Image Warping

Salvador Dalí, “The Persistence of Memory”

CSC320: Introduction to Visual Computing
Michael Guerzhoy

Many slides from Derek Hoiem, Alyosha Efros, Steve Seitz
Morphing

Blend from one object to other with a series of local transformations
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

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)

For linear transformations, we can represent $T$ as a matrix

$$p' = M p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} M$$
Parametric (global) warping

Examples of parametric warps:

- translation
- rotation
- aspect
- affine
- perspective
- cylindrical
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: \[ x' = a x \]
  \[ y' = b y \]

- Or, in matrix form:
  \[
  \begin{bmatrix}
  x' \\
  y'
  \end{bmatrix}
  =
  \begin{bmatrix}
  a & 0 \\
  0 & b
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \]

  scaling matrix \( S \)

What is the transformation from \((x', y')\) to \((x, y)\)?
2-D Rotation

\[
x' = x \cos(\theta) - y \sin(\theta)
\]

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

Polar coordinates…

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

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:

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

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?

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

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

2D Scale around (0,0)?

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

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

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

2D Rotate around (0,0)?

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

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

2D Shear?

\[ x' = x + k_x \times y \]
\[ y' = k_y \times x + y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  1 & k_x \\
  k_y & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
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?

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

NO!

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
- Closed under composition

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

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

Q: How can we represent translation in matrix form?

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

**Homogeneous coordinates**

- represent coordinates in 2 dimensions with a 3-vector

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

\[
\xrightarrow{\text{homogeneous coords}}
\]

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

2D Points $\rightarrow$ Homogeneous Coordinates
- Append 1 to every 2D point: $(x \ y) \rightarrow (x \ y \ 1)$

Homogeneous coordinates $\rightarrow$ 2D Points
- Divide by third coordinate $(x \ y \ w) \rightarrow (x/w \ y/w)$

Special properties
- Scale invariant: $(x \ y \ w) = k \times (x \ y \ w)$
- $(x, y, 0)$ represents a point at infinity
- $(0, 0, 0)$ is not allowed

Scale Invariance

![Graph showing scale invariance with points (2,1,1) or (4,2,2) or (6,3,3) on a Cartesian plane.](image)
Homogeneous Coordinates

Q: How can we represent translation in matrix form?

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

A: Using the rightmost column:

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

Homogeneous Coordinates

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

\(t_x = 2\)
\(t_y = 1\)
Basic 2D transformations as 3x3 matrices

Translate

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

Scale

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

Rotate

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

Shear

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} =
\begin{bmatrix}
    1 & \beta_x & 0 \\
    \beta_y & 1 & 0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]
Matrix Composition

Transformations can be combined by matrix multiplication

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

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

Does the order of multiplication matter?
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
- Closed under composition

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

Projective transformations are combos of
  • 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
  • Projective matrix is defined up to a scale (8 DOF)
2D image transformations

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Matrix</th>
<th># D.O.F.</th>
<th>Preserves:</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation</td>
<td>$[I \</td>
<td>\ t]_{2\times3}$</td>
<td>2</td>
<td>orientation + ⋯</td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>$[R \</td>
<td>\ t]_{2\times3}$</td>
<td>3</td>
<td>lengths + ⋯</td>
</tr>
<tr>
<td>similarity</td>
<td>$[sR \</td>
<td>\ t]_{2\times3}$</td>
<td>4</td>
<td>angles + ⋯</td>
</tr>
<tr>
<td>affine</td>
<td>$[A]_{2\times3}$</td>
<td>6</td>
<td>parallelism + ⋯</td>
<td></td>
</tr>
<tr>
<td>projective</td>
<td>$[\tilde{H}]_{3\times3}$</td>
<td>8</td>
<td>straight lines</td>
<td></td>
</tr>
</tbody>
</table>
Image warping

- Given a coordinate transform \((x', y') = T(x, y)\) and a source image \(f(x, y)\), how do we compute a transformed image \(g(x', y')\)?
Forward warping

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

\((x',y') = T(x,y)\) in the second image

What is the problem with this approach?

- Send each pixel \(f(x,y)\) to its corresponding location

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

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

\[(x,y) = T_i(x',y')\] in the first image

- Get each pixel \(g(x',y')\) from its corresponding location

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

A: Interpolate color value from neighbors
   - nearest neighbor, bilinear, Gaussian, bicubic
   - E.g. scipy.interpolate.interp2d
Forward vs. inverse warping

• Q: which is better?

• A: Usually inverse—eliminates holes
  – however, it requires an invertible warp function
Recovering Transformations

- What if we know $f$ and $g$ and want to recover the transform $T$?
  - willing to let user provide correspondences
  - How many do we need?
Translation: # correspondences?

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

\[ T(x,y) \]

\[
\begin{bmatrix}
1 & 0 & p_x' - p_x \\
0 & 1 & p_y' - p_y \\
0 & 0 & 1
\end{bmatrix}
\]
Euclidian: # correspondences?

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

• How many DOF?
• How many correspondences needed for affine?
Projective: # correspondences?

• How many DOF?
• How many correspondences needed for projective?