Computer Vision - Lecture 5

Structure Extraction

30.10.2014

Bastian Leibe
RWTH Aachen
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

- 3D Reconstruction

- Motion and Tracking
Topics of This Lecture

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

- **Fitting as template matching**
  - Distance transform
  - Chamfer matching
  - Application: traffic sign detection

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

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

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

Gaussian

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

Derivative of Gaussian

\[ \frac{\partial}{\partial x} h_{\sigma}(u, v) \]

Laplacian of Gaussian

\[ \nabla^2 h_{\sigma}(u, v) \]

- \( \nabla^2 \) is the Laplacian operator:

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]

Slide credit: Kristen Grauman
Designing an Edge Detector

- Criteria for an “optimal” edge detector:
  - **Good detection**: the optimal detector should minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges).
  - **Good localization**: the edges detected should be as close as possible to the true edges.
  - **Single response**: the detector should return one point only for each true edge point; that is, minimize the number of local maxima around the true edge.

![Diagram showing true edge, poor robustness to noise, poor localization, and too many responses.](image)
Primary edge detection steps

1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization
   - Determine which local maxima from filter output are actually edges vs. noise
   - Thresholding, thinning

- Two issues
  - At what scale do we want to extract structures?
  - How sensitive should the edge extractor be?
The apparent structures differ depending on Gaussian’s scale parameter.

- Larger values: larger-scale edges detected
- Smaller values: finer features detected
Sensitivity: Recall Thresholding

- Choose a threshold $t$
- Set any pixels less than $t$ to zero (off).
- Set any pixels greater than or equal $t$ to one (on).

\[
F_T[i,j] = \begin{cases} 
1, & \text{if } F[i,j] \geq t \\
0, & \text{otherwise}
\end{cases}
\]
Gradient Magnitude Image
Thresholding with a Lower Threshold
Thresholding with a Higher Threshold
Canny Edge Detector

- Probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization.

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
The Canny Edge Detector

Original image (Lena)

Slide credit: Kristen Grauman
The Canny Edge Detector

Gradient magnitude

Slide credit: Kristen Grauman
The Canny Edge Detector

How to turn these thick regions of the gradient into curves?
Non-Maximum Suppression

- Check if pixel is local maximum along gradient direction, select single max across width of the edge
  - Requires checking interpolated pixels p and r
  - \( \Rightarrow \) Linear interpolation based on gradient direction

Source: Forsyth & Ponce
The Canny Edge Detector

Problem: pixels along this edge didn’t survive the thresholding.

Thinning
(non-maximum suppression)

Slide credit: Kristen Grauman
Solution: Hysteresis Thresholding

- **Hysteresis**: A lag or momentum factor
- **Idea**: Maintain two thresholds $k_{\text{high}}$ and $k_{\text{low}}$
  - Use $k_{\text{high}}$ to find strong edges to start edge chain
  - Use $k_{\text{low}}$ to find weak edges which continue edge chain
- **Typical ratio of thresholds** is roughly
  \[ \frac{k_{\text{high}}}{k_{\text{low}}} = 2 \]
Hysteresis Thresholding

Original image

High threshold (strong edges)

Low threshold (weak edges)

Hysteresis threshold

Source: L. Fei-Fei
courtesy of G. Loy
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
Object Boundaries vs. Edges

Background
Texture
Shadows

Slide credit: Kristen Grauman
B. Leibe
Edge Detection is Just the Beginning...

- Berkeley segmentation database:  
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/
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 template matching
  - Distance transform
  - Chamfer matching
  - Application: traffic sign detection
- Fitting as parametric search
  - Line detection
  - Hough transform
  - Extension to circles
  - Generalized Hough transform
Fitting as Template Matching

- We’ve already seen that correlation filtering can be used for template matching in an image.

- Let’s try this idea with “edge templates”.
  - Example: traffic sign detection in (grayvalue) video.
Edge Templates

- Correlation filtering
  
  Correlation between edge pixels in template and image

  \[ D_{\text{corr}}(x, y) = - \sum_{u,v} T[u, v] I[x + u, y + v] \]

  Unfortunately, this doesn’t work at all... Why?

  ⇒ Zero correlation score if the edge template is 1 pixel off...
Edge Templates

- Better: Chamfer Distance
  - Average distance to nearest edge pixel
    \[ D_{Chamfer}(x, y) = \frac{1}{|T|} \sum_{u,v: T[u,v] = 1} d_t(x + u, y + v) \]
  - More robust to small shifts and size variations.

- How can we compute this efficiently?
How Can This Be Made Efficient?

- Fast edge-based template matching
  - Distance transform of the edge image

Value at \((x,y)\) tells how far that position is from the nearest edge point (or other binary image structure)

\[
>> \text{help bwdist}
\]
Distance Transform

- Image reflecting distance to nearest point in point set (e.g., edge pixels, or foreground pixels).

4-connected adjacency

8-connected adjacency
Distance Transform Algorithm (1D)

- Two-pass O(n) algorithm for 1D $L_1$ norm

1. **Initialize:** For all $j$
   - $D[j] \leftarrow 1_P[j]$ \hspace{1cm} // 0 if $j$ is in $P$, infinity otherwise

2. **Forward:** For $j$ from 1 up to $n-1$
   - $D[j] \leftarrow \min( D[j], D[j-1]+1 )$ 

3. **Backward:** For $j$ from $n-2$ down to 0
   - $D[j] \leftarrow \min( D[j], D[j+1]+1 )$

Adapted from D. Huttenlocher
Distance Transform Algorithm (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to the left
    - Bwd pass finds closest below and to the right

Adapted from D. Huttenlocher
Chamfer Matching

- Chamfer Distance
  - Average distance to nearest feature
    \[ D_{\text{chamfer}}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t) \]
  - This can be computed efficiently by correlating the edge template with the distance-transformed image

Edge image

Distance transform image
Chamfer Matching

- Efficient implementation
  - Instead of correlation, sample fixed number of points on template contour.
  - Chamfer score boils down to series of DT lookups.
    \[
    D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)
    \]
  - Computational effort independent of scale.

Edge image  Distance transform image

B. Leibe  [D. Gavrila, DAGM’99]
Chamfer Matching Results

Edge image
B. Leibe

Distance transform image
[D. Gavrila, DAGM’99]
Chamfer Matching for Pedestrian Detection

- Organize templates in tree structure for fast matching
Chamfer Matching for Pedestrian Detection

[D. Gavrila, V. Philomin, ICCV’99, PAMI’07]
Summary Chamfer Matching

• **Pros**
  - Fast and simple method for matching edge-based templates.
  - Works well for matching upright shapes with little intra-class variation.
  - Good method for finding candidate matches in a longer recognition pipeline.

• **Cons**
  - Chamfer score averages over entire contour, not very discriminative in practice.
    ⇒ Further verification needed.
  - Low matching cost in cluttered regions with many edges.
    ⇒ Many false positive detections.
  - In order to detect rotated & rescaled shapes, need to match with rotated & rescaled templates ⇒ can get very expensive.
Topics of This Lecture

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

• Fitting as template matching
  - Distance transform
  - Chamfer matching
  - Application: traffic sign detection

• Fitting as parametric search
  - Line detection
  - Hough transform
  - Extension to circles
  - Generalized Hough transform
Fitting as Search in Parametric Space

- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can’t tell whether a point belongs to a given model just by looking at that point.
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features
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?

Slide credit: Kristen Grauman
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?

Slide credit: Kristen Grauman

B. Leibe
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 will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.

• Ok if some features not observed, as model can span multiple fragments.
Fitting Lines

- 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 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.
  - 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.
Polar Representation for Lines

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

\[ x \cos \theta - y \sin \theta = d \]

- Point in image space \(\Rightarrow\) Sinusoid segment in Hough space

\(d\) : perpendicular distance from line to origin
\(\theta\) : angle the perpendicular makes with the \(x\)-axis
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 \( d = x \cos \theta - y \sin \theta \)

**Hough line demo**

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

Slide credit: Steve Seitz

B. Leibe
Example: HT for Straight Lines

Image space
edge coordinates

Votes
Bright value = high vote count
Black = no votes
Real-World Examples
Showing longest segments found

Slide credit: Kristen Grauman

B. Leibe
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)

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
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...

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 a fixed 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 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**

![Image of Hough Transform for Circles](image_url)
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

![Image space](image-space.png) ![Hough space](hough-space.png)

Slide credit: Kristen Grauman
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.

Slide credit: Kristen Grauman
Example: Detecting Circles with Hough

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

Original detections

Edges

Votes: Quarter

Slide credit: Kristen Grauman

B. Leibe

Coin finding sample images from: Vivek Kwatra
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.
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 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]
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.

Displacement vectors for model points
Example: Generalized Hough Transform
Example: Generalized Hough Transform

Range of voting locations for test point
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 $\theta = \theta^*$
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

• We’ll hear more about this method in lecture 14...
References and Further Reading

• Background information on edge detection can be found in Chapter 8 of

  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

  D.H. Ballard & C.M. Brown,  
  *Computer Vision*, Prentice Hall, 1982  
  (available from the class homepage)

• Try the Hough Transform demo at
  http://www.dis.uniroma1.it/~iocchi/slides/icra2001/java/hough.html