Hough Transform

Outline
• What is Hough transform
• Line detection problem
  • Straight-line Hough transform
• Circle detection problem
  • Circle Hough transform

Hough Transform
• A tool that allows recognition of global patterns in an image space by recognition of local patterns (ideally a point) in a transformed parameter space
• Global patterns that can be recognized:
  • Lines, circles, polynomials, etc

Line Detection Problem
• Objective:
  • Find straight lines from an image
  • Edge detection algorithms only provide isolated edge pixels
• Slope-intercept form of line equation:
  • $y = mx + b$
  • Need to calculate parameter $m$ & $b$

Naive Approach
• for each pair edge pixels $<p1, p2>$ {
  • determine the line $l$ that passes $p1$ & $p2$;
  • initialize count to 0;
  • for any other edge pixel $p$ {
    • calculate the distance between $p$ & $l$;
    • if the distance is small enough {
      • count ++;
    }
  }
  • output line $l$ if the count is high enough;
}

Each pair of points defines a straight line
• If there is enough edge pixels on this line, then it is a detected line
• Complexity: $O(n^2)$
• There are $n^2$ pair of points
• For each pair we need to compute $n$ distances

Alternative Approach
• For each edge pixel:
  • Find all possible lines that pass through it
  • Plot the parameters of these lines in a parameter space
  • Find the intersections in the parameter space
  • Use slope-intercept parameter space:
    • The parameters of all lines that pass a give pixel form a line

Algebraic form of line:
• $Ax + By + C = 0$
• Slope:
  • $m = -A/B$
  • $b = -C/B$
  • $y = mx + b$
Slope-Intercept Parameter Space

<table>
<thead>
<tr>
<th>Y</th>
<th>X</th>
<th>Line equation</th>
<th>Parameter equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>3=m×1+b</td>
<td>b=-m+3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2=m×2+b</td>
<td>b=-2m+2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>5=m×2+b</td>
<td>b=-2m+5</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0=m×4+b</td>
<td>b=-4m</td>
</tr>
</tbody>
</table>

Find the Intersections

- 3 lines intersect at location (-1, 4) in the parameter space
- Suggest that there are 3 edge pixels on straight line: y = -x + 4
- Other intersections exist due to noisy edge pixels
- Problems:
  - Need \( w^2 \) intersection tests for \( w \) lines
  - The precision issue for numerical calculation

Quantize the Parameter Space

- Divide parameter space into accumulator cells:
  - Represent the space using a 2D array of counters
  - Initialize the counters to 0
  - For each line, find out all the cells it passes through and increase the counters for these cells by 1
  - The result can be visualized as a 2D histogram
- Need to be careful with the number of cells used:
  - Too many (resolution too fine):
    - The same intersection can be in different cells
  - Too few (resolution too coarse):
    - Different intersections may fall in the same cell

Limitations of Slope-Intercept Space

- The slope approaches infinity as the line gets vertical
- Impossible to create an a large enough 2D array
- Normal form of a line equation:
  \( x \cos \theta + y \sin \theta = \rho \)
  - \( \rho \): length of the perpendicular from the origin to the line
  - \( \theta \): angle from the x-axis to the perpendicular

Straight-line Hough Transform

- Create a counter array of size \( \sqrt{M^2 + N^2} \times K \) for the angle-distance space
  - \( M \) & \( N \): width & height of the image
  - \( \sqrt{M^2 + N^2} \) is the maximum distance between origin & any line in the image
  - \( K \): number of samples used for sampling the angle space
- For each edge pixel at \((x, y)\):
  - Vary \( \theta \) from 0 to \( \frac{(\theta - 3) \times \pi}{2} \)
  - Calculate \( \rho = x \cos \theta + y \sin \theta \)
  - Increase the counter at \((\rho, \theta)\) in the array
- Look for high values in the 2D counter array:
  - The indices of these values correspond to parameters of lines in the original image

Example of Straight-line Hough Transform
Circle Detection Problem

- Objective:
  - Find a circle from isolated edge pixels
- Circle equation:
  - $(x - a)^2 + (y - b)^2 = r^2$
  - 3 parameters: $a$, $b$, & $r$
- Detecting arbitrary circles requires too much calculation
  - Normally fix one of the parameters and vary the other two

Detecting Circles with Known Radius

- Assume the radius is known to be $R$
  - Only need to search for parameter $a$ & $b$
- For edge pixel $(x, y)$, we have:
  - $(x - a)^2 + (y - b)^2 = R^2$
  - Thus, the centers $(a, b)$ of all circles that pass $(x, y)$ lie on a circle in the parameter space

Circle Hough Transform for Circles with Known Radius

- Create a 2D counter array of size $M \times N$
- For each edge pixel whose coordinate is $(x, y)$:
  - Plot a circle with radius $R$ around point $(x, y)$
  - *Increase the values of the counters in the array that are passed by the circle*
- Look for the high values in the 2D counter arrays:
  - The locations of these values are the centers of possible circles with radius $R$ in the original image

Example of Circle Hough Transform