Computer vision – Edge detection

Elad Osherov Nov 2015 Slides credit Lihi Zelnik-Manor
**Problem:**

In automated analysis of digital images, a sub-problem often arises of detecting simple shapes, such as straight lines, circles or ellipses.
Primitives detection

Solution?

Edge detection
Why edges?

- We know edges are special
What can cause an edge

- Surface normal discontinuity
- Depth discontinuity
- Illumination discontinuity
- Surface color discontinuity
What can cause an edge

- Reflectance change: appearance information, texture
- Change in surface orientation: shape
- Depth discontinuity: object boundary
- Cast shadows

Applications and Algorithms in CV

Tutorial 3: Edge detection
Finding straight lines

• An edge detector can be used as a pre-processing stage to obtain points (pixels) that are on the desired curve in the image space.

• Due to imperfections in either the image data or the edge detector, there may be missing points on the desired curves as well as spatial deviations between the ideal line/circle/ellipse and the noisy edge points as they are obtained from the edge detector.

• It is often non-trivial to group the extracted edge features to an appropriate set of lines, circles or ellipses.
Fitting in a parametric space

Goal:
- Choose a parametric model to represent a set of features
- Possibly multiple models are required

Main questions:
- What model represents this set of features best?
- How many models are there?
- Which features belong to each model?

Computational complexity:
- Typically we cannot examine all possible models
Difficulty of line fitting in images

- Which points are on which line?
- How many lines?
- Noisy edge detection:
  - Clutter
  - Missed parts
  - Points are only approximately along the line
Grouping & Fitting methods

2 approaches for grouping & fitting

1. Global optimization / Search for parameters
   - Least squares fit
   - Total least squares fit
   - Robust least squares
   - Iterative closest point (ICP)
   - Etc.

2. Hypothesize and test
   - Hough transform
   - Generalized Hough transform
   - RANSAC
   - Etc.
Grouping & Fitting methods

1. Global optimization / Search for parameters
   - Least squares fit
Least square line fitting

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

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

\[
\frac{dE}{dP} = 2A^TAp - 2A^Ty = 0
\]

\[
A^TAp = A^Ty \Rightarrow p = \left(A^TA\right)^{-1}A^Ty
\]

Matlab: \(p = A \setminus y;\)
Least square line fitting

**Question**: Will least squares work in this case?
- Fails completely for vertical lines

**Question**: is LSQ invariant to rotation?
- No. same edge, different parameters

![Least square line fitting diagram](image)
Grouping & Fitting methods

2 approaches for grouping & fitting

1. Global optimization / Search for parameters

   Total least squares fit
Total least square

Distance between a point \((x_i, y_i)\) and a line \(ax+by+c=0\):

\[
d = \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}}
\]

If \((a^2+b^2)=1\) then: \(d = |ax_i + by_i + c|

Unit normal: 
\(N=(a, b)\)
Distance between a point \((x_i, y_i)\) and a line \(ax+by+c=0\):

\[
d = \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}}
\]

If \((a^2+b^2=1)\) then: \(d = |ax_i + by_i + c|\)

Find \((a, b, c)\) to minimize the sum of squared perpendicular distances

\[
E = \sum_{i=1}^{n} (ax_i + by_i + c)^2
\]
Total least square

\[
\frac{\partial E}{\partial c} = \sum_{i=1}^{n} -2(ax_i + by_i + c) = 0 \quad \Rightarrow \quad c = -\frac{a}{n} \sum_{i=1}^{n} x_i - \frac{b}{n} \sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}
\]

\[
E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}
\]

minimize \( \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \) \quad s.t. \( \mathbf{p}^T \mathbf{p} = 1 \) \quad \Rightarrow \quad minimize \( \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}} \)

*Solution is eigenvector corresponding to smallest eigenvalue of \( \mathbf{A}^T \mathbf{A} \)
Recap: Two Common Optimization Problems

Problem statement

1. minimize $\|Ax - b\|^2$
   - least squares solution to $Ax = b$

Solution

1. $x = (A^T A)^{-1} A^T b$
2. $x = A \backslash b \ (\text{matlab})$

Problem statement

1. minimize $x^T A^T A x$ s.t. $x^T x = 1$
2. minimize $\frac{x^T A^T A x}{x^T x}$

Solution

1. $[v, \lambda] = \text{eig}(A^T A)$
2. $\lambda_1 < \lambda_{2..n} : x = v_1$
Least square line fitting

Least squares fit to the red points
Least square line fitting:: robustness to noise

Least squares fit with an outlier

Squared error heavily penalizes outliers
Least square line fitting:: 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.
Grouping & Fitting methods

2 approaches for grouping & fitting

2. Hypothesize and test

*Hough transform*
The purpose of the Hough transform is to address the problem of primitives detection by performing an explicit **voting** procedure over a set of parameterized image objects.
Voting

• Problem:
  • We cannot try all possible models.

• Solution by voting:
  • Features (points) vote for the model they are compatible with.
  • Search for models with lots of votes.

• Key ideas:
  • Noise and clutter votes are inconsistent, so will not harm model selection.
  • Ok if not all points are present as long as model gets enough votes.
Hough transform:: Detecting straight lines

A line \( y = mx + b \) can be described as the set of all points comprising it: \((x_1, y_1), (x_2, y_2)\)...

Instead, a straight line can be represented as a point \((b, m)\) in the parameter space.

**Question:** does \( \{b,m\} \) (slope and intercept) indeed the best parameter space?

**Answer:** No! \( m \) can get infinite values if the line is vertical.
The Hough Transform

- Transformation from image space \((x, y)\) to Hough space \((m, b)\)
- A line in the image corresponds to a point in Hough space
  - Image \(\rightarrow\) Hough:
    Given a set of points \((x, y)\) find all \((m, b)\) such that \(y = mx + b\)
  - Hough \(\rightarrow\) Image:
    A point \((x_0, y_0)\) in the image is the solution of \(b = -x_0m + y_0\)
- The line that contain both points \((x_0, y_0)\) and \((x_1, y_1)\) is the intersection of the lines \(b = -x_0m + y_0\) and \(b = -x_1m + y_1\)
Hough transform:: Line parameterization

Hough transform is calculated in polar coordinates

The parameter $r$ represents the distance between the line and the origin

$\theta$ is the angle of the vector from the origin to this closest point
**Hough transform:** Line parameterization

Using this parameterization, the equation of the line can be written as:

\[
y = \left(-\frac{\cos \theta}{\sin \theta}\right)x + \left(\frac{r}{\sin \theta}\right)
\]

which can be rearranged to:  
\[
r = x \cos \theta + y \sin \theta
\]
**Question**: Is there a straight line connecting the 3 dots?
**Hough Solution:**

1. For each data point, a number of lines are plotted going through it, all at different angles ($\theta$) (shown as solid lines).

2. Calculate $\tau$ For each solid line from step 1 (shown as dashed lines).

3. For each data point, organize the results in a table.

### Example

<table>
<thead>
<tr>
<th>Angle</th>
<th>Dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>30</td>
<td>69.6</td>
</tr>
<tr>
<td>60</td>
<td>81.2</td>
</tr>
<tr>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>120</td>
<td>40.6</td>
</tr>
<tr>
<td>150</td>
<td>0.4</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Angle</th>
<th>Dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>57.1</td>
</tr>
<tr>
<td>30</td>
<td>79.5</td>
</tr>
<tr>
<td>60</td>
<td>80.5</td>
</tr>
<tr>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>120</td>
<td>23.4</td>
</tr>
<tr>
<td>150</td>
<td>-19.5</td>
</tr>
</tbody>
</table>
Hough Solution:

4. Draw the lines received for each data point in the Hough space

5. Intersection of all lines indicate a line passing through all data points (remember: each point in Hough space represent a line in the original space)
Hough transform:: Example

features

votes

Applications and Algorithms in CV

Tutorial 3: Edge detection

Hough Transform
Hough transform:: other shapes
Hough Example On a Real Image

Applications and Algorithms in CV

Tutorial 3: Edge detection

Hough Transform
Hough Example On a Real Image

Showing longest segments found
Hough transform:: effect of noise
Hough transform: effect of noise

features

votes
Hough transform:: random points

features

votes
Random points:: Dealing with noise

Grid resolution tradeoff
- Too coarse: large votes obtained when too many different lines correspond to a single bucket
- Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets

Increment neighboring bins
- Smoothing in accumulator array

Try to get rid of irrelevant features
- Take only edge points with significant gradient magnitude
Random points:: conclusions

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

**Bad**
1. Some sensitivity to noise
2. Bin size trades off between noise tolerance, precision, and speed/memory
3. Not suitable for more than a few parameters (grid size grows exponentially)
(RANdom SAmple Consensus): Learning technique to estimate parameters of a model by random sampling of observed data
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
4. **Reevaluate** the fit according to the inliers

Repeat 1-4 until the best model is found with high confidence. Return the model with the most inliers.
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
4. **Reevaluate** the fit according to the inliers

Repeat 1-4 until the best model is found with high confidence Return the model with the most inliers
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
4. **Reevaluate** the fit according to the inliers

Repeat 1-4 until the best model is found with high confidence. Return the model with the most inliers.

**RANSAC::** line fitting example
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
4. **Reevaluate** the fit according to the inliers

Repeat 1-4 until the best model is found with high confidence. Return the model with the most inliers.

**RANSAC:: line fitting example**

\[ N_I = 6 \]
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
4. Reevaluate the fit according to the inliers

Repeat 1-4 until the best model is found with high confidence Return the model with the most inliers
**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
4. **Reevaluate** the fit according to the inliers

**Repeat 1-4 until the best model is found with high confidence** Return the model with the most inliers

**RANSAC:: line fitting example**

\[ N_I = 14 \]
**RANSAC:: parameter selection**

- \( s \): initial number of points (Typically minimum number needed to fit the model)
- \( N \): number of RANSAC iterations
- \( p \): probability of choosing \( s \) inliers at least in some iteration
- \( \omega \): probability of choosing an inlier
- \( \omega^s \): probability that all \( s \) points are inliers
- \( 1 - \omega^s \): probability that at least one point is an outlier
- \( 1 - p = (1 - \omega^s)^N \): probability that the algorithm never selects a set of \( s \) inliers

\[
N = \log \left( \frac{1 - p}{1 - \omega^s} \right)
\]
RANSAC:: Simple line fitting example

• Fitting a simple line

A data set with many outliers for which a line has to be fitted

Fitted line with RANSAC; outliers have no influence on the result
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