Today

• **Fitting** models (lines) to points, i.e. find the parameters of a model that best fits the data
  – Least squares
  – Hough transform
  – RANSAC

• **Matching** = finding correspondences between points, i.e. find the parameters of the transformation that best aligns points

• Homework 2 is due 10/08
Fitting

- Want to associate a model with observed features

For example, the model could be a line, a circle, or an arbitrary shape.
Example: Line fitting

• Why fit lines?
  Many objects characterized by presence of straight lines

• Why aren’t we done just by running edge detection?
Difficulty of line fitting

- Extra edge points (clutter), multiple models:
  - which points go with which line, if any?

- Only some parts of each line detected, and some parts are missing:
  - how to find a line that bridges missing evidence?

- Noise in measured edge points, orientations:
  - how to detect true underlying parameters?
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

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

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

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

Modified from Svetlana Lazebnik
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
Voting

• It’s not feasible to check all combinations of features by fitting a model to each possible subset.

• Voting is a general technique where we let the features vote for all models that are compatible with it.
  – Cycle through features, cast votes for model parameters.
  – Look for model parameters that receive a lot of votes.

• Noise & clutter features?
  – They will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.
Fitting lines: Hough transform

• Given points that belong to a line, what is the line?
• How many lines are there?
• Which points belong to which lines?

• **Hough Transform** is a voting technique that can be used to answer all of these questions.

  **Main idea:**
  1. Record vote for each possible line on which each edge point lies.
  2. Look for lines that get many votes.
Finding lines in an image: Hough space

Connection between image \((x,y)\) and Hough \((m,b)\) spaces

- A line in the image corresponds to a point in Hough space

\[ y = m_0 x + b_0 \]
Finding lines in an image: Hough space

Connection between image \((x,y)\) and Hough \((m,b)\) spaces

- A line in the image corresponds to a point in Hough space.
- What does a point \((x_0, y_0)\) in the image space map to?
  - Answer: the solutions of \(b = -x_0 m + y_0\)
  - This is a line in Hough space.

- To go from image space to Hough space:
  - given a set of points \((x,y)\), find all \((m,b)\) such that \(y = mx + b\)
Finding lines in an image: Hough space

What are the line parameters for the line that contains both $(x_0, y_0)$ and $(x_1, y_1)$?

- It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$
Finding lines in an image: Hough algorithm

How can we use this to find the most likely parameters \((m, b)\) for the most prominent line in the image space?

- Let each edge point in image space *vote* for a set of possible parameters in Hough space.
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.
Hough transform

Silvio Savarese
Parameter space representation

- Problems with the (m,b) space:
  - Unbounded parameter domains
  - Vertical lines require infinite m
Problems with the (m,b) space:

- Unbounded parameter domains
- Vertical lines require infinite m

Alternative: polar representation

Each point (x,y) will add a sinusoid in the (θ,ρ) parameter space

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


Use a polar representation for the parameter space

\[ x \cos \theta + y \sin \theta = \rho \]
Algorithm outline

- Initialize accumulator H to all zeros
- For each feature point \((x,y)\) in the image
  - For \(\theta = 0\) to \(180\)
    - \(\rho = x \cos \theta + y \sin \theta\)
    - \(H(\theta, \rho) = H(\theta, \rho) + 1\)
  - end
- Find the value(s) of \((\theta^*, \rho^*)\) where \(H(\theta, \rho)\) is a local maximum
- The detected line in the image is given by \(\rho^* = x \cos \theta^* + y \sin \theta^*\)

H: accumulator array (votes)

Svetlana Lazebnik
Hough transform example
Impact of noise on Hough

Image space edge coordinates

Votes

Silvio Savarese
Impact of noise on Hough

Image space
dge coordinates

Votes

What difficulty does this present for an implementation?
Impact of noise on Hough

Noisy data

Need to adjust grid size or smooth
Impact of noise on Hough

Here, everything appears to be “noise”, or random edge points, but we still see peaks in the vote space.
Algorithm outline

- Initialize accumulator $H$ to all zeros
- For each feature point $(x, y)$ in the image
  
  For $\theta = 0$ to $180$
  
  $\rho = x \cos \theta + y \sin \theta$
  
  $H(\theta, \rho) = H(\theta, \rho) + 1$

- Find the value(s) of $(\theta, \rho)$ where $H(\theta, \rho)$ is a local maximum
- The detected line in the image is given by $\rho = x \cos \theta + y \sin \theta$
Incorporating image gradients

• Recall: when we detect an edge point, we also know its gradient direction
• But this means that the line is uniquely determined!

• Modified Hough transform:

For each edge point \((x, y)\)

\[
\theta = \text{gradient orientation at } (x, y)
\]

\[
\rho = x \cos \theta + y \sin \theta
\]

\[
H(\theta, \rho) = H(\theta, \rho) + 1
\]

end
Hough transform for circles

- Circle: center \((a,b)\) and radius \(r\)
  \[
  (x_i - a)^2 + (y_i - b)^2 = r^2
  \]

- For a fixed radius \(r\), unknown gradient direction
Hough transform for circles

- Circle: center (a,b) and radius r

\[(x_i - a)^2 + (y_i - b)^2 = r^2\]

- For a fixed radius r, unknown gradient direction

Image space

Hough space

Intersection: most votes for center occur here.
Hough transform for circles

- Circle: center \((a,b)\) and radius \(r\)

\[
(x_i - a)^2 + (y_i - b)^2 = r^2
\]

- For an unknown radius \(r\), unknown gradient direction
Hough transform for circles

- Circle: center \((a, b)\) and radius \(r\)

\[
(x_i - a)^2 + (y_i - b)^2 = r^2
\]

- For an unknown radius \(r\), unknown gradient direction
Hough transform for circles

For every edge pixel \((x,y)\):
  For all \(a\):
    For all \(b\):
      \[r = \sqrt{(x - a)^2 + (y - b)^2}\]
      \[H[a,b,r] += 1\]
    end
  end
end
Hough transform for circles

- Circle: center \((a,b)\) and radius \(r\)
  \[
  (x_i - a)^2 + (y_i - b)^2 = r^2
  \]
- For an unknown radius \(r\), \textbf{known} gradient direction
Hough transform for circles

• A circle with radius $r$ and center $(a, b)$ can be described as:

\[ x = a + r \cos(\theta) \]
\[ y = b + r \sin(\theta) \]
Hough transform for circles

For every edge pixel \((x,y)\) :

For each possible radius value \(r\):

For each possible gradient direction \(\theta\):

// or use estimated gradient at \((x,y)\)

\[
\begin{align*}
a &= x - r \cos(\theta) \quad \text{// column} \\
b &= y - r \sin(\theta) \quad \text{// row}
\end{align*}
\]

\[H[a,b,r] \quad \text{+=} \quad 1\]

\[
x = a + r \cos(\theta) \\
y = b + r \sin(\theta)
\]
Example: detecting circles with Hough

Original

Edges

Votes: Penny

Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Kristen Grauman, images from Vivek Kwatra
Example: detecting circles with Hough

Original detections

Edges

Votes: Quarter

Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).
Example: iris detection

- Hemerson Pistori and Eduardo Rocha Costa
Voting: practical tips

• Minimize irrelevant tokens first

• Choose a good grid / discretization
  - **Too coarse**: large votes obtained when too many different lines correspond to a single bucket
  - **Too fine**: miss lines because points that are not exactly collinear cast votes for different buckets

• Vote for neighbors, also (smoothing in accumulator array)

• Use direction of edge to reduce parameters by 1

• To read back which points voted for “winning” peaks, keep tags on the votes
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

• What if we want to detect arbitrary shapes?

Intuition:

Now suppose those colors encode gradient directions…
Generalized Hough transform

Define a model shape by its boundary points and a reference point.

**Offline procedure:**

At each boundary point, compute displacement vector: \( \mathbf{r} = \mathbf{a} - \mathbf{p}_i \).

Store these vectors in a table indexed by gradient orientation \( \theta \).
Detection procedure:

For each edge point:
• Use its gradient orientation \( \theta \) to index into stored table
• Use retrieved \( r \) vectors to vote for reference point

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.
Generalized Hough transform

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

**Template**

**Model**
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.

Test image

Model
Generalized Hough for object detection

- 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

Svetlana Lazebnik
Generalized Hough for object detection

- 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

Svetlana Lazebnik
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering (more on this later in the course)
Generalized Hough transform

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

**Template**

**Model**
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry
3. For each codebook entry, store all positions it was found, relative to object center
Hough transform: pros and cons

**Pros**

- All points are processed independently, so can cope with occlusion, gaps
- Some robustness to noise: noise points unlikely to contribute *consistently* to any single bin
- Can detect multiple instances of a model in a single pass

**Cons**

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: can be tricky to pick a good grid size

Kristen Grauman
Today

• **Fitting** models (lines) to points, i.e. *find the parameters of a model that best fits the data*
  – Least squares
  – Hough transform
  – RANSAC

• **Matching** = finding correspondences between points, i.e. *find the parameters of the transformation that best aligns points*

• Homework 2 is due 10/08
Outliers

- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn’t belong to the line we are fitting.
Outliers affect least squares fit
Outliers affect least squares fit
RANSAC

• RANdom Sample Consensus

• **Approach:** we want to avoid the impact of outliers, so let’s look for “inliers”, and use those only.

• **Intuition:** if an outlier is chosen to compute the current fit, then the resulting line won’t have much support from rest of the points.
RANSAC: General form

• **RANSAC loop:**

1. Randomly select a *seed group* of *s* points on which to base model estimate

2. Fit model to these *s* points

3. Find *inliers* to this model (i.e., points whose distance from the line is less than *t*)

4. If there are *d* or more inliers, re-compute estimate of model on all of the inliers

5. Repeat *N* times

• Keep the model with the largest number of inliers

Modified from Kristen Grauman and Svetlana Lazebnik
RANSAC for line fitting example

Source: R. Raguram
RANSAC for line fitting example

Source: R. Raguram
1. Randomly select minimal subset of points
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

RANSAC for line fitting example

Uncontaminated sample

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
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
How to choose parameters?

• 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: $e$)

• Number of sampled points $s$
  – Minimum number needed to fit the model

• Distance threshold $\delta$
  – Choose $\delta$ so that a good point with noise is likely (e.g., prob=0.95) within threshold

$$N = \log(1-p) / \log(1-(1-e)^s)$$

Explanation in Szeliski 6.1.4

<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>4</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>
RANSAC pros and cons

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

• Cons
  • Lots of parameters to tune
  • Doesn’t work well for low inlier ratios (too many iterations, or can fail completely)
  • Can’t always get a good initialization of the model based on the minimum number of samples

• Common applications
  • Image stitching
  • Relating two views