Line, edge, blob and corner detection

Dmitri Melnikov

MTAT.03.260 Pattern Recognition and Image Analysis

April 5, 2011
Introduction

2 Line detection

3 Edge detection

4 Blob detection

5 Corner detection
Introduction

- Detecting edges is fundamental in computer vision and image processing.
- Segmentation is a process that subdivides an image into its constituent regions or objects.
- Segmentation accuracy determines further results of image analysis procedures.
- Edge and line detection techniques are used as a base for more complicated segmentation algorithms.
Detection of discontinuities

- The idea is to find points in a digital image where the brightness changes sharply – has discontinuities.
- To look for discontinuities, run the mask through the image, computing the sum of products of mask coefficients with the gray levels under the mask.

<table>
<thead>
<tr>
<th>w₁</th>
<th>w₂</th>
<th>w₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>w₄</td>
<td>w₅</td>
<td>w₆</td>
</tr>
<tr>
<td>w₇</td>
<td>w₈</td>
<td>w₉</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Z₁</th>
<th>Z₂</th>
<th>Z₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z₄</td>
<td>Z₅</td>
<td>Z₆</td>
</tr>
<tr>
<td>Z₇</td>
<td>Z₈</td>
<td>Z₉</td>
</tr>
</tbody>
</table>

- The response of the mask at any point in the image is:

\[ R = w₁z₁ + w₂z₂ + \cdots + w₉z₉ = \sum_{i=9}^{9} w_i z_i \]
Let $R_1, R_2, R_3$ and $R_4$ denote the responses from these 4 masks.

If, after running the four masks through the image, $|R_i| > |R_j|, \forall j \neq i$ then the point is likely to be associated with line in the direction of mask $i$.

Alternatively, just use a single mask when looking for lines of one direction.
Edge detection

- Edge is a set of connected pixels that lie on the boundary between two regions.

- Edge detection is based on the measure of gray-level discontinuity at a point. Edge is a "local" concept.

- Unlike edges, a boundary forms a closed path. It is a "global" concept.
First derivative

- The first derivative is positive at the points of transition into and out of the ramp.
- It is constant for points in the ramp.
- It is zero in areas of constant gray levels.

The magnitude of the first derivative can be used to detect the presence of an edge at a point in an image.
The second derivative is positive at the transition associated with the dark side of the edge.

It is negative at the transition associated with the light side of the edge.

It is zero along the ramp and in the areas of constant gray level.

Second derivative produces two values for every edge in an image. Zero-crossing – midpoint of the imaginary straight line joining the extreme positive and negative values. Can be used for finding centers of thick edges.
We define a point in an image as being an edge point if its first-order derivative is greater than some threshold.

A set of such connected points is an edge.

First-order derivatives in an image are computed using the gradient.

Second-order derivatives in an image are computed using the Laplacian.
The gradient of an image $f(x, y)$ at point $(x, y)$ is defined as the vector

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

It points in the direction of maximum rate of change of $f$ at $(x, y)$. The magnitude of this vector is

$$\nabla f = \text{mag}(\nabla f) = \left[ G_x^2 + G_y^2 \right]^{\frac{1}{2}}$$

To simplify computation, it can be approximated as $\nabla f = |G_x| + |G_y|$

The direction angle of the gradient vector at $(x, y)$ is

$$\alpha(x, y) = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$
To obtain the partial derivatives, Roberts, Prewitt or Sobel operators can be used.

Roberts first-order derivative approximation at point $z_5$

$$G_x = (z_9 - z_5), \ G_y = (z_8 - z_6)$$

Can be implemented with $2 \times 2$ masks
### Prewitt operators

\[
\begin{array}{ccc}
  z_1 & z_2 & z_3 \\
  z_4 & z_5 & z_6 \\
  z_7 & z_8 & z_9 \\
\end{array}
\]

\[
G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)
\]

\[
G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_6)
\]

\[
\begin{array}{ccc}
  -1 & -1 & -1 \\
  0 & 0 & 0 \\
  1 & 1 & 1 \\
\end{array}
\quad
\begin{array}{ccc}
  -1 & 0 & 1 \\
  -1 & 0 & 1 \\
  -1 & 0 & 1 \\
\end{array}
\]

-1 -1 -1
0 0 0
1 1 1

-1 0 1
-1 0 1
-1 0 1
Sobel operators

\[
G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)
\]
\[
G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_6)
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
The Laplacian of $f(x,y)$ is a second-order derivative defined as

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

Two most common digital approximations in practice are

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$

$$\nabla^2 f = 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8 + z_9)$$

Laplacian is usually not used in its original form for edge detection. Very sensitive to noise.
Laplacian of a Gaussian (LoG)

Laplacian combined with smoothing can find edges using zero-crossing.

The Gaussian function blurs the image and $\sigma$ determines the degree of blurring.

$$h(r) = -\exp\left(-\frac{r^2}{2\sigma^2}\right), \text{ where } r^2 = x^2 + y^2$$

The Laplacian of $h$ (the second-order derivative of $h$ with respect to $r$) is

$$\nabla^2 h(r) = -\left[\frac{r^2 - \sigma^2}{\sigma^4}\right] \exp\left(-\frac{r^2}{2\sigma^2}\right)$$
Laplacian of a Gaussian (LoG)
Canny Edge Detection Algorithm

1 **Smoothing**: Image blurring to remove noise.

2 **Finding gradients**: Edges consist of points with large gradient magnitudes.

3 **Non-maximum suppression**: Only local maxima should be marked as edges.

4 **Double thresholding**: Potential edges are determined by thresholding.

5 **Edge tracking by hysteresis**: Final edges are determined by suppressing all weak edges that are not connected to strong edges.
Canny Edge Detection Algorithm

(a) Original

(b) Smoothed

(a) Gradient values

(b) Edges after non-maximum suppression
Canny Edge Detection Algorithm

(a) Double thresholding  (b) Edge tracking by hysteresis  (c) Final output
The most common blob detector is based on the Laplacian of the Gaussian (LoG).

However, using fixed Gaussian kernel, the response is dependent on the size of the blob.

The magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is ”matched” to the scale of the blob.
Blob detection - LoG

- Scale-normalized Laplacian operator \( \nabla^2_{\text{norm}} g = \sigma^2 \left( \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right) \)

- The 2D Laplacian is given by

\[
(x^2 + y^2 - 2\sigma^2) \exp \left( -\frac{(x^2 + y^2)}{2\sigma^2} \right)
\]

- For a binary circle of radius \( r \), the Laplacian is maximum at \( \sigma = \frac{r}{\sqrt{2}} \)
Blob detection - LoG

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales.
2. Find local maxima/minima of Laplacian response in scale-space.
3. Apply threshold.
Blob detection - DoH

Determinant of the Hessian

Like before, given \( L(x, y; t) = g(x, y, t) \ast f(x, y) \)

and the Hessian matrix \( HL(x, y; t) = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix} \)

we can consider the scale-normalized determinant of the Hessian

\[
detHL(x, y; t) = t^2(L_{xx}L_{yy} - L_{xy}^2)
\]

Looking for scale-space local maxima of this operator, we obtain the blob points.
Corner detection

- An interest point is a point that has a well-defined position and can be robustly detected. A corner is one such interest point.
- A corner is an intersection of two edges.
- In the region around a corner, image gradient has two or more dominant directions.
- Shifting a small window in any direction gives a large change in intensity at corner points.
Harris Corner detector

- Change of intensity for the shift \([x, y]\) is:

\[
S(x, y) = \sum_{u, v} w(u, v) [I(u + x, v + y) - I(u, v)]^2
\]

Where \(w\) is a window function.

Window function \(w(x, y) = \) 

- 1 in window, 0 outside
- Gaussian
Harris Corner detector

- \( I(u + x, v + y) \) can be approximated using Taylor expansion:

\[
I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y
\]

\[
S(x, y) = \sum_{u,v} w(u, v) [I_x(u, v)x + I_y(u, v)y]^2
\]

- Now, \( S(x, y) \) can be written in matrix form as

\[
S(x, y) \approx (x \ y) A \begin{pmatrix} x \\ y \end{pmatrix}, \text{ where}
\]

\[
A = \sum_{u,v} w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}
\]

- Let \( \lambda_1, \lambda_2 \) be the eigenvalues of matrix \( A \).
Harris Corner detector

We classify image points based on the values of $\lambda_1$ and $\lambda_2$ as follows

1. If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ then it is a flat region.
2. If $\lambda_1 \approx 0$ and $\lambda_2$ is large, then an edge is found.
3. If $\lambda_1$ and $\lambda_2$ are large, a corner is found.

However, eigenvalues are expensive to compute, so we use the function

$$R = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2 = det(A) - \alpha \text{trace}^2(A),$$

where $\alpha$ is around 0.05.
Harris Corner detector

Algorithm

1. Compute partial derivatives $I_x$ and $I_y$.

2. Compute the matrix $A$ in a window around each pixel.

3. Compute function $R$ at each pixel.

4. Find local maxima of $R$ (using non-maximum suppression).

5. Apply threshold.
- SUSAN - Smallest Univalue Segment Assimilating Nucleus.
- Method for edge and corner detection.
- No image derivatives used.
- Not sensitive to noise.
part of original image

find USAN area for each image position

USAN area / pixels
1. Mask $M$ is placed around the center pixel. Usually, $|M| = 37$.

2. Calculate the USAN area – pixels within the mask which have similar brightness to the nucleus.

   $$c(m) = \exp \left( \frac{- (I(m) - I(m_0))^6}{t} \right), \forall m \in M$$

   $$n(M) = \sum_{m \in M} c(m)$$

3. The response of the SUSAN operator is

   $$R(M) = \begin{cases} g - n(M) & \text{if } n(M) < g \\ 0 & \text{otherwise} \end{cases}, \text{ where } g = \frac{n_{\text{max}}}{2}$$

4. Test for false positives by finding the USAN’s centroid.