Announcements

- Programming Assignment 0 graded.
- Programming Assignment 2 is out! (Due: April 16)

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?
Pinhole camera model

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

Home-made pinhole camera

Why so blurry?

Some sample results from PA0

6-month exposure

Semih Yagcioglu
Some sample results from PA0

Abdurrahman Bayrak

Figure © Stephen E. Palmer, 2002

Dimensionality Reduction Machine (3D to 2D)

3D world

2D image

Projection can be tricky...

Figures © Stephen E. Palmer, 2002

Projection can be tricky...
Projective Geometry

What is lost?
• Length

Length is not preserved

Projective Geometry

What is lost?
• Length
• Angles

Projective Geometry

What is preserved?
• Straight lines are still straight
**Geometric properties of projection**

- Points go to points
- Lines go to lines
- Planes go to whole image or half-planes.
- Polygons go to polygons
- Degenerate cases
  - line through focal point to point
  - plane through focal point to line

**Vanishing points and lines**

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

Vertical vanishing point (at infinity)

Photo by Criminisi

http://www.webexhibits.org/sciartperspective/tylerperspective.html

Vanishing points and lines

Photo from online Tate collection

Vanishing points and lines

Vanishing points and lines

Photo from online Tate collection

Vanishing points and lines

http://www.webexhibits.org/sciartperspective/tylerperspective.html
Vanishing points and lines

- Each set of parallel lines (in direction) meets at a different point
  - The vanishing point for this direction
- Sets of parallel lines on the same plane lead to collinear vanishing points.
  - The line is called the horizon for that plane

What if you photograph a brick wall head-on?

Perspective projection of that line

\[
x'(t) = \frac{f \cdot (x_0 + at)}{z_0}
\]

\[
y'(t) = \frac{f \cdot y_0}{z_0}
\]

All bricks have same \(z_0\). Those in same row have same \(y_0\)

Thus, a brick wall, photographed head-on, gets rendered as set of parallel lines in the image plane.

Other projection models: Orthographic projection

A very large focal length camera approximates the orthographic projection.

\[(x, y, z) \rightarrow (x, y)\]
Other projection models:
Weak perspective

• Issue
  – perspective effects, but not over the scale of individual objects
  – collect points into a group at about the same depth, then divide each point by the depth of its group
  – Adv: easy
  – Disadv: only approximate

\[(x, y, z) \rightarrow \left( \frac{fx}{z_0}, \frac{fy}{z_0} \right)\]

Weak perspective projection

Qingming Festival by the Riverside    Zhang Zeduan ~900 AD

Weak perspective projection

An Ottoman miniature from the Surname-ı Vehbi    Abdulcelil Levni 1720

Three camera projections

3-d point   2-d image position

(1) Perspective: \((x, y, z) \rightarrow \left( \frac{fx}{z}, \frac{fy}{z} \right)\)

(2) Weak perspective: \((x, y, z) \rightarrow \left( \frac{fx}{z_0}, \frac{fy}{z_0} \right)\)

(3) Orthographic: \((x, y, z) \rightarrow (x, y)\)
Homogeneous coordinates

Is this a linear transformation?
- no—division by $z$ is nonlinear

**Trick:** add one more coordinate:

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

Converting from homogeneous coordinates

\[
\begin{bmatrix}
  x \\
  y \\
  w \\
\end{bmatrix} \Rightarrow \begin{bmatrix}
  x/w \\
  y/w \\
  w \\
\end{bmatrix}
\]

Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates:

\[
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & -1/d & 0 \\
  0 & 0 & 1/f & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  f \\
\end{bmatrix} = \begin{bmatrix}
  x \\
  y \\
  z/f \\
  z \\
\end{bmatrix} \Rightarrow \begin{bmatrix}
  x/f \\
  y/f \\
\end{bmatrix}
\]

This is known as **perspective projection**
- The matrix is the projection matrix

Perspective Projection Example

1. Object point at $(10, 6, 4)$, $d=2$

\[
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & -1/2 & 0 \\
  0 & 0 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  f \\
\end{bmatrix} = \begin{bmatrix}
  10 \\
  6 \\
  4 \\
  1 \\
\end{bmatrix} = \begin{bmatrix}
  10 \\
  6 \\
  -2 \\
\end{bmatrix}
\]

\[\Rightarrow x' = -5, \ y' = -3\]

2. Object point at $(25, 15, 10)$

\[
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & -1/2 & 0 \\
  0 & 0 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  f \\
\end{bmatrix} = \begin{bmatrix}
  25 \\
  15 \\
  10 \\
  1 \\
\end{bmatrix} = \begin{bmatrix}
  25 \\
  15 \\
  -5 \\
\end{bmatrix}
\]

\[\Rightarrow x' = -5, \ y' = -3\]

Perspective projection is not 1-to-1!
Orthographic Projection

- Special case of perspective projection
  - Distance from the COP to the image plane is infinite
- Also called parallel projection
- What’s the projection matrix?

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= \begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\Rightarrow (x, y)
\]

Orthographic ("telecentric") lenses

Scaled Orthographic Projection

- Special case of perspective projection
  - Object dimensions are small compared to distance to camera
- Also called “weak perspective”
- What’s the projection matrix?

\[
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}
= \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & s & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]
2D Transformations

Example: translation

\[ x' = x + t \]

\[ x' = \begin{bmatrix} I & t \end{bmatrix} \bar{x} \]

\[ \bar{x}' = \begin{bmatrix} I & t \end{bmatrix} \bar{x} \]

\[ \bar{x}' = \begin{bmatrix} I & t \end{bmatrix} \begin{bmatrix} 0 & 0 & t_x & t_y \\ 0 & 1 & 0 & 1 \end{bmatrix} \bar{x} \]
When we take a picture

We let $f$ to take care of this as well. Unit of $f$ is pixel/world unit.

World Unit: e.g. Meters
Image Unit: Pixels

When we take a picture

We let $f$ to take care of this as well. Unit of $f$ is pixel/world unit.

World Unit: e.g. Meters
Image Unit: Pixels

$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

When we take a picture

We let $f$ to take care of this as well. Unit of $f$ is pixel/world unit.

World Unit: e.g. Meters
Image Unit: Pixels

$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

When we take a picture

We let $f$ to take care of this as well. Unit of $f$ is pixel/world unit.

World Unit: e.g. Meters
Image Unit: Pixels

$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$
Intrinsic parameters: from idealized world coordinates to pixel values

Perspective projection

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

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

May be pixels are not square

\[ u = \alpha \frac{x}{z} \]

\[ v = \beta \frac{y}{z} \]

But “pixels” are in some arbitrary spatial units

\[ u = \alpha \frac{x}{z} \]

\[ v = \alpha \frac{y}{z} \]

We don’t know the origin of our camera pixel coordinates

\[ u = \alpha \frac{x}{z} + u_0 \]

\[ v = \beta \frac{y}{z} + v_0 \]
Intrinsic parameters

May be skew between camera pixel axes

\[ u = \alpha \frac{x}{z} + u_0 \]
\[ v = \beta \frac{y}{z} + v_0 \]

Using homogeneous coordinates, we can write this as:

\[
\begin{pmatrix}
\alpha & -\alpha \cot(\theta) & u_0 \\
0 & \frac{\beta}{\sin(\theta)} & v_0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
\]

In pixels \[ \vec{p} = K \vec{Cp} \]
Perspective projection

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

\(K\) (intrinsic)

(converts from 3D rays in camera coordinate system to pixel coordinates)

in general, \(K = \begin{bmatrix}
-f & s & c_x \\
0 & -f & c_y \\
0 & 0 & 1
\end{bmatrix}\) (upper triangular matrix)

\(Q\): aspect ratio (1 unless pixels are not square)

\(S\): skew (0 unless pixels are shaped like rhombi/parallelograms)

\((c_x, c_y)\): principal point ((0,0) unless optical axis doesn’t intersect projection plane at origin)

Camera parameters

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - Skew (non-rectangular pixels)

Extrinsic Parameters

- How do we get the camera to “canonical form”? (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)

Step 1: Translate by \(-c\)

Extrinsic Parameters

- How do we get the camera to “canonical form”? (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)

Step 1: Translate by \(-c\)

How do we represent translation as a matrix multiplication?

\[
T = I_{3 \times 3} - c
\]
Extrinsic Parameters

- How do we get the camera to “canonical form”?  
  (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)

\[
R = \begin{bmatrix}
    u^T \\
    v^T \\
    w^T 
\end{bmatrix}
\]

3x3 rotation matrix

Step 1: Translate by \(-c\)
Step 2: Rotate by \(R\)

Extrinsic Parameter

Projection matrix

\[
\Pi = K \begin{bmatrix}
    R & -Rc \\
\end{bmatrix}
\]

\(\Pi = K \begin{bmatrix}
    R & -Rc \\
\end{bmatrix}\) in book’s notation

\[\Pi q = (x, y, z, 1)\] in homogeneous image coordinates

Slide credit: S. Seitz
Distortion

Radial distortion of the image
- Caused by imperfect lenses
- Deviations are most noticeable for rays that pass through the edge of the lens

Correcting radial distortion

from Helmut Dersch

Slide credit: S. Seitz
Modeling Radial Distortion

- Project \((\tilde{x}, \tilde{y}, \tilde{z})\) to “normalized” image coordinates
  \[
  x_n' = \frac{\tilde{x}}{\tilde{z}} \quad y_n' = \frac{\tilde{y}}{\tilde{z}}
  \]

- Apply radial distortion
  \[
  r^2 = x_n'^2 + y_n'^2
  x_d' = x_n'(1 + \kappa_1 r^2 + \kappa_2 r^4)
  y_d' = y_n'(1 + \kappa_1 r^2 + \kappa_2 r^4)
  \]

- Apply focal length \(f\) and translate image center
  \[
  x' = fx_d' + x_c
  y' = fy_d' + y_c
  \]

To model lens distortion
- Use above projection operation instead of standard projection matrix multiplication

Other Types of Distortion
- Lens Vignetting
- Chromatic Aberrations
- Lens Glare
Camera calibration

- Given n points with known 3D coordinates \( X_i \) and known image projections \( x_i \), estimate the camera parameters

\[
\lambda x_i = PX_i
\]

Two linearly independent equations

- \( P \) has 11 degrees of freedom (12 parameters, but scale is arbitrary)
- One 2D/3D correspondence gives us two linearly independent equations
- Homogeneous least squares
- 6 correspondences needed for a minimal solution

Note: for coplanar points that satisfy \( \Pi^T X = 0 \), we will get degenerate solutions (\( \Pi, 0,0 \)), \( (0, \Pi, 0) \), or \( (0,0,\Pi) \)
Camera calibration: Linear method

- Advantages: easy to formulate and solve
- Disadvantages
  - Doesn’t directly tell you camera parameters
  - Doesn’t model radial distortion
  - Can’t impose constraints, such as known focal length and orthogonality
- Non-linear methods are preferred
  - Define error as difference between projected points and measured points
  - Minimize error using Newton’s method or other non-linear optimization

Calibration Demo

Camera Calibration Toolbox for Matlab

http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

Calibration demo

Mosaics

Obtain a wider angle view by combining multiple images.
Mosaics: generating synthetic views

Can generate any synthetic camera view as long as it has the same center of projection!

Image reprojection

Basic question
- How to relate two images from the same camera center?
  - how to map a pixel from PP1 to PP2

Answer
- Cast a ray through each pixel in PP1
- Draw the pixel where that ray intersects PP2

Observation:
Rather than thinking of this as a 3D reprojection, think of it as a 2D image warp from one image to another.

Image reprojection: Homography

A projective transform is a mapping between any two PPs with the same center of projection
- rectangle should map to arbitrary quadrilateral
- parallel lines aren’t
- but must preserve straight lines
called Homography

\[
\begin{bmatrix}
wx' \\
wv' \\
w1
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
w
\end{bmatrix} \begin{bmatrix}
x' \\
y' \\
w1
\end{bmatrix} = \begin{bmatrix}
H
\end{bmatrix} \begin{bmatrix}
p
\end{bmatrix}
\]
To compute the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of $H$ are the unknowns...

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}$$

$x' = h_1^T p/h_3$

$y' = h_2^T p/h_3$

$h_1^T p - x'(h_3 p) = 0$

$h_2^T p - y'(h_3 p) = 0$

$$h = [a \ b \ c]$$

$$\text{Min } ||Lh||^2$$

s.t. $||h||^2 = 1$

$$\text{Min } h^T L^T L h$$

s.t. $||h||^2 = 1$

$$\text{Min } x^T A x$$

(A = $L^T L$, $x = h$)

s.t. $||x||^2 = 1$

Sol: eigen vector of $A$ corresponding to smallest eigen value

To apply a given homography $H$

- Compute $p' = Hp$ (regular matrix multiply)
- Convert $p'$ from homogeneous to image coordinates

$$p' = \begin{bmatrix} w' x' \\ w' y' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & x \\ * & * & * & y \\ * & * & * & 1 \end{bmatrix}$$
Given a coordinate transform and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?

**Image warping**

<table>
<thead>
<tr>
<th>Slide credit: A. Efros</th>
</tr>
</thead>
</table>

Send each pixel $f(x,y)$ to its corresponding location $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands “between” two pixels?

A: distribute color among neighboring pixels $(x',y')$ - Known as “splatting”

**Forward warping**

<table>
<thead>
<tr>
<th>Slide credit: A. Efros</th>
</tr>
</thead>
</table>

Send each pixel $f(x,y)$ to its corresponding location $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands “between” two pixels?

**Inverse warping**

<table>
<thead>
<tr>
<th>Slide credit: A. Efros</th>
</tr>
</thead>
</table>

Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from “between” two pixels?
Inverse warping

\[ f(x, y) \xrightarrow{T^{-1}} g(x', y') \]

Get each pixel \( g(x', y') \) from its corresponding location \( (x, y) = T^{-1}(x', y') \) in the first image.

Q: what if pixel comes from “between” two pixels?
A: Interpolate color value from neighbors
- nearest neighbor, bilinear…

Recap: How to stitch together a panorama (a.k.a. mosaic)?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation (homography) between second image and first using corresponding points.
  - Transform the second image to overlap with the first.
  - Blend the two together to create a mosaic.
  - (If there are more images, repeat)

Bilinear interpolation

Sampling at \( f(x, y) \):

\[
egin{align*}
(i, j + 1) & \quad (i + 1, j + 1) \\
(i, j) & \\
(x, y) & \quad (i + 1, j) \\
\alpha & \quad \beta
\end{align*}
\]

\[
f(x, y) = (1 - a)(1 - b) f[i, j] + a(1 - b) f[i + 1, j] + ab f[i + 1, j + 1] + (1 - a)b f[i, j + 1]
\]
Image rectification

Analyzing patterns and shapes

What is the shape of the b/w floor pattern?

The floor (enlarged)

Automatically rectified floor

From Martin Kemp *The Science of Art*

(Manual reconstruction)

Analyzing patterns and shapes

What is the (complicated) shape of the floor pattern?

St. Lucy Altarpiece, D. Veneziano

Automatically rectified floor

Slide credit: K. Grauman

Slide credit: A. Criminisi

Slide credit: A. Criminisi

Slide credit: A. Criminisi
Analyzing patterns and shapes

Automatic rectification

From Martin Kemp, *The Science of Art*
(manual reconstruction)

Slide credit: A. Criminisi