Lecture 6: Image transformations and alignment
Announcement

• New TA! Adarsh Kowdle

• Office hours: M 11-12, Ward Laboratory 112
Announcements

• Project 1 out, due Thursday, 9/24, by 11:59pm

• Quiz on Thursday, first 10 minutes of class

• Next week: guest lecturer, Prof. Pedro Felzenszwalb, U. Chicago
Announcements

• Project 2 will be released on Tuesday

• You can work in groups of two
  – Send me your groups by Friday evening
Readings

• Szeliski Chapter 3.5 (image warping), 9.1 (motion models)
Announcements

• A total of 3 late days will be allowed for projects
Project 1 questions
Last time: projection

\[(x, y, z) \rightarrow \left( -d\frac{x}{z}, -d\frac{y}{z}, -d \right) \rightarrow \left( -d\frac{x}{z}, -d\frac{y}{z} \right)\]
Projection is a matrix multiply using homogeneous coordinates:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -1/d & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix} =
\begin{bmatrix}
x \\
y \\
-z/d \\
1
\end{bmatrix} \Rightarrow \left(-\frac{d}{z}x, \quad -\frac{d}{z}y\right)
\]

divide by third coordinate

Equivalent to:

\[
\begin{bmatrix}
-d & 0 & 0 & 0 \\
0 & -d & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix} =
\begin{bmatrix}
-dx \\
-dy \\
-z \\
1
\end{bmatrix} \Rightarrow \left(-\frac{d}{z}x, \quad -\frac{d}{z}y\right)
\]
Perspective projection

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

\[K\] (intrinsics) (converts from 3D rays in camera coordinate system to pixel coordinates)

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

\(\alpha\) : 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)
Extrinsics

• 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 \(-c\)
Extrinsics

• 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)

\[
\begin{align*}
T &= \begin{bmatrix}
I_{3\times3} & -c \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{align*}
\]
Extrinsics

• 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 \(-c\)
Step 2: Rotate by \(R\)

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

3x3 rotation matrix
Extrinsics

• 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 \(-c\)
Step 2: Rotate by \(R\)

\[
R = \begin{bmatrix}
  u^T \\
  v^T \\
  w^T 
\end{bmatrix}
\]
\[ \Pi = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I_{3\times3} & -c \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Projection matrix

\[ \Pi q = (x, y, z, 1) \]

(in homogeneous image coordinates)
Perspective distortion

• What does a sphere project to?

Image source: F. Durand
Perspective distortion

- What does a sphere project to?
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
Modeling distortion

Project \((\hat{x}, \hat{y}, \hat{z})\) to “normalized” image coordinates

\[
x_n' = \frac{\hat{x}}{\hat{z}} \quad y_n' = \frac{\hat{y}}{\hat{z}}
\]

\[
r^2 = x_n'^2 + y_n'^2
\]

Apply radial distortion

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

Apply focal length, translate image center

\[
x' = f x_d' + x_c
\]
\[
y' = f y_d' + y_c
\]

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

• Lots of intriguing variants...
• (I’ll just mention a few fun ones)
360 degree field of view...

• Basic approach
  – Take a photo of a parabolic mirror with an orthographic lens (Nayar)
  – Or buy one a lens from a variety of omnicam manufacturers...
    • See http://www.cis.upenn.edu/~kostas/omni.html
Rotating sensor (or object)

Rollout Photographs © Justin Kerr

http://research.famsi.org/kerrmaya.html

Also known as “cyclographs”, “peripheral images”
Photofinish

The 2000 Sydney Olympic Games - 200m Women Final
Questions?
Today: Image transformations and alignment

Full screen panoramas (cubic):  http://www.panoramas.dk/
Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
Why Mosaic?

• Are you getting the whole picture?
  – Compact Camera FOV = 50 x 35°
  – Human FOV = 200 x 135°
Why Mosaic?

• Are you getting the whole picture?
  – Compact Camera FOV = 50 x 35°
  – Human FOV = 200 x 135°
  – Panoramic Mosaic = 360 x 180°
Mosaics: stitching images together
Readings

• Szeliski:
  – Chapter 3.5: Image warping
  – Chapter 5.1: Feature-based alignment
  – Chapter 8.1: Motion models
Image Warping

- image filtering: change *range* of image
  - \( g(x) = h(f(x)) \)

- image warping: change *domain* of image
  - \( g(x) = f(h(x)) \)
Image Warping

• image filtering: change range of image
  \[ g(x) = h(f(x)) \]

• image warping: change domain of image
  \[ g(x) = f(h(x)) \]
Parametric (global) warping

• Examples of parametric warps:
  
  - translation
  - rotation
  - aspect
  - affine
  - perspective
  - cylindrical
Parametric (global) warping

- Transformation $T$ is a coordinate-changing machine:
  $$p' = T(p)$$
- What does it mean that $T$ is global?
  - Is the same for any point $p$
  - Can be described by just a few numbers (parameters)
- Let’s consider linear xforms (can be represented by a 2D matrix):
  $$p' = Tp \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$
Common linear transformations

• Uniform scaling by $s$:

$$S = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

What is the inverse?
Common linear transformations

• Rotation by angle $\theta$ (about the origin)

\[
\mathbf{R} = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]

What is the inverse?

For rotations:
\[
\mathbf{R}^{-1} = \mathbf{R}^T
\]
2x2 Matrices

• What types of transformations can be represented with a 2x2 matrix?

2D mirror about Y axis?
\[
\begin{align*}
x' &= -x \\
y' &= y
\end{align*}
\]
\[
T = \begin{bmatrix}
-1 & 0 \\
0 & -1
\end{bmatrix}
\]

2D mirror over (0,0)?
\[
\begin{align*}
x' &= -x \\
y' &= -y
\end{align*}
\]
\[
T = \begin{bmatrix}
-1 & 0 \\
0 & -1
\end{bmatrix}
\]
2x2 Matrices

• What types of transformations can be represented with a 2x2 matrix?

2D Translation?

\[ \begin{align*}
    x' &= x + t_x \quad \text{NO!} \\
    y' &= y + t_y
\end{align*} \]

Translation is not a linear operation on 2D coordinates
All 2D Linear Transformations

• Linear transformations are combinations of ...
  – Scale,
  – Rotation,
  – Shear, and
  – Mirror

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

• Properties of linear transformations:
  – Origin maps to origin
  – Lines map to lines
  – Parallel lines remain parallel
  – Ratios are preserved
  – Closed under composition

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & e & f \\ c & d & g & h \\ i & j & k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]
Translation

• Solution: homogeneous coordinates to the rescue

\[ T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \]

\[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix} =
\begin{bmatrix}
x + t_x \\
y + t_y \\
1
\end{bmatrix}
\]
Affine transformations

\[ \mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \]

any transformation with last row \([0 \ 0 \ 1]\) we call an \textit{affine} transformation
Basic affine transformations

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Translate

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

2D in-plane rotation

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} =
\begin{bmatrix}
1 & \text{sh}_x & 0 \\
\text{sh}_y & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Shear

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} =
\begin{bmatrix}
-s & s & c_x \\
0 & -\alpha f & c_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

K = intrinsics matrix
Affine Transformations

• Affine transformations are combinations of ...
  – Linear transformations, and
  – Translations

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

• Properties of affine transformations:
  – Origin does not necessarily map to origin
  – Lines map to lines
  – Parallel lines remain parallel
  – Ratios are preserved
  – Closed under composition
Projective Transformations

• Projective transformations ...
  – Affine transformations, and
  – Projective warps

• Properties of projective transformations:
  – Origin does not necessarily map to origin
  – Lines map to lines
  – Parallel lines do not necessarily remain parallel
  – Ratios are not preserved
  – Closed under composition

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

\[ H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \]

Called a homography (or planar perspective map)
Image warping with homographies

image plane in front

black area where no pixel maps to
2D image transformations

These transformations are a nested set of groups
- Closed under composition and inverse is a member
Image Warping

- Given a coordinate xform \((x', y') = T(x, y)\) and a source image \(f(x, y)\), how do we compute an xformed image \(g(x', y') = f(T(x, y))\)?
Forward Warping

• Send each pixel $f(x)$ to its corresponding location $(x',y') = T(x,y)$ in $g(x',y')$
• What if pixel lands “between” two pixels?
Forward Warping

• Send each pixel $f(x,y)$ to its corresponding location $x' = h(x,y)$ in $g(x',y')$

• What if pixel lands “between” two pixels?

• Answer: add “contribution” to several pixels, normalize later (splatting)

• Can still result in holes

![Diagram showing forward warping](attachment:diagram.png)
Inverse Warping

- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x,y)$ in $f(x,y)$
- Requires taking the inverse of the transform
- What if pixel comes from “between” two pixels?
Inverse Warping

• Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$

• What if pixel comes from “between” two pixels?
• Answer: resample color value from interpolated (prefiltered) source image
Interpolation

• Possible interpolation filters:
  – nearest neighbor
  – bilinear
  – bicubic (interpolating)
  – sinc

• Needed to prevent “jaggies” and “texture crawl”
  (with prefiltering)
Questions?

• 3-minute break
Back to mosaics

• How to we align the images?
Creating a panorama

• Basic Procedure
  – Take a sequence of images from the same position
    • Rotate the camera about its optical center
  – Compute transformation between second image and first
  – Transform the second image to overlap with the first
  – Blend the two together to create a mosaic
  – If there are more images, repeat
If we capture all 360° of rays, we can create a 360° panorama.

The basic operation is *projecting* an image from one plane to another.

The projective transformation is scene-INDEPENDENT.
  - This depends on all the images having the same optical center.
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
What is the transformation?

Translations are not enough to align the images.
What is the transformation?

How do we map image 2 onto image 1’s projection plane?

\[
\begin{bmatrix}
X_1 \\
Y_1 \\
Z_1
\end{bmatrix}
= K_2^{-1}
\begin{bmatrix}
x_1 \\
y_1 \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
X_2 \\
Y_2 \\
Z_2
\end{bmatrix}
= R_2^T K_2^{-1}
\begin{bmatrix}
x_1 \\
y_1 \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
x_2 \\
y_2 \\
1
\end{bmatrix}
\sim K_1 R_2^T K_2^{-1}
\begin{bmatrix}
x_1 \\
y_1 \\
1
\end{bmatrix}
\]

3x3 homography
Can we use homography to create a 360 panorama?
Panoramas

• What if you want a 360° field of view?

mosaic Projection Sphere
Spherical projection

- Map 3D point \((X, Y, Z)\) onto sphere
  \[
  (\tilde{x}, \tilde{y}, \tilde{z}) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}} (X, Y, Z)
  \]

- Convert to spherical coordinates
  \[
  (\sin \theta \cos \phi, \sin \phi, \cos \theta \cos \phi) = (\tilde{x}, \tilde{y}, \tilde{z})
  \]

- Convert to spherical image coordinates
  \[
  (\tilde{x}, \tilde{y}) = (s \theta, s \phi) + (\tilde{x}_c, \tilde{y}_c)
  \]
    - \(s\) defines size of the final image
      - often convenient to set \(s = \text{camera focal length}\)
Spherical reprojection

- Map image to spherical coordinates
  - need to know the focal length
Aligning spherical images

• Suppose we rotate the camera by $\theta$ about the vertical axis
  – How does this change the spherical image?
Aligning spherical images

• Suppose we rotate the camera by $\theta$ about the vertical axis
  – How does this change the spherical image?
    – Translation by $\theta$
  – This means that we can align spherical images by translation
Questions?