CS6670: Computer Vision
Noah Snavely

Lecture 6: Image Warping and Projection
Readings

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

• Project 1 assigned, due next Friday 2/18
Dimensionality Reduction Machine
(3D to 2D)

What have we lost?

- Angles
- Distances (lengths)
Projection properties

• Many-to-one: any points along same ray map to same point in image
• Points $\rightarrow$ points
• Lines $\rightarrow$ lines (collinearity is preserved)
  – But line through focal point projects to a point
• Planes $\rightarrow$ planes (or half-planes)
  – But plane through focal point projects to line
Projection properties

- Parallel lines converge at a vanishing point
  - Each direction in space has its own vanishing point
  - But parallels parallel to the image plane remain parallel
2D to 2D warps

• Let’s start with simpler warps that map images to other images

• Examples of 2D $\rightarrow$ 2D warps:

  - translation
  - rotation
  - aspect
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):

$$p' = Tp$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$
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?

\[
x' = -x \\
y' = y
\]

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

2D mirror across line y = x?

\[
x' = y \\
y' = x
\]

\[
T = \begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]
2x2 Matrices

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

**2D Translation?**

\[
\begin{align*}
x' &= x + t_x \\
y' &= y + t_y
\end{align*}
\]

NO!

Translation is not a linear operation on 2D coordinates
Homogeneous coordinates

Trick: add one more coordinate:

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

homogeneous image coordinates

Converting \textit{from} homogeneous coordinates

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

• Solution: homogeneous coordinates to the rescue

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

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

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

any transformation with last row \([0 \ 0 \ 1]\) we call an affine transformation

\[ \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \]
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

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Scale

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  1 & sh_x & 0 \\
  sh_y & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Shear
Modeling projection

• Projection equations
  – Compute intersection with PP of ray from (x,y,z) to COP
  – Derived using similar triangles (on board)

\[(x, y, z) \rightarrow (-d \frac{x}{z}, -d \frac{y}{z}, -d)\]

• We get the projection by throwing out the last coordinate:

\[(x, y, z) \rightarrow (-d \frac{x}{z}, -d \frac{y}{z})\]
Modeling projection

- Is this a linear transformation?
  - no—division by \( z \) is nonlinear

Homogeneous coordinates to the rescue!

Homogeneous image coordinates

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

Homogeneous scene coordinates

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

Converting *from* homogeneous coordinates

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

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

Projection is a matrix multiply using homogeneous coordinates:

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

divide by third coordinate

This is known as perspective projection

- The matrix is the projection matrix

- (Can also represent as a 4x4 matrix – OpenGL does something like this)
Perspective Projection

• How does scaling the projection matrix change the transformation?

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

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

- Special case of perspective projection
  - Distance from the COP to the PP is infinite

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= \begin{bmatrix}
x \\
y \\
1
\end{bmatrix} \Rightarrow (x, y)
Orthographic projection
Perspective projection
Perspective distortion

• What does a sphere project to?

Image source: F. Durand
Perspective distortion

- The exterior columns appear bigger
- The distortion is not due to lens flaws
- Problem pointed out by Da Vinci
Perspective distortion: People
Distortion

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

No distortion  Pin cushion  Barrel
Correcting radial distortion

from Helmut Dersch
Distortion

(a) Orthoscopic

(b) Barrel

(c) Pin-cushion
Modeling distortion

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

- \(x'_n = \hat{x}/\hat{z}\)
- \(y'_n = \hat{y}/\hat{z}\)

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

- \(x' = fx'_d + x_c\)
- \(y' = fy'_d + y_c\)

• To model lens distortion
  – Use above projection operation instead of standard projection matrix multiplication
Other types of 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](http://www.cis.upenn.edu/~kostas/omni.html)
Tilt-shift

http://www.northlight-images.co.uk/article_pages/tilt_and_shift_ts-e.html

Tilt-shift images from Olivo Barbieri and Photoshop imitations
Camera parameters

• How can we model the geometry of a camera?

Two important coordinate systems:
1. *World* coordinate system
2. *Camera* coordinate system
Camera parameters

• To project a point \((x,y,z)\) in \textit{world} coordinates into a camera
• First transform \((x,y,z)\) into \textit{camera} coordinates
• Need to know
  – Camera position (in world coordinates)
  – Camera orientation (in world coordinates)
• The project into the image plane
  – Need to know camera \textit{intrinsics}
Camera parameters

A camera is described by several parameters:
- Translation \( T \) of the optical center from the origin of world coords
- Rotation \( R \) of the image plane
- focal length \( f \), principle point \((x'_c, y'_c)\), pixel size \((s_x, s_y)\)
- blue parameters are called “extrinsics,” red are “intrinsics”

Projection equation

\[
\begin{pmatrix}
    x \\
    y \\
    1
\end{pmatrix} = \begin{pmatrix}
    s_x & 0 & x'_c \\
    0 & s_y & y'_c \\
    0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
    X \\
    Y \\
    Z
\end{pmatrix} = \Pi \mathbf{X}
\]

- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

\[
\Pi = \begin{pmatrix}
    -fs_x & 0 & x'_c \\
    0 & -fs_y & y'_c \\
    0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0
\end{pmatrix} \begin{pmatrix}
    R_{3x3} & 0_{3x1} & I_{3x3} & T_{3x1} \\
    0_{1x3} & 1 & 0_{1x3} & 1
\end{pmatrix}
\]

- The definitions of these parameters are not completely standardized
  - especially intrinsics—varies from one book to another
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 \(-\mathbf{c}\)
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)

Step 1: Translate by \(-c\)

How do we represent translation as a matrix multiplication?

\[
T = \begin{bmatrix}
I_{3 \times 3} & -c \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
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 \(-\mathbf{c}\)
Step 2: Rotate by \(\mathbf{R}\)

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

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 \(R\)
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)
Focal length

• Can think of as “zoom”

• Also related to field of view
\[ \Pi = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3 \times 3} & -c \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Projection matrix

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

(in homogeneous image coordinates)
Questions?

• 3-minute break
Image 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 alignment

Image taken from same viewpoint, just rotated.

Can we line them up?
Image alignment

Why don’t these image line up exactly?
What is the geometric relationship between these two images?
What is the geometric relationship between these two images?
Is this an affine transformation?
Where do we go from here?

\[
\begin{bmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{bmatrix}
\]

affine transformation

what happens when we mess with this row?
Projective Transformations aka Homographies aka Planar Perspective Maps

\[
H = \begin{bmatrix}
    a & b & c \\
    d & e & f \\
    g & h & 1 \\
\end{bmatrix}
\]

Called a *homography* (or *planar perspective map*)
Homographies

• Example on board