CAP 4453
Robot Vision

Dr. Gonzalo Vaca-Castaño
gonzalo.vacacastano@ucf.edu
Credits

• Some slides come directly from these sources:
  • Ioannis (Yannis) Gkioulkas (CMU)
  • Noah Snavely (Cornell)
  • Marco Zuliani
Short Review from last class
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>
</table>
| translation     | \[
\begin{bmatrix}
I & t
\end{bmatrix}_{2 \times 3}
\] | 2        | orientation + ⋯            |      |
| rigid (Euclidean) | \[
\begin{bmatrix}
R & t
\end{bmatrix}_{2 \times 3}
\] | 3        | lengths + ⋯               |      |
| similarity      | \[
\begin{bmatrix}
sR & t
\end{bmatrix}_{2 \times 3}
\] | 4        | angles + ⋯                |      |
| affine          | \[
\begin{bmatrix}
A
\end{bmatrix}_{2 \times 3}
\] | 6        | parallelism + ⋯           |      |
| projective      | \[
\begin{bmatrix}
\tilde{H}
\end{bmatrix}_{3 \times 3}
\] | 8        | straight lines             |      |
Projective transformations (aka homographies)

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
Robot Vision

10. Image warping II
Outline

• Linear algebra
• Image transformations
• 2D transformations.
• Projective geometry 101.
• Transformations in projective geometry.
• Classification of 2D transformations.
• **Determining unknown 2D transformations.**
• Determining unknown image warps.
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?
Simple case: translations

How do we solve for $(x_t, y_t)$?
Simple case: translations

Displacement of match $i = (x'_i - x_i, y'_i - y_i)$

\[
(x_t, y_t) = \left( \frac{1}{n} \sum_{i=1}^{n} x'_i - x_i, \frac{1}{n} \sum_{i=1}^{n} y'_i - y_i \right)
\]
Another view

\[ \begin{align*}
  x_i + x_t &= x_i' \\
  y_i + y_t &= y_i'
\end{align*} \]

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?
Another view

- Problem: more equations than unknowns
  - “Overdetermined” system of equations
  - We will find the least squares solution

\[
\begin{align*}
    x_i + x_t &= x'_i \\
    y_i + y_t &= y'_i
\end{align*}
\]
Least squares formulation

• For each point
  \[(x_i, y_i)\]
  \[x_i + x_t = x_i'\]
  \[y_i + y_t = y_i'\]

• we define the *residuals* as

  \[r_{x_i}(x_t) = (x_i + x_t) - x_i'\]
  \[r_{y_i}(y_t) = (y_i + y_t) - y_i'\]
Least squares formulation

- Goal: minimize sum of squared residuals

\[ C(x_t, y_t) = \sum_{i=1}^{n} (r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2) \]

- “Least squares” solution
- For translations, is equal to mean (average) displacement
Least squares formulation

- Can also write as a matrix equation

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots \\
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
x_t \\
y_t
\end{bmatrix}
= 
\begin{bmatrix}
x'_1 - x_1 \\
y'_1 - y_1 \\
x'_2 - x_2 \\
y'_2 - y_2 \\
\vdots \\
x'_n - x_n \\
y'_n - y_n
\end{bmatrix}
\]

\[
A \quad t = \quad b
\]
Least squares

\[ \mathbf{At} = \mathbf{b} \]

• Find \( t \) that minimizes

\[ \| \mathbf{At} - \mathbf{b} \|^2 \]

• To solve, form the *normal equations*

\[ \mathbf{A}^T \mathbf{At} = \mathbf{A}^T \mathbf{b} \]

\[ t = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \]
Solving the linear system

Convert the system to a linear least-squares problem:

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

Expand the error:

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

Minimize the error:

Set derivative to 0

\[(A^TA)x = A^Tb\]

Solve for \(x\)

\[x = (A^TA)^{-1}A^Tb\]

In Phyton:

```python
import numpy as np
x, resid, rank, s = np.linalg.lstsq(A, b)
```

Note: You almost never want to compute the inverse of a matrix.
Least Squares Error

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

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

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

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

Least Squares Error

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

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

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

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

Important: We will see a different procedure for dealing with homographies!

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?

unknowns
point correspondences
\[ x' = Mx \]

• 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?

How do we solve this for $M$?
Affine transformations

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

- How many unknowns?
- How many equations per match?
- How many matches do we need?
Affine transformations

• Residuals:

\[ r_{x_i}(a, b, c, d, e, f) = (ax_i + by_i + c) - x'_i \]
\[ r_{y_i}(a, b, c, d, e, f) = (dx_i + ey_i + f) - y'_i \]

• Cost function:

\[
C(a, b, c, d, e, f) = \sum_{i=1}^{n} \left( r_{x_i}(a, b, c, d, e, f)^2 + r_{y_i}(a, b, c, d, e, f)^2 \right)
\]
Affine transformations

- Matrix form

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_2 & y_2 & 1 \\
  \vdots \\
  x_n & y_n & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_n & y_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
  e \\
  f \\
\end{bmatrix}
= 
\begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_2 \\
  y'_2 \\
  \vdots \\
  x'_n \\
  y'_n \\
\end{bmatrix}
\]

\[
A_{2n \times 6} \quad t_{6 \times 1} = b_{2n \times 1}
\]
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:**

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

Stack equations from point correspondences:

**Notation in system form:**

\[
b = Ax
\]
General form of linear least squares

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

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

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

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:

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

Minimize the error:

Set derivative to 0

\[ (A^T A)x = A^T b \]

Solve for \( x \)

\[ x = (A^T A)^{-1} A^T b \]

Note: You almost never want to compute the inverse of a matrix.

In Phyton:

```python
import numpy as np
x, resid, rank, s = np.linalg.lstsq(A, b)
```
Linear least squares estimation only works when the transform function is ?
Linear least squares estimation only works when the transform function is **linear! (duh)**

Also doesn’t deal well with outliers (next class !!!)
Homographies

To unwarp (rectify) an image

- solve for homography $H$ given $p$ and $p'$
- solve equations of the form: $wp' = Hp$
  - linear in unknowns: $w$ and coefficients of $H$
  - $H$ is defined up to an arbitrary scale factor
  - how many points are necessary to solve for $H$?
Create point correspondences

Given a set of matched feature points \( \{p_i, p'_i\} \) find the best estimate of \( H \) such that

\[
P' = H \cdot P
\]

How many correspondences do we need?
Determining the homography matrix

Write out linear equation for each correspondence:

\[ P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]
Determining the homography matrix

Write out linear equation for each correspondence:

\[ P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]

Expand matrix multiplication:

\[ x' = \alpha(h_1x + h_2y + h_3) \]
\[ y' = \alpha(h_4x + h_5y + h_6) \]
\[ 1 = \alpha(h_7x + h_8y + h_9) \]
Determining the homography matrix

Write out linear equation for each correspondence:

\[ P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]

Expand matrix multiplication:

\[
\begin{align*}
x' &= \alpha(h_1x + h_2y + h_3) \\
y' &= \alpha(h_4x + h_5y + h_6) \\
1 &= \alpha(h_7x + h_8y + h_9)
\end{align*}
\]

Divide out unknown scale factor:

\[
\begin{align*}
x'(h_7x + h_8y + h_9) &= (h_1x + h_2y + h_3) \\
y'(h_7x + h_8y + h_9) &= (h_4x + h_5y + h_6)
\end{align*}
\]

How do you rearrange terms to make it a linear system?
Just rearrange the terms

\[ x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3) \]
\[ y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6) \]

\[ h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0 \]
\[ h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0 \]
Solving for homographies

\[
x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}
\]
\[
y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}
\]

\[
\begin{bmatrix}
x_i & y_i & 1 & 0 & 0 & 0 & -x'_i & x_i & -x'_iy_i & -x'_i \\
0 & 0 & 0 & x_i & y_i & 1 & -y'_ix_i & -y'_iy_i & -y'_i \\
\end{bmatrix}
\begin{bmatrix}
h_{00} \\
h_{01} \\
h_{02} \\
h_{10} \\
h_{11} \\
h_{12} \\
h_{20} \\
h_{21} \\
h_{22} \\
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]
Determining the homography matrix

Re-arrange terms:

\[ h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0 \]
\[ h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0 \]

Re-write in matrix form:

\[ \mathbf{A}_i \mathbf{h} = 0 \]

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

\[ \mathbf{h} = [ h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6 \ h_7 \ h_8 \ h_9 ]^\top \]
Solving for homographies

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & -x'_1 & x_1 & -x'_1 y_1 & -x'_1 \\
0 & 0 & 0 & 1 & -y'_1 & x_1 & -y'_1 y_1 & -y'_1 \\
\vdots \\
x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n
\end{bmatrix}
\begin{bmatrix}
h_{00} \\
h_{01} \\
h_{02} \\
h_{10} \\
h_{11} \\
h_{12} \\
h_{20} \\
h_{21} \\
h_{22}
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
0
\end{bmatrix}
\]

\[A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 & x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 & x_1 & -y'_1 y_1 & -y'_1 \\ \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}
\quad \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}\]

Defines a least squares problem: minimize \( \|Ah - 0\|^2 \)

- Since \( h \) is only defined up to scale, solve for unit vector \( \hat{h} \)
- Solution: \( \hat{h} = \) eigenvector of \( A^T A \) with smallest eigenvalue
- Works with 4 or more points
Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$Ah = 0$$

$$
\begin{bmatrix}
-x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\
0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y'
\end{bmatrix}
\begin{bmatrix}
-h_1 \\
h_2 \\
h_3 \\
h_4 \\
h_5 \\
h_6 \\
h_7 \\
h_8 \\
h_9
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
$$

*Homogeneous* linear least squares problem
Reminder: Determining affine transformations

Affine transformation:

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

Vectorize transformation parameters:

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

Stack equations from point correspondences:

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

Notation in system form: \[Ax = b\]
Reminder: Determining affine transformations

Convert the system to a linear least-squares problem:

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

Expand the error:

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

Minimize the error:

Set derivative to 0

\[ (A^T A)x = A^T b \]

Solve for \( x \)

\[ x = (A^T A)^{-1} A^T b \]

In Python:

```python
import numpy as np
x, res, rnk, s = np.linalg.lstsq(A, b)
```

Note: You almost never want to compute the inverse of a matrix.
Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$Ah = 0$$

Homogeneous linear least squares problem
• How do we solve this?
Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$Ah = 0$$

$${\begin{bmatrix}
-x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\
0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \\
-x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\
0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \\
\vdots
\end{bmatrix}}$$

$$= \begin{bmatrix}
h_1 \\
h_2 \\
h_3 \\
h_4 \\
h_5 \\
h_6 \\
h_7 \\
h_8 \\
h_9
\end{bmatrix}$$

*Homogeneous* linear least squares problem

- Solve with SVD
Singular Value Decomposition

\[ A = U \Sigma V^T \]

\[ = \sum_{i=1}^{9} \sigma_i u_i v_i^T \]
Singular Value Decomposition

\[ A = U \Sigma V^{-1} \]

\[ \Sigma = \begin{bmatrix}
\sigma_1 \\
\sigma_2 \\
\vdots \\
\sigma_N
\end{bmatrix} \]

U, V = orthogonal matrix

\[ \sigma_i = \sqrt{\lambda_i} \]

\( \sigma \) = singular value
\( \lambda \) = eigenvalue of \( A^t A \)
General form of total least squares

\( E_{\text{TLS}} = \sum_i (a_i x)^2 \)

\( = \| A x \|^2 \) (matrix form)

\( \| x \|^2 = 1 \) constraint

\[
\begin{align*}
\text{minimize} & \quad \| A x \|^2 \\
\text{subject to} & \quad \| x \|^2 = 1
\end{align*}
\]

Solution is the eigenvector corresponding to smallest eigenvalue of \( A^T A \)

Solution is the column of \( V \) corresponding to smallest singular value

\( A = U \Sigma V^T \) (equivalent)

(Warning: change of notation. \( x \) is a vector of parameters!)
Homogeneous Linear Least Squares problem

\[ Ax = 0 \]

\[ A = U \Sigma V^T = \sum_{i=1}^{9} \sigma_i u_i v_i^T \]

- If the homography is *exactly determined*, then \( \sigma_9 = 0 \), and there exists a homography that fits the points exactly.
- If the homography is *overdetermined*, then \( \sigma_9 \geq 0 \). Here \( \sigma_9 \) represents a “residual” or goodness of fit.
- We will not handle the case of the homography being *underdetermined*. 
Solving for H using DLT

Given \( \{x_i, x'_i\} \) solve for H such that \( x' = Hx \)

1. For each correspondence, create 2x9 matrix \( A_i \)

2. Concatenate into single 2n x 9 matrix \( A \)

3. Compute SVD of \( A = U\Sigma V^\top \)

4. Store singular vector of the smallest singular value \( h = v_i \)

5. Reshape to get \( H \)
Recap: Two Common Optimization Problems

<table>
<thead>
<tr>
<th>Problem statement</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimize $|Ax - b|^2$</td>
<td>$x = (A^T A)^{-1} A^T b$</td>
</tr>
<tr>
<td>least squares solution to $Ax = b$</td>
<td></td>
</tr>
</tbody>
</table>

```python
import numpy as np
x, resid, rank, s = np.linalg.lstsq(A, b)
```

<table>
<thead>
<tr>
<th>Problem statement</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimize $x^TA^TAx$ s.t. $x^Tx = 1$</td>
<td>$[v, \lambda] = \text{eig}(A^T A)$</td>
</tr>
<tr>
<td>non-trivial lsq solution to $Ax = 0$</td>
<td>$\lambda_1 &lt; \lambda_{2..n} : x = v_1$</td>
</tr>
</tbody>
</table>
Derivation using Least squares

\[ Ah = 0 \]

The sum squared error can be written as:

\[ f(h) = \frac{1}{2} (Ah - 0)^T (Ah - 0) \]
\[ f(h) = \frac{1}{2} (Ah)^T (Ah) \]
\[ f(h) = \frac{1}{2} h^T A^T A h. \]

Taking the derivative of \( f \) with respect to \( h \) and setting the result to zero,

\[ \frac{d}{dh} f = 0 = \frac{1}{2} (A^T A + (A^T A)^T) h \]
\[ 0 = A^T A h. \]

h should equal the eigenvector of \( B = A^T A \) that has an eigenvalue of zero

\[ B\vec{h} = \lambda \vec{h} \]

(or, in the presence of noise the eigenvalue closest to zero)
Outline

• Linear algebra
• Image transformations
• 2D transformations.
• Projective geometry 101.
• Transformations in projective geometry.
• Classification of 2D transformations.
• Determining unknown 2D transformations.
• 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?

\[ T(x, y) \]

\[ f(x, y) \rightarrow g(x', y') \]
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.
- 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?

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

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

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

- Rotation Scale and Translation Mapping

The mapped points do not have integer coordinates!
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 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?

\[ f(x,y) \quad g(x',y') \]

\[ T^{-1}(x,y) \]
Inverse warping

Pixel may come from between two points

- How do we determine its intensity?

✓ Use interpolation
Inverse warping

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

✓ Use interpolation
- Nearest Neighbors
- Bilinear
- Cubic
- Lanczos

\[ f(x, y) \quad T^{-1}(x, y) \quad g(x’, y’) \]
Nearest Neighbor Interpolation

The question
Let $x$ and $y$ be the integer coordinates of the lattice. What is the value of $f$ at $\begin{bmatrix} p & q \end{bmatrix}^T$?

Nearest Neighbour Answer
$\hat{f}(p, q) = f(\text{round}(p), \text{round}(q))$
Problem with NN interpolation

Point Sampled: Aliasing!  Correctly Bandlimited
Bilinear Interpolation

The question
Let $x$ and $y$ be the integer coordinates of the lattice. What is the value of $f$ at $[p \quad q]^T$?

- $F_{0,0} \overset{\text{def}}{=} f(x, y)$
- $F_{1,0} \overset{\text{def}}{=} f(x + 1, y)$
- $F_{0,1} \overset{\text{def}}{=} f(x, y + 1)$
- $F_{1,1} \overset{\text{def}}{=} f(x + 1, y + 1)$
- $\Delta x \overset{\text{def}}{=} p - x$ and $\Delta y \overset{\text{def}}{=} q - y$
Bilinear Interpolation

The question
Let \( x \) and \( y \) be the integer coordinates of the lattice. What is the value of \( f \) at \([ x, y ]^T \)?

- Linear interpolation in the \( x \) direction:
  \[
  f_y(\Delta x) = (1 - \Delta x)F_{0,0} + \Delta x F_{1,0} \\
  f_{y+1}(\Delta x) = (1 - \Delta x)F_{0,1} + \Delta x F_{1,1}
  \]

- Linear interpolation in the \( y \) direction:
  \[
  \hat{f}(p, q) = (1 - \Delta y)f_y + \Delta y f_{y+1}
  \]
Bilinear Interpolation

The question
Let \( x \) and \( y \) be the integer coordinates of the lattice. What is the value of \( f \) at \( [ p \quad q ]^T \)?

Bilinear Interpolation Answer
Note that \( \hat{f}(p, q) \) “passes through” the samples.

\[
\hat{f}(p, q) = (1 - \Delta y)(1 - \Delta x)F_{0,0} + (1 - \Delta y)\Delta x F_{1,0} + \Delta y(1 - \Delta x)F_{0,1} + \Delta y\Delta x F_{1,1}
\]
Forward vs inverse warping

Suppose we have two images.

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

$T(x, y)$

$T^{-1}(x, y)$

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?

\[ T(x, y) \]
\[ T^{-1}(x, y) \]

• Inverse warping eliminates holes in target image
• Forward warping does not require existence of inverse transform
Warping with different transformations

- translation
- affine
- pProjective (homography)
View warping

original view  synthetic top view  synthetic side view

What are these black areas near the boundaries?
Virtual camera rotations

original view

synthetic rotations
Image rectification

two original images

rectified and stitched
Understanding geometric patterns

What is the pattern on the floor?

magnified view of floor
Understanding geometric patterns

What is the pattern on the floor?

magnified view of floor  rectified view  reconstruction from rectified view
Understanding geometric patterns

Very popular in renaissance drawings (when perspective was discovered)

rectified view of floor

reconstruction
A weird drawing

Holbein, “The Ambassadors”
A weird drawing

Holbein, “The Ambassadors”

What’s this???
A weird drawing
Holbein, “The Ambassadors”
rectified view
skull under anamorphic perspective
A weird drawing

Holbein, “The Ambassadors”

DIY: use a polished spoon to see the skull
Panoramas from image stitching

1. Capture multiple images from different viewpoints.

2. Stitch them together into a virtual wide-angle image.
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).
Questions?