CS5760: Computer Vision
RANSAC

Reading

• Szeliski (2nd edition): Chapter 8.1
Announcements

• Project 2 due Thursday, February 22, by 8pm on GitHub
  – Report due Thursday, Feb 22 by 8pm on CMSX
• Take-home midterm to be released after February Break
  – To be distributed in class at 2:40pm Thursday, Feb 29
  – Due Tuesday, March 5 by 1:25pm (beginning of class)
  – Open book, open note (but no Google or ChatGPT)
  – To be done on your own
• No class on Tuesday (February Break)
Recap: 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?
Today: we tie up any loose ends and finish image alignment
Outliers
Robustness

• Let’s consider the problem of linear regression

Problem: Fit a line to these datapoints

Least squares fit

• How can we fix this?
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
RAndon 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 w/ 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 at least 99% probability
  – How many rounds do we need?
Suppose we are running RANSAC to compute a translation on a problem with 20% outliers, and we want to find the correct answer with 99% probability. How many rounds do we need?
\[ 0.2^n \leq 1 - 0.99 = 0.01 \]

\[
N \log 0.2 \leq \log 0.01
\]

\[
N \geq \frac{\log 0.01}{\log 0.2} \geq 2.86
\]
RANSAC: Another view

- 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?
RANSAC

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

• General version:
  1. Randomly choose $s$ samples
     • Typically $s = \text{minimum sample size that lets you fit a model}$
  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 $p$

\[ N \geq \frac{\log(1 - p)}{\log(1 - (1 - e)^s)} \]

<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)
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?

• To figure this out, we need to know what a camera is
360 panorama
Questions?