Introduction to Computational and Biological Vision

CS 202-1-5261

Computer Science Department, BGU

Ohad Ben-Shahar
Drawing skills 101
Edge detection

Why look for edges? Physical causes for edges

Depth (boundary) discontinuities
Edge detection

Why look for edges? Physical causes for edges

Surface (normal) discontinuities
Edge detection

Why look for edges? Physical causes for edges

Illumination discontinuities
Edge detection

Why look for edges? Physical causes for edges

Reflectance discontinuities
**Edge detection**

A (desired) edge detection mapping (labeling)

\[ E[I(x, y)]: R^2 \rightarrow \Lambda = \{0, 1, 2, 3, 4, 5, \ldots \} \]

\[ E[I(x, y)] = \begin{cases} 
1 & (x,y) \text{ is a boundary point} \\
2 & (x,y) \text{ is a surface discontinuity point} \\
3 & (x,y) \text{ is an illumination discontinuity point} \\
4 & (x,y) \text{ is a reflectance discontinuity point} \\
\vdots & \\
0 & \text{Otherwise (non edge point)} 
\end{cases} \]

Q1: Is there enough information in the image to implement \( E[\cdot] \)?

Q2: Is \( E[\cdot] \) all we need to know about edge point?

Q3: Can we really consider edge point in isolation (i.e., individually)?
Edge detection

Edge appearance in images
**Edge detection**

**Edge appearance in images**

- **Step edge**
- **Ramp edge**
- **Bright line**
- **Dark line**

Orthogonal cross section $(x_0, y_0)$
**Edge detection**

Step edge localization and detection – continuous signals

**Edge detection heuristic #1**

Given $I(x)$, its edge points occur at the (local) maxima of $|I'(x)|$
**Edge detection**

Step edge localization and detection – continuous signals

Edge detection heuristic #2

Given \( I(x) \), its edge points occur at the zero crossing of \( I''(x) \)
**Edge detection**

Step edge localization and detection – discrete signals

\[
I'(x) \approx \frac{I(x+h) - I(x)}{h} + o(h)
\]

\[
I'(x) = \frac{I(x+h) - I(x-h)}{2h} + o(h^2)
\]

\[
I'(x) = \frac{I(x-2h) - 8I(x-h) + 8I(x+h) - I(x+2h)}{12h} + o(h^4)
\]
Edge detection

Step edge localization and detection – discrete signals

\[ I''(x) = \frac{I(x-h) - 2I(x) + I(x+h)}{h^2} + o(h^2) \]

\[ I''(x) = \frac{-I(x-2h) + 16I(x-h) - 30I(x) + 16I(x+h) - I(x+2h)}{12h^2} + o(h^4) \]

Edge detection via (central) differences operation is an instance of convolution!!
Edge detection

Step edge localization and detection – discrete signals

![Graphs showing edge detection and second-order central differences](image)
Edge detection

Step edge localization and detection – discrete signals

![Graphs showing edge detection and central differences](image-url)
Edge detection

Step edge localization and detection – discrete signals
Edge detection

Step edge localization and detection – discrete signals

\[ I(x) \]

\[ I'(x) \]
Edge detection

Step edge localization and detection – discrete signals
**Edge detection**

Gradient-based edge detection in images

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

**Proposition:**

The gradient vector points in the direction where the image changes "the most".

**Edge detection heuristic #3**

Given \( I(x,y) \), its edge points occur at local (directional) maxima of \( |\nabla I| \)
**Edge detection**

Gradient-based edge detection in images

\[
|\nabla I| = L_2[\nabla I] = \sqrt{I_x^2 + I_y^2}
\]

\[
|\nabla I| = L_1[\nabla I] = |I_x| + |I_y|
\]

\[
I_x \approx \frac{I(x+1, y) - f(x-1, y)}{2}
\]

\[
I_y \approx \frac{I(x, y+1) - f(x, y-1)}{2}
\]
Edge detection

Gradient-based edge detection in images

\[ \theta(x, y) = \tan^{-1}\left( \frac{I_y}{I_x} \right) \]

\[ \theta^\perp(x, y) = \tan^{-1}\left( -\frac{I_x}{I_y} \right) \]
**Edge detection**

Laplacian-based edge detection in images

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

**Edge detection heuristic #4**

Given \( I(x,y) \), its edge points occur at the zero crossing of \( \Delta I \)
**Edge detection**

Laplacian-based edge detection in images

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

\[
\frac{\partial^2 I}{\partial x^2} \approx \frac{I(x-h, y) - 2I(x, y) + I(x+h, y)}{h^2}
\]

\[
\frac{\partial^2 I}{\partial y^2} \approx \frac{I(x, y-h) - 2I(x, y) + I(x, y+h)}{h^2}
\]

\[
\nabla^2 I \approx \frac{I(x-h, y) + I(x, y-h) - 4I(x, y) + I(x+h, y) + I(x, y+h)}{h^2}
\]
Edge detection

Detecting 2D edge-like structures

\[ I(x, y) \quad |I_x| \quad |I_y| \quad |\nabla I| \]
Edge detection

Detecting 2D edge-like structures

\[ I(x, y) = |I_x| \quad |I_y| \quad |\nabla I| \]
Edge detection

Detecting 2D edge-like structures

$I(x, y)$

$\nabla^2 I$

$ZC[\nabla^2 I]$
Edge detection

Detecting 2D edge-like structures

$I(x, y)$

$\nabla^2 I$

$ZC[\nabla^2 I]$
Edge detection

Detecting 2D edge-like structures

$I(x, y)$  \hspace{1cm} $\nabla^2 I$  \hspace{1cm} $ZC[\nabla^2 I]$
Edge detection

Signal vs. Noise

Noise:

Irrelevant or meaningless data that is mixed into the signal, usually due to imperfect performance of the measurement device.

Basic noise models:

• Additive \( \tilde{I}(x) = I(x) + \eta(x) \)

• Salt & Pepper
  \[
  \tilde{I}(x) = \begin{cases} 
  I(x) & \text{with probability } 1-p \\
  \min & \text{with probability } p/2 \\
  \max & \text{with probability } p/2 
  \end{cases}
  \]
**Edge detection**

**Signal vs. Noise**

- $\sigma = 0.001$
- $\sigma = 0.003$
- $\sigma = 0.005$
- $\sigma = 0.007$
- $p = 0.05$
- $p = 0.15$
- $p = 0.25$
- $p = 0.35$
Edge detection

Detecting 2D edge-like structures

$I(x, y)$

$\left| I_x \right|$

$\left| I_y \right|$

$\nabla I$

$\nabla^2 I$

$ZC[\nabla^2 I]$
Edge detection

Detecting 2D edge-like structures

$I(x, y)$

$|I_x|$  

$|I_y|$  

$\nabla^2 I$  

$ZC[\nabla^2 I]$
Edge detection

Image denoising

Signal changes slowly (most of the time)

Noise changes fast, zero mean

What can reduce noise without affecting signal too much?

Smoothing

Averaging

Weighted sum

Convolution
Edge detection

Average-based image denoising
Edge detection

Average-based image denoising

Original  Window size 5  Window size 9  Window size 17
Edge detection

Detecting 2D edge-like structures

$I(x, y)$

$|\nabla I|$

$ZC[\nabla^2 I]$
Edge detection
Detecting 2D edge-like structures

With smoothing vs. Without smoothing

$I(x, y)$
$|\nabla I|$
Edge detection

Combining smoothing and differentiation for edge detection

\[
( I \ast K_s ) \ast K_d
\]

\[
I \ast ( K_s \ast K_d )
\]
Edge detection

Combining smoothing and differentiation for edge detection

\[
(I \ast (K_{sd})) \ast K_d
\]

The derivative of convolution theorem

\[
\frac{\partial}{\partial x_i} (g \ast h) = g \ast \frac{\partial}{\partial x_i} h = \frac{\partial}{\partial x_i} g \ast h
\]
**Edge detection**

**Sobel edge detector**

\[
K_x = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \ast \begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}
\]

\[
K_y = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \ast \begin{bmatrix} -1 & -1 \\ +1 & +1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}
\]

\[
I_x = I \ast K_x
\]

\[
I_y = I \ast K_y
\]

\[
|\nabla I| = \sqrt{I_x^2 + I_y^2}
\]
Edge detection

Sobel edge detector – results on noisy images
Edge detection

Sobel edge detector – results on noisy images
**Edge detection**

**Gaussian-based smoothing**

\[
G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}
\]

\[
G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}
\]
**Edge detection**

**Gaussian-based smoothing**

Why Gaussian (as opposed to, say, simple uniform averaging in a window)?

1. Contribution of neighbors weighted by distance
2. Isotropic (doesn’t prefer one direction over the other)
3. Symmetric about the origin.
4. Infinitely differentiable (as smooth as it gets)
5. Fourier transform of a Gaussian is another Gaussian (in the frequency domain)
6. Convolution of two Gaussians is a Gaussian: $G_{\sigma_1} \ast G_{\sigma_2} = G \sqrt{g_1^2 + g_2^2}$
7. Gaussians are separable: $G_\sigma(x, y) = G_\sigma(x) \cdot G_\sigma(y)$
8. $N^{th}$ derivative of a Gaussian can be approximated by an appropriately subtracting two $(N-1)^{th}$ derivatives.
**Edge detection**

Gaussian-based smoothing

\[ \mathcal{F}[G_\sigma(x)] = C_\sigma \cdot G_{1/\sigma}(\omega) \]

**Linear systems and filtering**

The convolution theorem/property

\[ x(t) \xrightarrow{\mathcal{F}} X(\omega) \quad x(t) * h(t) \xrightarrow{\mathcal{F}} X(\omega) \cdot H(\omega) \quad h(t) \xrightarrow{\mathcal{F}} H(\omega) \]

**Linear systems and filtering**

Linear filtering and the Modulation Transfer Function (MTF)

- Low pass filter
- Band pass filter
- High pass filter
Edge detection

Gaussian-based smoothing

\[ \mathcal{F} \left[ \delta_{\Delta} \left( x - \frac{\Delta}{2} \right) \right] = \text{sinc} (\Delta \cdot \omega) = \frac{\sin (\Delta \cdot \omega)}{\Delta \cdot \omega} \]

The convolution theorem/property

\[ \mathcal{F}[G_\sigma(x)] = C_\sigma \cdot G_{1/\sigma}(\omega) \]

Linear systems and filtering

The convolution theorem/property

\[ \mathcal{F} \left[ \frac{\sin(\omega \cdot \Delta)}{\omega \cdot \Delta} \right] = \frac{\sin (\omega \cdot \Delta)}{\omega \cdot \Delta} \]

Linear filtering and the Modulation Transfer Function (MTF)

Low pass filter

Band pass filter

High pass filter
Edge detection

Gaussian-based smoothing

\[
\frac{d^N}{dx^N} G_\sigma(x) \approx \sum_i w_i^N \cdot \frac{d^{N-1}}{dx^{N-1}} G_\sigma(x - t_i^N) \approx \ldots \approx \sum_i w_i^1 \cdot G_\sigma(x - t_i^1)
\]
Edge detection

Laplacian of Gaussian (LOG) and Lateral Inhibition
**Edge detection**

Marr-Hildreth edge detector (1980)

\[ MH(I) = R_{|\nabla I|}(ZC[\nabla^2 (I * G_\sigma)]) \]
**Edge detection**

Marr-Hildreth edge detector (1980)

\[
MH(I) = R_{|\nabla I|}(ZC[ (I \ast G_\sigma) \ast L])
\]
**Edge detection**

Marr-Hildreth edge detector (1980)

\[
MH(I) = R_{\|\nabla I\|}(ZC[I \ast (L \ast G_{\sigma})])
\]
Marr-Hildreth edge detector (1980)

\[ MH(I) = R_{|\nabla I|}(ZC[I * (\text{LOG}_{\sigma})]) \]
**Edge detection**

Marr-Hildreth edge detector (1980)

\[
MH(I) = R_{\nabla I} \left( ZC[I \ast LOG_\sigma] \right)
\]

\[
LOG_\sigma(x, y) = a \left( 2 - \frac{x^2 + y^2}{\sigma^2} \right) e^{-\frac{(x^2+y^2)}{2\sigma^2}}
\]

The Mexican Hat Operator
Edge detection

Marr-Hildreth edge detector (1980)

\[ * G_{\sigma_1} \quad * G_{\sigma_2} \quad * G_{\sigma_{n-1}} \quad * G_{\sigma_n} \]
Edge detection

Marr-Hildreth edge detector (1980)

Larger $\sigma$

Larger threshold
Edge detection

Marr-Hildreth edge detector (1980)
Edge detection

Marr-Hildreth edge detector (1980)
Edge detection

Marr-Hildreth edge detector (1980)
Edge detection

Marr-Hildreth edge detector (1980)
Edge detection

Marr-Hildreth edge detector (1980)
Edge detection

Marr-Hildreth edge detector (1980)
**Edge detection**

The Canny edge detector (1986)

Main steps:

1. Smooth the image with a Gaussian and estimate gradient direction and magnitude.
2. Thin filter response via nonmaxima suppression in the direction of the gradient.
3. Trace and link edge points to curves using hysteresis
Edge detection

The Canny edge detector

Nonmaxima-suppression in gradient direction

For each pixel $p$:

1. Determine immediate grid neighbors in gradient direction (use interpolation if needed)

2. Unless $|\nabla (I * G_\sigma)|_p$ is greater than both neighbors, reset its magnitude to zero
**Edge detection**

The Canny edge detector

**Edge curve construction via threshold hysteresis**

1. Define two thresholds $\tau_1 < \tau_2$
2. Threshold magnitude response by $\tau_1$
3. For each unlabeled pixel $p$ with gradient magnitude $\geq \tau_2$
   - Label $p$ as an edge point
   - Check for an immediate unlabeled neighbor $q$ of $p$ with gradient magnitude $\geq \tau_1$
     - Set $p=q$
4. End
5. Repeat until no more unlabeled pixels
The Canny edge detector

Clean:

Additive noise:

Salt and Pepper noise:

\( \tau_2 = 0.1 \quad \tau_2 = 0.3 \quad \tau_2 = 0.5 \quad \tau_2 = 0.7 \)
Edge detection

The Canny edge detector

Clean:

Additive noise:

Salt and Pepper noise:

\[ \tau_2 = 0.1 \]  \[ \tau_2 = 0.3 \]  \[ \tau_2 = 0.5 \]  \[ \tau_2 = 0.7 \]
Edge detection

Canny (Clean)

Canny (Noisy)
Edge detection

Canny (Clean)

Canny (Noisy)
Desired: $E[I(x, y)]: R^2 \rightarrow \Lambda = \{0, 1, 2, 3, 4, 5, \ldots \}$

Given these edge detection results, is edge detection solved?

Is it the right information/features to look for in the first place?
**Edge detection**

Edge detection performance measures

Performance criteria:

- Probability of false positives
- Probability of misses
- Error in spatial localization (e.g., in distance units)
- Error in angular specificity
- Tolerance to noise
- Ability to cope with non-curve-like edge features (corners, junctions, …)
Edge detection

Edge detector performance measures

Example: Ability to cope with non curve-like edge features (corners, junctions, …)
Edge detection

Edge detector performance measures

Example: Ability to cope with non curve-like edge features (corners, junctions, …)

Logical/Linear edge detector (1995)
**Edge detection**

Edge detector performance measures

Pratt’s Figure of Merit (1991)

\[
FM = \frac{1}{\max(I_D, I_I)} \sum_{i=1}^{I_D} \frac{1}{1 + d_i \alpha^2}
\]

\( I_D \) — Number of detected edge points

\( I_I \) — Number of ideal edge points

\( d_i \) — Distance of detected edge point \( i \) from its ideal counterpart

\( \alpha \) — Penalty constant for displaced edge points
Edge detection

Edge detector performance measures

Precision/Recall curves

Precision ($p$) = \% detected points that are correct

Recall ($p$) = \% of correct points that are detected

Ground truth

“Soft” detected map (Canny)
**Edge detection**

Edge detector performance measures

Precision/Recall curves

Precision: $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$

Recall: $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$

$F(p) = \frac{2}{\frac{1}{\text{Precision}(p)} + \frac{1}{\text{Recall}(p)}} = 2 \cdot \frac{\text{Precision}(p) \cdot \text{Recall}(p)}{\text{Precision}(p) + \text{Recall}(p)}$

$AUC = \int_0^1 \text{Recall}(\text{Precision}) \, d\text{Precision}$
Edge detection

Edge detector performance measures

<table>
<thead>
<tr>
<th>Method</th>
<th>F-measure</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>.79</td>
<td>-</td>
</tr>
<tr>
<td>Canny</td>
<td>.49</td>
<td>20</td>
</tr>
<tr>
<td>Sobel</td>
<td>.40</td>
<td>100</td>
</tr>
<tr>
<td>BEL</td>
<td>.63</td>
<td>1/50</td>
</tr>
<tr>
<td>gPb</td>
<td>.71</td>
<td>1/720</td>
</tr>
<tr>
<td>SketchToken</td>
<td>.73</td>
<td>1.5</td>
</tr>
<tr>
<td>StructuredForest</td>
<td>.71</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 1. The F-measure score and the speed comparison of edge detectors.

Figure 1. The precision and recall curve of edge detectors.
**Edge detection**

A final note about *denoising*

Q: Can we do better than local filtering in the image plane?
Edge detection

“Patch collaboration” – Block Matching and 3D filtering (BM3D; 2007)

1 - Match image blocks to reference patch
2 - Collect matched blocks
3 - Consider as a 3D signal
4 - 3D transform
5 - 3D filtering
6 - 3D inverse filtering
7 - Block-wise estimate
8 - Aggregate and restore
Edge detection

Estimate Aggregation in BM3D

\[ I(p) = \sum_{b \in \text{block}} w_b \cdot I_b (p) \]
Edge detection

BM3D Results
Edge detection

BM3D Results