Computer Vision – Lecture 4

Structure Extraction

29.04.2019

Bastian Leibe
Visual Computing Institute
RWTH Aachen University
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de
Course Outline

• Image Processing Basics
  ➢ Image Formation
  ➢ Binary Image Processing
  ➢ Linear Filters
  ➢ Edge & Structure Extraction

• Segmentation

• Local Features & Matching

• Object Recognition and Categorization

• Deep Learning

• 3D Reconstruction
Topics of This Lecture

• Recap: Edge detection
  - Image gradients
  - Canny edge detector

• Fitting as parametric search
  - Line detection
  - Hough transform
  - Extension to circles
  - Generalized Hough transform
Recap: The Gaussian Pyramid

Low resolution

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]
\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]
\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]
\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]
\[ G_0 = \text{Image} \]

High resolution

Source: Irani & Basri
Recap: Derivatives and Edges…

1st derivative

Maxima of first derivative

2nd derivative

“zero crossings” of second derivative
Recap: 2D Edge Detection Filters

- \( \nabla^2 \) is the Laplacian operator:
  \[
  \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}
  \]

\[h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}\]

Gaussian

Derivative of Gaussian

Laplacian of Gaussian

See Exercise 2.2!
Recap: Canny Edge Detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

• MATLAB:
  >> edge(image, ‘canny’);
  >> help edge

adapted from D. Lowe, L. Fei-Fei
Edges vs. Boundaries

Edges are useful signals to indicate occluding boundaries, shape.

Here the raw edge output is not so bad…

…but quite often boundaries of interest are fragmented, and we have extra “clutter” edge points.

Slide credit: Kristen Grauman
Edge Detection is Just the Beginning…

Image  |  Human segmentation  |  Gradient magnitude

• Berkeley segmentation database: [http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/](http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/)

Source: L. Lazebnik
Fitting

- Want to associate a model with observed features

For example, the model could be a line, a circle, or an arbitrary shape.

[Figure from Marszalek & Schmid, 2007]

Slide credit: Kristen Grauman
Topics of This Lecture

• Recap: Edge detection
  - Image gradients
  - Canny edge detector

• Fitting as parametric search
  - Line detection
  - Hough transform
  - Extension to circles
  - Generalized Hough transform
Example: Line Fitting

- Why fit lines?
  - Many objects are characterized by presence of straight lines

- Wait, 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?
Fitting Lines

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

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

• Main idea:
  1. Vote for all possible lines on which each edge point could lie.
  2. Look for line candidates that get many votes.
  3. Noise features will cast votes too, *but* their votes should be inconsistent

Slide adapted from Kristen Grauman
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.
  - 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

- Connection between image \((x, y)\) and Hough \((m, b)\) spaces
  - A line in the image corresponds to a point in Hough space.
  - To go from image space to Hough space:
    - Given a set of points \((x, y)\), find all \((m, b)\) such that \(y = mx + b\)
  - What does a point \((x_0, y_0)\) in the image space map to?
    - Answer: the solutions of \(b = -x_0m + y_0\)
    - This is a line in Hough space.
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
    \[
    b = -x_0m + y_0 \quad \text{and} \quad b = -x_1m + y_1
    \]
Finding Lines in an Image: Hough Space

- 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.

Slide credit: Steve Seitz
Polar Representation for Lines

- Issues with usual \((m, b)\) parameter space: can take on infinite values, undefined for vertical lines.

\[
\begin{align*}
x \cos \theta - y \sin \theta &= d \\
d : \text{perpendicular distance from line to origin} \\
\theta : \text{angle the perpendicular makes with the } x\text{-axis}
\end{align*}
\]

- Point in image space
  \Rightarrow \text{Sinusoid segment in Hough space}

Slide adapted from Steve Seitz
Hough Transform Algorithm

Using the polar parameterization:
\[ x \cos \theta + y \sin \theta = d \]

Basic Hough transform algorithm
1. Initialize \( H[d, \theta] = 0 \).
2. For each edge point \((x, y)\) in the image
   for \( \theta = 0 \) to 180 // some quantization
   \[ d = x \cos \theta + y \sin \theta \]
   \[ H[d, \theta] += 1 \]
3. Find the value(s) of \((d, \theta)\) where \( H[d, \theta] \) is maximal.
4. The detected line in the image is given by \( d = x \cos \theta + y \sin \theta \)

- Time complexity (in terms of number of votes)?

Slide credit: Steve Seitz
Example: HT for Straight Lines

Image space

edge coordinates

 Votes

Bright value = high vote count
Black = no votes

Slide credit: David Lowe
Real-World Examples
Showing longest segments found

Slide credit: Kristen Grauman
Impact of Noise on Hough Transform

What difficulty does this present for an implementation?

Slide credit: David Lowe
Impact of Noise on Hough Transform

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

Slide credit: David Lowe
Extensions

Extension 1: Use the image gradient

1. same

2. for each edge point $I[x, y]$ in the image

   $\theta = \text{gradient at } (x, y)$

   $d = x \cos \theta + y \sin \theta$

   $H[d, \theta] += 1$

3. same

4. same

(Reduces degrees of freedom)
Extensions

Extension 1: Use the image gradient
1. same
2. for each edge point $I[x, y]$ in the image
   compute unique $(d, \theta)$ based on image gradient at $(x, y)$.
   $H[d, \theta] += 1$
3. same
4. same

(Reduces degrees of freedom)

Extension 2
- Give more votes for stronger edges (use magnitude of gradient)

Extension 3
- Change the sampling of $(d, \theta)$ to give more/less resolution

Extension 4
- The same procedure can be used with circles, squares, or any other shape
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

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

Slide credit: Kristen Grauman
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

Slide credit: Kristen Grauman
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 \(r\), *known* gradient direction
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
      
      \[a = x - r \cos(\theta)\]
      
      \[b = y + r \sin(\theta)\]
      
      \[H[a, b, r] += 1\]
Example: Detecting Circles with Hough

Crosshair indicates results of Hough transform, bounding box found via motion differencing.
Example: Detecting Circles with Hough

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

Slide credit: Kristen Grauman

Coin finding sample images from: Vivek Kwatra
Example: Detecting Circles with Hough

- Original detections
- Edges
- Votes: Quarter

Coin finding sample images from: Vivek Kwatra

Slide credit: Kristen Grauman
Voting: Practical Tips

• Minimize irrelevant tokens first (take edge points with significant gradient magnitude)
• 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 some points that are not exactly collinear cast votes for different buckets
• Vote for neighbors, also (smoothing in accumulator array)
• Utilize direction of edge to reduce free parameters by 1
• To read back which points voted for “winning” peaks, keep tags on the votes.

Slide credit: Kristen Grauman
Hough Transform: Pros and Cons

Pros

• All points are processed independently, so can cope with occlusion
• 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: hard to pick a good grid size
Generalized Hough Transform

- What if we want to detect arbitrary shapes defined by boundary points and a reference point?

At each boundary point, compute displacement vector:

\[ r = a - p_i. \]

For a given model shape: store these vectors in a table indexed by gradient orientation \( \theta \).

[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

Slide credit: Kristen Grauman
Generalized Hough Transform

To detect the model shape in a new image:

• For each edge point
  - Index into table with its gradient orientation \( \theta \)
  - Use retrieved \( r \) vectors to vote for position of reference point

• Peak in this Hough space is reference point with most supporting edges

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

Say we’ve already stored a table of displacement vectors as a function of edge orientation for this model shape.
Example: Generalized Hough Transform

Now we want to look at some edge points detected in a new image, and vote on the position of that shape.
Example: Generalized Hough Transform

Range of voting locations for test point

Slide credit: Svetlana Lazebnik
Example: Generalized Hough Transform

Range of voting locations for test point

Slide credit: Svetlana Lazebnik
Example: Generalized Hough Transform

Votes for points with $\theta = \uparrow$
Example: Generalized Hough Transform

Displacement vectors for model points
Example: Generalized Hough Transform

Range of voting locations for test point
Example: Generalized Hough Transform

Votes for points with $\theta =$
Application in Recognition

• Instead of indexing displacements by gradient orientation, index by “visual codeword”.

Application in Recognition

- Instead of indexing displacements by gradient orientation, index by “visual codeword”.

Test image

- Possible mechanism for designing object detectors…
References and Further Reading

- Background information on edge detection can be found in Chapter 3 of the Szeliski book or in Chapter 8 of Forsyth & Ponce.

  R. Szeliski
  Computer Vision – Algorithms and Applications
  Springer, 2010

  D. Forsyth, J. Ponce,
  Computer Vision – A Modern Approach.
  Prentice Hall, 2003

- Read Ballard & Brown’s description of the Generalized Hough Transform in Chapter 4.3 of

  (available from the class webpage)