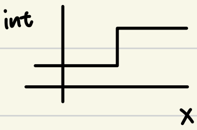
edge detection
1st step from low-level to medium level
spatial 2D filter:

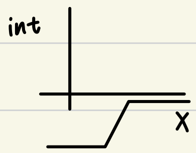
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

horizontal sobel vertical 45° edge

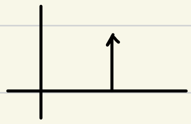
ideal edge:

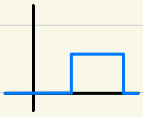


ramp edge:

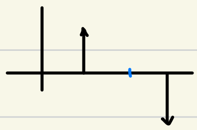


1st derivative: look for big absolute value

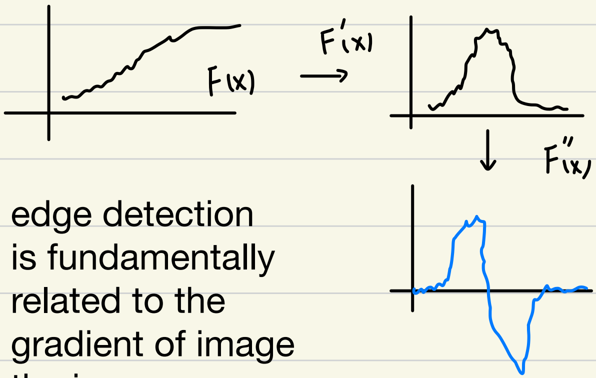




2nd derivative: look for sign change (aka zero crossing

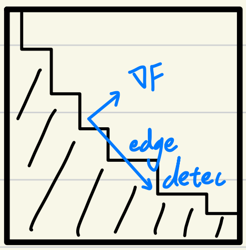


real world



edge detection is fundamentally related to the gradient of image the image:

$$\nabla F = \text{grad } F = \begin{bmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \approx \begin{bmatrix} F(x+1,y) - F(x,y) \\ F(x,y+1) - F(x,y) \end{bmatrix}$$



∇F is perpendicular to edge

$$M(x,y) = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2} = \sqrt{g_x^2 + g_y^2}$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{\partial F}{\partial y} \Big/ \frac{\partial F}{\partial x}\right)$$

we can do things like:

$M(x,y) > T$

$|\alpha(x,y) - \theta| < T_1$ and $M(x,y) > T_2$

laplacian of Gaussian, aka Marr-Hildneth or Mexican Hat Filter

an edge detector that be tuned to edges at different scales:
big operators: large-scale/blurry edges
small operators: small-scale/fine detail

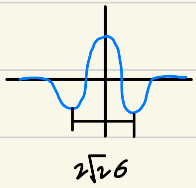
$G(x,y)$ Gaussian Filter

$= e^{-\frac{x^2+y^2}{2\sigma^2}}$

$$\nabla^2 G(x,y) = \frac{\partial^2}{\partial x^2} G(x,y) + \frac{\partial^2}{\partial y^2} G(x,y)$$

$$= \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

the negative of this function looks like:



idea: - Gaussian smooths the image down to a certain scale
- Laplacian finds edges at that scale
- look for zero-crossings of log operation (since it's like a 2nd derivative)

$$g(x,y) = [\nabla^2 G(x,y)] * F(x,y) = \nabla^2 [G(x,y) * F(x,y)]$$

could also find the scale σ at which a given edge is not significant; filter on that
We can approximate the log with a difference of Gaussian (dog)

$$dog(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{(x^2+y^2)}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{(x^2+y^2)}{2\sigma_2^2}} \qquad \sigma_1 > \sigma_2$$

used a lot in computer vision, sift

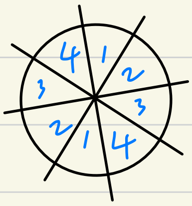
Canny edge detector

basic steps: 1) smooth image with Gaussian filter

2) compute $M(x,y)$, $\alpha(x,y)$

3) apply non-maxima suppression to $M(x,y)$ to obtain a new gradient image $g_N(x,y)$

idea: quantize $\alpha(x,y)$ into 4 bins



17	15	30
10	25	22
27	15	26

$M(x,y)$

2	1	2
2	1	1
2	2	1

$\alpha(x,y)$

if $M(x,y)$ is greater than both its neighbour in the quantized edge detection,

$g_N(x,y) = M(x,y)$, o/w $g_N(x,y) = 0$

4) detect and link edges

$g_H(x,y) = g_N(x,y) \geq T_H$ strong edges

$g_L(x,y) = g_N(x,y) \leq T_L$ weak edges

Final Map: $g_H(x,y) \cup$ all edge pixels in $g_L(x,y)$ that are adjacent to at least one pixel of $g_H(x,y)$

$T_H \approx 2\times$ or $3\times$ T_L