Computational Vision

Lecture 1.2: Human Vision

Edge Detection and Filtering

Hamid Dehghani

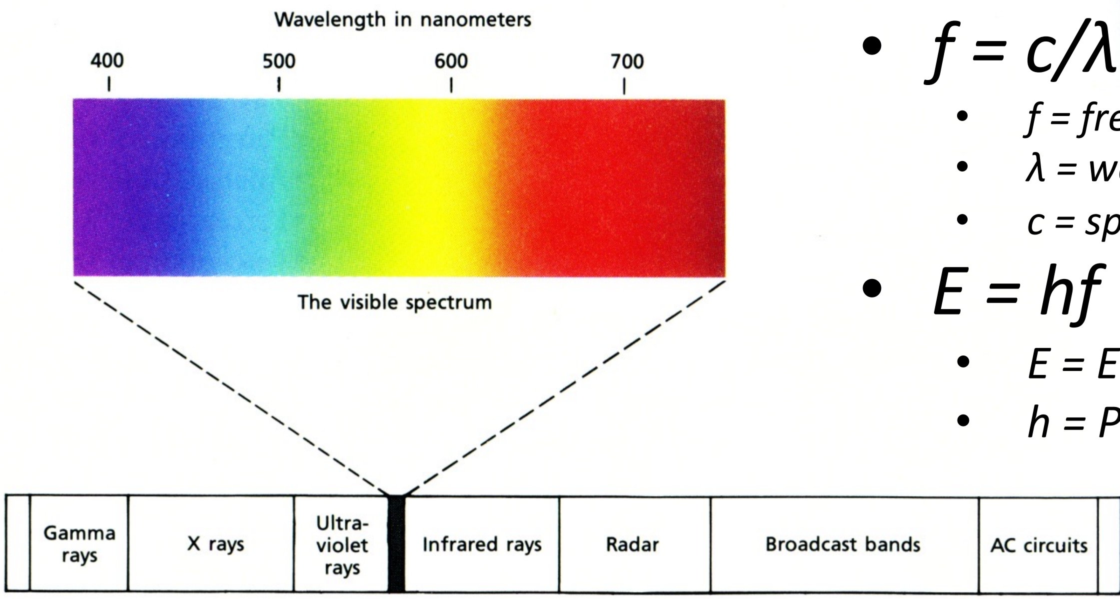
Office: CS 241

Aims

- Human Vision: Colour!
- Intensity Images
- Edge Detection
  - Convolution
- Noise Reduction

Visible Light

- Humans perceive electromagnetic radiation with wavelengths 380-760nm (1 nm = 10^{-9} m)



- $f = c/\lambda$
  - f = *frequency (Hz)*
  - λ = *wavelength (m)*
  - c = *speed of light* ($2.998 \times 10^{8}\ ms^{-1}$)
- $E = hf$
  - E = *Energy (J)*
  - h = *Plank's constant* (6.623×10^{34} *Js*)

Colour

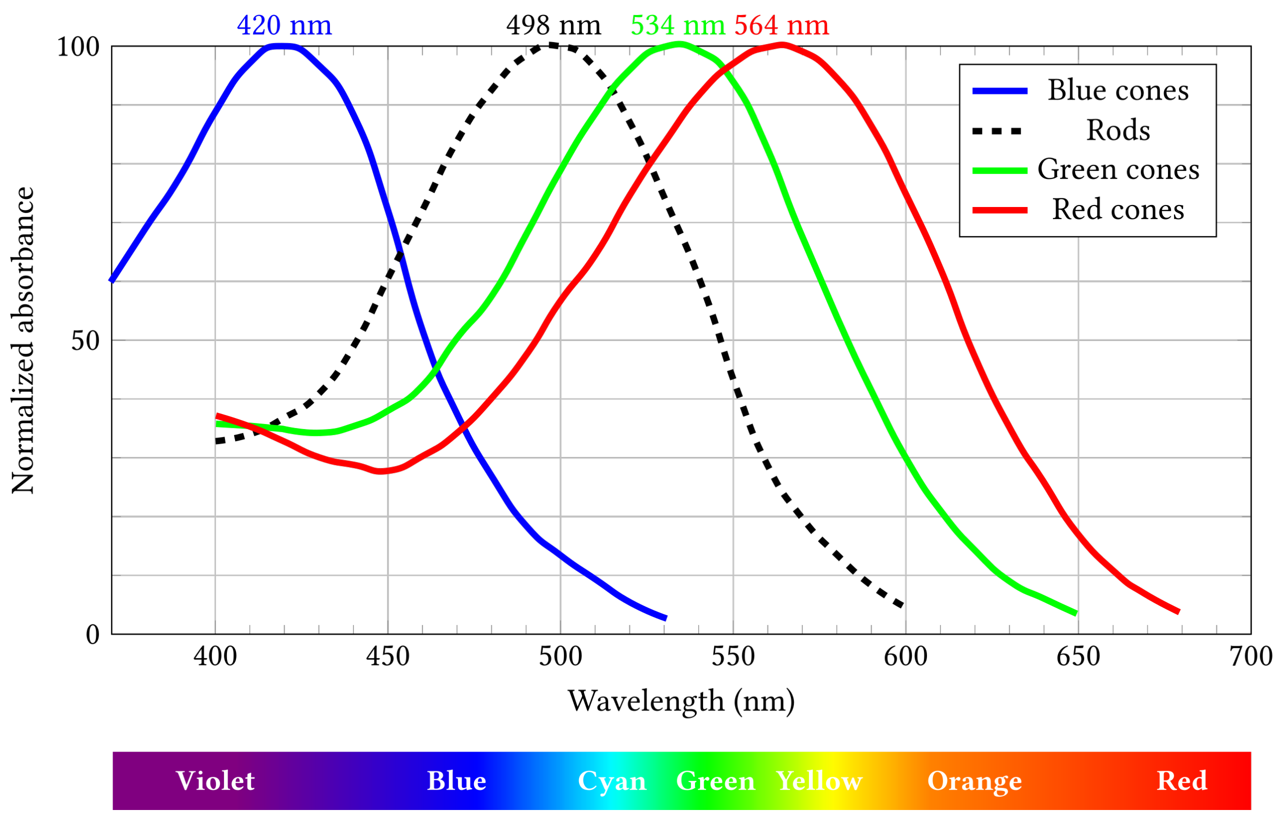
- Objects selectively absorb some wavelengths (colours) and reflect others
- Human retinas contain three different kinds of cones to provide very elaborate form of vision
  - Gives ability to distinguish different forms of same objects
    - Fruits
    - Camouflage

Colour mixing

- Many forms of colour vision proposed
  - Until recently some hard to disapprove
- 1802: Proposed that the eye has three different types of receptors, each sensitive to a single hue (Young, a British physicist)
  - By the fact that any colour can be produced by appropriate mixing of the three primary colours.
- This became known as Trichromatic (three-colour) theory.

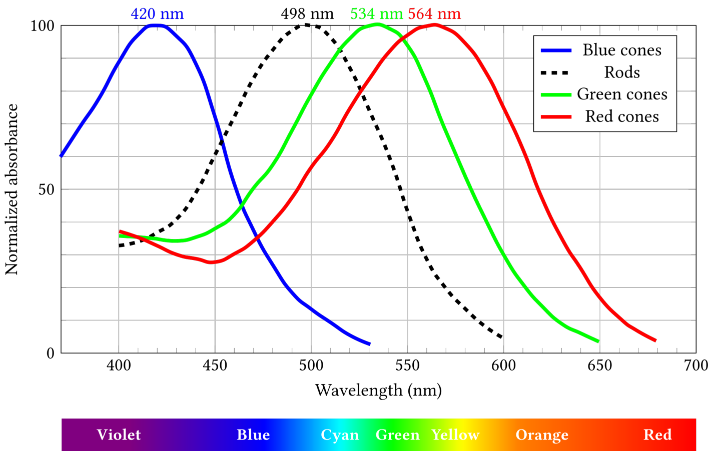
Trichromatic Coding

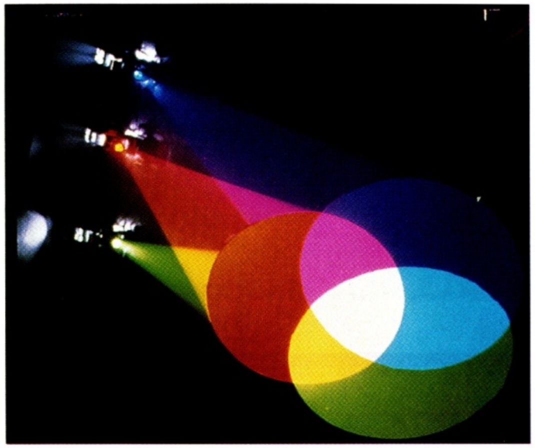
- Retina contains approximately equal numbers of red and green cones, but only 8% are blue cones



Trichromatic Coding

- Explains
  - How we discriminate wavelengths 2nm in difference
  - How we can match a mixture of wavelengths to a single colour
  - Some types of colour blindness
- Does not account for colour blending:
  - Some colour blend while others don't!
  - One can imaging Bluish-green or Yellowish-green, But NOT Greenish-red or Bluish-yellow!





Colour mixing

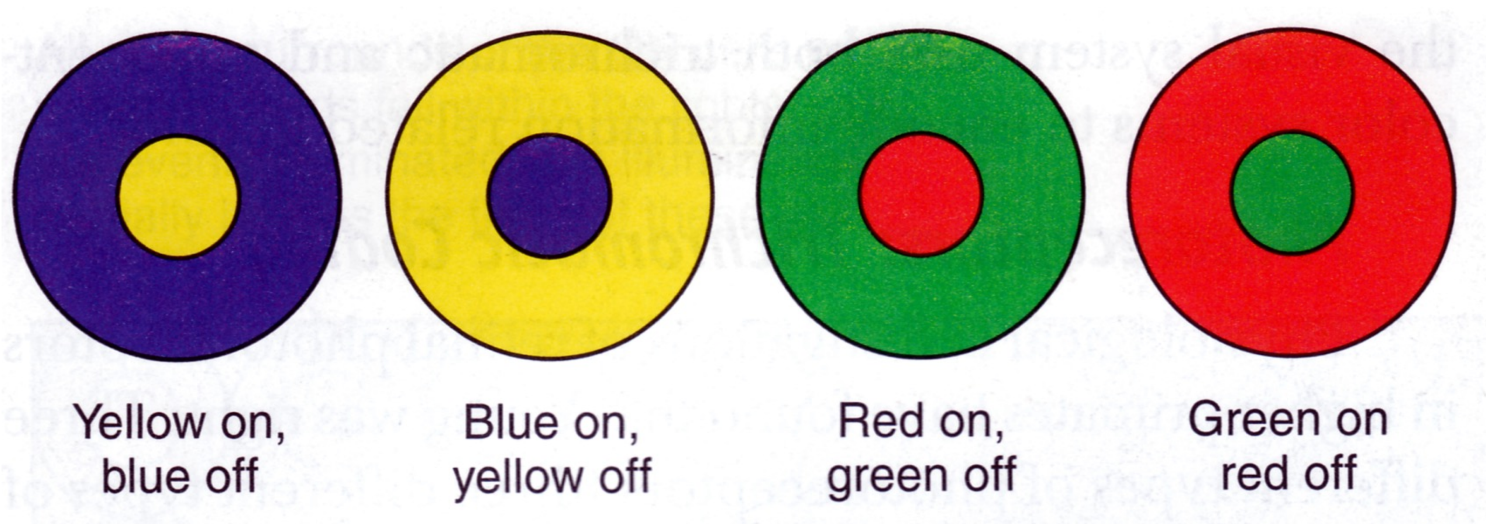
- Many forms of colour vision proposed
  - Until recently some hard to disapprove
- 1930s: Hering (German Physiologist) suggested colour may be represented in visual system as ‘opponent colours’
- Yellow, Blue, Red and Green – Primary colours
  - Trichromatic theory cannot explain why yellow is a primary colour

Opponent Process Coding

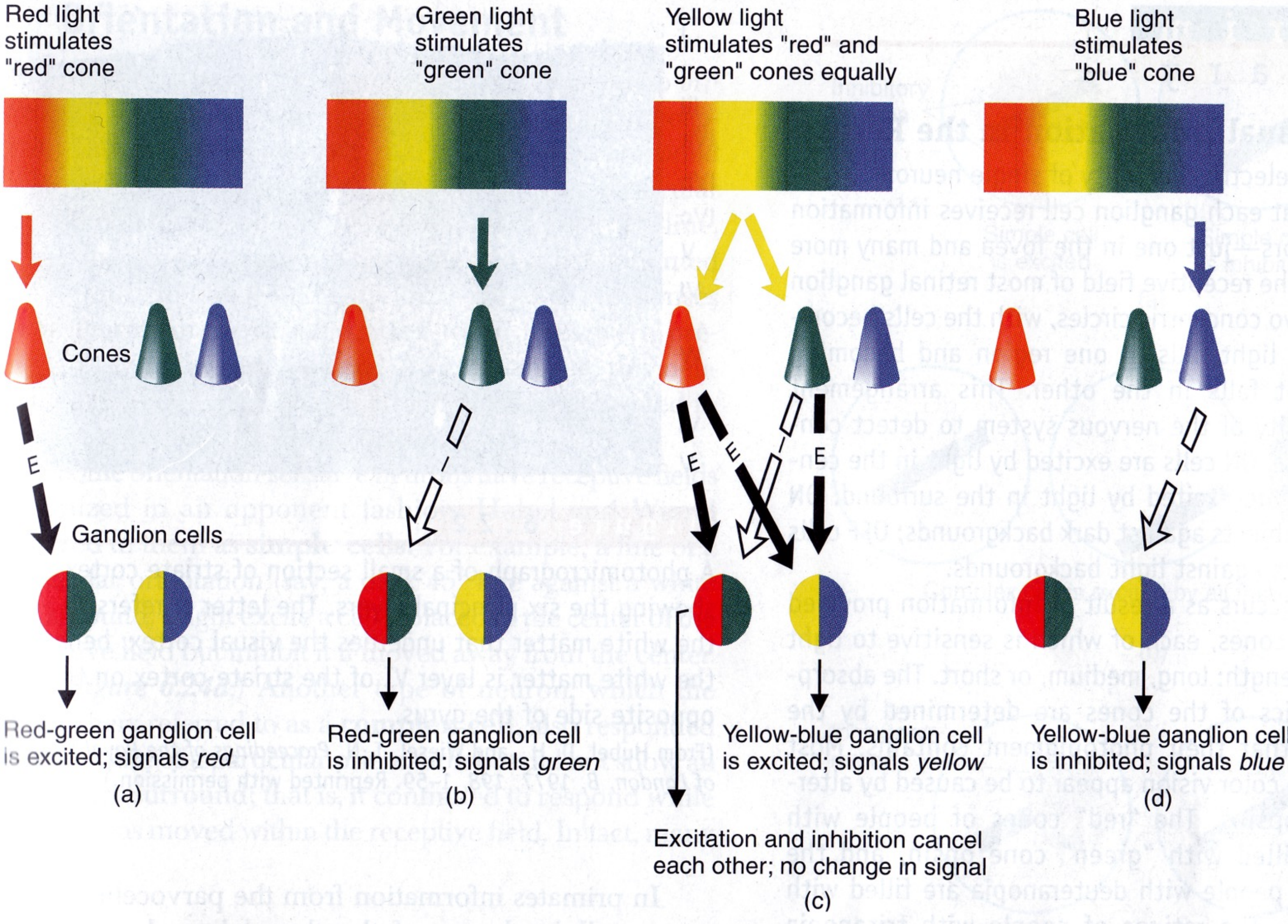
- Yellow, Blue, Red and Green – Primary colours
  - Trichromatic theory cannot explain why yellow is a primary colour
  - Also some colours blend and others do not!
    - Bluish green, yellowish green, orange (red and yellow), purple (red and blue) OK
    - Reddish green?? Bluish Yellow??
      - Opposite to each other

Opponent Process Coding

- Neurons respond to pairs of primary colours
  - Red-Green & Yellow-Blue
- Some respond in centre-surround fashion
- Response characteristics determined by appropriate ganglion cells connections

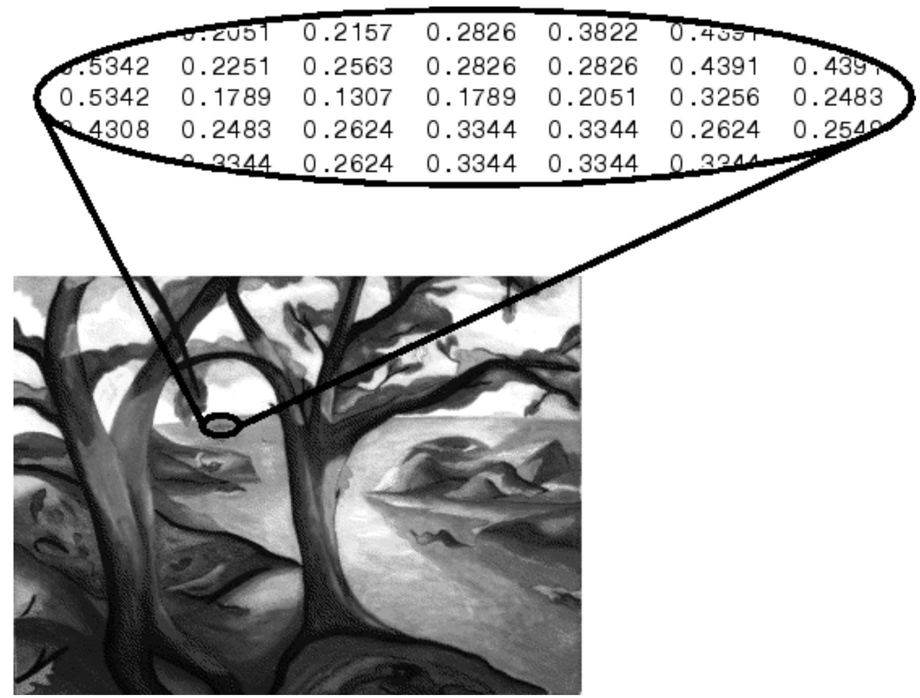


Opponent Process Coding



Intensity Images

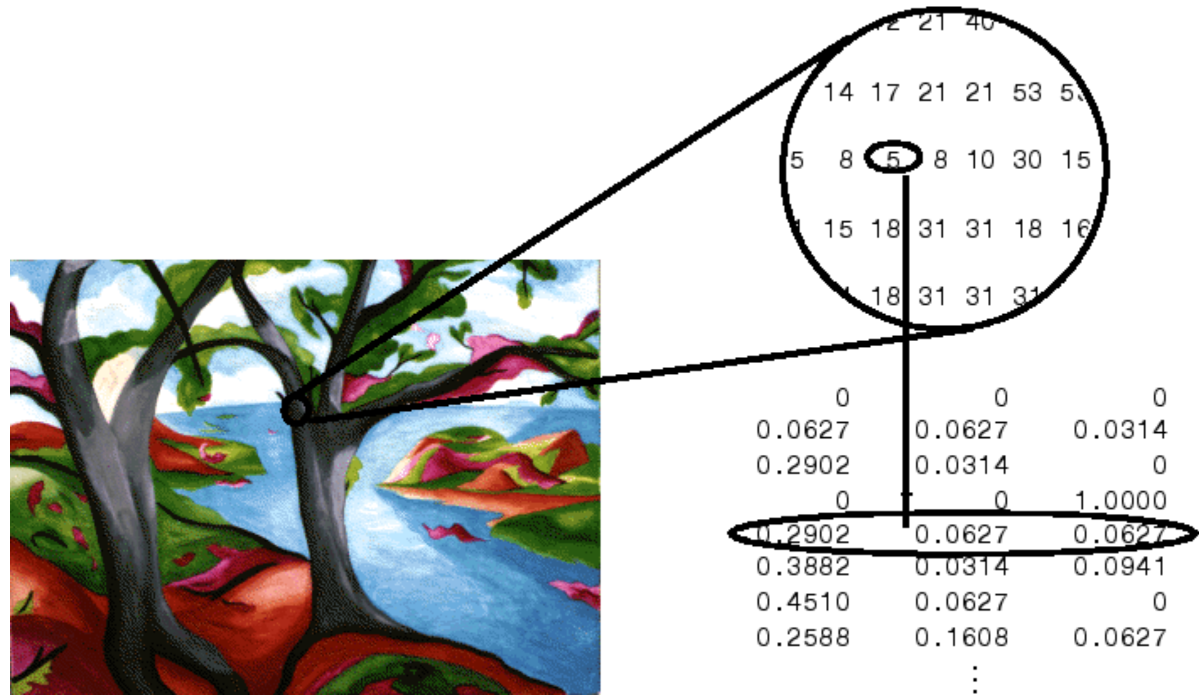
- An intensity image is a data matrix, whose values represent intensities within some range.
- represented as a single matrix, with each element of the matrix corresponding to one image pixel
- In matlab: To display an intensity image, use the imagesc ("image scale") function



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].

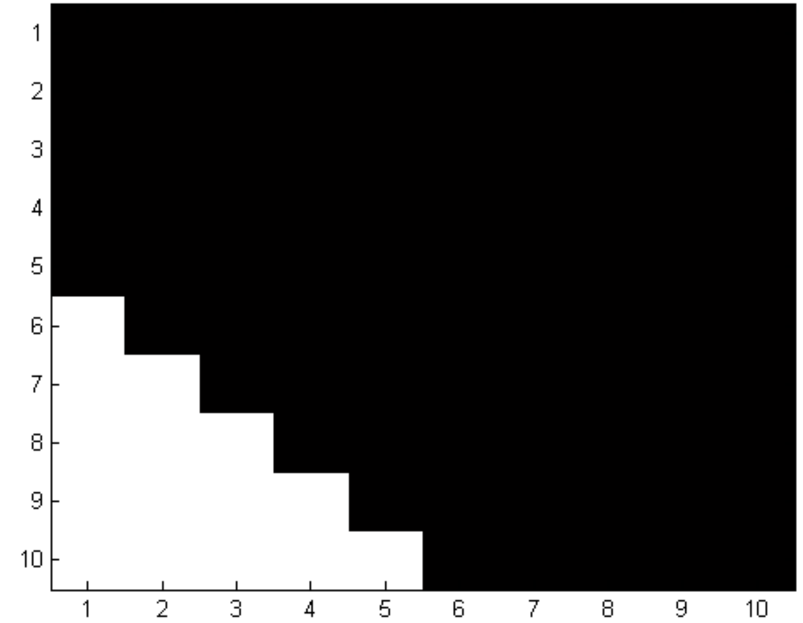
- Each row of map specifies the red, green, and blue components of a single color.



Intensity gradients

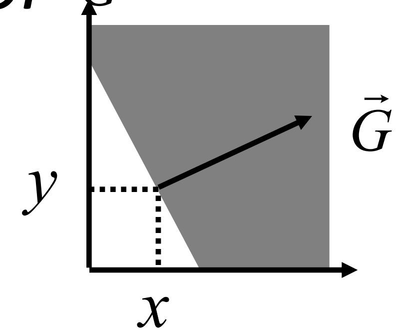
- The image is a function mapping coordinates to intensity *f(x,y)*

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



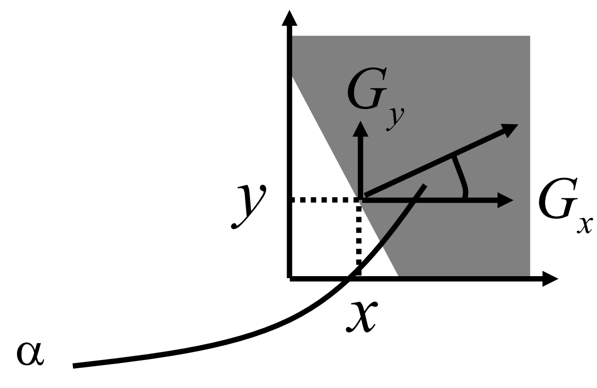
Intensity gradients

- 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

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

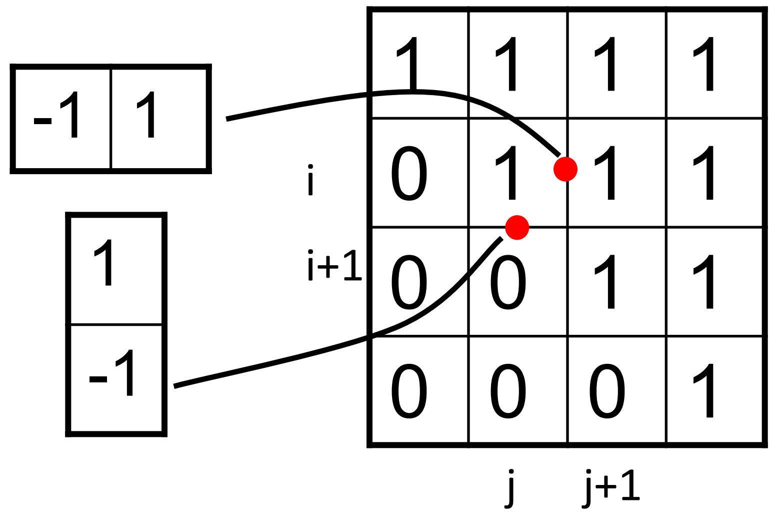
direction

Approximating the gradient

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

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

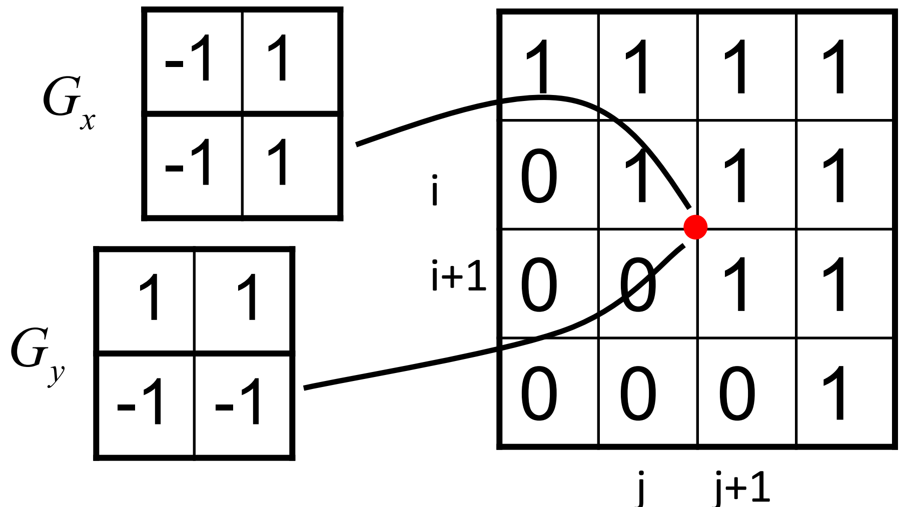
$$G_y \cong f[i, j] - f[i+1, j]$$



- 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

$$G_x \quad \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array} \qquad G_y \quad \begin{array}{|c|c|}\hline 0 & -1 \\ \hline 1 & 0 \\ \hline\end{array}$$

- Sobel

$$G_x \quad \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \qquad G_y \quad \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$$

Approximating the gradient

- Sobel

G_x

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

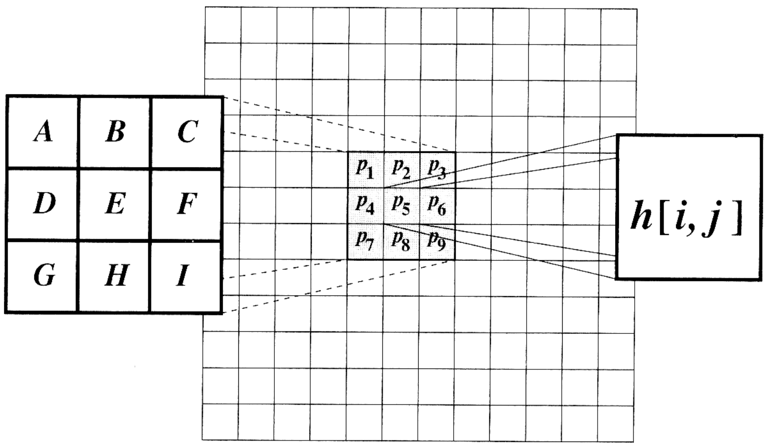
G_y

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

	1	1	1	1
i	0	1	1	1
i+1	0	0	1	1
	0	0	0	1
		j	j+1	

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

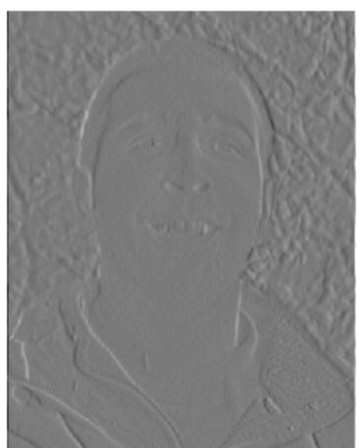


What do these filters do

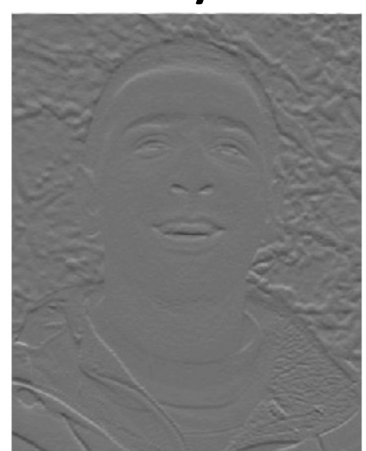
- Steps:
  - Take image
  - Convolve mask with image for each direction
    - Calculate derivatives Gx and Gy
  - Calculate magnitude = $M(\vec{G}) = \sqrt{G_x^2 + G_y^2}$

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

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=5

i=3

	9	14	4	3	
	10	16	3	-2	
	4	17	12	-4	
	0	10	19	2	
	0	3	19	12	

Noise

- It turns out we will need to remove noise
- There are many noise filters
- We can implement most of them using the idea of convolution again
- e.g. Mean filter

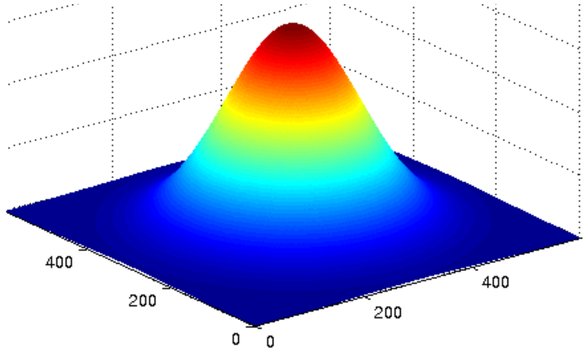
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

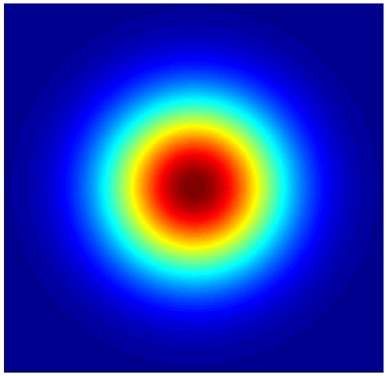
Noise filtering

- We can use convolution to remove noise as we mentioned, e.g. mean filter

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

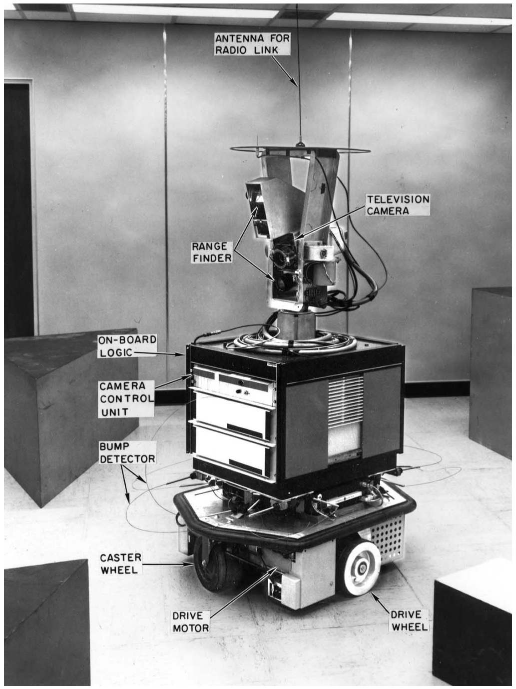
- This is a linear filter
- The most widely used is Gaussian filtering



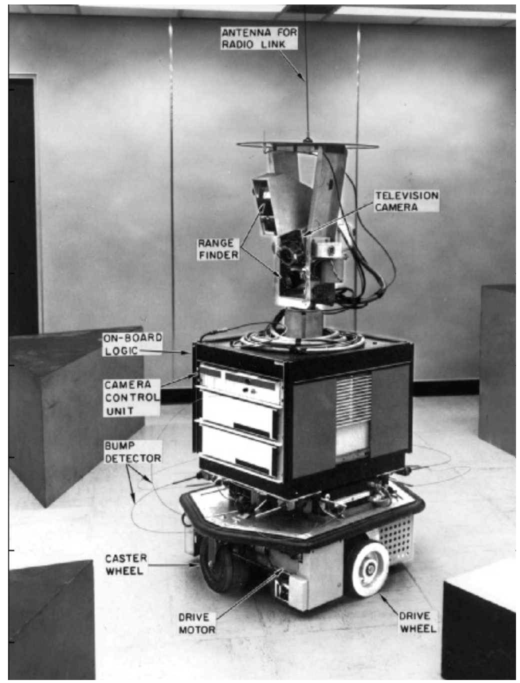


0	.01	.02	.01	0
.01	.06	.11	.06	.01
.02	.11	.16	.11	.02
.01	.06	.11	.06	.01
0	.01	.02	.01	0

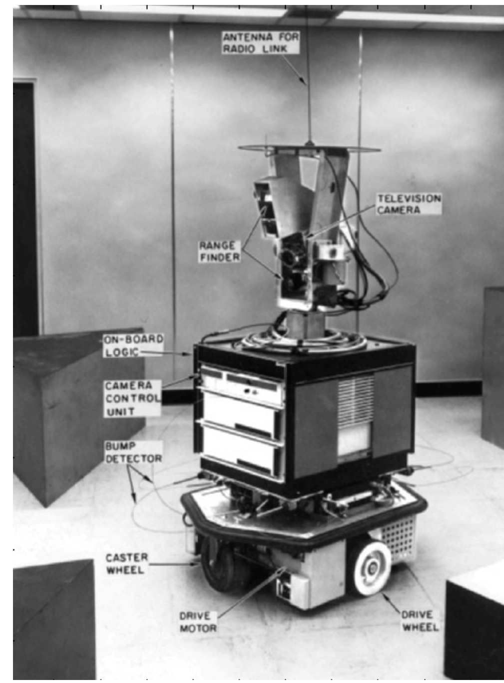
Effect of mean filtering



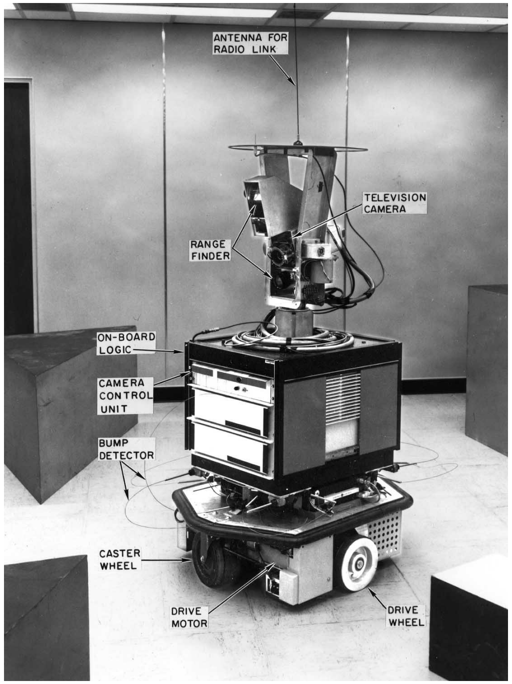
Original



3x3 filter



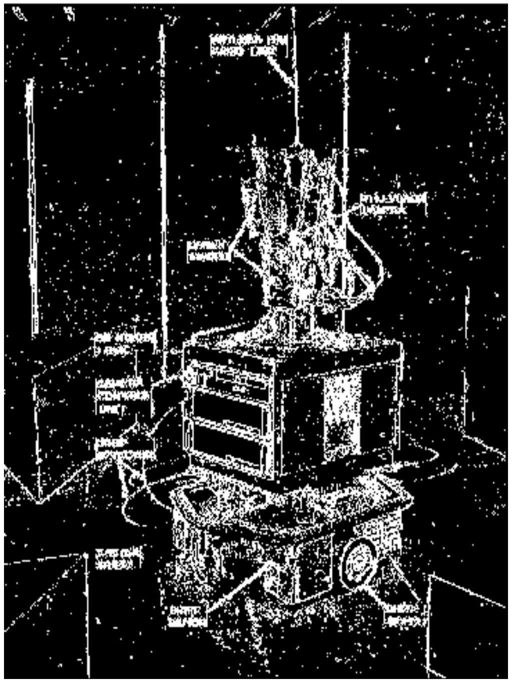
5x5 filter

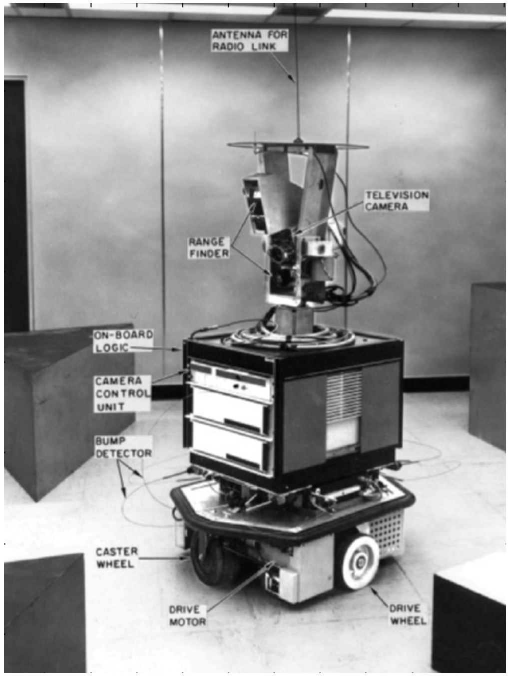


Horizontal Sobel operator

Abs(G_x)

Threshold=30



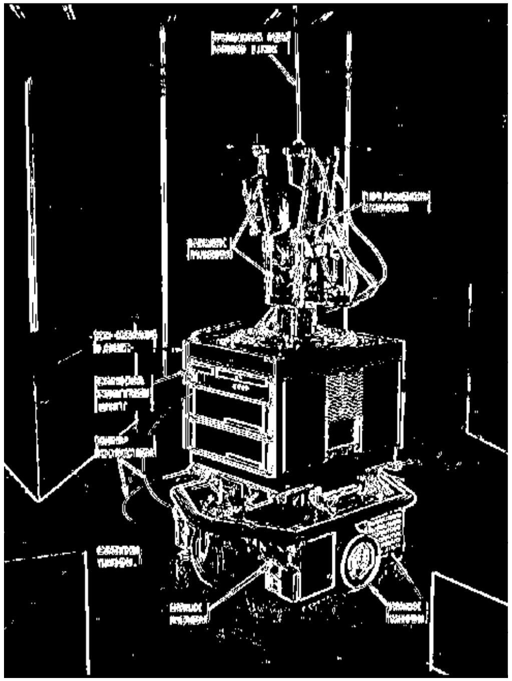


5x5 Mean Filter

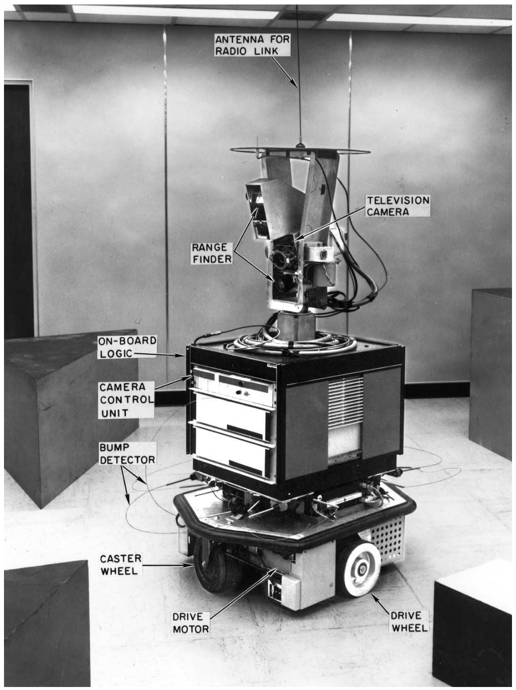
Horizontal Sobel operator

Abs(G_x)

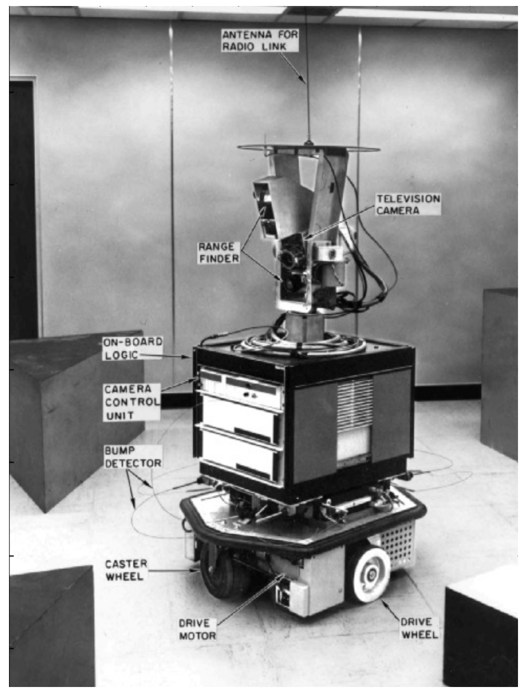
Threshold=30



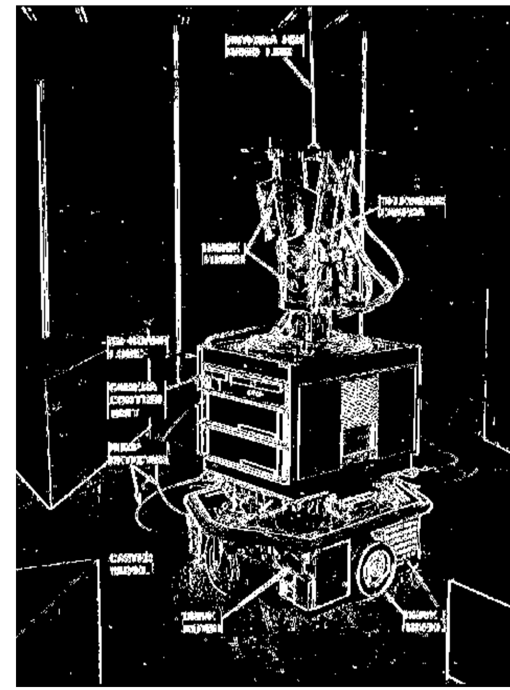
Effect of Gaussian filtering



Original



5x5 filter



Horizontal Sobel Operator

Abs(G_x)

Threshold = 30

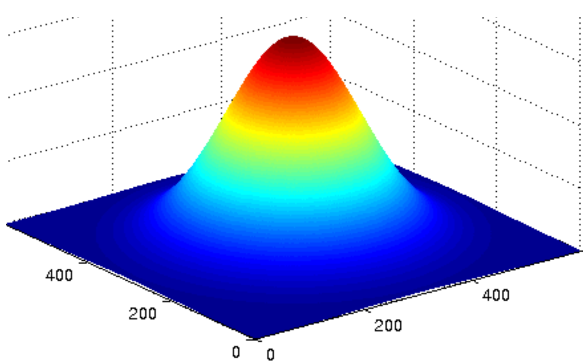
Sequenced filters

We can replace a 2D Gaussian filter with 2, 1D Gaussian filters in sequence

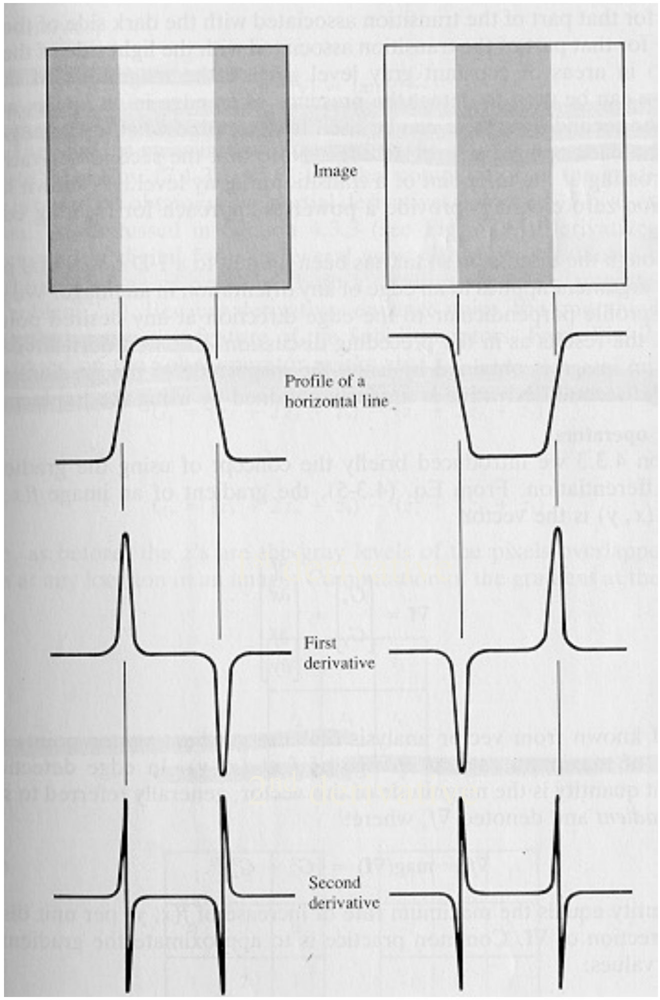
0.003	.0133	.0219	.0133	0.003
.0133	.0596	.0983	.0596	.0133
.0219	.0983	.1621	.0983	.0219
.0133	.0596	.0983	.0596	.0133
0.003	.0133	.0219	.0133	0.003

.0545	.2442	.4026	.2442	.0545

.0545
.2442
.4026
.2442
.0545



Laplacian Operator



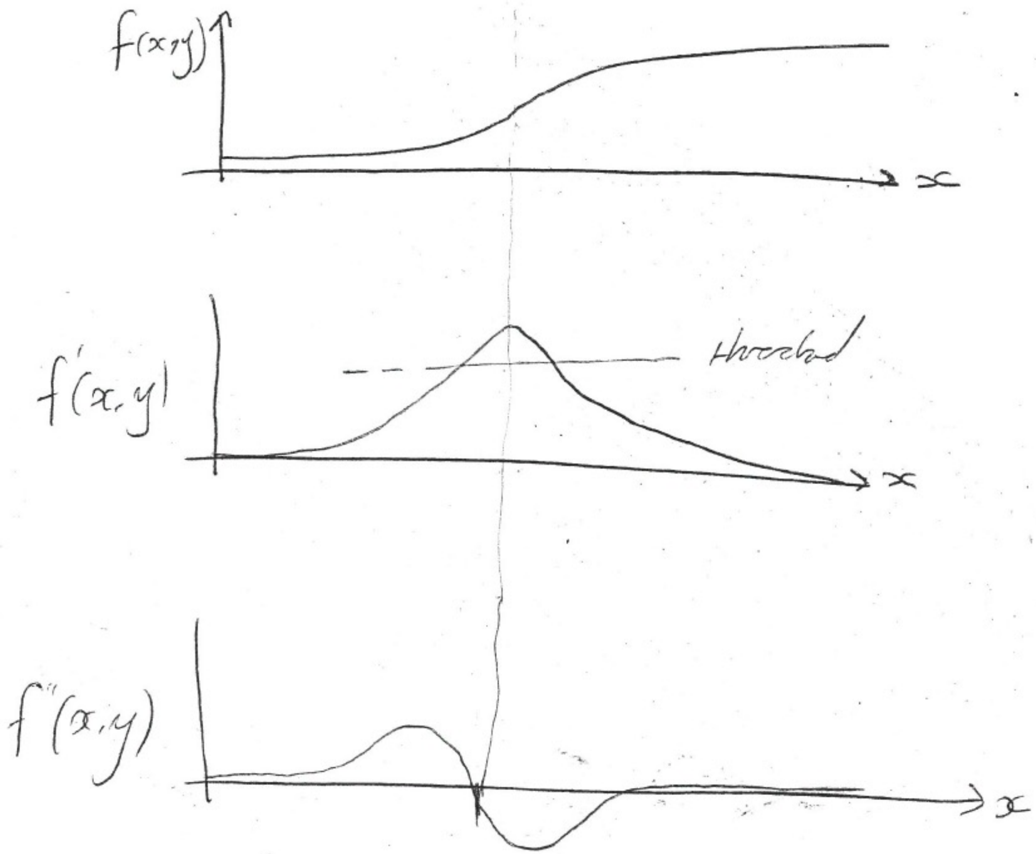
Laplacian Operator

- 1^{st} derivative results in too many edge points
- Better approach → points that only have local maxima in gradient → edge points

ie peak in 1^{st} derivative & zero crossing in 2^{nd} derivative.

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

Laplacian Operator



Laplacian Operator

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial G_x}{\partial x}$$

$$= \frac{\partial\left(f[i,j+1] - f[i,j]\right)}{\partial x}$$

$$= \frac{\partial f[i,j+1]}{\partial x} - \frac{\partial f[i,j]}{\partial x}$$

$$= \left(f[i,j+2] - f[i,j+1]\right) - \left(f[i,j+1] - f[i,j]\right)$$

$$= f[i,j+2] - 2f[i,j+1] + f[i,j]$$

but centred at $[i,j+1]$, so replace $[j]$ with $[j-1]$

Laplacian Operator

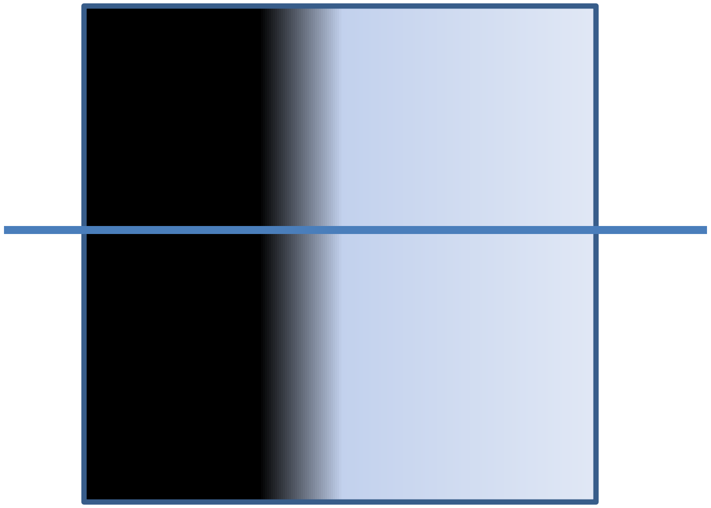
but centred at $[i, j+1]$, so replace $[j]$ with $[j-1]$

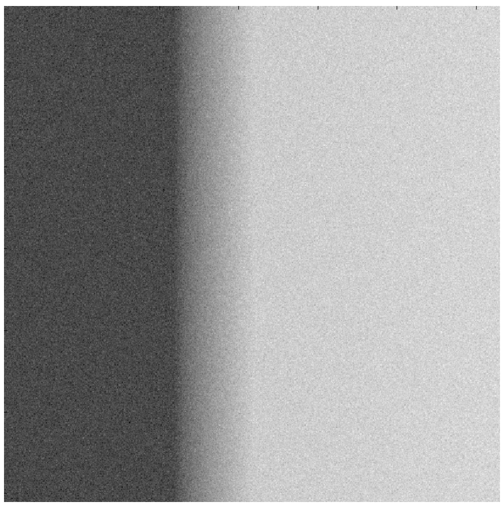
$$\frac{\partial^2 f}{\partial x^2} = f[i, j+1] - 2f[i, j] + f[i, j-1]$$

$$\therefore \frac{\partial^2 f}{\partial y^2} = f[i+1, j] - 2f[i, j] + f[i-1, j]$$

$$\therefore \nabla^2 \simeq \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplacian Operator





Highly Directed Work

- Laplacian of Gaussian
- Gaussian (Canny) edge detection

Labs/Tutorial

- See Group Number assignment on Canvas