CS4501: Introduction to Computer Vision

Hough Transform and SIFT Features

Various slides from previous courses by:
D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL), S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook). J. M. Frahm (UNC), V. Ordonez (UVA).
Last Class – Interest Points

- Corner Detection - Harris
- Blob Detection – Laplacian of Gaussian / Difference of Gaussians (DoG)
Today’s Class

• Blog Detection – Difference of Gaussians
• SIFT Feature descriptor – Feature Matching
• Hough Transform -> For Line Detection
How to do Line Detection?
How to do Line Detection?
Idea: Sobel First!

![Image showing the idea of Sobel First with coordinates x = 12px, x = 38px, x = 68px, y = 16px, y = 20px, y = 56px]
Idea: Sobel First!
Idea: Then Count Pixels that support each line hypothesis.
Idea: Then Count Pixels that support each line hypothesis.
Problem with this?

count (sum) → count (sum)

x = 12px  x = 38px  x = 68px
y = 16px
y = 20px
y = 56px
What if you’re given this instead?
How do we get this?
Furthermore. How do we get this!? 

\[ y = -2x + 30 \]

\[ y = -2x + 20 \]

\[ y = -2x + 10 \]

\[ y = x + 10 \]

\[ y = x + 7 \]

\[ y = x - 5 \]
Hough transform

• An early type of voting scheme for Detecting Lines

• General outline:
  • Discretize *parameter space* into bins
  • For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
  • Find bins that have the most votes

Hough Transform: Let’s again apply Sobel first
Hough Transform: Let’s again apply Sobel first
Then let’s count but now in a 2D array

\[ y = mx + b \]

Count all points intersecting with all lines with \( m = (0, \infty) \), \( b = [-B_{-\infty}, B_{\text{max}}] \)
• Problems with the (m,b) space:
  • Unbounded parameter domains
  • Vertical lines require infinite m
Parameter space representation

- Problems with the \((m,b)\) space:
  - Unbounded parameter domains
  - Vertical lines require infinite \(m\)
- Alternative: *polar representation*

\[
x \cos \theta + y \sin \theta = \rho
\]
Then let’s count but now in a 2D array

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

Count all points intersecting with all lines with \( \rho = (-\text{diagonal, diagonal}), \theta = [0, 180] \)
Hough Transform 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
- 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
    - $\rho = x \cos \theta + y \sin \theta$
• Each point \((x, y)\) in Image Space will add a sinusoid in the Hough Transform \((\theta, \rho)\) parameter space.
Basic illustration

features

votes
Hough Transform for an Actual Image
Edges using threshold on Sobel’s magnitude
Hough Transform (High Resolution)

\[ \rho = -\sqrt{h^2 + w^2} \]

\[ \rho = 0 \]

\[ \rho = \sqrt{h^2 + w^2} \]
Hough Transform (After threshold)

\[ \rho = -\sqrt{h^2 + w^2} \]

\[ \rho = 0 \]

\[ \rho = \sqrt{h^2 + w^2} \]
Hough Transform (After threshold)

\[ \rho = -\sqrt{h^2 + w^2} \]

\[ \rho = 0 \]

\[ \rho = \sqrt{h^2 + w^2} \]

\[ \theta = -90^0 \quad \theta = 0 \quad \theta = 90^0 \]

Vertical lines
Hough Transform (After threshold)

\[
\rho = \sqrt{h^2 + w^2}
\]

\[
\rho = -\sqrt{h^2 + w^2}
\]

\[
\rho = 0
\]

\[
\theta = -90^0 \quad \theta = 0 \quad \theta = 90^0
\]

Vertical lines
Hough Transform with Non-max Suppression

\[
\rho = -\sqrt{h^2 + w^2}
\]

\[
\rho = 0
\]

\[
\rho = \sqrt{h^2 + w^2}
\]

\(\theta = -90^0 \quad \theta = 0 \quad \theta = 90^0\)
Back to Image Space – with lines detected

\[ y = -\frac{\cos\theta}{\sin\theta} x + \frac{\rho}{\sin\theta} \]

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

\[
\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}
\]

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

image space

Hough parameter space

\[(x, y) + r \nabla I(x, y)\]

\[(x, y) - r \nabla I(x, y)\]
Hough transform for circles

- Conceptually equivalent procedure: for each \((x,y,r)\), draw the corresponding circle in the image and compute its “support.”

Is this more or less efficient than voting with features?
Basic idea

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*

Blob filter

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]
Blob detection

- Find maxima \textit{and minima} of blob filter response in space \textit{and scale}

Source: N. Snavely
Blob at multiple scales – Option 1
Apply Non-Max Suppression – Show blobs as circles
Scale-space blob detector: Example
Scale-space blob detector: Example
Blog at Multiple Scales: Option 2

sigma = 11.9912
Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space

Slide by Svetlana Lazebnik
Scale-space blob detector: Example
Efficient implementation

- Approximating the Laplacian with a difference of Gaussians:

\[ L = \sigma^2 \left( G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right) \]  
(Laplacian)

\[ DoG = G(x, y, k\sigma) - G(x, y, \sigma) \]  
(Difference of Gaussians)
Efficient implementation

Gaussian Pyramid – DoG pyramid


Figure from Workload analysis and efficient OpenCL-based implementation of SIFT algorithm on a smartphone

*Guohui Wang, Blaine Rister, Joseph R. Cavallaro*
Gaussian Pyramid


Figure from Workload analysis and efficient OpenCL-based implementation of SIFT algorithm on a smartphone

*Guohui Wang, Blaine Rister, Joseph R. Cavallaro*
Same results
Locations + Scales + Orientations

Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram
From keypoint detection to keypoint representation (feature descriptors)

Figure by Svetlana Lazebnik
SIFT descriptors

- Inspiration: complex neurons in the primary visual cortex

From keypoint detection to keypoint representation (feature descriptors)

Compare SIFT feature vectors instead
SIFT Feature Matching

Rice Hall at UVA
JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan Dat Nguyen, Ming-Ming Cheng
GMS: Grid-based Motion Statistics for Fast, Ultra-robust Feature Correspondence IEEE CVPR, 2017
The method has been integrated into OpenCV library (see xfeatures2d in opencv_contrib).
A hard keypoint matching problem

NASA Mars Rover images
Answer below (look for tiny colored squares...)

NASA Mars Rover images with SIFT feature matches
Figure by Noah Snavely
Feature Descriptors Zoo

• SIFT (under a patent) Proposed around 1999
• SURF (under a patent too – I think)
• BRIEF
• ORB (seems free as it is OpenCV’s preferred)
• BRISK
• FREAK
• FAST
• KAZE
• LIFT (Most recently proposed at ECCV 2016)
• New ones? I haven’t kept track
<table>
<thead>
<tr>
<th>TITLE</th>
<th>CITED BY</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinctive image features from scale-invariant keypoints</td>
<td>45496</td>
<td>2004</td>
</tr>
<tr>
<td>DG Lowe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>International journal of computer vision 60 (2), 91-110</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object recognition from local scale-invariant features</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DG Lowe</td>
<td>14817</td>
<td>1999</td>
</tr>
<tr>
<td>International Conference on Computer Vision, 1999, 1150-1157</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Questions?