Image Warping (Szeliski 3.6.1)

http://www.jeffrey-martin.com

cs129: Computational Photography
James Hays, Brown, Fall 2012
Image Transformations

image filtering: change **range** of image

\[ g(x) = T(f(x)) \]

image warping: change **domain** of image

\[ g(x) = f(T(x)) \]
Image Transformations

image filtering: change range of image

\[ g(x) = T(f(x)) \]

image warping: change domain of image

\[ g(x) = f(T(x)) \]
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 and parametric?

- Is the same for any point \( p \)
- Can be described by just a few numbers (parameters)

Let’s represent \( T \) as a matrix:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  x \\
  y
\end{bmatrix} \begin{bmatrix}
  m_{11} & m_{12} \\
  m_{21} & m_{22}
\end{bmatrix}
\]

\( \mathbf{p}' = \mathbf{M}\mathbf{p} \)
Scaling

Scaling a coordinate means multiplying each of its components by a scalar.

Uniform scaling means this scalar is the same for all components.
**Scaling**

*Non-uniform scaling*: different scalars per component:

$X \times 2, \quad Y \times 0.5$
Scaling

Scaling operation:

\[ x' = ax \]
\[ y' = by \]

Or, in matrix form:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= 
\begin{bmatrix}
  a & 0 \\
  0 & b
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

What's inverse of \( S \)?
2-D Rotation

\[ x' = x \cos(\theta) - y \sin(\theta) \]
\[ y' = x \sin(\theta) + y \cos(\theta) \]
2-D Rotation

\[
x = r \cos (\phi)
\]
\[
y = r \sin (\phi)
\]
\[
x' = r \cos (\phi + \theta)
\]
\[
y' = r \sin (\phi + \theta)
\]

Trig Identity…
\[
x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)
\]
\[
y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)
\]

Substitute…
\[
x' = x \cos(\theta) - y \sin(\theta)
\]
\[
y' = x \sin(\theta) + y \cos(\theta)
\]
2-D Rotation

This is easy to capture in matrix form:

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

\( R \)

Even though \( \sin(\theta) \) and \( \cos(\theta) \) are nonlinear functions of \( \theta \),

- \( x' \) is a linear combination of \( x \) and \( y \)
- \( y' \) is a linear combination of \( x \) and \( y \)

What is the inverse transformation?

- Rotation by \(-\theta\)
- For rotation matrices \( R^{-1} = R^T \)
2x2 Matrices

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

2D Identity?

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

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

2D Scale around (0,0)?

\[ x' = s_x * x \]
\[ y' = s_y * y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  s_x & 0 \\
  0 & s_y
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2x2 Matrices

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

2D Rotate around (0,0)?

\[
x' = \cos \Theta \cdot x - \sin \Theta \cdot y \\
y' = \sin \Theta \cdot x + \cos \Theta \cdot y
\]

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

2D Shear?

\[
x' = x + s h_x \cdot y \\
y' = s h_y \cdot x + y
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} =
\begin{bmatrix}
1 & s h_x \\
 s h_y & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]
2x2 Matrices

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

2D Mirror about Y axis?

\[
\begin{align*}
x' &= -x \\
y' &= y
\end{align*}
\]

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

2D Mirror over (0,0)?

\[
\begin{align*}
x' &= -x \\
y' &= -y
\end{align*}
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
-1 & 0 \\
0 & -1
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix}
\]
2x2 Matrices

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

2D Translation?

$$x' = x + t_x$$

$$y' = y + t_y$$

Only linear 2D transformations can be represented with a 2x2 matrix.
Linear transformations are combinations of …

- Scale,
- Rotation,
- Shear, and
- Mirror

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

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

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
a & b & e & f & i & j \\
c & d & g & h & k & l
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix}
\]
Q: How can we represent translation as a 3x3 matrix?

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

Homogeneous coordinates
- represent coordinates in 2 dimensions with a 3-vector

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

\[
\begin{bmatrix}
  \text{homogenous coords} \\
  x \\
  y \\
  1
\end{bmatrix}
\]
Homogeneous Coordinates

Add a 3rd coordinate to every 2D point

- \((x, y, w)\) represents a point at location \((x/w, y/w)\)
- \((x, y, 0)\) represents a point at infinity
- \((0, 0, 0)\) is not allowed

Convenient coordinate system to represent many useful transformations

\((2,1,1)\) or \((4,2,2)\) or \((6,3,3)\)
Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix?

\[
x' = x + t_x
\]

\[
y' = y + t_y
\]

A: Using the rightmost column:

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

Example of translation

Homogeneous Coordinates

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} = \begin{bmatrix}
  x + t_x \\
  y + t_y \\
  1
\end{bmatrix}
\]
Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

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

Translate

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

Rotate

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

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
Matrix Composition

Transformations can be combined by matrix multiplication

\[
\begin{bmatrix}
    x' \\
    y' \\
    w'
\end{bmatrix} = \begin{bmatrix}
    1 & 0 & tx \\
    0 & 1 & ty \\
    0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    \cos \Theta & -\sin \Theta & 0 \\
    \sin \Theta & \cos \Theta & 0 \\
    0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    sx & 0 & 0 \\
    0 & sy & 0 \\
    0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    x \\
    y \\
    w
\end{bmatrix}
\]

\[p' = T(t_x, t_y) \quad R(\Theta) \quad S(s_x, s_y) \quad p\]
Affine Transformations

Affine transformations are combinations of …
- Linear transformations, and
- Translations

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
- Models change of basis

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

Will the last coordinate $w$ always be 1?
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
  • Models change of basis

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

2D image transformations

These transformations are a nested set of groups
  • Closed under composition and inverse is a member
Matlab Demo

“help imtransform”
Recovering Transformations

What if we know \( f \) and \( g \) and want to recover the transform \( T \)?

- e.g. better align images from Project 1
- willing to let user provide correspondences
  - How many do we need?
Translation: # correspondences?

How many correspondences needed for translation?
How many Degrees of Freedom?
What is the transformation matrix?

\[
\mathbf{M} = \begin{bmatrix}
1 & 0 & p'_x - p_x \\
0 & 1 & p'_y - p_y \\
0 & 0 & 1
\end{bmatrix}
\]
Euclidian: # correspondences?

How many correspondences needed for translation+rotation?
How many DOF?
Affine: # correspondences?

How many correspondences needed for affine?
How many DOF?
How many correspondences needed for projective?
How many DOF?
Given a coordinate transform \((x',y') = T(x,y)\) and a source image \(f(x,y)\), how do we compute a transformed image \(g(x',y') = f(T(x,y))\)?
Forward warping

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?
Forward warping

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”
   – Check out griddata in Matlab
Inverse warping

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

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, Gaussian, bicubic
   - Check out interp2 in Matlab
Forward vs. inverse warping

Q: which is better?

A: usually inverse—eliminates holes
  • however, it requires an invertible warp function—not always possible...