Geometric Image Transformations

- Image Warping
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 image warps:

translation
rotation
aspect
affine
perspective
cylindrical
Transformation Function

Transform the geometry of an image to a desired geometry
Definition: Image Warping

**Source Image**: Image to be used as the reference. The geometry of this image is no changed.

**Target Image**: this image is obtained by transforming the reference image.

\((x,y)\): coordinates of points in the reference image

\((u,v)\): coordinates of points in the target image

\(f,g\) or \(F,G\): x and y components of a transformation function
Definition: Image Warping

**Control points:** Unique points in the reference and target images. The coordinates of corresponding control points in images are used to determine a transformation function.
A Transformation Function

Used to compute the corresponding points

\[ u = f(x, y) \]
\[ v = g(x, y) \]
\[ x = F(u, v) \]
\[ y = G(x, v) \]
Affine Transform

Have the form:
\[ u = ax + by + c \]
\[ v = dx + ey + f \]

In matrix notation:
\[ \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]

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

- A combination of 2-D scale, rotation, shear, and translation transformations.
- Allows a square to be distorted into any parallelogram.
- 6 degrees of freedom (a, b, c, d, e, f)
- Inverse is of same form (is also affine). Given by inverse of 3X3 matrix above.
- Good when controlling a warp with triangles, since 3 points in 2D determined the 6 degrees of freedom.
Projective Transform
(a.k.a “perspective”)

Have the form:

\[
\begin{align*}
    u &= (ax+by+c)/(gx+hy+i) \\
    v &= (dx+ey+f)/(gx+hy+i)
\end{align*}
\]

In matrix notation:

\[
\begin{bmatrix}
    uq \\
    vq \\
    q
\end{bmatrix} =
\begin{bmatrix}
    a & b & c \\
    d & e & f \\
    g & h & i
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\quad \text{and} \quad
u = uq / q, \quad v = vq / q
\]

- Linear numerator & denominator
- If \(g=h=0\), then you get affine as a special case
- Allow a square to be distorted into any quadrilateral
- 8 degrees of freedom (a-h). We can choose \(i=1\), arbitrarily
- Inverse is of same form (is also projective).
- Good when controlling a warp with quadrilaterals, since 4 points in 2D determine the 8 degrees of freedom
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:

- $X \times 2$
- $Y \times 0.5$
Scaling

Scaling operation:

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

Or, in matrix form:

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

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \text{scaling matrix } S
\]

What's inverse of S?

\[ x' \text{ same as } u \]
\[ \text{in other slides} \]
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…
\[
x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta) \\
y' = r \sin(\phi) \sin(\theta) + r \cos(\phi) \cos(\theta)
\]

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:

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

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, \(\text{det}(R) = 1\) so \( R^{-1} = R^T \)
2x2 Matrices

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

2D Identity?

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

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

2D Scale around (0,0)?

\[
\begin{align*}
x' &= s_x \times x \\
y' &= s_y \times y
\end{align*}
\]

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

\[
x' = x + sh_x \cdot y \\
y' = sh_y \cdot x + y
\]

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

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

2D Mirror about Y axis?

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

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

2D Mirror over \((0,0)\)?

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

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  -1 & 0 \\
  0 & -1
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
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.
All 2D Linear Transformations

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

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

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  a & b & e & f \\
  c & d & g & h \\
  i & j & k & l
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
Linear Transformations as Change of Basis

Any linear transformation is a basis!!!

- What's the inverse transform?
- How can we change from any basis to any basis?
- What if the basis are orthogonal?

\[ p = 4i + 3j = (4, 3) \]

\[ p' = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} p \]

\[ v = (v_x, v_y) \]

\[ u = (u_x, u_y) \]

\[ p_x' = 4u_x + 3v_x \]
\[ p_y' = 4u_y + 3v_y \]

Any linear transformation is a basis!!!

- What's the inverse transform?
- How can we change from any basis to any basis?
- What if the basis are orthogonal?
Homogeneous Coordinates

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

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

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

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

Translation
Translation

Example of translation

Homogeneous Coordinates

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

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

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
Given a coordinate transform function $f, g$ or $F, G$ and source image $S(x,y)$, how do we compute a transformed image $T(u,v)$?
Forward Warping

Forward warping algorithm:

for $y = y_{\text{min}}$ to $y_{\text{max}}$

for $x = x_{\text{min}}$ to $x_{\text{max}}$

$u = f(x,y)$; $v = g(x,y)$

copy pixel at source $S(x,y)$ to $T(u,v)$
Forward Warping

Forward warping algorithm:

for \( y = y_{\text{min}} \) to \( y_{\text{max}} \)

for \( x = x_{\text{min}} \) to \( x_{\text{max}} \)

\[ u = f(x,y); \ v = g(x,y) \]

copy pixel at source \( S(x,y) \) to \( T(u,v) \)

Any problems for forward warping?
Forward Warping

Q: What if the transformed pixel is located between pixels?
Q: What if the transformed pixel is located between pixels?

A: Distribute color among neighboring pixels
   - known as “splatting”
Forward Warping

- Iterate over source, sending pixels to destination
- Some source pixels maps to the same dest. pixel
- Some dest. pixels may have no corresponding source
- Holes in reconstruction
- Must splat etc.

for \( y = y_{\text{min}} \) to \( y_{\text{max}} \)
  for \( x = x_{\text{min}} \) to \( x_{\text{max}} \)
    \[ u = f(x,y); \ v = g(x,y) \]
    copy pixel at source \( S(x,y) \) to \( T(u,v) \)
Forward Warping

- Iterate over source, sending pixels to destination
- Some source pixels map to the same dest. pixel
- Some dest. pixels may have no corresponding source
- Holes in reconstruction
- Must splat etc.

\[
\text{for } y = y_{\text{min}} \text{ to } y_{\text{max}} \\
\text{for } x = x_{\text{min}} \text{ to } x_{\text{max}} \\
u = f(x,y); v = g(x,y) \\
\text{copy pixel at source } S(x,y) \text{ to } T(u,v)
\]
Forward Warping

- Iterate over source, sending pixels to destination
- Some source pixels map to the same dest. pixel
- Some dest. pixels may have no corresponding source
- Holes in reconstruction
- Must splat etc.

for $y = y_{\text{min}}$ to $y_{\text{max}}$
for $x = x_{\text{min}}$ to $x_{\text{max}}$
  $u = f(x,y); v = g(x,y)$
  copy pixel at source $S(x,y)$ to $T(u,v)$

- How to remove the holes?
Inverse Warping

Inverse warping algorithm:

for \( v = v_{\text{min}} \) to \( v_{\text{max}} \)
  for \( u = u_{\text{min}} \) to \( u_{\text{max}} \)
    \( x = F(u,v); \ y = G(u,v) \)
    copy pixel at source \( S(x,y) \) to \( T(u,v) \)
Inverse Warping

Q: What if pixel comes from “between” two pixels?

A: Interpolate color values from neighboring pixels
Inverse Warping

• Iterate over dest., finding pixels from source
• Non-integer evaluation source image, resample
• May oversample source
• But no holes
• Simpler, better than forward mapping

\[
\begin{align*}
&\text{for } v = v_{\min} \text{ to } v_{\max} \\
&\quad \text{for } u = u_{\min} \text{ to } u_{\max} \\
&\quad \quad x = F(u,v); y = G(u,v) \\
&\quad \quad \text{copy pixel at source } S(x,y) \text{ to } T(u,v)
\end{align*}
\]
Resampling Filter

Image warping requires resampling: converting a digital signal from one sampling grid to another. It’s 2-D signal resampling.

If you copy pixels (“point sampling”), as in the previous pseudocode, results are ugly.

As in 1-D resampling, filtering must be used:

- if the mapping scales up (stretches), the operation is called upsampling or interpolation. *If done poorly, you get rastering (e.g. pixel replication).*
- if the mapping scales down (squeezes), the operation is called downsampling or decimation. *If done poorly, you get aliasing (e.g. moire).*

High quality resampling with arbitrary scale factors requires careful use of low pass filters of variable support (width). But for a non-affine warp, a shift-variant filter is needed -- these are not widely studied in the signal processing literature.

Good resampling for scale factors near 1 (not scaling up or down much) can be done with bilinear interpolation. OK for most image warping.
This is a 2D signal reconstruction problem!
Point Sampling

Nearest neighbor

- Copies the color of the pixel with the closest integer coordinate
- A fast and efficient way to process textures if the size of the target is similar to the size of the reference
- Otherwise, the result can be a chunky, aliased, or blurred image.
Bilinear Filter

Weighted sum of four neighboring pixels
Bilinear Filter

Sampling at $S(x,y)$:

$$S(x,y) = (1-a)(1-b)S(i,j) + a(1-b)S(i+1,j) + (1-a)bS(i,j+1) + abS(i+1,j+1)$$
Bilinear Filter

Sampling at $S(x,y)$:

$$S(x,y) = a \cdot b \cdot S(i,j) + a \cdot (1-b) \cdot S(i+1,j) + (1-a) \cdot b \cdot S(i,j+1) + (1-a) \cdot (1-b) \cdot S(i+1,j+1)$$

To optimize the above, do the following:

$$S_i = S(i,j) + a \cdot (S(i,j+1)-S(i))$$
$$S_j = S(i+1,j) + a \cdot (S(i+1,j+1)-S(i+1,j))$$
$$S(x,y) = S_i + b \cdot (S_j - S_i)$$
Bilinear Filter

\[ y \]

\[(i,j) \quad (i,j+1) \]
\[
\begin{array}{ccc}
(i+1,j) & & (i+1,j+1) \\
\end{array}
\]

\[
\begin{array}{c}
\text{a} \quad \text{b} \\
\end{array}
\]

\[
\begin{array}{c}
x \quad y \\
\end{array}
\]
Inverse Warping and Resampling

Inverse warping algorithm:

for \( v = v_{\text{min}} \) to \( v_{\text{max}} \)

for \( u = u_{\text{min}} \) to \( u_{\text{max}} \)

float \( x = F(u,v) \); float \( y = G(u,v) \);

\( T(u,v) = \text{resample_source}(x,y,w); \)
Computing Affine Transformations between Sets of Matching Points

Given 3 matching pairs of points, the affine transformation can be computed through solving a simple matrix equation.

\[
\begin{bmatrix}
  u_1 & u_2 & u_3 \\
  v_1 & v_2 & v_3 \\
  1 & 1 & 1
\end{bmatrix}
= \begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_1 & x_2 & x_3 \\
  y_1 & y_2 & y_3 \\
  1 & 1 & 1
\end{bmatrix}
\]