RANSAC: Fitting Transforms with Outliers
Reading

• Szeliski: Chapter 6.1
Goals

• Understand the Random Sample Consensus (RANSAC) algorithm.
• Be prepared to implement RANSAC for fitting image coordinate transforms using matches that may contain outliers.
Announcements
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
Fitting a Homography: TL;DM

- For each feature match \((x_i, y_i) \rightarrow (x'_i, y'_i)\), fill in 2 rows of \(A\):

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

• For each feature match \((x_i, y_i) \rightarrow (x_i', y_i')\), fill in 2 rows of A

• Solve the homogeneous least squares problem
  \[
  \min_h ||Ah||^2:
  \]
  – Take the SVD of A to get U, S, and V.
  – Let \(h\) be the right singular vector of A whose singular value is smallest.
  – Let \(h\) be the column of V (row of \(V^T\)) whose column index is the same as that of the smallest diagonal entry of S.
Solving for homographies

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

\[
\begin{bmatrix}
  h_{00} \\
  h_{01} \\
  h_{02} \\
  h_{10} \\
  h_{11} \\
  h_{12} \\
  h_{20} \\
  h_{21} \\
  h_{22}
\end{bmatrix} \text{ is only defined up to scale, solve for unit vector } \hat{h}
\]

\[
\begin{bmatrix}
  A \\
  0
\end{bmatrix}
\]

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
## Recap: Two Common Optimization Problems

<table>
<thead>
<tr>
<th>Problem statement</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{minimize} \quad |Ax - b|^2 ] least squares solution to ( Ax = b )</td>
<td>[ x = \left( A^T A \right)^{-1} A^T b ] np.linalg.lstsq(A, b)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem statement</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{minimize} \quad x^T A^T Ax \quad \text{s.t.} \quad x^T x = 1 ] non-trivial lsq solution to ( Ax = 0 )</td>
<td>[ U, \Sigma, V = \text{svd}(A) ] [ x \leftarrow v_{\arg \min_i \Sigma[i,i]} ] [ U, s, V = \text{np.linalg.svd}(A) ]</td>
</tr>
</tbody>
</table>
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?
Code: fitting affine transformations

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

\[
A \quad 2n \times 6 \\
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_2 & y_2 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  x_n & y_n & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_n & y_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
  e \\
  f \\
\end{bmatrix}
= \begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_2 \\
  y'_2 \\
  \vdots \\
  x'_n \\
  y'_n \\
\end{bmatrix}
\]

\[
t \quad 6 \times 1 \\
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_2 & y_2 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  x_n & y_n & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_n & y_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
  e \\
  f \\
\end{bmatrix}
= \begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_2 \\
  y'_2 \\
  \vdots \\
  x'_n \\
  y'_n \\
\end{bmatrix}
\]

\[
b \quad 2n \times 1 \\
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_2 & y_2 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  x_n & y_n & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_n & y_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
  e \\
  f \\
\end{bmatrix}
= \begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_2 \\
  y'_2 \\
  \vdots \\
  x'_n \\
  y'_n \\
\end{bmatrix}
\]
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? Which one is the “best”? 
Translations
RAndom SAmple Consensus

Select *one* match at random, count *inliers*
Random Sample Consensus

Select another match at random, count inliers
RAndon 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 (hopefully small) number of outliers will (hopefully) 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?
• **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?
RANSAC

Set threshold so that, e.g., 95% of the Gaussian lies inside that radius.
• Back to linear regression
• How do we generate a hypothesis?
RANSAC

• 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>$\begin{bmatrix} I &amp; t \end{bmatrix}_{2\times3}$</td>
<td>2</td>
<td>orientation $+$ $\cdots$</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 $+$ $\cdots$</td>
<td></td>
</tr>
<tr>
<td>similarity</td>
<td>$\begin{bmatrix} sR &amp; t \end{bmatrix}_{2\times3}$</td>
<td>4</td>
<td>angles $+$ $\cdots$</td>
<td></td>
</tr>
<tr>
<td>affine</td>
<td>$A_{2\times3}$</td>
<td>6</td>
<td>parallelism $+$ $\cdots$</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>
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?