Lecture 8
Image Segmentation

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2016
Outline

- Fundamentals
- Isolated point detection
- Edge detection
- Segmentation based on thresholding
- Analytic element detection by Hough transform
Fundamentals

- Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image
- It is one of the most difficult tasks in image processing
- Typically the first step in any automated computer vision application
Fundamentals—Segmentation Examples

Input image

Segmentation result
Fundamentals—Segmentation Examples
Fundamentals

There are three basic types of grey level discontinuities that we tend to look for in digital images:

- Points
- Lines
- Edges

We typically find discontinuities using masks and correlation.
Outline

• Fundamentals
• Isolated point detection
• Edge detection
• Segmentation based on thresholding
• Analytic element detection by Hough transform
Isolated Point Detection

- The detection of isolated points embedded in areas of constant or nearly constant intensity in an image can be fulfilled by using Laplacian operator

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

Laplacian kernel for convolution
Isolated Point Detection

- The detection of isolated points embedded in areas of constant or nearly constant intensity in an image can be fulfilled by using Laplacian operator

The output can be obtained using

\[ g(x, y) = \begin{cases} 
1, & \text{if } |R(x, y)| \geq T \\
0, & \text{otherwise}
\end{cases} \]

where \( T \) is a threshold, and \( R(x, y) \) is the image’s response to Laplacian kernel
Isolated Point Detection—An Example

There is an isolated black point in the input image.

How to detect it?

X-ray image of turbine blade
Isolated Point Detection—An Example

Laplacian filtering result

Point detection result after thresholding

Source code for this demo is available on our course website

Lin ZHANG, SSE, 2016
Outline

- Fundamentals
- Isolated point detection
- Edge detection
  - General concepts
  - Basic edge detection
  - More advanced techniques
- Segmentation based on thresholding
- Analytic element detection by Hough transform
General concepts

• What are edges?
  • Edges are pixels where the image function changes abruptly

• Why edge detection is useful?
  • Neurological and psychophysical research suggests that locations in the image in which the function value changes abruptly are important for image perception
  • Such a process will lead to a significant reduction of image data, however, does not undermine understanding of the content of the image (interpretation) in many cases
General concepts

• Main causes of edges
  • Depth discontinuity
    – one surface occludes another
  • Surface orientation discontinuity
    – the edge of a block
  • reflectance discontinuity
    – texture or color changes
  • illumination discontinuity
    – shadows
There are 3 fundamental steps for edge detection

- Image smoothing for noise reduction
- Detection of edge points. This is a local operation that extracts from an image all points that are potential candidates to become edge points
- Edge localization. This step is to select from the candidate edge points only the points that are true members of the set of points comprising an edge
Outline

• Fundamentals
• Isolated point detection
• Edge detection
  • General concepts
  • Basic edge detection
  • More advanced techniques
• Segmentation based on thresholding
• Analytic element detection by Hough transform
Basic Edge Detection

- Edges can be regarded as the extrema points of the first-order derivative.
Basic Edge Detection

• In 2D case, first order derivative is characterized by gradient

Definition of gradient

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

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]

\[ \theta = \arctan 2 \left( \frac{\partial f}{\partial y}, \frac{\partial f}{\partial x} \right) \]

where \( \theta \) is the direction of the gradient

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \]

\[ \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right] \]
Basic Edge Detection

• Look for points where the gradient magnitude is a maximum along the direction perpendicular to the edge
• The direction perpendicular to the edge can be estimated using the direction of the gradient
Basic Edge Detection

• For discrete case, 2D gradients are usually computed by using various kinds of gradient operators.

• Image’s gradient can be obtained by filtering the image with the gradient operator.

\[
\begin{align*}
\text{Sobel operator} & \quad h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \\
\text{Prewitt operator} & \quad h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}
\end{align*}
\]
Basic Edge Detection—An Example (Sobel)

Input image

\[
\frac{\partial f}{\partial x}
\]

\[
\frac{\partial f}{\partial y}
\]

Gradient magnitude
Basic Edge Detection

• Derivative with smoothing

Consider a single row or column of the image

• Plotting intensity as a function of position gives a signal

Where is the edge?
Basic Edge Detection

• Derivative with smoothing
  • Finite difference filters respond strongly to noise
    – Image noise results in pixels that look very different from their neighbors
    – Generally, the larger the noise the stronger the response
  • What is to be done?
    – Smoothing the image should help, by forcing pixels different from their neighbors (=noise pixels?) to look more like neighbors
Basic Edge Detection

- Derivative with smoothing

To find edges, look for peaks in $\frac{d}{dx}(f * g)$.
Basic Edge Detection

- Differentiation of convolution has the following property:
  \[ \frac{d}{dx} (f * g) = f * \frac{d}{dx} g \]
- This saves us one operation:
• Fundamentals
• Isolated point detection
• Edge detection
  • General concepts
  • Basic edge detection
  • More advanced techniques
• Segmentation based on thresholding
• Analytic element detection by Hough transform
Marr-Hildreth Edge Detector (Zero-crossings of LoG)

Edges can be regarded as the extrema points of the first-order derivative

They correspond to zero-crossings of the 2\textsuperscript{nd}-order derivative
Marr-Hildreth Edge Detector (Zero-crossings of LoG)

• In 2D case, 2\textsuperscript{nd}-order derivative can be characterized by Laplacian operator $\nabla^2 f$

• However, Laplacian operator is sensitive to noise; we need to apply a Gaussian filtering to $f$ before using $\nabla^2$

Since,

$$\left( \nabla^2 (G_\sigma \ast f) \right)(x, y) = \left( \left( \nabla^2 G_\sigma \right) \ast f \right)(x, y)$$

we can compute $\nabla^2 G_\sigma(x, y)$ offline
Marr-Hildreth Edge Detector (Zero-crossings of LoG)

If \( \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \), \( G_\sigma(x, y) = e^{\frac{x^2 + y^2}{2\sigma^2}} \)

Then,

\[
\nabla^2 G_\sigma(x, y) = \left( \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) \exp\left( -\frac{x^2 + y^2}{2\sigma^2} \right)
\]

Laplacian of Gaussian (LoG) operator

How to verify?
Marr-Hildreth Edge Detector (Zero-crossings of LoG)

Note: the LoG function can be approximated by a difference of Gaussian (DoG)

\[
DoG(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}
\]

with \( \sigma_1 > \sigma_2 \)

As suggested by Marr and Hildreth, \( \sigma_1 : \sigma_2 \) is set as 1.6:1 to provides a closer “engineering” approximation to the LoG function
Marr-Hildreth Edge Detector (Zero-crossings of LoG)

An example of LoG filter

```
logFunction = fspecial('log',51, 8);
figure;
surfl(logFunction);
shading interp
colormap(gray);
figure;imshow(logFunction,[]);
```

LoG filter is also called as *Mexican Hat* due to its shape
Marr-Hildreth Edge Detector (Zero-crossings of LoG)

- The Marr-Hildreth edge detection algorithm may be summarized as follows
  1. Filter the input image with a Gaussian low-pass filter
  2. Compute the Laplacian of the result in step 1
  3. Find the zero-crossings of the image from step 2
     (Of course, steps 1 and 2 can be combined by using one operator LoG)
Marr-Hildreth Edge Detector (Zero-crossings of LoG)

- How to identify the zero-crossing points?

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_4$</td>
<td>$P$</td>
<td>$P_5$</td>
</tr>
<tr>
<td>$P_6$</td>
<td>$P_7$</td>
<td>$P_8$</td>
</tr>
</tbody>
</table>

If $p$ is a zero-crossing point, the **signs** of at least two of its opposing neighboring pixels must differ.

Usually, we also require that the absolute value of their difference must also exceed a pre-defined threshold before we can call $p$ a zero-crossing pixel (see the example in the next page).
Marr-Hildreth Edge Detector (Zero-crossings of LoG)
Canny Edge Detection [1]

- Although the algorithm is more complex, the performance of the Canny edge detector is superior in general to the edge detectors discussed before.

- It can be summarized as the following steps:
  1. Smooth the input image with a Gaussian filter.
  2. Compute the gradient magnitude and angle maps.
  3. Apply nonmaxima suppression to the gradient magnitude map.
  4. Use double thresholding and connectivity analysis to detect and link edges.

Canny Edge Detection [1]

Original image
Canny Edge Detection [1]

Gradient magnitude map
Canny Edge Detection [1]

Thinning (non-maxima suppression)
Canny Edge Detection [1]

- Non-maxima suppression
  - Check if gradient magnitude at pixel location \((i, j)\) is local maximum along gradient direction
Canny Edge Detection [1]

- Non-maxima suppression

**Algorithm**

For each pixel \((i, j)\) do:

if \(\text{magn}(i, j) < \text{magn}(i_1, j_1)\) or \(\text{magn}(i, j) < \text{magn}(i_2, j_2)\)

then \(I_N(i, j) = 0\)

else \(I_N(i, j) = \text{magn}(i, j)\)

Warning: requires checking interpolated pixels \(p\) and \(r\)
Implementation Tips

In Matlab, edge detection can be completed by using the built-in function “edge”

```matlab
im = imread('tongji.bmp');
edgeResult = edge(im,'canny');
imwrite(edgeResult,'canny.bmp');
figure;
imshow(edgeResult,[]);
```
Implementation Tips

In Matlab, edge detection can be completed by using the built-in function “edge”

Edge detection result by using “canny”
Outline

- Fundamentals
- Isolated point detection
- Edge detection
- Segmentation based on thresholding
  - Foundation
  - Basic global thresholding
  - Otsu’s optimum global thresholding
  - Variable thresholding
- Analytic element detection by Hough transform
Because of its intuitive properties, simplicity of implementation, and computational speed, image thresholding enjoys a central role in image segmentation.

Suppose that image $f(x, y)$ comprises of light objects on a dark background, the objects can be separated out by

$$g(x, y) = \begin{cases} 1, & \text{if } f(x, y) > T \\ 0, & \text{otherwise} \end{cases}$$
Foundation—Thresholding Example

- Imagine a poker playing robot that needs to visually interpret the cards in its hand.

![Original Image](original.png)

![Thresholded Image](thresholded.png)

Original Image

Thresholded Image
Foundation—Thresholding Example

- If you get the threshold wrong, the results can be disastrous
Foundation—the role of noise in thresholding

Without additional processing, we have little hope of finding a suitable threshold for segmenting the third image.
Foundation—the role of illumination

Without additional processing, we have little hope of finding a suitable threshold for segmenting the third image.
Foundation

- The success of intensity thresholding is related to the width and depth of the valley(s) separating the histogram modes
- The key factors are
  - The separation between peaks (the further apart the peaks are, the better the chances of separating the modes)
  - The noise content in the image (the modes broaden as noise increases)
  - The uniformity of the illumination
  - The uniformity of the reflectance properties of the image
Outline

• Fundamentals
• Isolated point detection
• Edge detection
• Segmentation based on thresholding
  • Foundation
    • Basic global thresholding
    • Otsu’s optimum global thresholding
    • Variable thresholding
  • Analytic element detection by Hough transform
Basic Global Thresholding

- Partition the image histogram using a single global threshold
- The success of this technique very strongly depends on how well the histogram can be partitioned
Basic Global Thresholding

The basic global threshold, $T$, is calculated as follows:

1. Select an initial estimate for $T$ (typically the average grey level in the image)
2. Segment the image using $T$ to produce two groups of pixels: $G_1$ consisting of pixels with grey levels $> T$ and $G_2$ consisting of pixels with grey levels $\leq T$
3. Compute the average intensity values $m_1$ and $m_2$ for the pixels in $G_1$ and $G_2$, respectively
4. Compute a new threshold $T = \left( m_1 + m_2 \right) / 2$
5. Repeat steps 2 through 4 until the difference between values of $T$ in successive iterations is smaller than a predefined parameter $\Delta T$
Basic Global Thresholding—An Example

Using the basic global thresholding algorithm, threshold = 125.4
Basic Global Thresholding—An Example

Segmentation result

Source code for this demo is available on our course website

Lin ZHANG, SSE, 2016
Outline

• Fundamentals
• Isolated point detection
• Edge detection
• Segmentation based on thresholding
  • Foundation
  • Basic global thresholding
    • Otsu’s optimum global thresholding
    • Variable thresholding
• Analytic element detection by Hough transform
Otsu’s Optimum Global Thresholding

• Otsu’s method can choose the optimum threshold in the sense that it maximizes the *between-class variance*

Let $L$ denote the distinct intensity levels of an image with the size $M \times N$ and $n_i$ denote the number of pixels with intensity $i$. Then, the normalized histogram has components

$$p_i = \frac{n_i}{MN}, \quad \sum_{i=0}^{L-1} p_i = 1, \quad p_i \geq 0$$

Suppose that a threshold $k$ is chosen such that $C_1$ is the set of pixels with levels $[0, 1, \ldots, k]$ and $C_2$ is the set of pixels with levels $[k+1, \ldots, L-1]$
Otsu’s Optimum Global Thresholding

We want to find a $k$ such that it *maximizes* the “between class variance”

$$\sigma_B^2 = P_1(k)[m_1(k) - m_G]^2 + P_2(k)[m_2(k) - m_G]^2$$

where

- $P_1(k)$ is the probability of set $C_1$ occurring  $P_1(k) = \sum_{i=0}^{k} p_i$
- $P_2(k)$ is the probability of set $C_2$ occurring and  $P_2(k) = 1 - P_1(k)$
- $m_1(k)$ and $m_2(k)$ are the mean intensities of $C_1$ and $C_2$
- $m_G$ is the global mean,  $m_G = \sum_{i=0}^{L-1} i p_i$

Thus, the optimum threshold is

$$k^* = \arg \max_{0 \leq k \leq L-1} \sigma_B^2(k)$$
Otsu’s Optimum Global Thresholding

Otsu’s algorithm can be summarized as follows

1. Compute the normalized histogram of the input image. Denote the components of the histogram by $p_i, i = 0, 1, \ldots, L-1$
2. Compute the global mean $m_G$
3. For each $0 \leq k \leq L-1$, compute $\sigma_B^2$
4. Obtain the optimal threshold $k^*$ that maximize $\sigma_B^2$
5. Use $k^*$ to segment the image

Implementation Tips

In Matlab, “graythresh” computes Otsu’s threshold
Otsu’s Optimum Global Thresholding—An Example

Basic global threshold  Otsu’s global threshold
Outline

• Fundamentals
• Isolated point detection
• Edge detection
• Segmentation based on thresholding
  • Foundation
  • Basic global thresholding
  • Otsu’s optimum global thresholding
    • Variable thresholding
• Analytic element detection by Hough transform
Variable Thresholding

- Factors such as noise and nonuniform illumination play a major role in the performance of a thresholding algorithm.
- In some cases (even seems very simple), a global threshold does not work well.
- That’s why we want to have a variable thresholding scheme.
Variable Thresholding based on Local Image Properties

• This basic approach uses the standard deviation and mean of the pixels in a neighborhood of every pixel in an image

• For each pixel, a distinct threshold will be computed

Let $\sigma_{xy}$ and $m_{xy}$ denote the standard deviation and mean of the pixels in a neighbourhood, centered at $(x, y)$

Local thresholds are defined as

$$ T_{xy} = a\sigma_{xy} + bm_{xy} $$

where $a$ and $b$ are nonnegative constants

The segmented image is computed as

$$ g(x, y) = \begin{cases} 
1, & \text{if } f(x, y) > T_{xy} \\
0, & \text{otherwise}
\end{cases} $$
Variable Thresholding—Moving Average

- Moving average is a special case of the local thresholding, which is based on computing a moving average along scan lines of an image.
- It is especially useful in document processing.
- The scanning is carried out line by line in a zigzag pattern to reduce illumination bias.
Variable Thresholding—Moving Average

- Moving average at the pixel $k$ is formed by averaging the intensities of that pixel and its $n-1$ preceding neighbors.

Suppose we have a 5*5 image, $n = 4$

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$b_3$</td>
<td>$b_4$</td>
<td>$b_5$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$c_2$</td>
<td>$c_3$</td>
<td>$c_4$</td>
<td>$c_5$</td>
</tr>
<tr>
<td>$d_1$</td>
<td>$d_2$</td>
<td>$d_3$</td>
<td>$d_4$</td>
<td>$d_5$</td>
</tr>
<tr>
<td>$e_1$</td>
<td>$e_2$</td>
<td>$e_3$</td>
<td>$e_4$</td>
<td>$e_5$</td>
</tr>
</tbody>
</table>

Step 1: reform it in a line following a zigzag way

| $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | ... |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
Variable Thresholding—Moving Average

- Moving average at the pixel $k$ is formed by averaging the intensities of that pixel and its $n-1$ preceding neighbors.

Suppose we have a 5*5 image, $n = 4$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$a_3$</td>
<td>$a_4$</td>
<td>$a_5$</td>
</tr>
<tr>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$b_3$</td>
<td>$b_4$</td>
<td>$b_5$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$c_2$</td>
<td>$c_3$</td>
<td>$c_4$</td>
<td>$c_5$</td>
</tr>
<tr>
<td>$d_1$</td>
<td>$d_2$</td>
<td>$d_3$</td>
<td>$d_4$</td>
<td>$d_5$</td>
</tr>
<tr>
<td>$e_1$</td>
<td>$e_2$</td>
<td>$e_3$</td>
<td>$e_4$</td>
<td>$e_5$</td>
</tr>
</tbody>
</table>

Step 2: for each position, compute the local average as the threshold

$$\frac{(a_4 + a_5 + b_5 + b_4)}{4}$$
Variable Thresholding—Moving Average

• Moving average at the pixel $k$ is formed by averaging the intensities of that pixel and its $n-1$ preceding neighbors

Suppose we have a 5*5 image, $n = 4$

\[
\begin{array}{cccc}
  a_1 & a_2 & a_3 & a_4 & a_5 \\
  b_1 & b_2 & b_3 & b_4 & b_5 \\
  c_1 & c_2 & c_3 & c_4 & c_5 \\
  d_1 & d_2 & d_3 & d_4 & d_5 \\
  e_1 & e_2 & e_3 & e_4 & e_5 \\
\end{array}
\]

Step 3: reform the local average “line” into the original matrix form to get the local threshold map $m(x, y)$

Segment the image as

\[
g(x, y) = \begin{cases} 
1, & \text{if } f(x, y) > K \cdot m(x, y) \\
0, & \text{otherwise}
\end{cases}
\]
Variable Thresholding—Moving Average

• An example
  
  • A handwritten text shaded by a spot intensity pattern; this form of shading is typical of images obtained with a photographic flash
Variable Thresholding—Moving Average

• An example

Segmentation result of global thresholding using Otsu’s method
Variable Thresholding—Moving Average

• An example

Segmentation result of local thresholding using moving averages
Variable Thresholding—Moving Average

• Another example

• Text image corrupted by a sinusoidal intensity variation typical of the variation that may occur when the power supply in a document scanner is not grounded properly
Variable Thresholding—Moving Average

- Another example

Segmentation result of global thresholding using Otsu’s method
Variable Thresholding—Moving Average

• Another example

Segmentation result of local thresholding using moving averages
Outline

• Fundamentals
• Isolated point detection
• Edge detection
• Segmentation based on thresholding
• Analytic element detection by Hough transform
What can Hough transform do?

- Find lines or curves in an input image. Such an image might be the output of an edge detector discussed in the previous lectures.
Voting schemes

• Let each feature vote for all the models that are compatible with it
• Hopefully the noise features will not vote consistently for any single model
• Missing data doesn’t matter as long as there are enough features remaining to agree on a good model
Hough transform

- An early type of voting scheme

- 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

Parameter space representation

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

\[ y = m_0 x + b_0 \]
What does a point \((x_0, y_0)\) in the image space map to in the Hough space?
Parameter space representation

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

- Where is the line that contains both \((x_0, y_0)\) and \((x_1, y_1)\)?
Parameter space representation

- Where is 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 \(b = -x_1m + y_1\)
Parameter space representation

• Problems with the \((m, b)\) space:
  • Unbounded parameter domain
  • Vertical lines require infinite \(m\)
Parameter space representation

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

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

Each point will add a sinusoid in the \((\theta, \rho)\) parameter space
Algorithm outline

• Initialize accumulator $H$ to all zeros

• For each edge point $(x, y)$ in the image
  
  For $\theta = -90$ to $90$
  
  $\rho = x \cos \theta + y \sin \theta$
  
  $H(\theta, \rho) = H(\theta, \rho) + 1$
  
  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 $\rho = x \cos \theta + y \sin \theta$
Basic illustration

features

votes
A real example

- Two lines correspond to two peaks in the parameter space

Figure 5.2  (a) An image containing two lines, sampled irregularly, and several random points. (b) Plot of the counters in the corresponding parameter space (how many points contribute to each cell \((m, n)\)). Notice that the main peaks are obvious, but there are many secondary peaks.
Incorporating image gradients

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

• Modified Hough transform:

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

\[ \theta = \text{gradient orientation at } (x, y) \]
\[ \rho = x \cos \theta + y \sin \theta \]
\[ H(\theta, \rho) = H(\theta, \rho) + 1 \]

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]
\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
Hough transform for finding lines in Matlab

- Matlab has 3 related routines to identify line segments in an image by using Hough transform
  - \([H, \text{theta}, \text{rho}] = \text{hough}(BW)\) implements the Standard Hough Transform (SHT)
  - \(\text{peaks} = \text{houghpeaks}(H, \text{numpeaks})\) After you compute the Hough transform, you can use the houghpeaks to find peak values in the parameter space. These peaks represent potential lines in the input image
  - \(\text{lines} = \text{houghlines}(BW, \text{theta}, \text{rho}, \text{peaks})\) After you identify the peaks in the Hough transform, you can use the houghlines to find the endpoints of the line segments corresponding to peaks in the Hough transform. This function automatically fills in small gaps in the line segments
Hough transform for finding lines in Matlab
Hough transform for circles

• How many dimensions will the parameter space have?

\[(x - a)^2 + (y - b)^2 = r^2\]

Three parameters \(a, b\) and \(r\) \(\Rightarrow\)

Accumulator data structure is 3D

In fact, an arbitrary curve can be represented by the equation

\[f(x, a) = 0\], where \(a\) is the vector of curve parameters
Hough transform – General Procedure

Algorithm: Curve detection using Hough transform
1. Quantize parameter space within the limits of parameters \( \mathbf{a} \). The dimensionality \( n \) of the parameter space is given by the number of parameters of the vector \( \mathbf{a} \).
2. Form an \( n \)-dimensional accumulator array \( A(\mathbf{a}) \) with structure matching the quantization of parameter space; set all elements to zero.
3. For each image point \((x_1, x_2)\) in the appropriately thresholded gradient image, increase all accumulator cells \( A(\mathbf{a}) \) if \( f(x, \mathbf{a}) = 0 \)
   \[
   A(\mathbf{a}) = A(\mathbf{a}) + 1
   \]
4. Local maxima in the accumulator array \( A(\mathbf{a}) \) correspond to realizations of curves \( f(x, \mathbf{a}) \) that are present in the original image.
Hough transform—Circles detection

Example

Original Image

Edges detected by Canny edge detector
Hough transform—Circles detection

Example

Hough Transform of the edge detected image

Detected circles
Hough transform: Discussion

• Pros
  • Can deal with non-locality and occlusion
  • Can detect multiple instances of a model
  • Some robustness to noise: noise points unlikely to contribute consistently to any single bin

• Cons
  • Complexity of search time increases exponentially with the number of model parameters
  • Non-target shapes can produce spurious peaks in parameter space
  • It’s hard to pick a good grid size
Thanks for your attention