Features

Feature types:

1) global : e.g. color distribution (histograms)

2) local : e.g. edges or corners

Goal:

characterize images
and/or
local neighborhoods

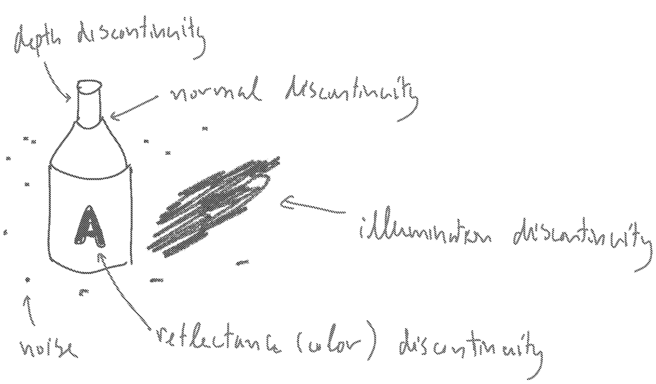
Edge detection

Requirements:

- correspond to scene elements
- invariant (illumination, pose, viewpoint, scale)
- reliable detection

Edge detection

edge = location with change in image



Edge detection steps

1) smooth to reduce noise (without affecting edges)
2) Enhance edges
3) Detect edges
4) Localize edges

Image gradient

Image: $I(x,y)$

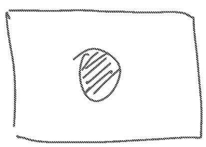
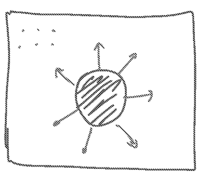


Image gradient:

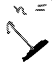
$$\nabla I(x,y) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$



$$= \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

Edge detection using gradients

"edgle" = edge element



magnitude: $|\nabla I| = \sqrt{I_x^2 + I_y^2}$

angle : $\theta = \tan^{-1}\left(\frac{I_y}{I_x}\right)$

* Edge detection:

$$E(i,j) = \begin{cases} 1 & \text{if } |\nabla I(i,j)| > \tau \\ 0 & \text{otherwise} \end{cases}$$

edge map ↗

Forward differences

$$\frac{\partial}{\partial x} I(x,y) = \frac{I(x+h,y) - I(x,y)}{h} \qquad h=1$$

$$= I(x+1,y) - I(x,y)$$

$$\Delta_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

Central differences

$$\frac{\partial}{\partial I} I(x,y) = \frac{I(x+h,y) - I(x-h,y)}{2h}$$

$$= I(x+1,y) - I(x-1,y)$$

$$\Delta_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

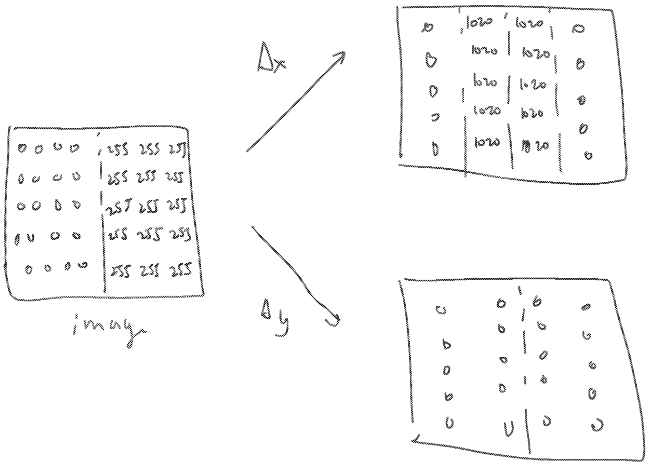
Sobel filter

Smooth and then take derivative

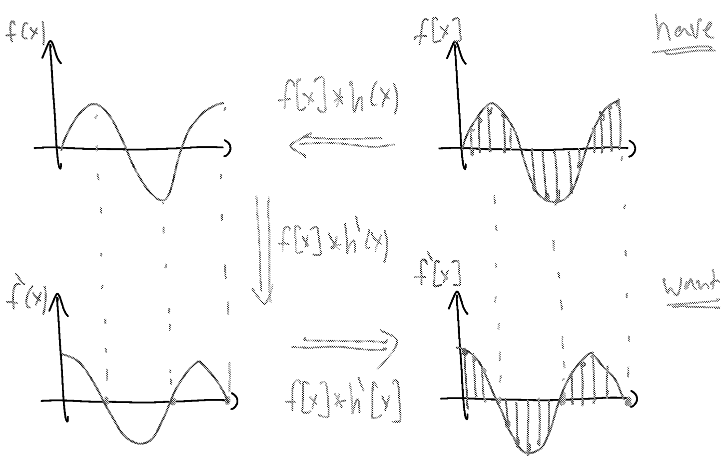
$$\Delta_x = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\Delta_y = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Example



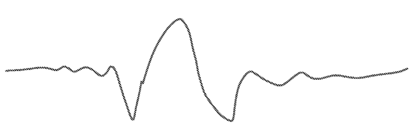
More accurate derivatives



Perfect interpolator

Perfect interpolator: under certain conditions (nyquist) perfect reconstruction. $f_s \geq 2 f_m$

$$h(x) = \text{sinc}(x) = \frac{\sin(\pi x / T)}{(\pi x / T)}$$



infinite tails

approximate $h(x)$ with $G(x)$

Accurate derivatives

$I_x = I * G_x$ $\qquad f[x] * G[x]$

$I_y = I * G_y$

Using separable property of Gaussians:

1D convolution with horizontal Gaussian derivative

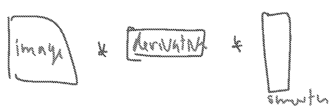
$I_x = I * G'[x] * G[y]$ ← 1D convolution with vertical Gaussian

$I_y = I * G'[y] * G[x]$ ← 1D convolution with horizontal Gaussian

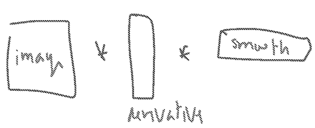
1D convolution with vertical Gaussian derivative

Accurate derivatives

$I_x = I * G'[x] * G[y]$

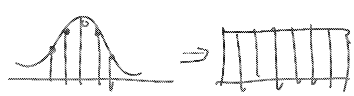


$I_y = I * G'[y] * G[x]$

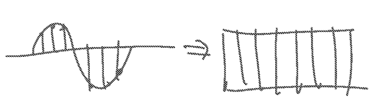


Gaussian derivatives

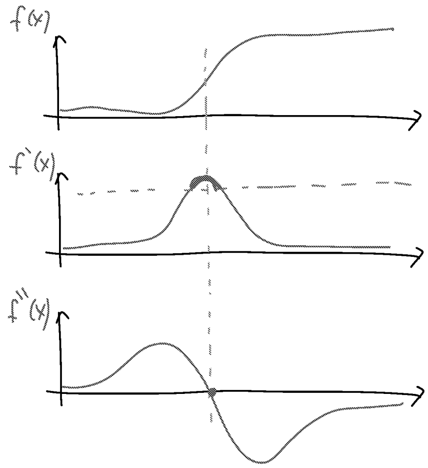
$$G(x) = e^{-\frac{x^2}{2\sigma^2}}$$



$$G'(x) = -\frac{\cancel{2}x}{\cancel{2}\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$



Laplacian of Gaussian



edge

edge by threshold

edge by zero crossing

2D second derivative as scalar quantity

Laplacian: $\Delta I = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = I_{xx} + I_{yy}$

$I_x = I(x+1) - I(x)$

$I_{xx} = (I(x+1) - I(x)) - (I(x) - I(x-1))$

$= I(x+1) - 2I(x) + I(x-1) \Rightarrow \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$

$I_{yy} = \cdots \Rightarrow \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$

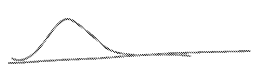
$I_{xx} + I_{yy} \Rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

Laplacian of Gaussian (LOG)

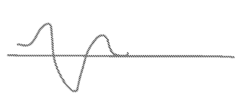
Smooth with a Gaussian before applying Laplacian

$H = \nabla^2 (I * G) = \underbrace{\nabla^2 G}_{\text{LoG}} * I$

$G = e^{-\frac{r^2}{2\sigma^2}} \quad (r^2 = x^2 + y^2)$



$\nabla^2 G = \frac{r^2 - 2\sigma^2}{\sigma^4} e^{-\frac{r^2}{2\sigma^2}}$



Edge detection using LOG

1) Compute LOG: $H = (\nabla^2 G) * I$

2) Threshold : $E(i,j) = \begin{cases} 0 & \text{if } H(i,j) < 0 \\ 1 & \text{if } H(i,j) \geq 0 \end{cases}$

3) Mark edges at transitions $0 \to 1$, $1 \to 0$

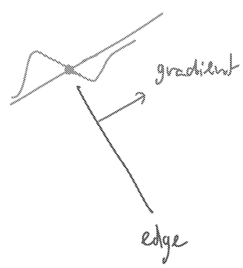
   (scan left-to-right and top-to-bottom)

Canny edge detection

Detect edges at zero crossings of second order directional derivative taken along the gradient

n = gradient

if $|n| > \tau$ detect edges at zero crossing of $\frac{\partial^2}{\partial n^2}(I * G)$



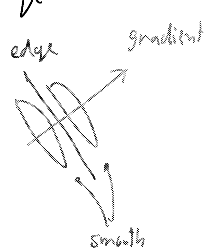
Canny edge detection

Comments:

1) attempt detection only if gradient magnitude is large enough ($|n| > \tau$)

2) smooth along edges to preserve edge

3) Alternative to zero crossing of $\frac{\partial^2}{\partial n^2}(I * G)$ is maximum of $\frac{\partial}{\partial n}(I * G)$



Directional derivative expression

gradient: $n = \nabla(I * G)$

directional derivative: $\frac{\partial}{\partial n}(I * G) \equiv n \cdot \nabla(I * G)$

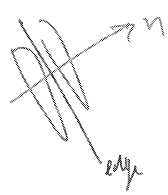
$$n \cdot \nabla(I * G) = \nabla(I * G) \cdot \nabla(I * G)$$
$$= |\nabla(I * G)|^2$$

Canny summary

If $|\nabla(I * G)| > \tau$ set edge at maximum of $|\nabla(I * G)|$ along the direction of $\nabla(I * G)$

Additional components:

1) non-maximum suppression
2) hysteresis thresholding



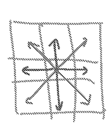
Non-maximum suppression

Need: local maximum of gradient magnitude in direction of gradient

$\nabla(I * G) \equiv (I_x, I_y)$

$\theta = tg^{-1}\left(\frac{I_y}{I_x}\right)$

$\theta^* = round\left(\frac{\theta}{45}\right) * 45$

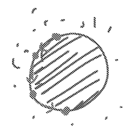


Compare neighbors after discretization

$$E(i,j) = \begin{cases} 1 & \text{if } \nabla(I * G) \text{ is a local maximum} \\ 0 & \text{otherwise} \end{cases}$$

Hysteresis thresholding

Use τ_H to start tracking and τ_L to continue ($\tau_H > \tau_L$)



1) Initialize array of visited pixels
   $v(i,j) = 0$
2) Scan image T-B, L-R:
   if $!v(i,j)$ && $|\nabla I| > \tau_H$ start tracking an edge
3) Search for additional neighbors in direction orthogonal to ∇I such that $|\nabla I| > \tau_L$

Summary edge detection

* Edge detection requires finding discrete image derivatives:
  - forward difference
  - central difference
  - Sobel
  - Gaussian derivatives

  first order derivatives

  - Zero crossing of LoG (2nd order)
  - Directional derivatives (Canny)