Lectures 6- 7-8: Edge Detection

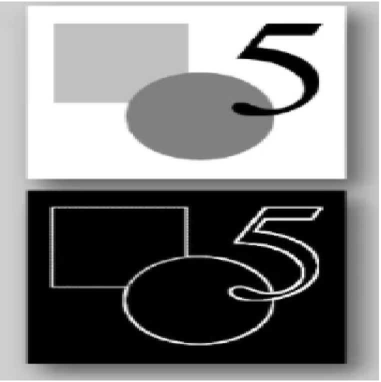
Dr. V Masilamani
masila@iiitdm.ac.in

Department of Computer Science and Engineering
IIITDM Kancheepuram
Chennai-127

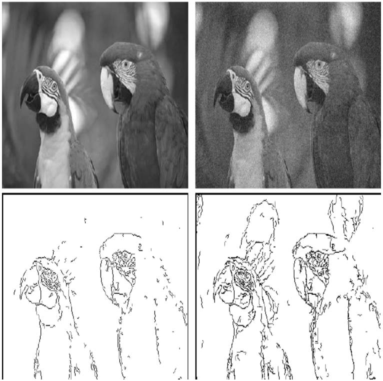
Overview

1. What is edge?
2. Why to detect edge?
3. Which features of object cause edges in its image
4. How to detect edge?
   - Edge detection using first derivative operators
   - Edge detection using second derivative operators
5. Pre-processing before applying edge detection operator
6. Some applications of edge detection
7. Some Classical Eldge Detection Algorithms
   - Canny Edge Detection Algorithm
   - Marr Hildreth Edge Detection Algorithm
8. Acknowledgements

What is edge



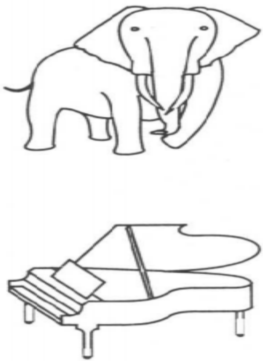
What is edge (cont.)



Why to detect edge?

- Edge in image provides structural information of object in the scene

Why to detect edge? (cont.)

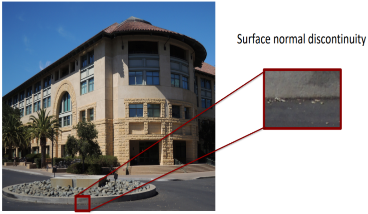


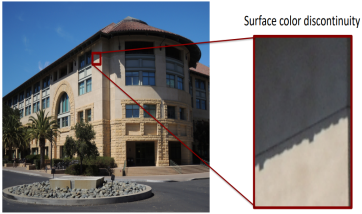
Why to detect edge? (cont.)

- Structural information of object is more important than the other information such as colour

Which features of object cause edges in its image

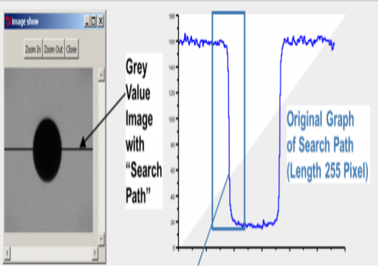
- Discontinuity in surface normal of the object
- Discontinuity in depth
- Discontinuity in colour of the surface of the object
- Discontinuity in light intensity

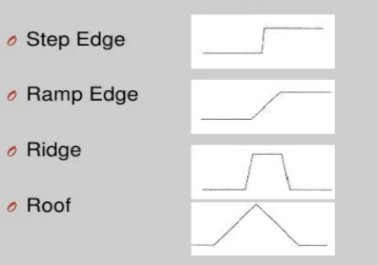


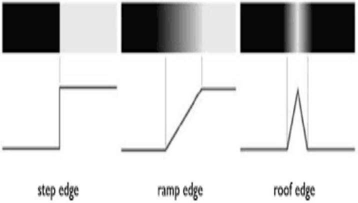


Models of edge

- Edge Point: If intensity profile changes at a point p significantly, then p is called as edge point
- Edge Map: Set of edge points is called as edge map



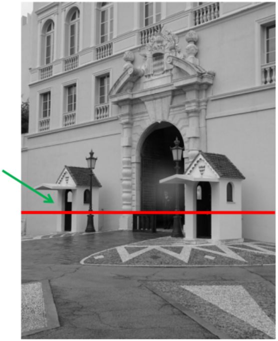


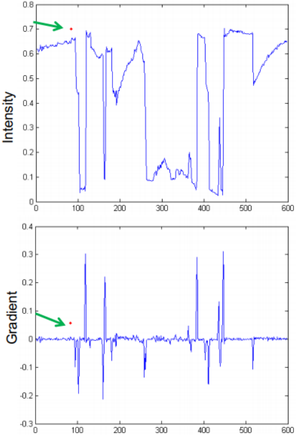


Edge detection using first derivative operators

- Let $f(x)$ be a function
- $$f'(x) = \lim_{h \to 0} (f(x+h) - f(x))/h$$
- Discrete Approximation 1: $f'(x) = f(x+1) - f(x)$ (Sub h=1)
  - $f'(x) = Cov(f(x), w(x))$, where $w = (1, -1)$ —(1)
    - $f = (5, 5, 5, 5, 5, 10, 10, 10, 10, 10)$
    - $f' = 0, 0, 0, 0, 5, 0, 0, 0, 0, 0$
  - An Approximation to (1), $f'(x) = Cov(f(x), w(x)$, where $w = (1, 0, -1)$ ——(2)
    - $f = (5, 5, 5, 5, 5, 10, 10, 10, 10, 10)$
    - $f' = 0, 0, 0, 0, 5, 5, 0, 0, 0, 0$
- Discrete Approximation 2: $f'(x) = f(x) - f(x-1)$ (Sub h=-1)

Intensity profile





- How to detect edge point in 1D signal f?
  $if(f'(x) > T_0)$ then x is an edge point

First Derivative Filters for Images

- Robert
  Along X direction

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

  Along Y direction

$$\begin{pmatrix} 1 & -1 \end{pmatrix}$$

  Along diagonal direction

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

  Along anti diagonal direction

Edge detection using first derivative operators (cont.)

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

- How to detect edge point in 2D signal f?
  - If $|\nabla(f(x,y))| > T_0$, then (x,y) is an edge pixel

Issues with Robert Filters

- Not center aligned
- Noise pixels are also detected as edge pixel

Approximating to make it center aligned, and remove noise before taking derivatives

- Instead of

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

, consider

$$\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

- Remove noise using sum filter before applying derivatives

Edge detection using first derivative operators (cont.)

- Prewitt filter along X-dir (to detect horizontal edge) is defined as

$$P_x = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

Prewitt in X- dir filter can be seen as convolution of smoothing and differentiating operator as follows.

$$\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

Edge detection using first derivative operators (cont.)

- Similarly along Y dir (to detect vertical edge), the Prewit filter will be

$$P_y = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Prewitt in Y-dir filter can be seen as convolution of smoothing and differentiating operator as follows.

$$\begin{pmatrix} 1 & 0 & -1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

- Horizontal Edge Detection: $f_x = E_h(f) = f * P_x$
- Vertical Edge Detection: $f_y = E_v(f) = f * P_y$

Edge detection using gradient:

Given function $f(x,y)$

Gradient vector $\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$

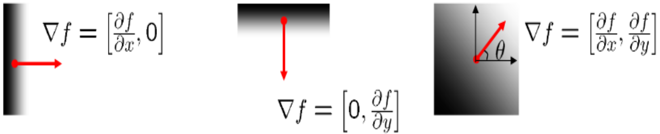
Gradient magnitude $|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2}$

Gradient direction $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

Edge detection using first derivative operators (cont.)

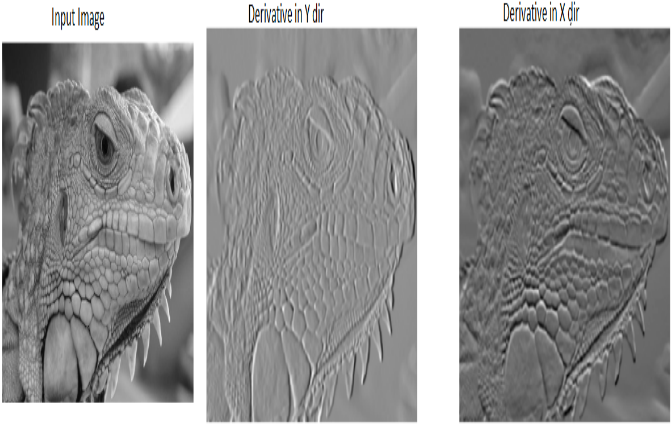
Edge detection using gradient:

The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



- Edge Detection:
  if($|\nabla(f(x,y))| > T$) then $E(x,y) = 1$
  else $E(x,y) = 0$

Edge detection using gradient:



Edge detection using first derivative operators (cont.)

- Sobel filter along X-dir (to detect horizontal edge) is defined as

$$S_x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Sobel in X- dir filter can be seen as convolution of smoothing and differentiating operator as follows.

$$\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} * \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$$

- Similarly along Y dir (to detect vertical edge), the Sobel filter will be

$$S_y = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

Sobel in Y-dir filter can be seen as convolution of smoothing and differentiating operator as follows.

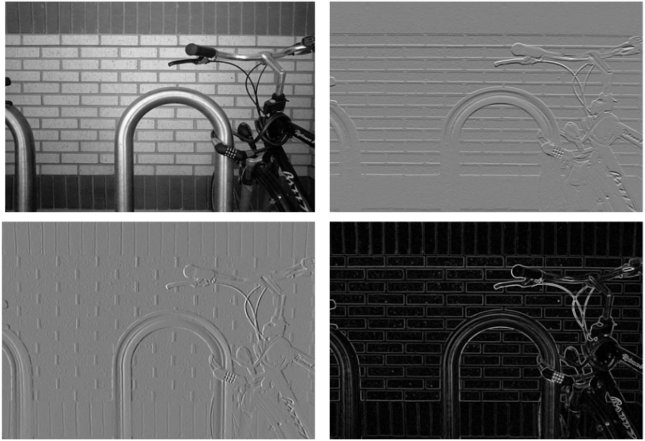
Edge detection using first derivative operators (cont.)

$$\begin{pmatrix} 1 & 0 & -1 \end{pmatrix} * \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

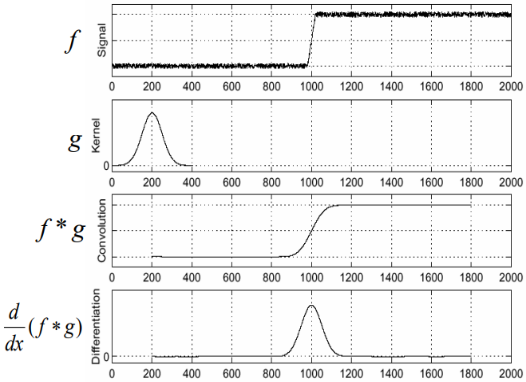
- Horizontal Edge Detection: $f_x = E_h(f) = f * S_x$
- Vertical Edge Detection: $f_y = E_v(f) = f * S_y$
- Edge Detection Using gradient:

  if($|\nabla(f(x, y))| > T$) then $E(x, y) = 1$
  else $E(x, y) = 0$

Edge detection using gradient:



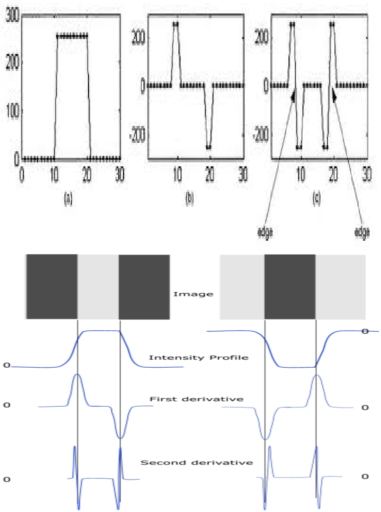
Summary of edge detection using gradient:



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

What are the issues with first derivative based filters

- Even when intensity profile is slowly varying at a point p, p may be be declared as edge point
- Localization of edge point is not effective



Laplacian Filter Design

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

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \rightarrow \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

x kernel

$$\frac{\partial^2 f}{\partial y^2} = f(y+1) + f(y-1) - 2f(y) \rightarrow \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

y kernel

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Edge detection using second derivative operators (cont.)

Properties of Lapacian filter

- The output of the filter is 0, when applied on homogeneous region
- The magnitude of output of the filter is high, when applied on edges
- **Zero Crossing**(+ve/-ve followed by -ve/+ve) indicates the presence of edges
- Zero Crossing can be used to located the edge point effectively

Other variants of Laplacian Filter -satisfying above properties

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

1	1	1
1	-8	1
1	1	1

Edge detection using second derivative operators (cont.)

2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8

A sample image containing a vertical step edge.

0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0

2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8

A sample image containing a vertical ramp edge.

0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Smoothing followed by Laplacian Filter Design -Laplacian of Gaussian Filter(LOG)

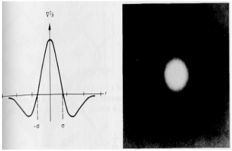
$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(σ determines the degree of smoothing, mask size increases with σ)

- It can be shown that:

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

$$\nabla^2 G(x,y) = \left(\frac{r^2-\sigma^2}{\sigma^4}\right)e^{-r^2/2\sigma^2}, \quad (r^2 = x^2 + y^2)$$



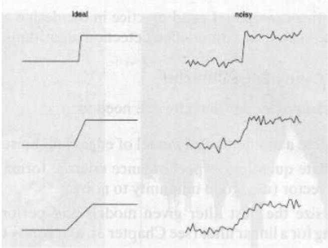
5 × 5 Laplacian of Gaussian mask

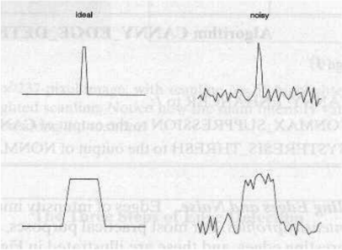
0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

17 × 17 Laplacian of Gaussian mask

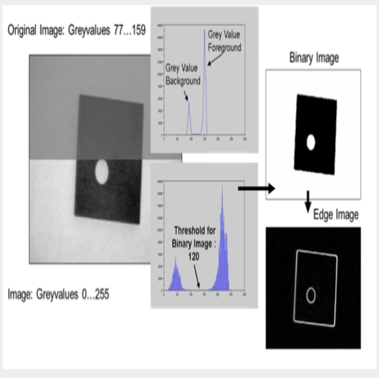
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	4	12	21	24	21	12	4	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0

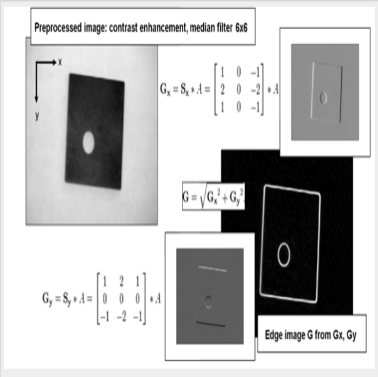
- To obtain accurate edges, do pre-processing such as
  - Noise removal
  - Contrast enhancement
  - Sharpen the image



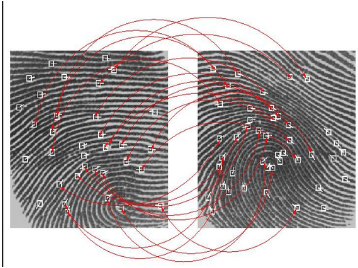


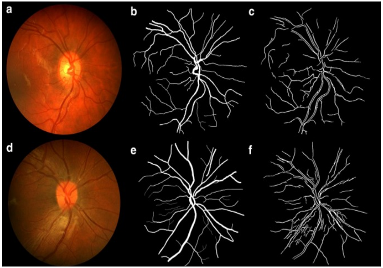
Pre-processing before applying edge detection operator (cont.)





- Object detection
- Measure computation





Input: Image f; Output: Edge Image E

1. Find $g = Conv(f, h)$, where h is the Gaussian Filter (Smoothing Step)
2. Find Gradient of g, say g
3. Find Magnitude M and Orientation θ of g
4. Find $S(x, y) = 0$ if $M(x, y) < M(x', x')$ for some (x', y') in the direction of θ ; $S(x, y) = M(x, y)$ otherwise (Non-Maximal Suppression Step)
5. $E(x, y) = 0$ if $S(x, y) < T_0$ $e(x, y) = 1$ if $S(x, y) > T_1$ (Double Thresholding)
6. $E(x, y) = 1$ if $E(x, y)$ is not yet defined, and $E(x', y') = 1$, where (x', y') is adjacent to (x, y) (Edge tracing step)

Step 0

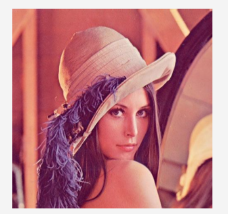


Figure 1: Original

Figure 2: Black and White

Step 1

Figure 3: Gaussian Blur

Figure 4: Gx

Figure 5: Gy

Figure 6: Gradient Magnitude

Step 3

Figure 7: Non Maximum Suppression with Interpolation

Figure 8: Step 5:Double Thresholding

Figure 9: Step 6:Edge Tracking

Figure 10: Step 7:Final Result from Canny Edge Detection Algorithm

Marr Hildreth Edge Detection Algorithm

Input: Image f; Output: Edge Images E(x,y. σ)

1. Find $g = Conv(f, h)$, where h is the Gaussian Filter with SD σ (Smoothing Step)
2. Find Laplacian of g, say $\Delta^2 g$
3. Ddefine $E(x, y, \sigma) = 1$ if Zero crossing occurs at (x, y) at the scale σ
   $E(x, y, \sigma) = 0$ otherwise

$\sigma = 1$ $\qquad$ $\sigma = 2$ $\qquad$ $\sigma = 3$

Kernels (h1)

[4 60 272 450 27 2 60 4]

[4 19 60 146 272 397 450 397 272 146 60 19 4]

[1 4 12 29 60 112 185 272 360 425 450 425 360 272 185 112 60 29 12 4 1]

Figure 11: Effect of scale(SD)

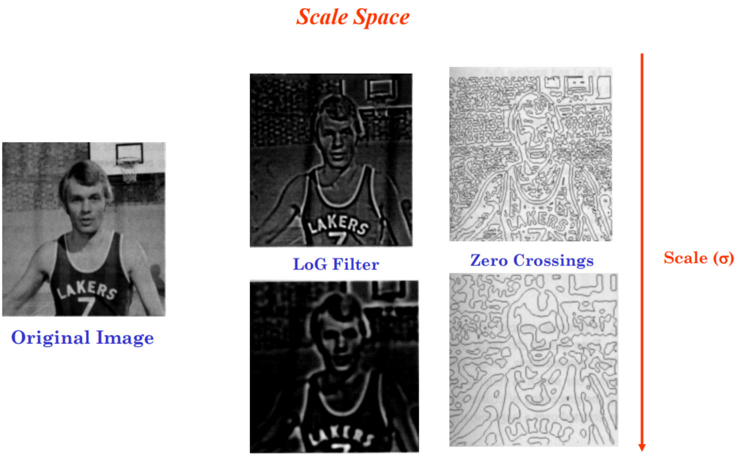


Figure 12: Scale Space

Image Sharpening using edge detection

Input: Image f; Output: g, a sharpened version of f

- Find Edge image e of f
- g(x,y)= f(x,y) +c e(x,y), for some c

Assignement 1:

- Capture your face image, say f
- Sharpen f using Canny edge detection
- Sharpen f using Marr Hildreth edge detection
- Note: choose c manually such that sharpening is good

Acknowledgements

- Images are downloaded from internet sources

Thank You! :)