2D transformations (a.k.a. warping)
Course announcements

• Homework 1 posted on course website.
  - Due on February 5\textsuperscript{th} at 23:59.
  - This homework is in Matlab.
  - How many of you have looked at/started/finished homework 1?

• Homework 2 will be posted tonight on the course website.
  - It will be due on February 19\textsuperscript{th} at 23:59 pm.
  - Start early because it is much larger and more difficult than homework 1.

• Second theory quiz on course website and due on February 10\textsuperscript{th}, at 23:59.

• Starting with quiz 3, release/due dates will be on Sunday.

• Reminder: All office hours are at the Smith Hall 200 conference room (not my office!).

• How was the first quiz?
Overview of today’s lecture

• Reminder: image transformations.
• 2D transformations.
• Projective geometry 101.
• Transformations in projective geometry.
• Classification of 2D transformations.
• Determining unknown 2D transformations.
• Determining unknown image warps.
Most of these slides were adapted from:

Reminder: image transformations
What is an image?

A (grayscale) image is a 2D function.

What is the range of the image function $f$?

A (grayscale) image is a 2D function.

Domain $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
What types of image transformations can we do?

Filtering

changes pixel values

Warping

changes pixel locations
What types of image transformations can we do?

Filtering

\[ G(x) = h\{F(x)\} \]

changes range of image function

Warping

\[ G(x) = F(h\{x\}) \]

changes domain of image function
Warping example: feature matching
Warping example: feature matching
Warping example: feature matching

- object recognition
- 3D reconstruction
- augmented reality
- image stitching

How do you compute the transformation?
Warping example: feature matching

Given a set of matched feature points:

\[ \{x_i, x'_i\} \]

point in one image \rightarrow point in the other image

and a transformation:

\[ x' = f(x; p) \]

transformation function \rightarrow parameters

find the best estimate of the parameters \[ p \]

What kind of transformation functions \( f \) are there?
2D transformations
2D transformations

- translation
- rotation
- aspect
- affine
- perspective
- cylindrical
2D planar transformations
2D planar transformations

• Each component multiplied by a scalar
• Uniform scaling - same scalar for each component

How would you implement scaling?
2D planar transformations

Each component multiplied by a scalar
Uniform scaling - same scalar for each component

$x' = ax$
$y' = by$

What’s the effect of using different scale factors?
2D planar transformations

- Each component multiplied by a scalar
- Uniform scaling - same scalar for each component

$x' = ax$
$y' = by$

Matrix representation of scaling:

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

Scaling matrix $S$
2D planar transformations

Shear

How would you implement shearing?
2D planar transformations

Shear

or in matrix form:

\[ x' = x + a \cdot y \]
\[ y' = b \cdot x + y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  1 & a \\
  b & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2D planar transformations

How would you implement rotation?

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

rotation around the origin

\[ x = \begin{bmatrix} x \\ y \end{bmatrix} \]
2D planar transformations

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

\[ y' = x \sin \theta + y \cos \theta \]

rotation around the origin
2D planar transformations

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

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

Substitute...
\[x' = x \cos(\theta) - y \sin(\theta)\]
\[y' = x \sin(\theta) + y \cos(\theta)\]
2D planar transformations

\[ x' = x \cos \theta - y \sin \theta \]
\[ y' = x \sin \theta + y \cos \theta \]

or in matrix form:
\[ \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \]
2D planar and linear transformations

\[ x' = f(x; p) \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = M \begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
# 2D planar and linear transformations

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>$M = \begin{bmatrix} s_x &amp; 0 \ 0 &amp; s_y \end{bmatrix}$</td>
</tr>
<tr>
<td>Flip across $y$</td>
<td>$M = \begin{bmatrix} -1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Rotate</td>
<td>$M = \begin{bmatrix} \cos \theta &amp; -\sin \theta \ \sin \theta &amp; \cos \theta \end{bmatrix}$</td>
</tr>
<tr>
<td>Flip across origin</td>
<td>$M = \begin{bmatrix} -1 &amp; 0 \ 0 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Shear</td>
<td>$M = \begin{bmatrix} 1 &amp; s_x \ s_y &amp; 1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Identity</td>
<td>$M = \begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>
2D translation

How would you implement translation?
2D translation

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

What about matrix representation?

\[
M = \begin{bmatrix}
? & ? \\
? & ?
\end{bmatrix}
\]
2D translation

$x' = x + t_x$

$y' = y + t_x$

What about matrix representation?

Not possible.
Projective geometry 101
Homogeneous coordinates

- Represent 2D point with a 3D vector

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

- Represent 2D point with a 3D vector
- 3D vectors are only defined up to scale
2D translation

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

What about matrix representation using homogeneous coordinates?
2D translation

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

What about matrix representation using heterogeneous coordinates?

\[
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix} \Rightarrow
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix} \quad M = \begin{bmatrix}
    1 & 0 & t_x \\
    0 & 1 & t_y \\
    0 & 0 & 1
\end{bmatrix}
\]
2D translation using homogeneous coordinates

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

\[t_x = 2\]
\[t_y = 1\]
Homogeneous coordinates

Conversion:

• heterogeneous → homogeneous

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

• homogeneous → heterogeneous

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

• scale invariance

\[
\begin{bmatrix} x & y & w \end{bmatrix}^T = \lambda \begin{bmatrix} x & y & w \end{bmatrix}^T
\]

Special points:

• point at infinity

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

• undefined

\[
\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}
\]
Projective geometry

image point in pixel coordinates \( \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \)

image point in homogeneous coordinates \( \mathbf{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \)

What does scaling \( \mathbf{X} \) correspond to?

\( \mathbf{X} \) is a projection of a point \( P \) on the image plane.
Transformations in projective geometry
2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

Translation:

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

Scaling:

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

Rotation:

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

Shearing:

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} \begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]
2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

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

scaling

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

rotation

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

shearing
2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

Translation:

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

Scaling:

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

Rotation:

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

Shearing:

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix}
    1 & \beta_x & 0 \\
    \beta_y & 1 & 0 \\
    0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]
2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

Translation

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

Rotation

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

Scaling

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

Shearing
Matrix composition

Transformations can be combined by matrix multiplication:

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

$$\mathbf{p}' = ? ? ? ? \mathbf{p}$$
Matrix composition

Transformations can be combined by matrix multiplication:

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

\[
p' = \text{translation}(t_x, t_y) \quad \text{rotation}(\theta) \quad \text{scale}(s, s) \quad p
\]

Does the multiplication order matter?
Classification of 2D transformations
Classification of 2D transformations
## Classification of 2D transformations

<table>
<thead>
<tr>
<th>Name</th>
<th>Matrix</th>
<th># D.O.F.</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation</td>
<td>$\begin{bmatrix} I &amp; t \end{bmatrix}$</td>
<td>?</td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>$\begin{bmatrix} R &amp; t \end{bmatrix}$</td>
<td>?</td>
</tr>
<tr>
<td>similarity</td>
<td>$\begin{bmatrix} sR &amp; t \end{bmatrix}$</td>
<td>?</td>
</tr>
<tr>
<td>affine</td>
<td>$\begin{bmatrix} A \end{bmatrix}$</td>
<td>?</td>
</tr>
<tr>
<td>projective</td>
<td>$\begin{bmatrix} \tilde{H} \end{bmatrix}$</td>
<td>?</td>
</tr>
</tbody>
</table>
Classification of 2D transformations

Translation:

\[
\begin{bmatrix}
1 & 0 & t_1 \\
0 & 1 & t_2 \\
0 & 0 & 1
\end{bmatrix}
\]

How many degrees of freedom?
Classification of 2D transformations

Euclidean (rigid): rotation + translation

\[
\begin{bmatrix}
  r_1 & r_2 & r_3 \\
  r_4 & r_5 & r_6 \\
  0 & 0 & 1
\end{bmatrix}
\]

Are there any values that are related?
Classification of 2D transformations

Euclidean (rigid): rotation + translation

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & r_3 \\
\sin \theta & \cos \theta & r_6 \\
0 & 0 & 1
\end{bmatrix}
\]

How many degrees of freedom?
Classification of 2D transformations

Euclidean (rigid): rotation + translation

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & r_3 \\
\sin \theta & \cos \theta & r_6 \\
0 & 0 & 1
\end{bmatrix}
\]

which other matrix values will change if this increases?
Classification of 2D transformations

what will happen to the image if this increases?

Euclidean (rigid): rotation + translation

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & r_3 \\
\sin \theta & \cos \theta & r_6 \\
0 & 0 & 1
\end{bmatrix}
\]
Classification of 2D transformations

Euclidean (rigid): rotation + translation

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & r_3 \\
\sin \theta & \cos \theta & r_6 \\
0 & 0 & 1
\end{bmatrix}
\]

what will happen to the image if this increases?
Classification of 2D transformations

Similarity: uniform scaling + rotation + translation

\[
\begin{bmatrix}
  r_1 & r_2 & r_3 \\
  r_4 & r_5 & r_6 \\
  0 & 0 & 1
\end{bmatrix}
\]

Are there any values that are related?
Classification of 2D transformations

Multiply these four by scale $s$

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & r_3 \\
\sin \theta & \cos \theta & r_6 \\
0 & 0 & 1
\end{bmatrix}
\]

Similarity: uniform scaling + rotation + translation

How many degrees of freedom?
Classification of 2D transformations

what will happen to the image if this increases?

Similarity:
uniform scaling + rotation + translation

\[
\begin{bmatrix}
  r_1 & r_2 & r_3 \\
  r_4 & r_5 & r_6 \\
  0 & 0 & 1
\end{bmatrix}
\]
Classification of 2D transformations

Affine transform:
uniform scaling + shearing
+ rotation + translation

\[
\begin{bmatrix}
    a_1 & a_2 & a_3 \\
    a_4 & a_5 & a_6 \\
    0 & 0 & 1
\end{bmatrix}
\]

Are there any values that are related?
Classification of 2D transformations

Affine transform:
uniform scaling + shearing
+ rotation + translation

\[
\begin{bmatrix}
  a_1 & a_2 & a_3 \\
  a_4 & a_5 & a_6 \\
  0 & 0 & 1
\end{bmatrix}
\]

Are there any values that are related?

\[
\begin{bmatrix}
  s r_1 & s r_2 \\
  s r_3 & s r_4
\end{bmatrix}
\begin{bmatrix}
  1 & h_1 \\
  h_2 & 1
\end{bmatrix}
= 
\begin{bmatrix}
  s r_1 + h_2 s r_2 & s r_2 + h_1 s r_1 \\
  s r_3 + h_2 s r_4 & s r_4 + h_1 s r_3
\end{bmatrix}
\]
Classification of 2D transformations

Affine transform:
uniform scaling + shearing
+ rotation + translation

\[
\begin{bmatrix}
  a_1 & a_2 & a_3 \\
  a_4 & a_5 & a_6 \\
  0 & 0 & 1
\end{bmatrix}
\]

How many degrees of freedom?

\[
\begin{bmatrix}
  sr_1 & sr_2 \\
  sr_3 & sr_4
\end{bmatrix}
\begin{bmatrix}
  1 & h_1 \\
  h_2 & 1
\end{bmatrix}
= \begin{bmatrix}
  sr_1 + h_2 sr_2 & sr_2 + h_1 sr_1 \\
  sr_3 + h_2 sr_4 & sr_4 + h_1 sr_3
\end{bmatrix}
\]
Affine transformations are combinations of
- arbitrary (4-DOF) 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 map to parallel lines
- ratios are preserved
- compositions of affine transforms are also affine transforms

Does the last coordinate w ever change?
Affine transformations

Affine transformations are combinations of
• arbitrary (4-DOF) 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 map to parallel lines
• ratios are preserved
• compositions of affine transforms are also affine transforms

Nope! But what does that mean?
How to interpret affine transformations here?

image point in pixel coordinates: \( \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \)

image point in heterogeneous coordinates: \( \mathbf{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \)

\( \mathbf{X} \) is a projection of a point \( \mathbf{P} \) on the image plane.
Projective transformations are combinations of

- affine transformations; and
- projective wraps

Properties of projective transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms

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

How many degrees of freedom?
Projective transformations

Projective transformations are combinations of
- affine transformations; and
- projective wraps

Properties of projective transformations:
- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms

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

8 DOF: vectors (and therefore matrices) are defined up to scale
How to interpret projective transformations here?

image point in pixel coordinates

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

image point in heterogeneous coordinates

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

\( X \) is a projection of a point \( P \) on the image plane

\( z = 1 \)
Determining unknown 2D transformations
Determining unknown transformations

Suppose we have two triangles: ABC and DEF.
Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
Determining unknown transformations

Suppose we have two triangles: ABC and DEF.
• What type of transformation will map A to D, B to E, and C to F?
• How do we determine the unknown parameters?

Affine transform: uniform scaling + shearing + rotation + translation

\[
\begin{bmatrix}
  a_1 & a_2 & a_3 \\
  a_4 & a_5 & a_6 \\
  0 & 0 & 1 \\
\end{bmatrix}
\]

How many degrees of freedom do we have?
Determining unknown transformations

Suppose we have two triangles: ABC and DEF.
• What type of transformation will map A to D, B to E, and C to F?
• How do we determine the unknown parameters?

\[ x' = Mx \]

unknowns

point correspondences

• One point correspondence gives how many equations?
• How many point correspondences do we need?
Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
- How do we determine the unknown parameters?

\[ \mathbf{x}' = \mathbf{Mx} \]

unknowns

point correspondences

How do we solve this for \( \mathbf{M} \)?
Least Squares Error

\[ E_{\text{LS}} = \sum_i \| f(x_i; p) - x'_i \|^2 \]
Least Squares Error

\[ E_{\text{LS}} = \sum_{i} \left\| f(x_i; p) - x'_i \right\|^2 \]
Least Squares Error

$$E_{LS} = \sum_{i} \left\| f(x_i; p) - x'_i \right\|^2$$

Euclidean (L2) norm

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Predicted location

Measured location

Squared!
Least Squares Error

\[ E_{LS} = \sum_i \| f(x_i; p) - x'_i \|^2 \]

Residual (projection error)
What is the free variable?
What do we want to optimize?

Least Squares Error

\[ E_{\text{LS}} = \sum_{i} \left\| f(x_i; p) - x'_i \right\|^2 \]
Find parameters that minimize squared error

\[ \hat{p} = \arg \min_p \sum_i \| f(x_i; p) - x'_i \|^2 \]
General form of linear least squares

\[ E_{\text{LLS}} = \sum_i |a_i x - b_i|^2 \]

\[ = \|Ax - b\|^2 \quad \text{(matrix form)} \]
Determining unknown transformations

Affine transformation:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  p_1 & p_2 & p_3 \\
  p_4 & p_5 & p_6
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Why can we drop the last line?

Vectorize transformation parameters:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  x & y & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x & y & 1 \\
  x' & y' & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x & y & 1
\end{bmatrix}
\begin{bmatrix}
  p_1 \\
  p_2 \\
  p_3 \\
  p_4 \\
  p_5 \\
  p_6
\end{bmatrix}
\]

Stack equations from point correspondences:

Notation in system form:

\[\begin{bmatrix}
  b \\
  A \\
  x
\end{bmatrix}\]
General form of linear least squares

(Warning: change of notation. $x$ is a vector of parameters!)

$$E_{LLS} = \sum_{i} |a_i x - b_i|^2$$

$$= \|Ax - b\|^2$$  (matrix form)

This function is quadratic.

*How do you find the root of a quadratic?*
Solving the linear system

Convert the system to a linear least-squares problem:

\[ E_{\text{LLS}} = \|Ax - b\|^2 \]

Expand the error:

\[ E_{\text{LLS}} = x^\top(A^\top A)x - 2x^\top(A^\top b) + \|b\|^2 \]

Minimize the error:

Set derivative to 0 \hspace{1cm} (A^\top A)x = A^\top b

Solve for \( x \) \hspace{1cm} x = (A^\top A)^{-1}A^\top b

In Matlab:

\[ x = A \backslash b \]

Note: You almost never want to compute the inverse of a matrix.
Linear least squares estimation only works when the transform function is linear.
Linear least squares estimation only works when the transform function is linear! (duh)

Also doesn’t deal well with outliers
Determining unknown image warps
Determining unknown image warps

Suppose we have two images.
- How do we compute the transform that takes one to the other?

\[
T(x, y) \
\text{from } f(x, y) \text{ to } g(x', y')
\]
Forward warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?

1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before
3. Send intensities $f(x, y)$ in first image to their corresponding location in the second image

later lecture
Forward warping

Suppose we have two images.

1. How do we compute the transform that takes one to the other?

   1. Form enough pixel-to-pixel correspondences between two images
   2. Solve for linear transform parameters as before
   3. Send intensities $f(x,y)$ in first image to their corresponding location in the second image

   What is the problem with this?
Forward warping

Pixels may end up between two points
- How do we determine the intensity of each point?

\[ f(x,y) \rightarrow T(x,y) \rightarrow g(x',y') \]
Forward warping

Pixels may end up between two points
• How do we determine the intensity of each point?
✓ We distribute color among neighboring pixels \((x',y')\) (“splatting”)

\[ f(x,y) \quad T(x,y) \quad g(x',y') \]

• What if a pixel \((x',y')\) receives intensity from more than one pixels \((x,y)\)?
Forward warping

Pixels may end up between two points
• How do we determine the intensity of each point?
✓ We distribute color among neighboring pixels \((x',y')\) (“splatting”)

• What if a pixel \((x',y')\) receives intensity from more than one pixels \((x,y)\)?
✓ We average their intensity contributions.
Inverse warping

Suppose we have two images.
• How do we compute the transform that takes one to the other?

1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before, then compute its inverse
3. Get intensities \( g(x', y') \) in the second image from point \((x, y) = T^{-1}(x', y')\) in first image

what is the problem with this?
Inverse warping

Pixel may come from between two points
• How do we determine its intensity?

\[ f(x,y) \]
\[ g(x',y') \]
\[ T^{-1}(x,y) \]
Inverse warping

Pixel may come from between two points
• How do we determine its intensity?
✓ Use interpolation

\[
f(x,y)
\quad T^{-1}(x,y)
\quad g(x',y')
\]

\[
f(x,y)
\quad T^{-1}(x,y)
\quad g(x',y')
\]

\[
\begin{align*}
&\text{Pixel may come from between two points} \\
&\quad \text{• How do we determine its intensity?} \\
&\quad \text{✓ Use interpolation}
\end{align*}
\]
Bilinear interpolation

1. Interpolate to find $R_2$
2. Interpolate to find $R_1$
3. Interpolate to find $P$

Grayscale example

In matrix form (with adjusted coordinates)

$$f(x, y) \approx [1 - x \ x] \begin{bmatrix} f(0, 0) & f(0, 1) \\ f(1, 0) & f(1, 1) \end{bmatrix} [1 - y \ y].$$

In Matlab:

call interp2
Suppose we have two images.
- How do we compute the transform that takes one to the other?

Pros and cons of each?
Forward vs inverse warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?

- Inverse warping eliminates holes in target image

- Forward warping does not require existence of inverse transform
References

Basic reading:
• Szeliski textbook, Section 3.6.

Additional reading:
• Hartley and Zisserman, “Multiple View Geometry in Computer Vision,” Cambridge University Press 2004. a comprehensive treatment of all aspects of projective geometry relating to computer vision, and also a very useful reference for the second part of the class.
• Richter-Gebert, “Perspectives on projective geometry,” Springer 2011. a beautiful, thorough, and very accessible mathematics textbook on projective geometry (available online for free from CMU’s library).