Reading

• Szeliski: Chapter 6.1
Announcements

• Project 2 is out
  – To be done in groups of 2
  – Due March 10

• Project 1 artifact voting
  – Please submit your votes by Wednesday at 11:59pm
Project 2 Demo
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>$\begin{bmatrix} I &amp; t \end{bmatrix}_{2\times3}$</td>
<td>2</td>
<td>orientation + ⋯</td>
<td>□</td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>$\begin{bmatrix} R &amp; t \end{bmatrix}_{2\times3}$</td>
<td>3</td>
<td>lengths + ⋯</td>
<td>◊</td>
</tr>
<tr>
<td>similarity</td>
<td>$\begin{bmatrix} sR &amp; t \end{bmatrix}_{2\times3}$</td>
<td>4</td>
<td>angles + ⋯</td>
<td>◊</td>
</tr>
<tr>
<td>affine</td>
<td>$\begin{bmatrix} A \end{bmatrix}_{2\times3}$</td>
<td>6</td>
<td>parallelism + ⋯</td>
<td>□</td>
</tr>
<tr>
<td>projective</td>
<td>$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$</td>
<td>8</td>
<td>straight lines</td>
<td>□</td>
</tr>
</tbody>
</table>
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

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

• Affine transformations are combinations of ...
  – 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 remain parallel
  – Ratios are preserved
  – Closed under composition
Homographies
Alternate formulation for homographies

\[
\begin{bmatrix}
    x'_i \\
y'_i \\
1
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
\end{bmatrix}
\begin{bmatrix}
x_i \\
y_i \\
1
\end{bmatrix}
\]

where the length of the vector \([h_{00} \ h_{01} \ldots \ h_{22}]\) is 1
Image Warping

- Given a coordinate xform \((x', y') = T(x, y)\) and a source image \(f(x, y)\), how do we compute an xformed image \(g(x', y') = f(T(x, y))\)?
Forward Warping

- Send each pixel \( f(x) \) to its corresponding location \((x',y') = T(x,y)\) in \( g(x',y') \)
- What if pixel lands “between” two pixels?
Forward Warping

• Send each pixel $f(x,y)$ to its corresponding location $x' = h(x,y)$ in $g(x',y')$

• What if pixel lands “between” two pixels?
• Answer: add “contribution” to several pixels, normalize later (splatting)

• Can still result in holes
Inverse Warping

• Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x,y)$ in $f(x,y)$
• Requires taking the inverse of the transform
• What if pixel comes from “between” two pixels?
Inverse Warping

- Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$
- What if pixel comes from “between” two pixels?
- Answer: *resample* color value from *interpolated* (prefiltered) source image
Interpolation

• Possible interpolation filters:
  – nearest neighbor
  – bilinear
  – bicubic (interpolating)
  – sinc

• Needed to prevent “jaggies” and “texture crawl”

  (with prefiltering)
Questions?
Computing transformations

- Given a set of matches between images A and B
  - How can we compute the transform T from A to B?
  - Find transform T that best “agrees” with the matches
Computing transformations
Simple case: translations

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

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

$$(x_t, y_t) = \left( \frac{1}{n} \sum_{i=1}^{n} x'_i - x_i, \frac{1}{n} \sum_{i=1}^{n} y'_i - y_i \right)$$
System of linear equations

- What are the knowns? Unknowns?
- How many unknowns? How many equations (per match)?

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

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

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

• Can also write as a matrix equation

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

\[
A \quad \mathbf{t} = \mathbf{b}
\]

\[2n \times 2 \quad 2 \times 1 \quad 2n \times 1\]
Least squares

\[ A t = b \]

• Find \( t \) that minimizes

\[ \| A t - b \|^2 \]

• To solve, form the *normal equations*

\[ A^T A t = A^T b \]

\[ t = (A^T A)^{-1} A^T b \]
Questions?
Least squares: linear regression

\[ y = mx + b \]

\((y_i, x_i)\)
Linear regression

Cost\((m, b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2\)
Linear regression

\[
\begin{pmatrix}
    x_1 & 1 \\
    x_2 & 1 \\
    \vdots \\
    x_n & 1
\end{pmatrix}
\begin{pmatrix}
    m \\
    b
\end{pmatrix}
=
\begin{pmatrix}
    y_1 \\
    y_2 \\
    \vdots \\
    y_n
\end{pmatrix}
\]
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

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

\[A_{2n \times 6} t_{6 \times 1} = b_{2n \times 1}\]
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 $H$?
Solving for homographies

\[
\begin{bmatrix}
x'_i \\
y'_i \\
1
\end{bmatrix}
= \begin{bmatrix}
h_{00} & h_{01} & h_{02} \\h_{10} & h_{11} & h_{12} \\h_{20} & h_{21} & h_{22}
\end{bmatrix}
\begin{bmatrix}
x_i \\
y_i \\
1
\end{bmatrix}
\]

\[
x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}
\]

\[
y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}
\]

Not linear!

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

\[
y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}
\]
Solving for homographies

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

\[
\begin{bmatrix}
  x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\
  0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \\
\end{bmatrix}
\begin{bmatrix}
  h_{00} \\
  h_{01} \\
  h_{02} \\
  h_{10} \\
  h_{11} \\
  h_{12} \\
  h_{20} \\
  h_{21} \\
  h_{22}
\end{bmatrix}
= \begin{bmatrix}
  0 \\
  0
\end{bmatrix}
\]
Solving for homographies

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

\[
A = \begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\
0 & 0 & 0 & x_n & y_n & 1 & -x'_n x_n & -x'_n y_n & -x'_n \\
0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \\
\end{bmatrix}
\]

\[
h = \begin{bmatrix}
h_{00} \\
h_{01} \\
h_{02} \\
h_{10} \\
h_{11} \\
h_{12} \\
h_{20} \\
h_{21} \\
h_{22} \\
\end{bmatrix}
\]

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

- 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
## Recap: Two Common Optimization Problems

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

| Problem 2                                                                 |          |
| minimize \( x^T A^T A x \) s.t. \( x^T x = 1 \)                              |          |
| non-trivial lsq solution to \( Ax = 0 \)                                       |          |
| **Solution 2**                                                                 |
| \( [v, \lambda] = \text{eig}(A^T A) \)                                       |          |
| \( \lambda_1 < \lambda_{2..n} : x = v_1 \)                                   |          |
Questions?
Image Alignment Algorithm

Given images A and B

1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

What could go wrong?
Outliers

outliers

inliers
Robustness

• Let’s consider a simpler example... linear regression

Problem: Fit a line to these datapoints

• How can we fix this?

Least squares fit
We need a better cost function...

• Suggestions?
Idea

• Given a hypothesized line
• Count the number of points that “agree” with the line
  – “Agree” = within a small distance of the line
  – I.e., the inliers to that line

• For all possible lines, select the one with the largest number of inliers
Counting inliers
Counting inliers

Inliers: 3
Counting inliers

Inliers: 20
How do we find the best line?

• Unlike least-squares, no simple closed-form solution

• Hypothesize-and-test
  – Try out many lines, keep the best one
  – Which lines?
Translations
RAndom SAmple Consensus

Select *one* match at random, count *inliers*
Select another match at random, count *inliers*
RAndom SAmple Consensus

Output the translation with the highest number of inliers
RANSAC

• Idea:
  – All the inliers will agree with each other on the translation vector; the (hopefuly small) number of outliers will (hopefuly) disagree with each other
  • RANSAC only has guarantees if there are < 50% outliers

  – “All good matches are alike; every bad match is bad in its own way.”
    – Tolstoy via Alyosha Efros
RANSAC

• **Inlier threshold** related to the amount of noise we expect in inliers
  – Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)

• **Number of rounds** related to the percentage of outliers we expect, and the probability of success we’d like to guarantee
  – Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
  – How many rounds do we need?
set threshold so that, e.g., 95% of the Gaussian lies inside that radius
RANSAC

- Back to linear regression
- How do we generate a hypothesis?
• Back to linear regression
• How do we generate a hypothesis?
RANSAC

• General version:
  1. Randomly choose $s$ samples
     • Typically $s =$ minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat $N$ times
  5. Choose the model that has the largest set of inliers
How many rounds?

• If we have to choose \( s \) samples each time
  – with an outlier ratio \( e \)
  – and we want the right answer with probability \( p \)

<table>
<thead>
<tr>
<th>( s )</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>19</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>34</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>17</td>
<td>26</td>
<td>57</td>
<td>146</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>7</td>
<td>16</td>
<td>24</td>
<td>37</td>
<td>97</td>
<td>293</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>8</td>
<td>20</td>
<td>33</td>
<td>54</td>
<td>163</td>
<td>588</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>26</td>
<td>44</td>
<td>78</td>
<td>272</td>
<td>1177</td>
</tr>
</tbody>
</table>

\( p = 0.99 \)

Source: M. Pollefeys
How big is $s$?

- For alignment, depends on the motion model
  - Here, each sample is a correspondence (pair of matching points)

<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>$[I \ t]_{2\times3}$</td>
<td>2</td>
<td>orientation + ⋯</td>
<td></td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>$[R \ t]_{2\times3}$</td>
<td>3</td>
<td>lengths + ⋯</td>
<td></td>
</tr>
<tr>
<td>similarity</td>
<td>$[sR \ t]_{2\times3}$</td>
<td>4</td>
<td>angles + ⋯</td>
<td></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>$[\vec{H}]_{3\times3}$</td>
<td>8</td>
<td>straight lines</td>
<td></td>
</tr>
</tbody>
</table>
RANSAC pros and cons

• Pros
  – Simple and general
  – Applicable to many different problems
  – Often works well in practice

• Cons
  – Parameters to tune
  – Sometimes too many iterations are required
  – Can fail for extremely low inlier ratios
  – We can often do better than brute-force sampling
Final step: least squares fit

Find average translation vector over all inliers
RANSAC

• An example of a “voting”-based fitting scheme
• Each hypothesis gets voted on by each data point, best hypothesis wins

• There are many other types of voting schemes
  – E.g., Hough transforms...
Panoramas

• Now we know how to create panoramas!

• Given two images:
  – Step 1: Detect features
  – Step 2: Match features
  – Step 3: Compute a homography using RANSAC
  – Step 4: Combine the images together (somehow)

• What if we have more than two images?
Can we use homographies to create a 360 panorama?

- In order to figure this out, we need to learn what a camera is
360 panorama
Questions?