 0 & 1 \\
\end{bmatrix}_{4\times4}
\] | 15    | straight lines|      |

Table 2.2  Hierarchy of 3D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The $3 \times 4$ matrices are extended with a fourth $[0^T \ 1]$ row to form a full $4 \times 4$ matrix for homogeneous coordinate transformations. The mnemonic icons are drawn in 2D but are meant to suggest transformations occurring in a full 3D cube.
Today’s agenda

- How biological vision understands geometry
- Brief history of geometric vision
- Geometric transformations
- Pinhole camera
- The Pinhole camera transformation

Reference: Szeliski 2.1, 2.2.3, 7.4
Reminder: Camera Obscura

- 5th century BC: principles of pinhole camera, a.k.a. camera obscura
  - China: 5th century BC
  - Greece: 4th century BC
  - Egypt: 11th century
  - Throughout Europe: from 11th century onwards

First mention …

Chinese philosopher Mozi (470 to 390 BC)

Greek philosopher Aristotle (384 to 322 BC)
Pinhole imaging

- Image plane
- Digital sensor (CCD or CMOS)
- Focal length $f$
- Barrier (diaphragm)
- Pinhole (aperture)
- Camera center (center of projection)
- Real-world object
What does the image on the sensor look like?

Each scene point contributes to only one sensor pixel.
Pinhole imaging

copy of real-world object (inverted and scaled)

real-world object
Bare-sensor imaging (without a pinhole camera)

All scene points contribute to all sensor pixels

What does the image on the sensor look like?

digital sensor (CCD or CMOS)

real-world object
Bare-sensor imaging (without a pinhole camera)

All scene points contribute to all sensor pixels
Cameras & Lenses

- **Focal length** determines the magnification of the image projected onto the image plane.
- **Aperture** determines the light intensity of that image pixels.

Source: wikipedia
What’s going on there?

The buildings look distorted and bending towards each other.
Beyond Pinholes: Radial Distortion

- Common in wide-angle lenses or for special applications (e.g., automotive)
- Creates a projective transformation
- Usually handled through solving for non-linear terms and then correcting image
Cameras & Lenses

What happens with a smaller aperture?

- Less light passes through
- Less diffraction effect and clearer image

Pinhole is the miniscule aperture, resulting in the least amount of light and clearest image.

Slide credit: J. Wu & S. Seitz
Today’s agenda

- How biological vision understands geometry
- Brief history of geometric vision
- Geometric transformations
- Pinhole camera
- The Pinhole camera transformation

Reference: Szeliski 2.1, 2.2.3, 7.4
For this course, we focus on the pinhole model.

- Similar to thin lens model in Physics: central rays are not deviated.
- Assumes lens camera in focus.
- Useful approximation but ignores important lens distortions.

Describing both lens and pinhole cameras
The pinhole camera

- Image plane
- Focal length $f$
- Camera center
- Real-world object
The (rearranged) pinhole camera

- Virtual image plane
- Real-world object
- Camera center
- Focal length $f$
The (rearranged) pinhole camera

What is the transformation $\mathbf{x} = \mathbf{PX}$?
Pinhole Camera Matrix

Because all transformations are done using homogeneous coordinate system, all transformations are correct up to some scale lambda

\[
\begin{bmatrix}
    x \\
    y \\
    z
\end{bmatrix} \sim \begin{bmatrix}
    p_1 & p_2 & p_3 & p_4 \\
    p_5 & p_6 & p_7 & p_8 \\
    p_9 & p_{10} & p_{11} & p_{12}
\end{bmatrix} \begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

- **image** coordinates: 3 x 1
- **camera matrix**: 3 x 4
- **world (camera)** coordinates: 4 x 1

"x = PX" 

\[
\lambda \hat{x}^I = P \hat{x}^C
\]
2D view of the (rearranged) pinhole camera

Similar Triangles:

\[
\frac{x}{\tilde{X}} = \frac{f}{\tilde{Z}}
\]
Pinhole Camera Matrix

Transformation from camera coordinates to image coordinates:

\[
[X \quad Y \quad Z]^T \rightarrow [fX/Z \quad fY/Z]^T
\]

General camera model in homogeneous coordinates:

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} \sim \begin{bmatrix}
p_1 & p_2 & p_3 & p_4 \\
p_5 & p_6 & p_7 & p_8 \\
p_9 & p_{10} & p_{11} & p_{12}
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

Pinhole camera has a much simpler projection matrix (assume only scaling):

\[
P = \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0 
\end{bmatrix}
\]

\[
\begin{bmatrix}
fX \\
fY \\
fZ
\end{bmatrix} \rightarrow \begin{bmatrix}
fX/Z \\
fY/Z
\end{bmatrix}
\]
Generalizing the camera matrix

In general, the camera and image have *different* coordinate systems.
Generalizing the camera matrix

In particular, the camera origin and image origin may be different:

Q. How does the camera matrix change?

$$\mathbf{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$
Generalizing the camera matrix

In particular, the camera origin and image origin may be different:

Q. How does the camera matrix change?

\[
P = \begin{bmatrix}
    f & 0 & p_x & 0 \\
    0 & f & p_y & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\]

Translate the camera origin to image origin.
Camera matrix decomposition

We can decompose the camera matrix like this:

\[
P = \begin{bmatrix}
  f & 0 & p_x \\
  0 & f & p_y \\
  0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
  f & 0 & \color{blue} p_x & 0 \\
  0 & f & \color{blue} p_y & 0 \\
  0 & 0 & 1 & 0 \\
\end{bmatrix}
\]
Camera matrix decomposition

We can decompose the camera matrix like this:

\[
P = \begin{bmatrix}
    f & 0 & p_x \\
    0 & f & p_y \\
    0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

(homogeneous) transformation from 2D to 2D, accounting for focal length \( f \) and origin translation

(homogeneous) perspective projection from 3D to 2D, assuming image plane at \( z = 1 \) and shared camera/image origin
Camera matrix decomposition

We can decompose the camera matrix like this:

\[
P = \begin{bmatrix}
f & 0 & p_x \\
0 & f & p_y \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

(homogeneous) transformation from 2D to 2D, accounting for focal length \(f\) and origin translation

(homogeneous) perspective projection from 3D to 2D, assuming image plane at \(z = 1\) and shared camera/image origin

Also written as: \( P = K[I|0] \)

where \( K = \begin{bmatrix}
f & 0 & p_x \\
0 & f & p_y \\
0 & 0 & 1
\end{bmatrix} \)

\( K \) is called the camera intrinsics
Generalizing the camera matrix

In general, there are 3 different coordinate systems (camera moves in the world).
World-to-camera coordinate transformation

Let's assume camera is at location $C^W$ in world coordinate system.

Q. What is $X^W$ in camera coordinate system?

Note: heterogeneous coordinates for now.
World-to-camera coordinate transformation

Why aren’t the points aligned?

$X^W - C^W$ translate
World-to-camera coordinate transformation

\[ \mathbf{R} (\mathbf{X}^W - \mathbf{C}^W) \]

rotate translate
Coordinate system transformation

In *heterogeneous* coordinates, we have:

\[ X^C = R (X^W - C^W) \]

Q. How do we write this transformation in homogeneous coordinates?
Coordinate system transformation

In heterogeneous coordinates, we have:

\[ X^c = R \ (X^w - C^w) \]

Q. How do we write this transformation in homogeneous coordinates?

\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1
\end{bmatrix} = \begin{bmatrix}
R & -RC \\
0 & 1
\end{bmatrix} \begin{bmatrix}
X_w \\
Y_w \\
Z_w \\
1
\end{bmatrix} \quad \text{or} \quad \tilde{X}^c = \begin{bmatrix}
R & -RC^w \\
0 & 1
\end{bmatrix} \tilde{X}^w
\]
Let’s update our camera transformation

The previous camera transformation we calculated is for homogeneous 3D coordinates in camera coordinate system:

\[ \mathbf{x}^I \sim K[I|0]\mathbf{x}^C \]

We also just derived:

\[ \mathbf{x}^C = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{C} \\ 0 & 1 \end{bmatrix} \mathbf{x}^W \]
Putting it all together

We can write everything into a single projection:

$$\mathbf{x}^I \sim \mathbf{K}[\mathbf{I}\mathbf{0}][\mathbf{R} \begin{bmatrix} -\mathbf{R} \\ \mathbf{C} \end{bmatrix}] \mathbf{x}^W = \mathbf{P}\mathbf{x}^W$$

The camera matrix now looks like:

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{R} \\ \mathbf{0} & 1 \end{bmatrix}$$

**intrinsic parameters** (3 x 3): correspond to camera internals (image-to-image transformation)

**perspective projection** (3 x 4): maps 3D to 2D points (camera-to-image transformation)

**extrinsic parameters** (4 x 4): correspond to camera externals (world-to-camera transformation)
Putting it all together

We can write everything into a single projection: \( \mathbf{x}^I \sim \mathbf{P} \mathbf{x}^W \)

The camera matrix now looks like:

\[
\mathbf{P} = \begin{bmatrix}
    f & 0 & p_x \\
    0 & f & p_y \\
    0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    \mathbf{R} \\
    \mathbf{t} \\
\end{bmatrix}
\]

Intrinsic parameters (3 x 3): correspond to camera internals (sensor not at \( f = 1 \) and origin shift)

Extrinsic parameters (3 x 4): correspond to camera externals (world-to-image transformation)
General pinhole camera matrix

\[ P = K[R|t] \quad \text{where} \quad t = -RC \]
General pinhole camera matrix

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

where \( t = -RC \)

\[
P = \begin{bmatrix}
f & 0 & p_x \\
0 & f & p_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
r_1 & r_2 & r_3 & t_1 \\
r_4 & r_5 & r_6 & t_2 \\
r_7 & r_8 & r_9 & t_3
\end{bmatrix}
\]

\( P \) are the intrinsic parameters, and \( R \) and \( t \) are the extrinsic parameters.

\[
R = \begin{bmatrix}
r_1 & r_2 & r_3 \\
r_4 & r_5 & r_6 \\
r_7 & r_8 & r_9
\end{bmatrix}
\]

\[
t = \begin{bmatrix}
t_1 \\
t_2 \\
t_3
\end{bmatrix}
\]

3D rotation \( R \) and 3D translation \( t \).
More general camera matrices

Non-square pixels, sensor may be skewed (causing focal length to be different along x and y).

\[
P = \begin{bmatrix}
\alpha_x & s & p_x \\
0 & \alpha_y & p_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
R \\
-RC
\end{bmatrix}
\]

Q. How many degrees of freedom?
Many other types of cameras

(a) 3D view
(b) orthography
(c) scaled orthography
(d) para-perspective
(e) perspective
(f) object-centered

Figure: R. Szeliski
Camera Models: Still an Active Area

Is everybody only using a 2400 years old model?

- More complex cameras: pinhole + distortion, fisheye, catadioptric, dashcams, underwater…

- **The Double Sphere Camera Model**, Usenko *et al* ECCV 2018 (commonly used in robotics, like in our [ICRA’22 paper](#))

- Learning Camera Models
  - **Neural Ray Surfaces**, Vasiljevic *et al*, 3DV 2020
Next time

Camera calibration