Model Fitting

Computer Vision
CS 143, Brown

James Hays

Slides from Silvio Savarese, Svetlana Lazebnik, and Derek Hoiem
Fitting: find the parameters of a model that best fit the data

Alignment: find the parameters of the transformation that best align matched points
Example: Computing vanishing points
Example: Estimating an homographic transformation
Example: Estimating “fundamental matrix” that corresponds two views
Example: fitting an 2D shape template
Example: fitting a 3D object model
Critical issues: noisy data
Critical issues: intra-class variability

“All models are wrong, but some are useful.” Box and Draper 1979
Critical issues: outliers
Critical issues: missing data (occlusions)
Fitting and Alignment

• Design challenges
  – Design a suitable **goodness of fit** measure
    • Similarity should reflect application goals
    • Encode robustness to outliers and noise
  – Design an **optimization** method
    • Avoid local optima
    • Find best parameters quickly
Fitting and Alignment: Methods

• Global optimization / Search for parameters
  – Least squares fit
  – Robust least squares
  – Iterative closest point (ICP)

• Hypothesize and test
  – Generalized Hough transform
  – RANSAC
Simple example: Fitting a line
Least squares line fitting

• Data: \((x_1, y_1), \ldots, (x_n, y_n)\)
• Line equation: \(y_i = mx_i + b\)
• Find \((m, b)\) to minimize

\[
E = \sum_{i=1}^{n} (y_i - mx_i - b)^2
\]

\[
E = \sum_{i=1}^{n} \left( \begin{bmatrix} 1 & \begin{bmatrix} m \\ b \end{bmatrix} \end{bmatrix} - y_i \right)^2 = \left[ \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right]^2 = \|Ap - y\|^2
\]

\[
\]

\[
\frac{dE}{dB} = 2A^T Ap - 2A^T y = 0
\]

\[
A^T Ap = A^T y \implies p = \left( A^T A \right)^{-1} A^T y
\]

Matlab: \(p = A \setminus y;\)

Modified from S. Lazebnik
Least squares: Robustness to noise

Least squares fit to the red points:
Least squares: Robustness to noise

Least squares fit with an outlier:

Problem: squared error heavily penalizes outliers
Search / Least squares conclusions

Good
• Clearly specified objective
• Optimization is easy (for least squares)

Bad
• Not appropriate for non-convex objectives
  – May get stuck in local minima
• Sensitive to outliers
  – Bad matches, extra points
• Doesn’t allow you to get multiple good fits
  – Detecting multiple objects, lines, etc.
Robust least squares (to deal with outliers)

General approach:
minimize

$$\sum_i \rho(u_i(x_i, \theta; \sigma))$$

$$u = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$u_i(x_i, \theta)$$ – residual of i\(^{th}\) point w.r.t. model parameters \(\theta\)
\(\rho\) – robust function with scale parameter \(\sigma\)

The robust function \(\rho\)

- Favors a configuration with small residuals
- Constant penalty for large residuals
Choosing the scale: Just right

The effect of the outlier is minimized
The error value is almost the same for every point and the fit is very poor.
Choosing the scale: Too large

Behaves much the same as least squares
Robust fitting is a nonlinear optimization problem that must be solved iteratively.

Least squares solution can be used for initialization.

Adaptive choice of scale: approx. 1.5 times median residual (F&P, Sec. 15.5.1)
Hypothesize and test

1. Propose parameters
   - Try all possible
   - Each point votes for all consistent parameters
   - Repeatedly sample enough points to solve for parameters

2. Score the given parameters
   - Number of consistent points, possibly weighted by distance

3. Choose from among the set of parameters
   - Global or local maximum of scores

4. Possibly refine parameters using inliers
Hough transform


Given a set of points, find the curve or line that explains the data points best

\[ y = m x + b \]

Hough space

Slide from S. Savarese
Hough transform
Hough transform


Issue: parameter space \([m,b]\) is unbounded…

Use a polar representation for the parameter space

\[ x \cos \theta + y \sin \theta = \rho \]
Hough transform - experiments

Noisy data

Issue: Grid size needs to be adjusted…
Generalized Hough transform

• We want to find a template defined by its reference point (center) and several distinct types of landmark points in stable spatial configuration
Generalized Hough transform

- Template representation: for each type of landmark point, store all possible displacement vectors towards the center.
Generalized Hough transform

- Detecting the template:
  - For each feature in a new image, look up that feature type in the model and vote for the possible center locations associated with that type in the model.
Application in recognition

- Index displacements by “visual codeword”

B. Leibe, A. Leonardis, and B. Schiele, *Combined Object Categorization and Segmentation with an Implicit Shape Model*, ECCV Workshop on Statistical Learning in Computer Vision 2004
Application in recognition

• Index displacements by “visual codeword”

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004
Hough transform conclusions

Good
• Robust to outliers: each point votes separately
• Fairly efficient (often faster than trying all sets of parameters)
• Provides multiple good fits

Bad
• Some sensitivity to noise
• Bin size trades off between noise tolerance, precision, and speed/memory
  – Can be hard to find sweet spot
• Not suitable for more than a few parameters
  – Grid size grows exponentially

Common applications
• Line fitting (also circles, ellipses, etc.)
• Object instance recognition (parameters are affine transform)
• Object category recognition (parameters are position/scale)
RANSAC  
(RANDOM SAMPLE CONSENSUS) 

Fischler & Bolles in ‘81.

**Algorithm:**
1. **Sample** (randomly) the number of points required to fit the model 
2. **Solve** for model parameters using samples 
3. **Score** by the fraction of inliers within a preset threshold of the model 

**Repeat** 1-3 until the best model is found with high confidence
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

**Repeat** 1-3 until the best model is found with high confidence
Algorithm:
1. **Sample** (randomly) the number of points required to fit the model (\#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

**Repeat** 1-3 until the best model is found with high confidence
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

**Repeat** 1-3 until the best model is found with high confidence
Choosing the parameters

- **Initial number of points** $s$
  - Typically minimum number needed to fit the model

- **Distance threshold** $t$
  - Choose $t$ so probability for inlier is $p$ (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. $\sigma$: $t^2 = 3.84\sigma^2$

- **Number of samples** $N$
  - Choose $N$ so that, with probability $p$, at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: $e$)

\[
\begin{align*}
\left\lceil \frac{-\log(1-p)}{\log(1-e)} \right\rceil &= 1-p \\
N &= \log \left( \frac{1-p}{\log(1-e)} \right)
\end{align*}
\]

<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>

Source: M. Pollefeys
RANSAC conclusions

Good
• Robust to outliers
• Applicable for larger number of parameters than Hough transform
• Parameters are easier to choose than Hough transform

Bad
• Computational time grows quickly with fraction of outliers and number of parameters
• Not good for getting multiple fits

Common applications
• Computing a homography (e.g., image stitching)
• Estimating fundamental matrix (relating two views)
What if you want to align but have no prior matched pairs?

- Hough transform and RANSAC not applicable

- Important applications

Medical imaging: match brain scans or contours

Robotics: match point clouds
Iterative Closest Points (ICP) Algorithm

Goal: estimate transform between two dense sets of points

1. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
2. **Estimate** transformation parameters
   - e.g., least squares or robust least squares
3. **Transform** the points in {Set 1} using estimated parameters
4. **Repeat** steps 1-3 until change is very small