Digital Image Processing

Lecture 10

Edge detection

Edge Detection

The first step towards Image understanding

from before:

SPATIAL 2D Filters

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

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

Sobel

for "Sobel-like operator" to detect edges at different angles

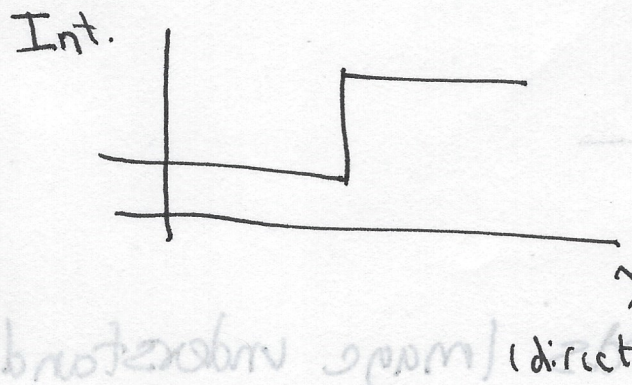
-2	-1	0
-1	0	1
0	1	2

Larger filters for different ~~angles~~ angles

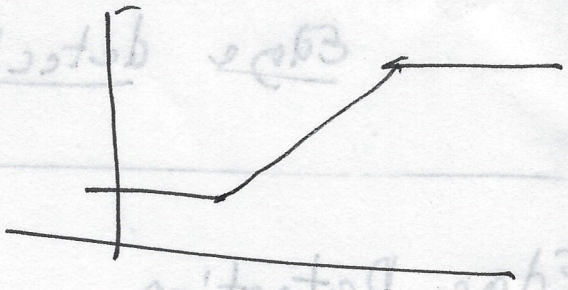
→ 5×5 for 30°

(2) So what is an edge?

Ideal Edge

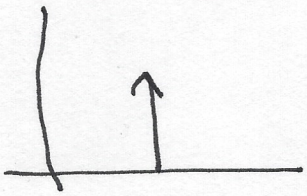


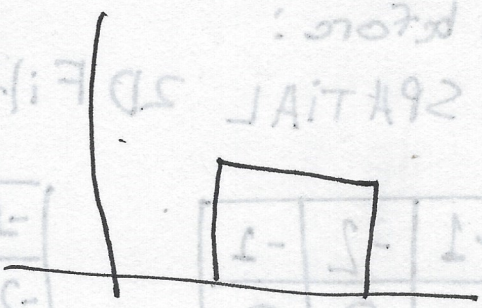
In Practice we have a ramp edge



1st derivative

Look for Big absolute values



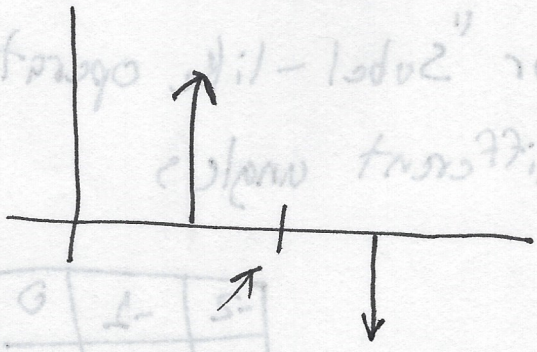


places where the first derivative is large can be used to detect edges

2nd derivative

Look for sign changes in the derivative

(zero crossing)

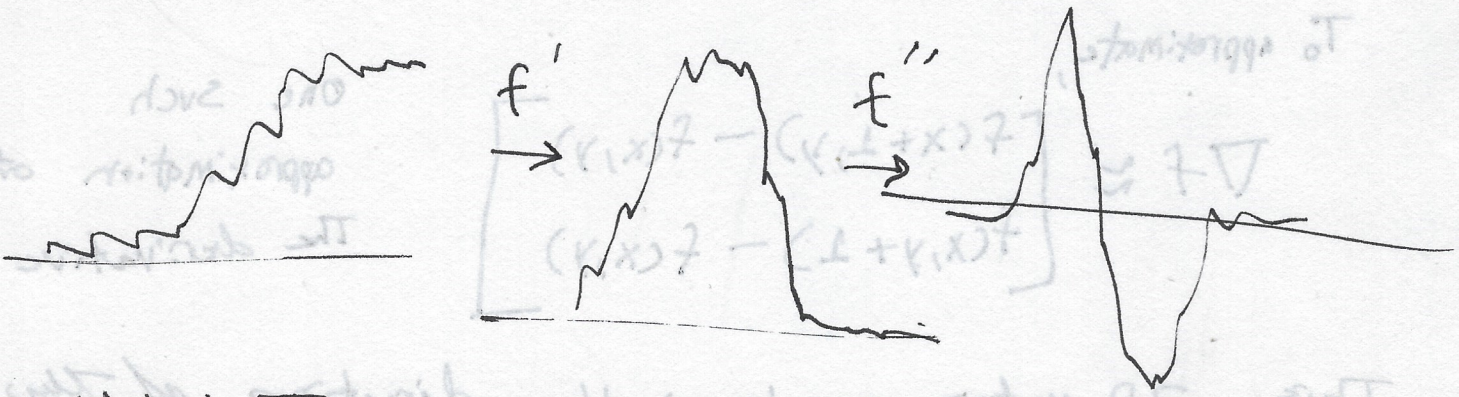


Look for the middle point where the sign changes

Finding Edges:

- places where 1st Derivative is Large
- places where 2nd Derivative changes sign
  (Also Known as : Zero-crossings)

for <u>real edges</u>



<u>Note</u>: Taking Derivatives Amplifies Noise

→ Noise can make it harder to find the edges using the criteria mentioned above

<u>A Solution</u>: LP Image first

Look at the Sobel filter,

We get the vertical Sobel filter by

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} [1\ 2\ 1]$$

find difference vertically ~~or~~ then Smooth horizontally (reduce Noise)

These are successive linear operators and can be applied in any order.
(Smooth Image then find Edges in other direction)

(4)

Edge Detection is fundamentally related To the Gradient of the Image

$$\nabla f = \text{grad } f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

To approximate,

$$\nabla f \simeq \begin{bmatrix} f(x+1,y) - f(x,y) \\ f(x,y+1) - f(x,y) \end{bmatrix}$$

one such approximation of The derivative

This 2D vector points in the direction of the highest rate of change of a function.
(where is the function changing the most?)

This is the gradient

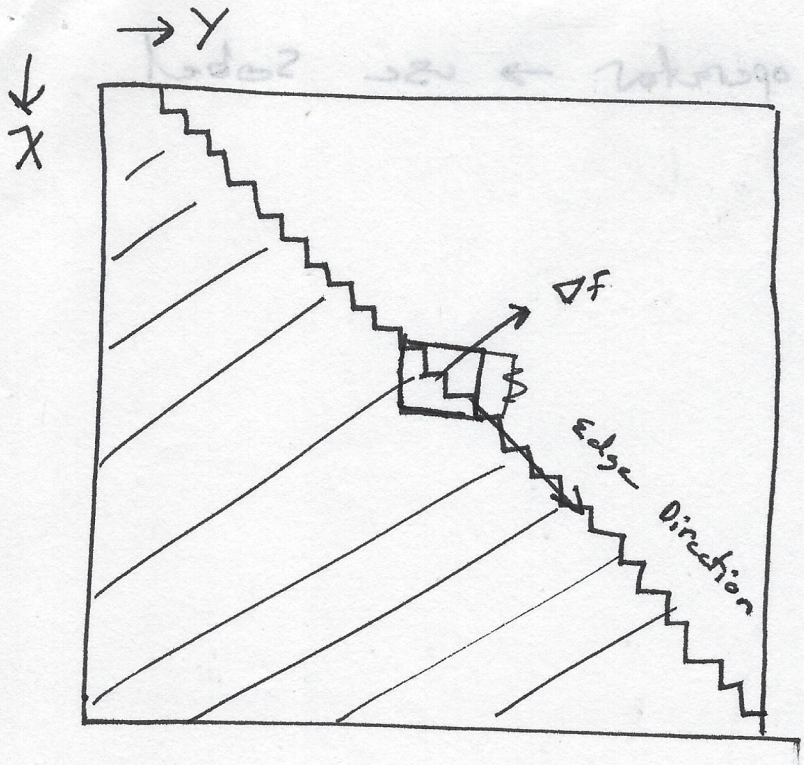
$$M(x,y) = \text{mag}(\nabla f) = \|\nabla f\|$$

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

$$\alpha(x,y) = \arctan\left(\frac{g_y}{g_x}\right)$$

MatLab: imgradientxy → g_x, g_y

imgradient → M, α



∇f will point in the direction perpendicular to the edge

if we start at the point of interest, which direction should we go to change our intensity the fastest?

(for this, naturally step across the edge)

→ MatLab at 12:00

Think cartesian differences

```
hx = [-1 1];
hy = [-1; 1];

gx = filter2(hx, im);
gy = filter2(hy, im);

figure
imshow(abs(gx), [])
colormap jet
colorbar
imshow(abs(gx) > 40)
colormap jet
colorbar
imshow(abs(gy), [])
```

blue color indicates not much action for edge detection, whereas yellow and red indicate edges

threshold for edges greater than 40

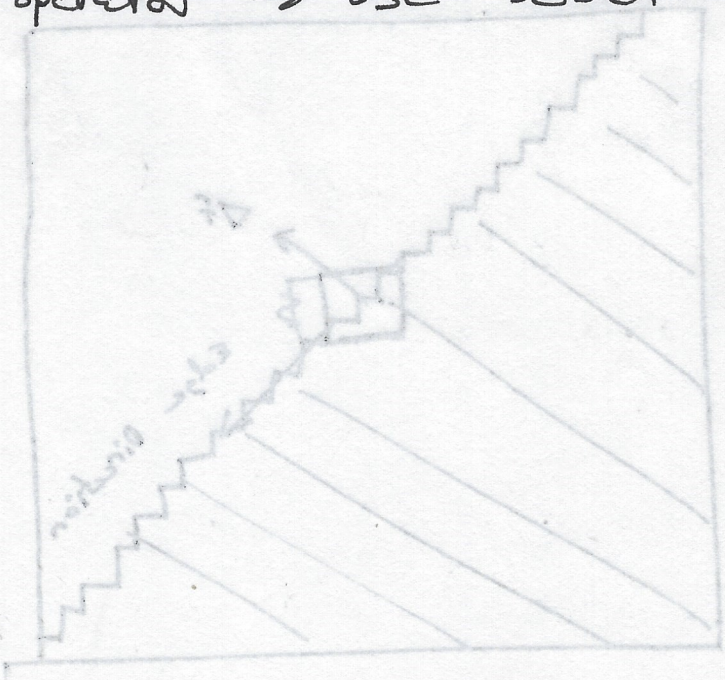
← strong edges for "up and down" activity

[Y component of ∇f]

(6) use a better edge operator → use Sobel

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

```
hy = hx';
gx = filter2(hx,im);
gy = filter2(hy,im);
figure
imshow(abs(gx),[]);
colormap jet
figure
imshow(abs(gy),[]);
colormap jet
```



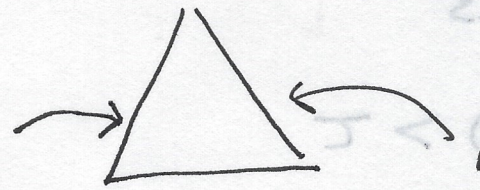
To get the magnitude and the angle

```
m = sqrt(gx.^2 + gy.^2);
a = atan(gy./gx)*180/pi;   % for degrees
figure
imshow(m,[])
colormap jet
figure
imshow(a,[])
colormap jet
colorbar
```

→ at 0°, we have a greenish color (for this example, vertical edges are green)

at horizontal edge we have a red/blue thing (depending on the wrap-around)

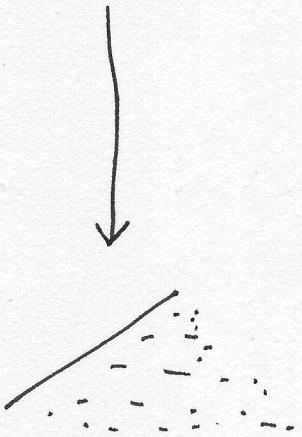
at the triangle (Nose of the Image)




```
e = (abs(a-30) < 10) & (m > 200);
imshow(e,[])
```

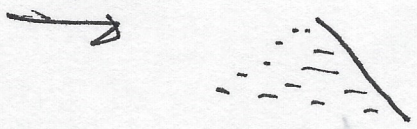
24:00

% Give us an Image such that the abs of the angle −30° is less than 10° (We want something close to 30°) and places where the magnitude is greater than a certain threshold.



for

```
e = (abs(a+30) < 10) & (m > 200)
```



- we can make preferential Edge operators (certain angle)

imgradientxy uses Sobel operators by default

We can Threshold The Gradient Magnitude To find Strong Edges

$$M(x,y) > \tau$$

We can refine / Preferentially Detect Edges at certain orientations

$$\left(|\alpha(x,y) - \hat{\alpha}| < \tau_1 \right) \& \left(M(x,y) > \tau_2 \right)$$

OTHER Edge Detectors

1) Laplacian of Gaussian (LoG)
   - MARR-HILDRETH
   - Mexican HAT FILTER

- Idea: Scale dependent Edge detection

AN Edge detector that can be tuned to Edges at Different scales.

Big operators: Large-Scale / Blurry Edges

Small ~~Scale~~ operators: Small-Scale / fine Detail

Gaussian:

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

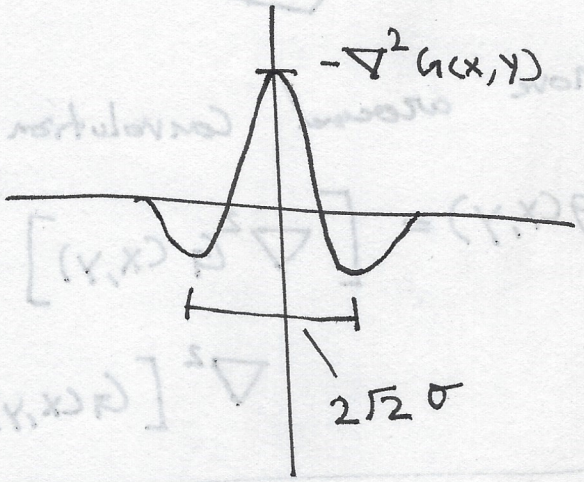
σ, small, points ; σ, large, spread out

We are smoothing the Image at some scale then the Laplacian operator finds Edges of the Image (after smoothing)

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

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

The negative of This function Looks Like:



Idea: Depending on σ, (To a certain scale)

The Gaussian Blurs the Image more/Less, and the Laplacian picks up the Edges that scale (Look for zero-crossing) → Due to Laplacian being a 2nd Derivative operator

(10)

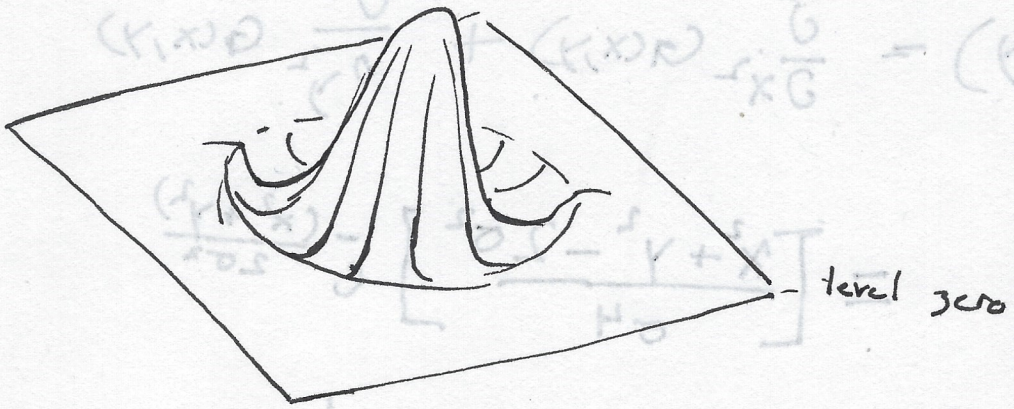
fspecial log

~32:00

h = fspecial ('log', [101 101], 10); ← σ

surf(-h, 'edgecolor', 'none')

colormap jet



→ Move around convolutions

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

$$\nabla^2 \left[G(x,y) * f(x,y)\right]$$

(demonstration around 37:00)

↓

Importance of blurring at different scales then edge finding

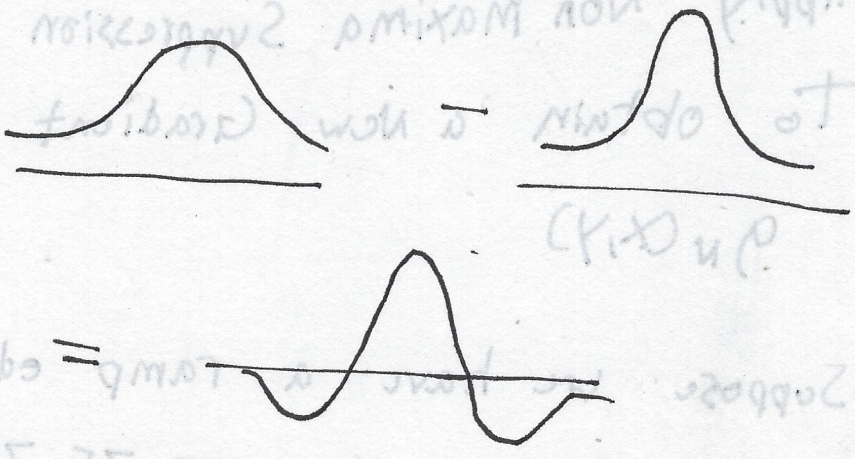
~~Side Note~~

Could ~~Ra~~ also find the scale (σ) AT which a Given Edge is most significant, filter on that

We can Approximate the LoG with a Difference of Gaussians (DoG)

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

$\sigma_1 > \sigma_2$



for computational reasons we may just want to filter the Image with two gaussians

```
h1 = fspecial('gaussian', 101, 20)
h2 = fspecial('gaussian', 101, 10)

surf(-(h1-h2), 'edgecolor', 'none')
```

USED alot in Computer Vision (SIFT)

(12)

CANNY Edge Detector

Basic Steps

1) Smooth Image with A Gaussian

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

3) Apply Non maxima Suppression To $M(x,y)$ to obtain a New Gradient Image $g_N(x,y)$

Suppose we have a ramp edge,

0 0 0 10 25 50 65 75 75 75 75

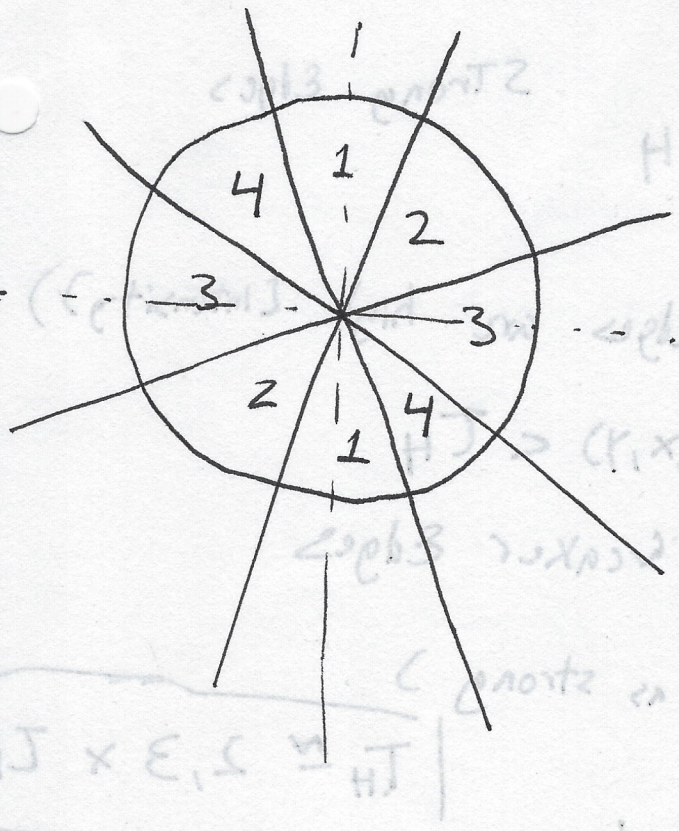
0 0 10 15 25 15 10 0 0 0 ...

Edge center — Non trivial edge response (magnitudes)

- If we are looking for the edge center, we care about the middle

- we want to suppress these Non trivial magnitudes and not include them in our Edge Gap

IDEA: Quantize $\alpha(x,y)$ into 4 ~~bits~~ BINS



Suppose we have angle and magnitude at Given pixel

17	15	30
10	25	22
27	15	26

$M(x,y)$

2	1	2
2	1	1
2	2	1

$\alpha(x,y)$

center pixel has a magnitude and Direction

IF $M(x,y)$ is Greater than Both ITS NEIGHBORS in the Quantized EDGE DIRECTION,

$g_N(x,y) = M(x,y)$, otherwise $g_N(x,y) = 0$

4) Detect and Link Edges

$$g_H(x,y) = g_N(x,y) \geq \tau_H$$

Strong Edges

(High Map)

(find places where the edges are high [intensity])

$$g_L(x,y) = \tau_L \leq g_N(x,y) < \tau_H$$

weaker Edges

(Low MAP)

(find edges that are not as strong)

$$\tau_H \approx 2,3 \times \tau_L$$

Final MAP: $g_H(x,y) \cup$

All edges in $g_L(x,y)$ That are adjacent to at least one pixel of $g_H(x,y)$

IDEA: Assign 2 thresholds, the pixels with very strong Intensities are all included in the final map.

The pixels that are somewhat strong are only included if they are close to a strong edge.

— This helps us bridge some gaps we may have missed

```
[bw, th] = edge(im, 'canny');
```

we have a threshold unspecified

try

```
[bw, th] = edge(im, 'canny', 0.4);
imshow(bw, [])
```

Canny is cleaner than Sobel 57:15