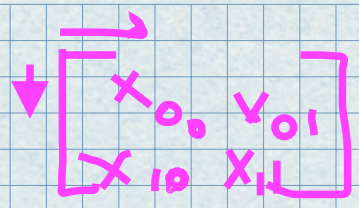
Görüntü İşleme Eğitimi



Blurs, contrast changes, color changes

Gaussian blurs, edge detection

but in the end It's all filters.

Convolution

$$y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t-\tau)\, d\tau$$

BLURS

Firstly, we will use KERNEL CONVOLUTION & that is the kind of the core of Gaussian blurs, mean blurs and edge detections.

Simple Technique

Kernel Conv. is just a process where we take a small grid of numbers and we pass them over the whole image transforming it based on what those numbers are.

0 ——— 255

KERNEL

x_{00}	x_{01}	x_{02}
x_{10}	x_{11}	x_{12}
x_{20}	x_{21}	x_{22}

3x3 Kernel size

IMG

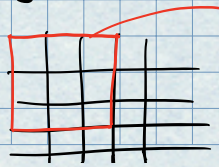
14	15	5	2	6
5	8	8	3	2
21	6	5	128	52
255	25	0	1	5
52	48	78	65	110

1) Mean Blur

* if all x are 1

So it is a mean blur.

How it works

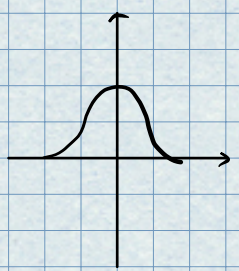


convolution and divide by matrix size } then it will go to a new image or it will blow our code

| | | |

* It can help the remove noises.

2) Gaussian Blur



σ (deviation)
(standart sapma)

Example

1	2	1
2	4	2
1	2	1

Image (Edge)

50	50	100	100	
50	50	100	100	
50	50	100	100	
50	50	100	100	
50	50	100	100	
50	50	100	100	

Mean Blur Kernel

50	50	100
50	50	100
50	50	100

$\frac{600}{9} = 66,6$

$\simeq 67$

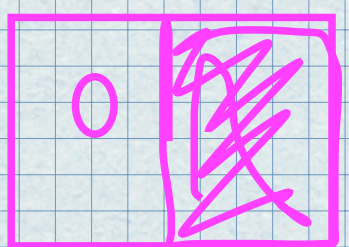
Gaussian Blur Kernel

50	100	100
100	200	200
50	100	100

$\frac{1000}{16} = 62,5$

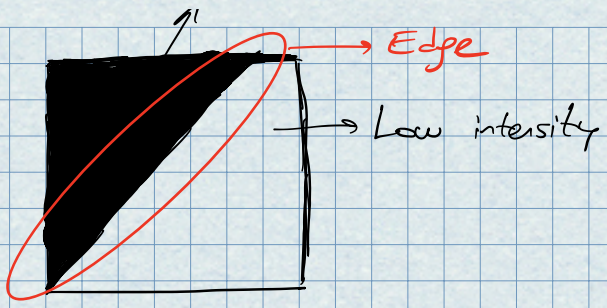
$\simeq 63$

1+2+1+2+4+2+1+2+1=16



SOBEL EDGE DETECTOR

High intensity



x direction kernel

$G_x =$

-1	0	1
-2	0	2
-1	0	1

50	50	100	100	
50	50	100	100	
50	50	100	100	
50	50	100	100	
50	50	100	100	
50	50	100	100	

G_x →

-50	0	100
-100	0	200
-50	0	100

$= 200$

G_y ↓

$G_y =$

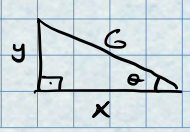
-1	-2	-1
0	0	0
1	2	1

50	100	100
0	0	0
-50	-100	-100

$= 0$

$$G = \underbrace{\sqrt{G_x^2 + G_y^2}}_{\text{Pythagoras theorem}}$$

$$\arctan\left(\frac{G_y}{G_x}\right) = \theta$$



But it is unusal in color. In the many areas it is a gray scale operator.

It can found so many unnecessary edges with technique so you can use gaussian blur first and then you can use sobel edge and the result will be better.

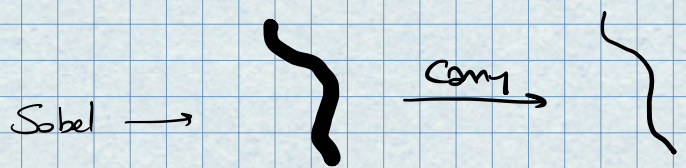
Image → grayscale → Gaussian blur → sobel edge → edge detected image

CANNY EDGE DETECTOR

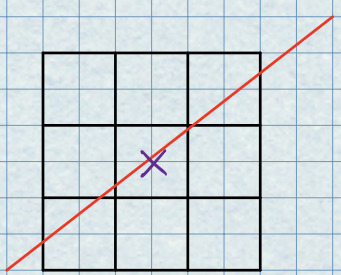
Published in 1986

It is using sobel but it has one more step.

Actually we don't want the thickness of our edges we want to know their directions.



canny calculates which neighbor pixel is greater than other with θ angle

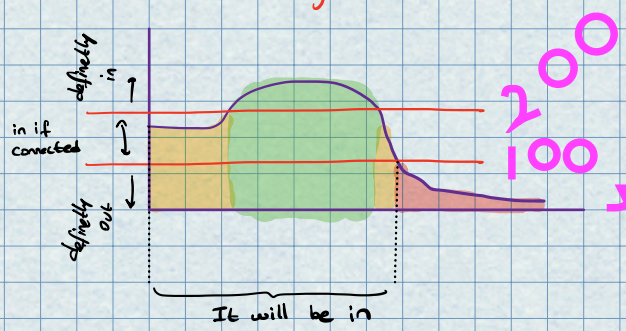


Hysterisis thresholding

If threshold is low than all edges and noises still be there

" " " high " we will lose some edges

One dimensional image



OpenCV → Java, C++, Python

RESIZING THE IMAGES (Bicubic, Nearest neighbour, Bilinear)

3×3 image

120	121	115
100	110	80
100	80	60

x →

y ↓

resize it →

8×8 nearest neighbour

120	120	121	121	121	115	115	115	115
100	100		110			80		
100			80			60		

First guess ?

$x \rightarrow 3x$

$y \rightarrow 3x$

IT IS REALLY BAD!

DON'T DO IT

Bilinear

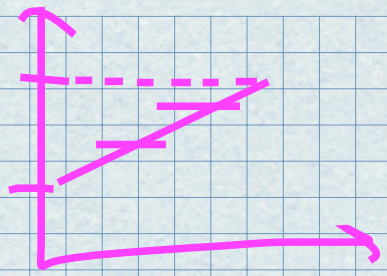
0 → 120

1/3

2/3

1 → 121

Ex



100 133 167 200

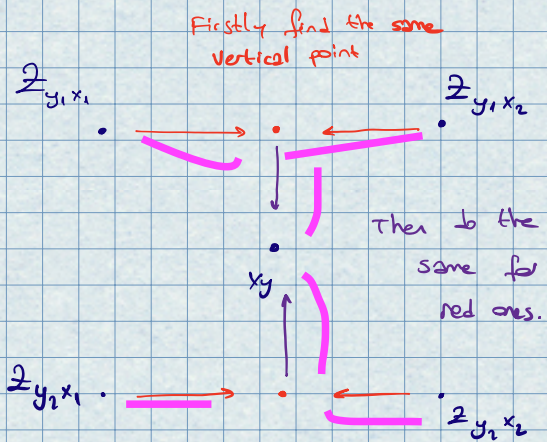
When we calculate x_{01}

$$x_{01} = \text{distance}\left(1 - \frac{|x_{01} - x_{00}|}{x_{03} - x_{00}}\right) \cdot x_{00} + \text{distance}\left(1 - \frac{|x_{01} - x_{03}|}{x_{03} - x_{00}}\right) \cdot x_{03}$$

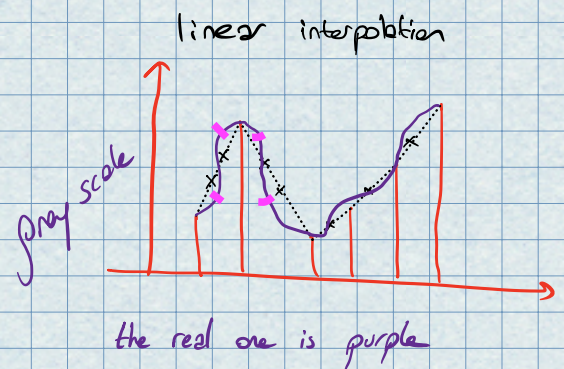
$$= \left(1 - \frac{1}{3}\right) \cdot 100 + \left(1 - \frac{2}{3}\right) \cdot 200$$

$$= \frac{200}{3} + \frac{200}{3} = \frac{400}{3} \simeq \frac{133}{5}$$

For example



So this is bilinear



x_0 is known
x_1 is known
} so we can use third degree polynomial. This is called called cubic interpolation

$$f(x) = ax^3 + bx^2 + cx + d$$

$$f'(x) = 3ax^2 + 2bx + c$$

$$
\begin{cases} f(0) = d \\ f(1) = a+b+c+d \\ f'(0) = c \\ f'(1) = 3a+2b+c \end{cases} \longrightarrow
\begin{cases} a = 2f(0) - f(1) + f'(0) + f'(1) \\ b = -3f(0) + 3f(1) - 2f'(0) - f'(1) \\ c = f'(0) \\ d = f(0) \end{cases}
$$