Computational Vision

Lecture 2.2: Edge Detection and Filtering

Hamid Dehghani

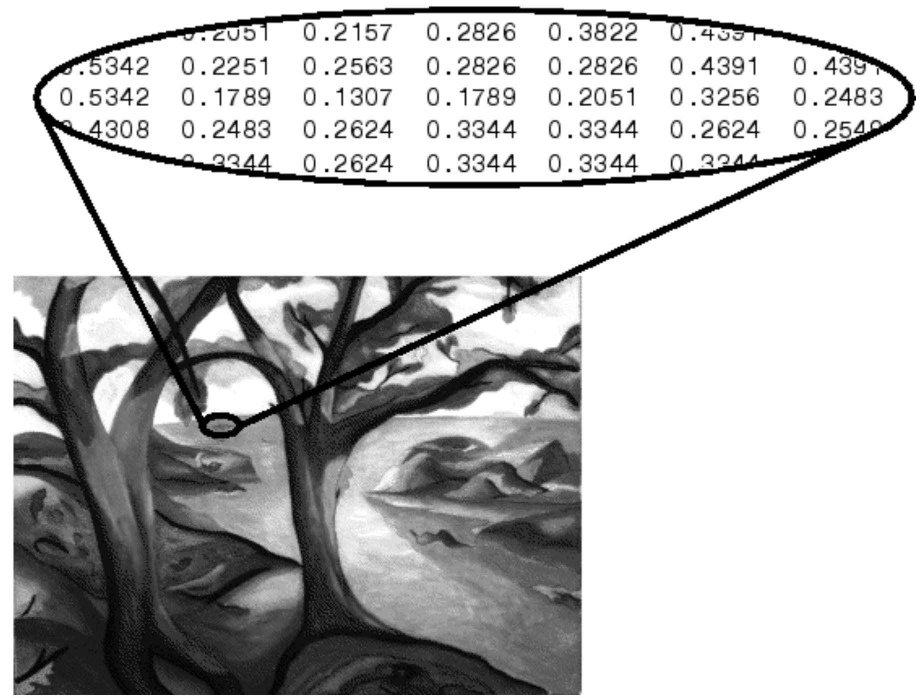
Office: UG38

Aims

- Intensity Images
- Edge Detection
- Convolution

Intensity Images

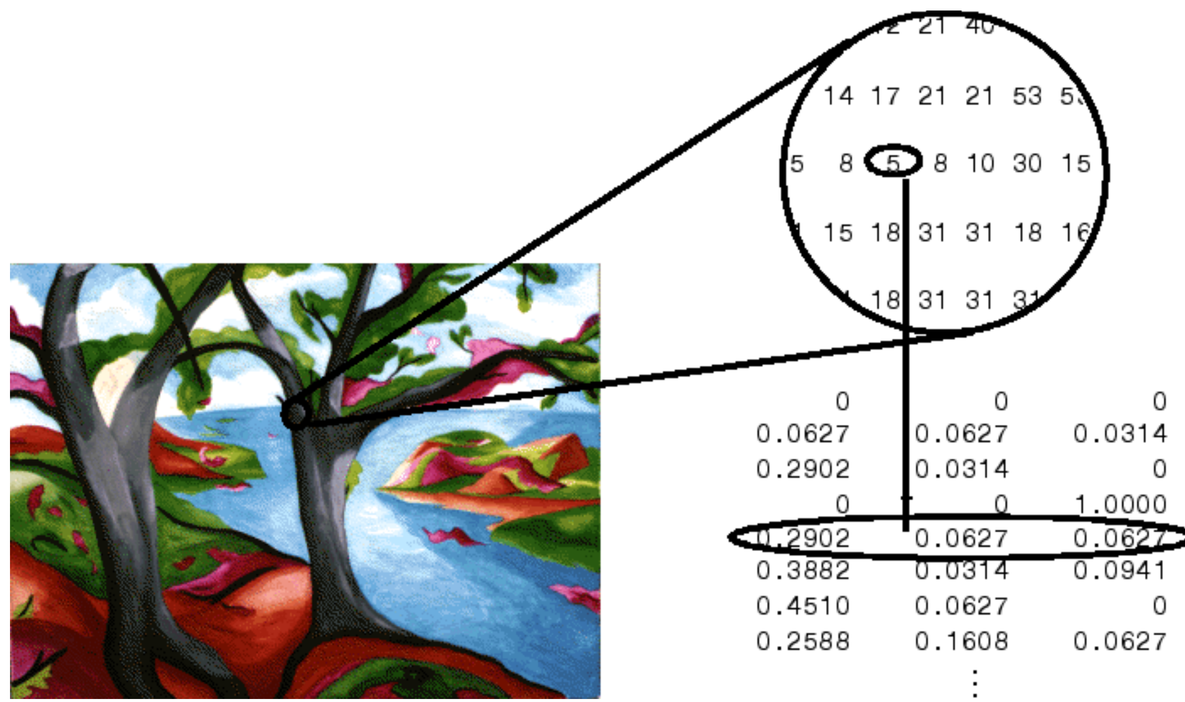
- An intensity image is a data matrix, whose values represent intensities within some range.
- represented as a single matrix, with each element of the matrix corresponding to one image pixel
- In matlab: To display an intensity image, use the imagesc ("image scale") function



Indexed Images

- An indexed image consists of a data matrix, X, and a colormap matrix, map.
- map is an m-by-3 array of class double containing floating-point values in the range [0, 1].

- Each row of map specifies the red, green, and blue components of a single color.



Guess the image

1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1

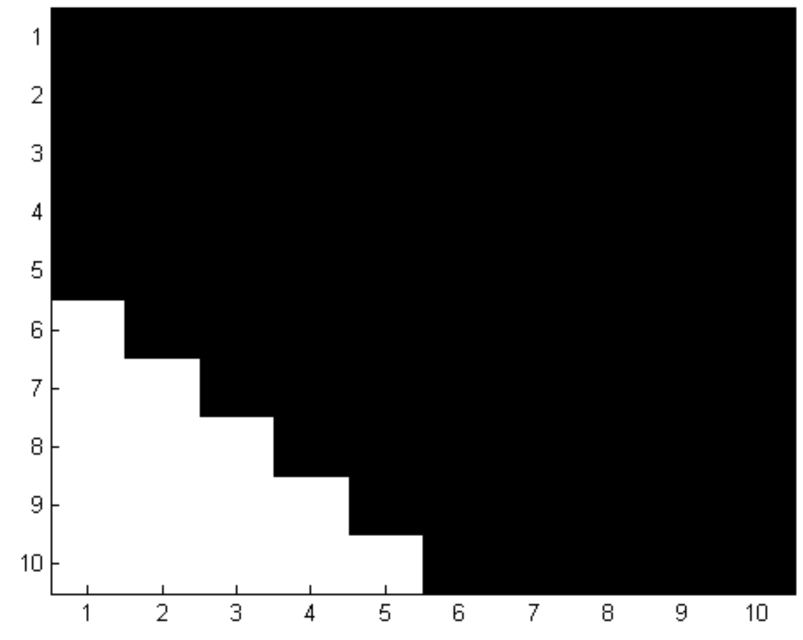
Guess the image

1	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0
0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1

Intensity gradients

- The image is a function mapping coordinates to intensity *f(x,y)*

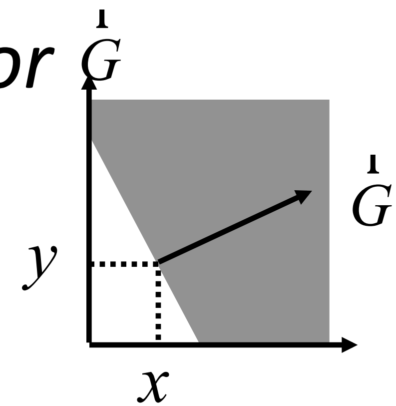
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1



Intensity gradients

- The image is a function mapping coordinates to intensity *f(x,y)*
- *The gradient of the intensity is a vector* $\vec{G}$

$$\vec{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix}$$



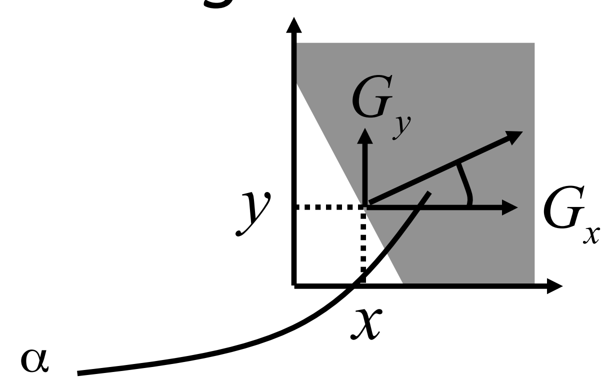
- *We can think of the gradient as having an x and a y component*

$$M(\vec{G}) = \sqrt{G_x^2 + G_y^2}$$

magnitude

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

direction

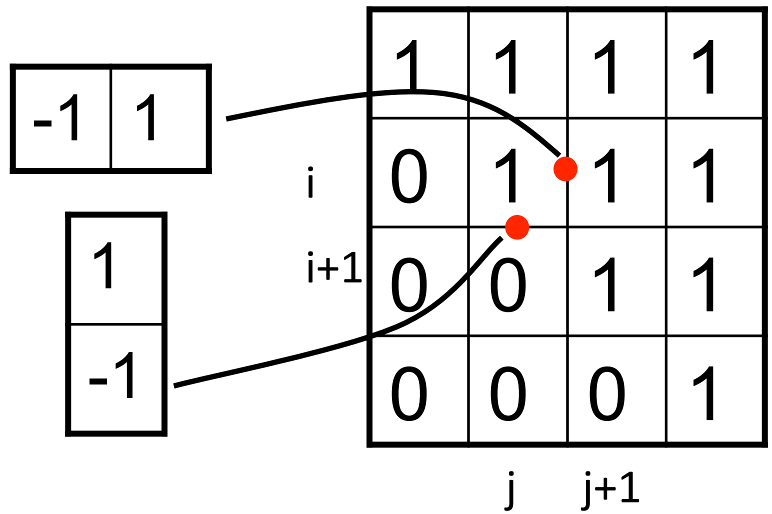


Approximating the gradient

- Our image is discrete with pixels indexed by i and j

$$G_x \cong f[i, j+1] - f[i, j]$$

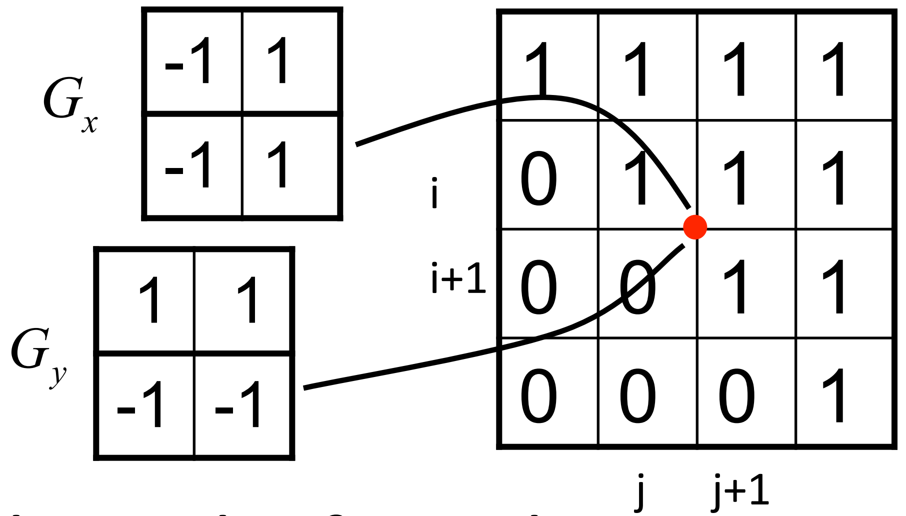
$$G_y \cong f[i, j] - f[i+1, j]$$



- We want to estimated in the same place

Approximating the gradient

- So we use a 2x2 mask instead



- For each mask of weights you multiply the corresponding pixel by the weight and sum over all pixels

Other edge detectors

- Roberts

G_x

1	0
0	-1

G_y

0	-1
1	0

- Sobel

G_x

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

G_y

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

Approximating the gradient

- Sobel

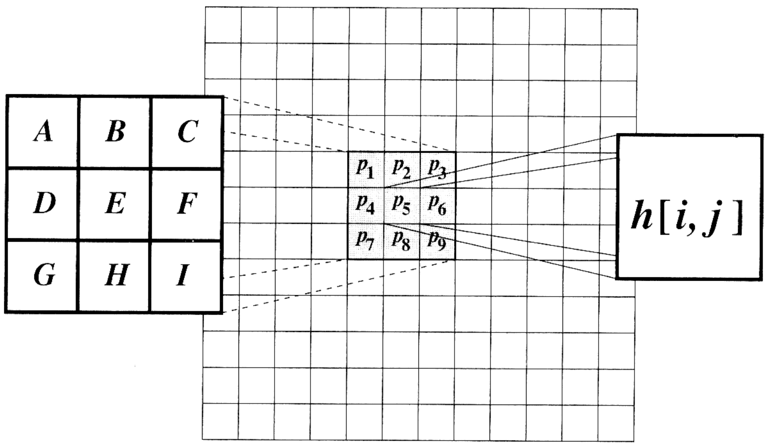
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{array}{c|cccc|}
 & & & & \\
 & 1 & 1 & 1 & 1 \\
i & 0 & 1 & 1 & 1 \\
i+1 & 0 & 0 & 1 & 1 \\
 & 0 & 0 & 0 & 1 \\
\hline
 & & j & j+1 &
\end{array}$$

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

Convolution

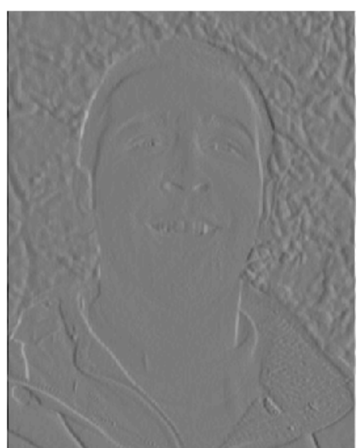
- Convolution is the computation of weighted sums of image pixels.
- For each pixel [i,j] in the image, the value h[i,j] is calculated by translating the mask to pixel [i,j] and taking the weighted sum of pixels in neighbourhood

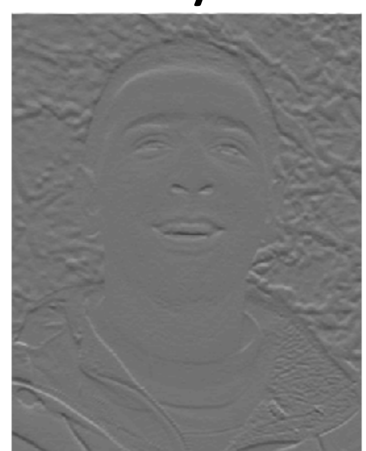


What do these filters do

- Steps:
  - Take image
  - Convolve mask with image for each direction
    - Calculate derivatives Gx and Gy
  - Calculate magnitude = $M(\vec{G}) = \sqrt{G_x^2 + G_y^2}$

Original


Gx


Gy


M


Filtering

- We could detect edges by calculating the intensity change (gradient) across the image

- We could implement this using the idea of filtering

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

0	1	1	3	4	5
0	0	2	3	3	4
0	0	4	6	3	5
0	0	0	4	4	3
0	0	0	3	5	2
0	0	0	0	5	5
0	0	0	0	4	3

Linear filtering: the algorithm

```
for i=2:image_height-1
 for j=2:image_width-1
```

$$\mathbf{A}_{out}(\mathbf{i},\mathbf{j}) = \sum_{y=-1}^{1}\sum_{x=-1}^{1}\mathbf{A}_{in}(i+y, j+x)\mathbf{M}(y+2, x+2)$$

```
 end
end
```

x+2

y+2

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

j+x

i+y

0	1	1	3	4	5
0	0	2	3	3	4
0	0	4	6	3	5
0	0	0	4	4	3
0	0	0	3	5	2
0	0	0	0	5	5
0	0	0	0	4	3

j

i

NB We count from the upper left,and in MATLAB we start at 1

Highly Directed Work

- Gaussian (Canny) edge detection
- Second order operators
- Thresholding