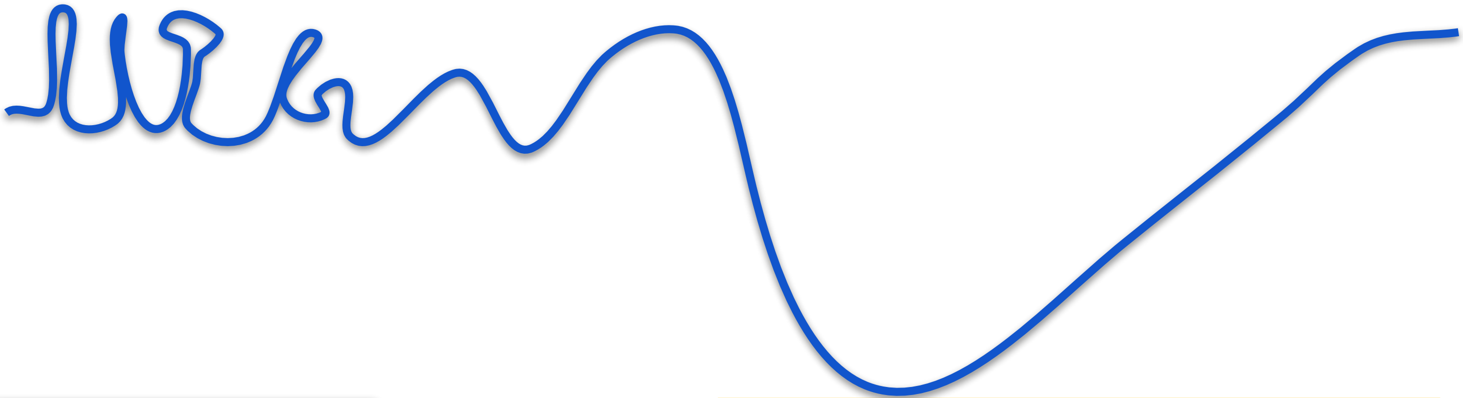
Computing with Signals



DA 623 | Jan - May 2024 | IIT Guwahati

Instructors: Neeraj Sharma

Lecture-17

Raise your hand to explain what information is this following code snippet providing?

```python
import matplotlib.pyplot as plt
import numpy as np

im = plt.imread('./rohini_godbole.jpeg')
print('Datatype: ', type(im))
print('Size of data:', im.shape)
```

```
Datatype:  <class 'numpy.ndarray'>
Size of data: (500, 600, 3)
```

Raise your hand to explain what information is this following code snippet providing?

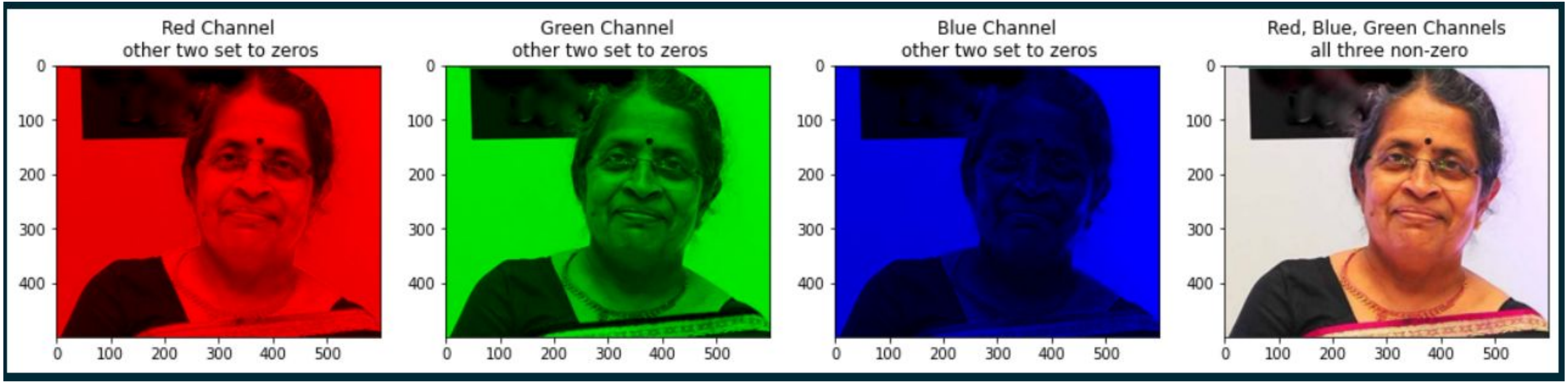
```python
import matplotlib.pyplot as plt
import numpy as np

im = plt.imread('./rohini_godbole.jpeg')
print('Datatype: ', type(im))
print('Size of data:', im.shape)
```

```
Datatype:  <class 'numpy.ndarray'>
Size of data: (500, 600, 3)
```

It is a numpy array - M x N x C, where

- M is the number of rows (image height)
- N is number of columns (image width)
- C is number of layers (image channels)



Rohini Godbole is an Indian physicist and academic specializing in elementary particle physics: field theory and phenomenology. She is currently a professor at the Centre for High Energy Physics, Indian Institute of Science, Bangalore.

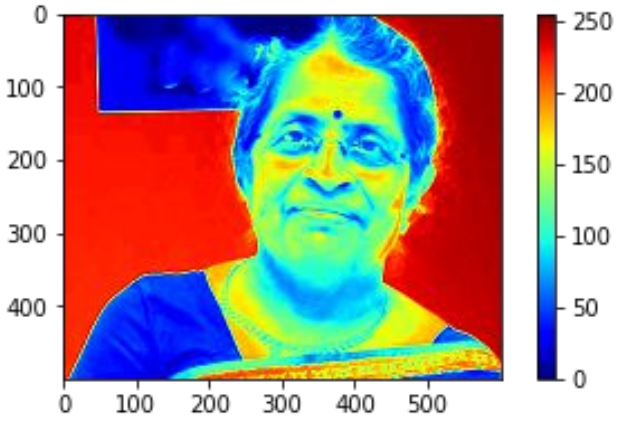
Raise your hand to explain what information is this following code snippet providing?

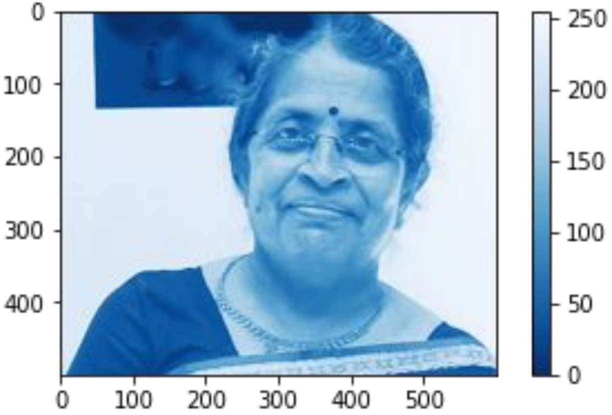
```python
def rgb2gray(im):
    """
    Converts RGB image to grayscale image
    """
    # write one line to compute the mean across the channel
    im_temp = im.mean(axis=2)
    return im_temp

# convert color image to grayscale
im_gs = rgb2gray(im)

# plot image
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=[7,7])
ax.imshow(im_gs, cmap='gray')
ax.set_title('Grayscale')
plt.show()
```

Choose colorbar with care
It impacts visualization





Perspective | Open access | Published: 28 October 2020

The misuse of colour in science communication

Fabio Crameri, Grace E. Shephard & Philip J. Heron

Nature Communications **11**, Article number: 5444 (2020) | Cite this article

Choose colorbar with care
It impacts visualization

Perspective | Open access | Published: 28 October 2020

The misuse of colour in science communication

Fabio Crameri, Grace E. Shephard & Philip J. Heron

Nature Communications **11**, Article number: 5444 (2020) | Cite this article

291k Accesses | **353** Citations | **1323** Altmetric | Metrics

Raise your hand to explain what information is this following code snippet providing?

- Choose `thres`
- Create `im_bw` from `im_gs` by setting all values $>$ `thres` to 255 and rest to 0.

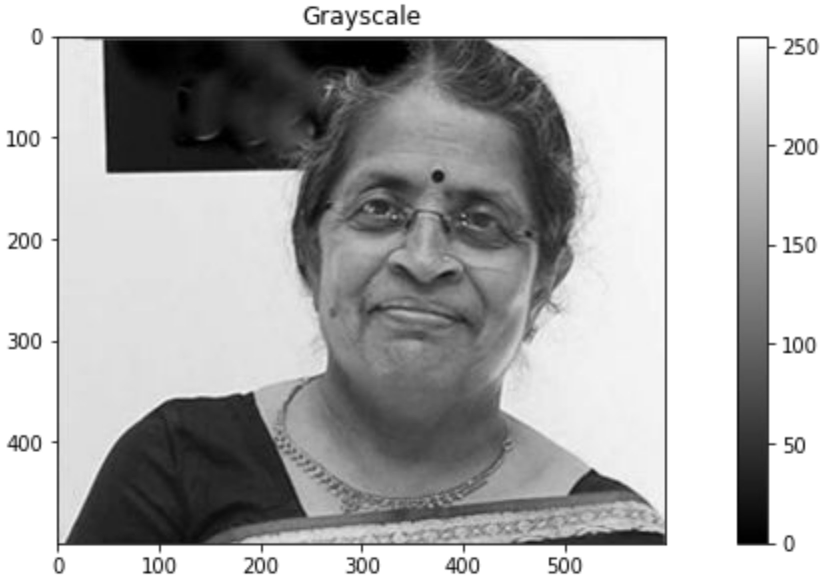
Raise your hand to explain what information is this following code snippet providing?

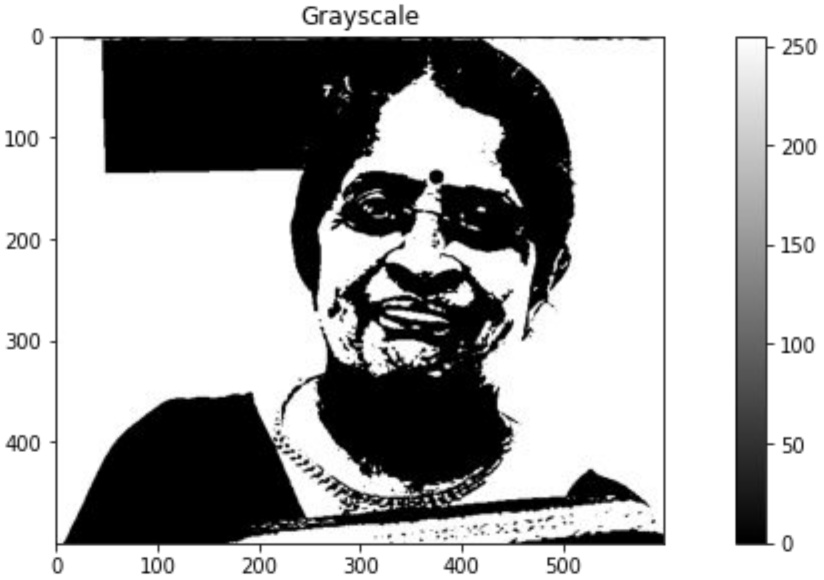
- Choose `thres`
- Create `im_bw` from `im_gs` by setting all values > `thres` to 255 and rest to 0.

```
thres = 255//2
im_bw = np.zeros(im_gs.shape)
im_bw[im_gs>thres] = 255
```

Raise your hand to explain what information is this following code snippet providing?

- Choose `thres`
- Create `im_bw` from `im_gs` by setting all values > `thres` to 255 and rest to 0.

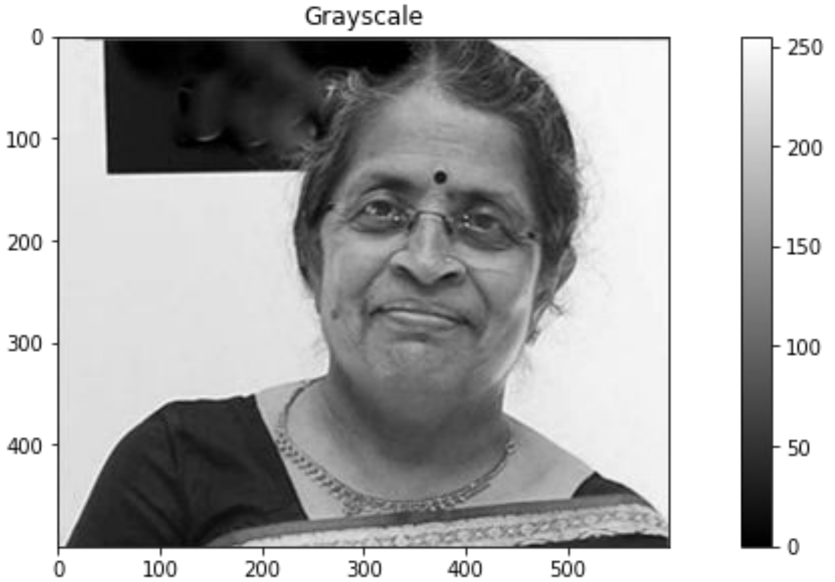


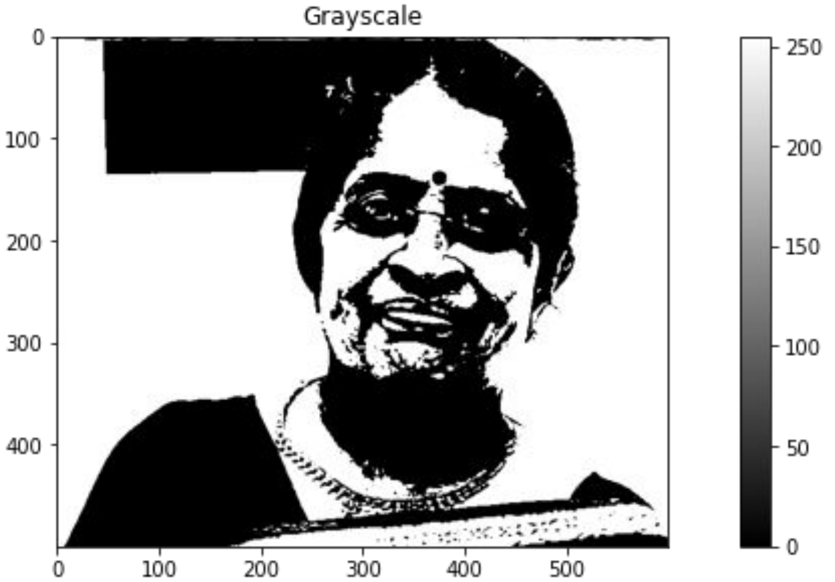


The black and white image is losing a lot of details.

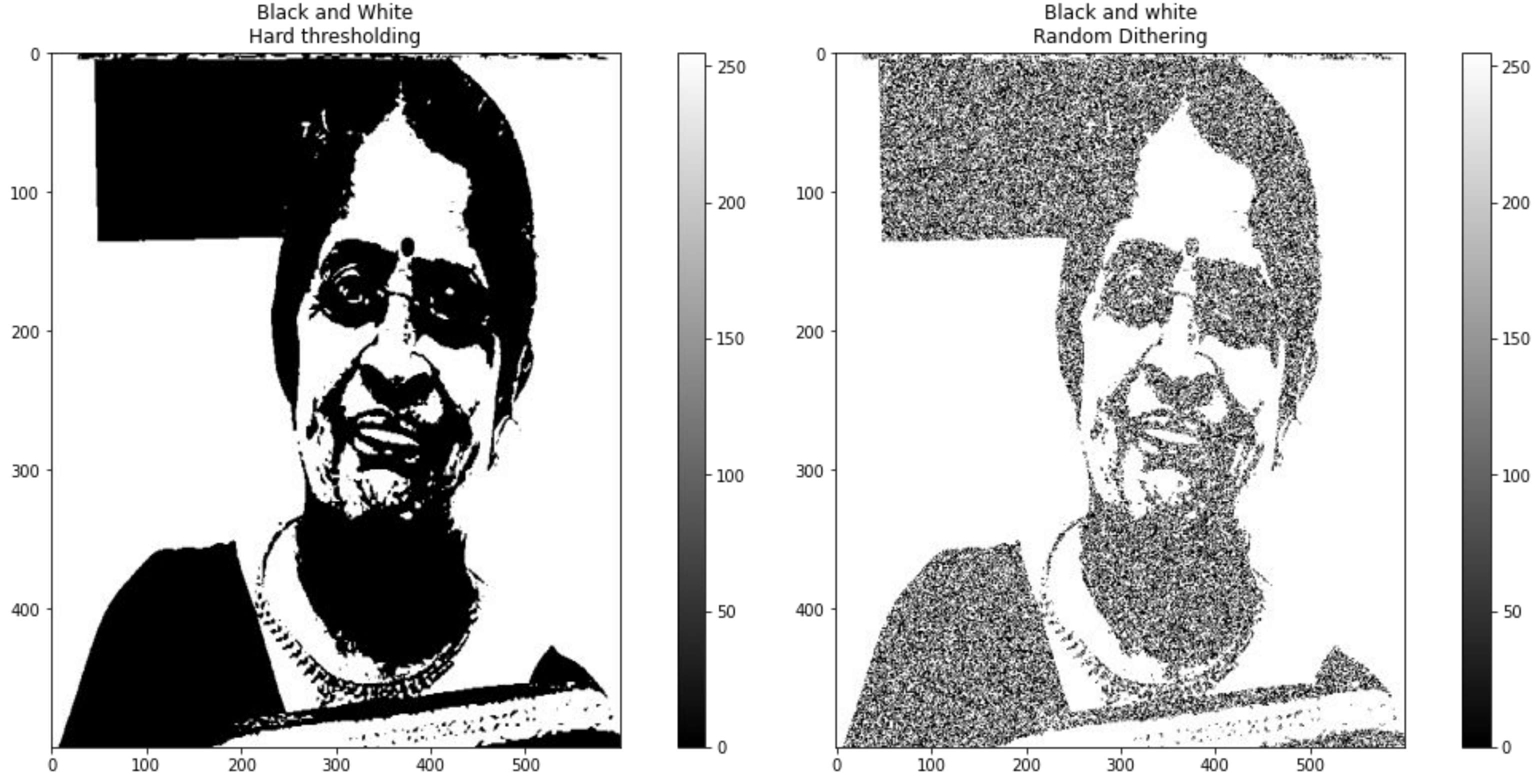
Can we improvise the black and white image?

- Choose `thres`
- Create `im_bw` from `im_gs` by setting all values > `thres` to 255 and rest to 0.





```python
# make binary image using random switching the 0 intensty pixels
im_rd = np.zeros(im_gs.shape)
im_rd[im_gs>thres] = 255
im_random = np.random.randint(0,2,im_gs.shape)
im_rd[im_gs<=thres] = 255*im_random[im_gs<=thres]
```



The black and white image is losing a lot of details.

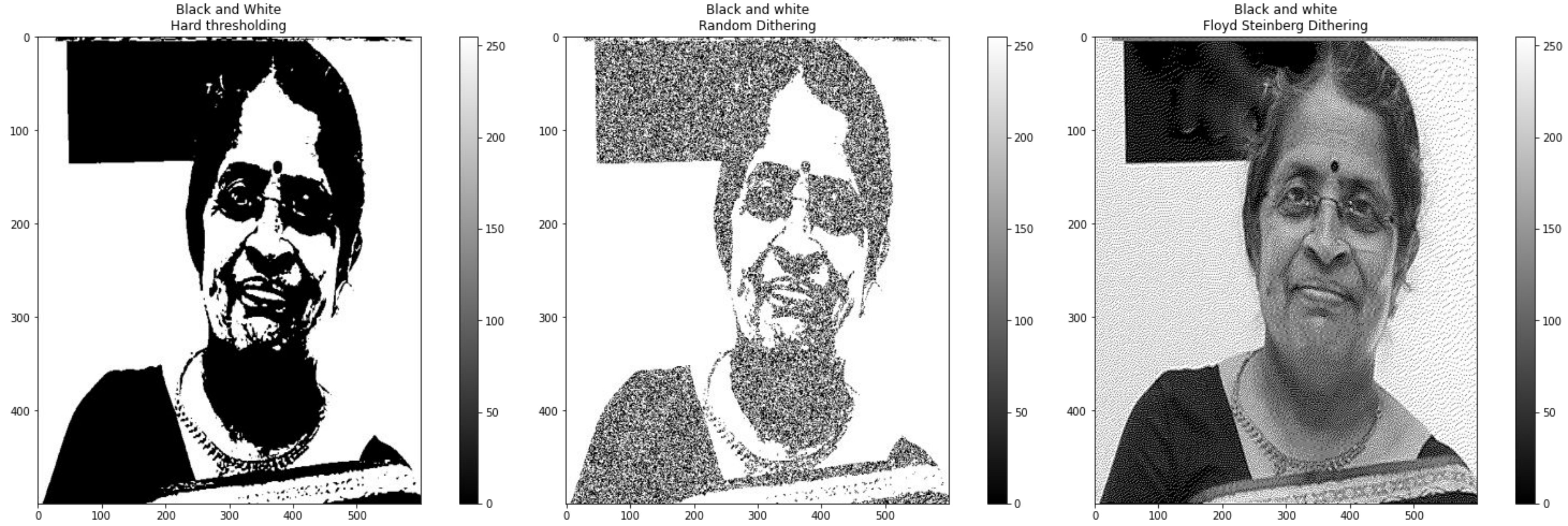
Can we improvise the black and white image?

```python
def floyd_steinberg(im_gs):
    """
    Implements the Floyd steinberg algorithm to create binary image from grayscale image
    """
    thres = 255//2
    im_temp = im_gs.copy()
    for row in range(0,im_gs.shape[0]-1):
        for col in range(1,im_gs.shape[1]-1):
            old = im_temp[row, col]
            if im_temp[row, col] > thres:
                im_temp[row, col] = 255
            else:
                im_temp[row, col] = 0

            quant_error = old - im_temp[row, col]
            im_temp[row, col+1] = im_temp[row, col+1] + quant_error * 7 /16
            im_temp[row+1, col-1] = im_temp[row+1, col-1] + quant_error * 3 /16
            im_temp[row+1, col] = im_temp[row+1, col] + quant_error * 5 /16
            im_temp[row+1, col+1] = im_temp[row+1, col+1] + quant_error * 1 /16

    im_temp[im_temp<thres] = 0
    im_temp[im_temp>=thres] = 255

    return im_temp
```


Black and White
Hard thresholding
Black and white
Random Dithering
Black and white
Floyd Steinberg Dithering
0
100
200
300
400
500
0
50
100
150
200
250

Floyd–Steinberg dithering

The black and white image is losing a lot of details.

Can we improvise the black and white image?

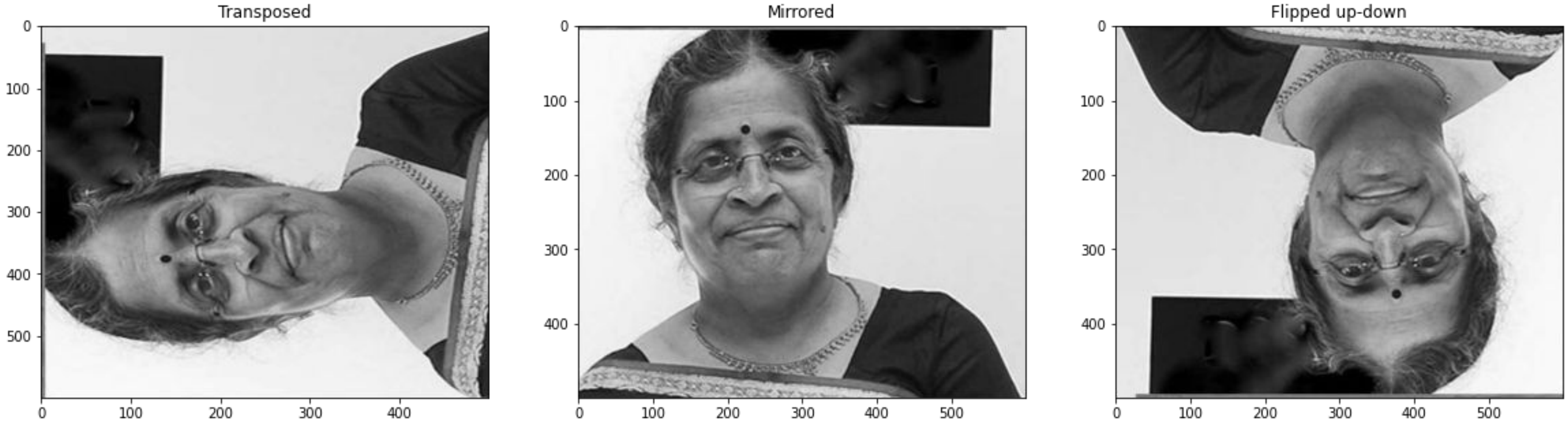
```python
def floyd_steinberg(im_gs):
    """
    Implements the Floyd steinberg algorithm to create binary image from grayscale image
    """
    thres = 255//2
    im_temp = im_gs.copy()
    for row in range(0,im_gs.shape[0]-1):
        for col in range(1,im_gs.shape[1]-1):
            old = im_temp[row, col]
            if im_temp[row, col] > thres:
                im_temp[row, col] = 255
            else:
                im_temp[row, col] = 0

            quant_error = old - im_temp[row, col]
            im_temp[row, col+1] = im_temp[row, col+1] + quant_error * 7 /16
            im_temp[row+1, col-1] = im_temp[row+1, col-1] + quant_error * 3 /16
            im_temp[row+1, col] = im_temp[row+1, col] + quant_error * 5 /16
            im_temp[row+1, col+1] = im_temp[row+1, col+1] + quant_error * 1 /16

    im_temp[im_temp<thres] = 0
    im_temp[im_temp>=thres] = 255

    return im_temp
```

Simple image manipulation.



Some of the slides in the following material is borrowed from the following excellent course material:

course by Prof. Shree K. Nayar, Columbia University

Reference: https://fpcv.cs.columbia.edu/Monographs (used here for educational purpose)

Binary Images

Binary Images

Binary Image: Can have only two values (0 or 1).
Simple to process and analyze.

Reference:
https://fpcv.cs.columbia.edu/Monographs

Making Binary Images

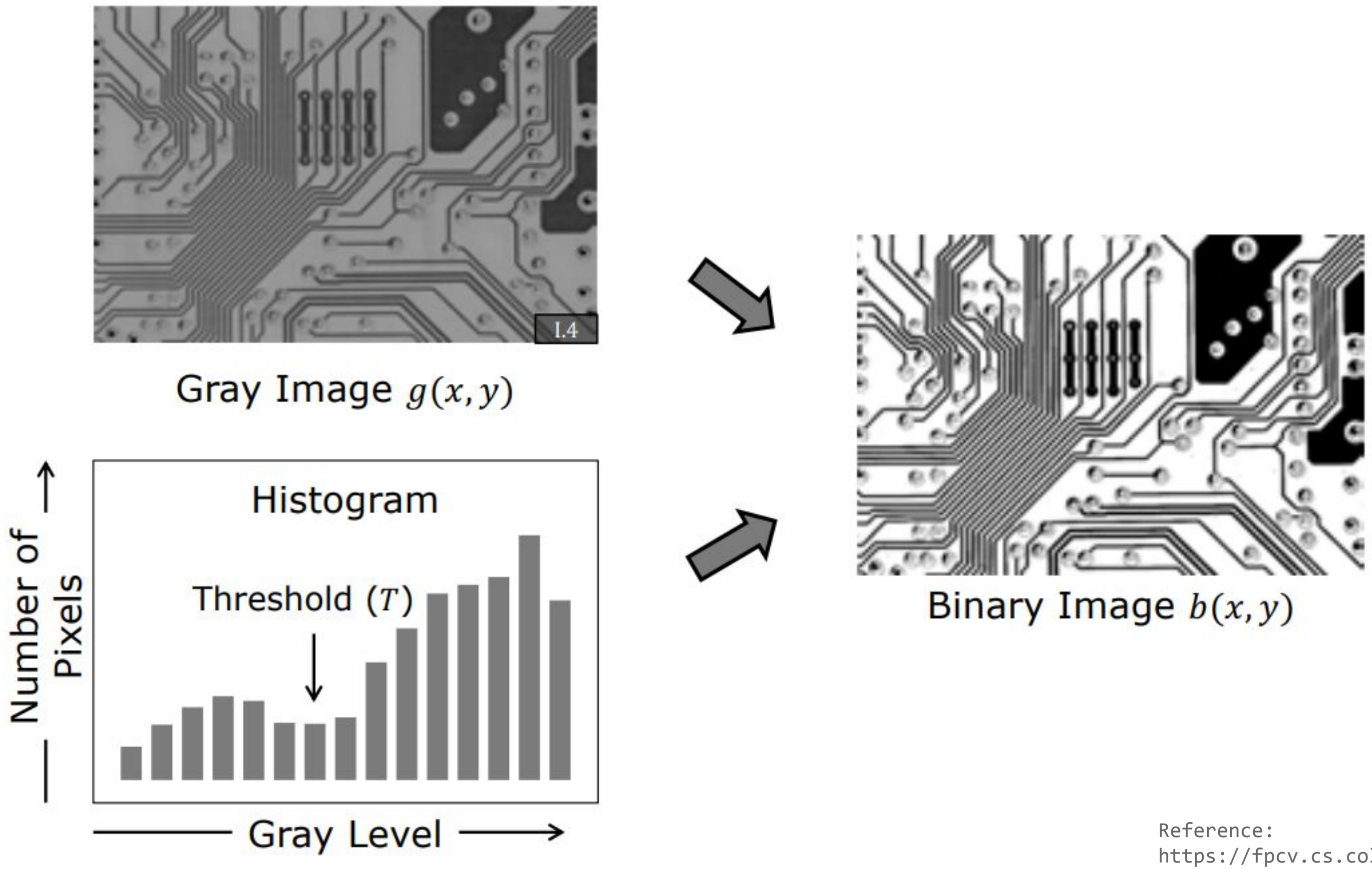
Binary Image $b(x,y)$: Usually obtained from Gray-level image $g(x,y)$ by Thresholding.

Characteristic Function:

$$b(x,y) = \begin{cases} 0, & g(x,y) < T \\ 1, & g(x,y) \geq T \end{cases}$$

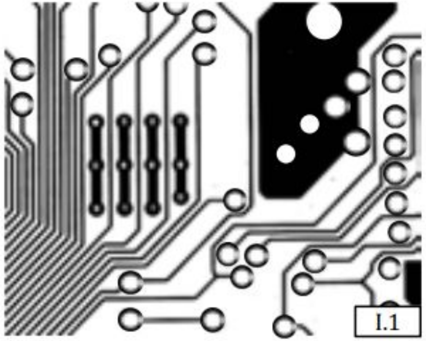
Reference:
https://fpcv.cs.columbia.edu/Monographs

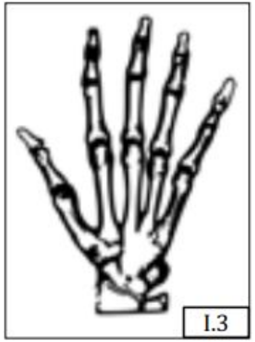
Selecting a Threshold (T)



Reference:
https://fpcv.cs.columbia.edu/Monographs

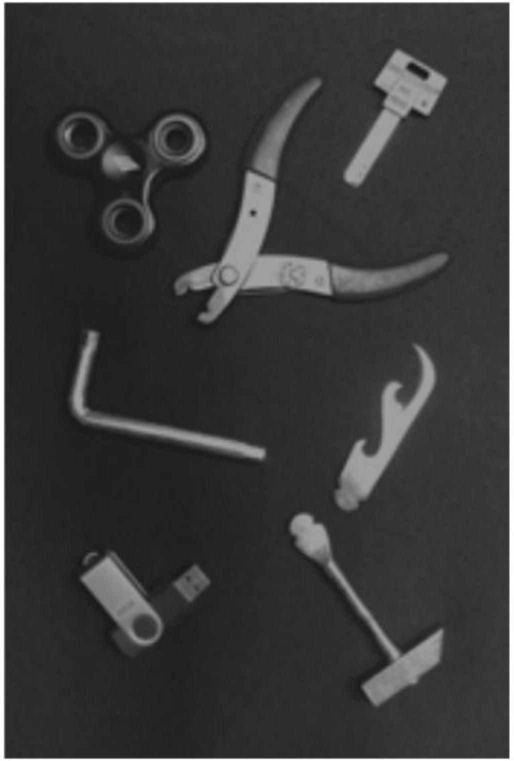
Examples of Binary Images





Reference:
https://fpcv.cs.columbia.edu/Monographs

Capturing a Binary Images



Backlighting

Reference:
https://fpcv.cs.columbia.edu/Monographs

Binary Images

Binary Image: Can have only two values (0 or 1).
Simple to process and analyze.

Topics:

(1) Geometric Properties

(2) Segmenting Binary Images

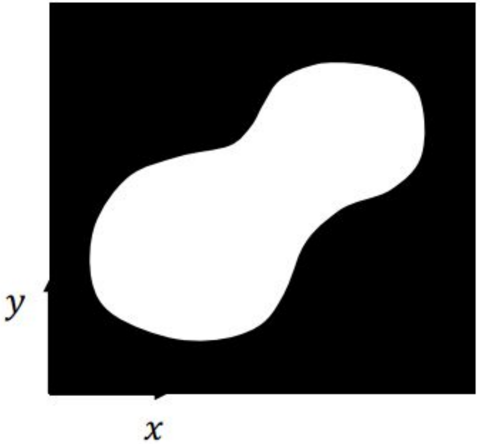
(3) Iterative Modification

Reference:
https://fpcv.cs.columbia.edu/Monographs

Geometric Properties of Binary Images

Assume:

- $b(x, y)$ is continuous
- Only one object

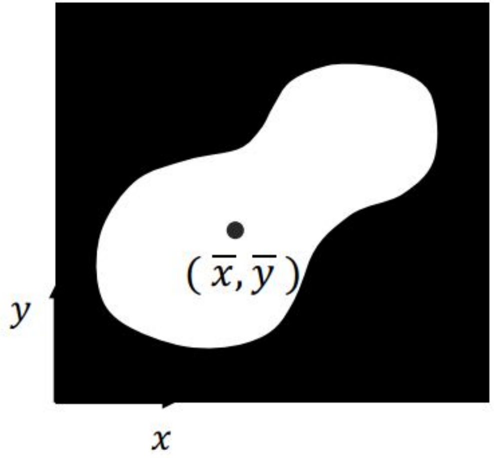


Reference:
https://fpcv.cs.columbia.edu/Monographs

Area and Position

Area: (Zeroth Moment)

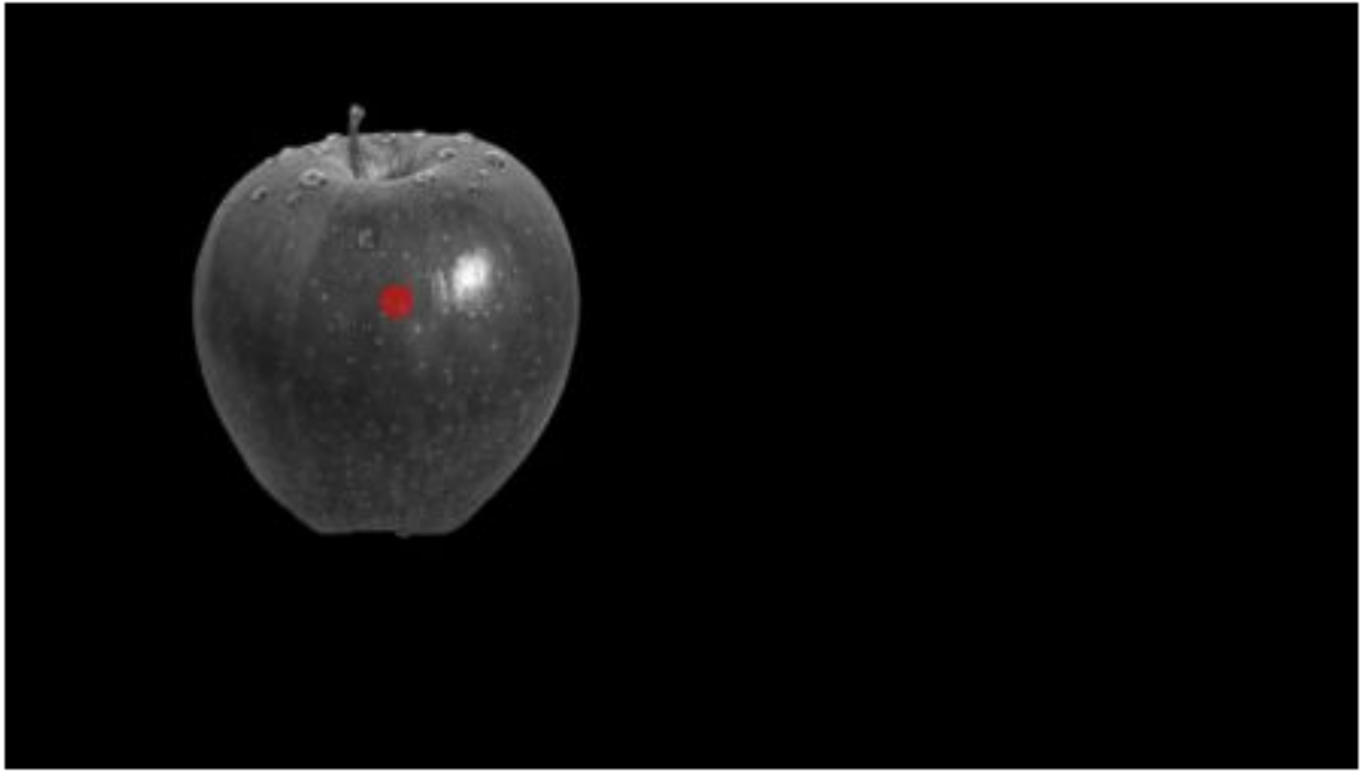
$$A = \iint_I b(x,y)\,dx\,dy$$



Position: Center of Area (First Moment)

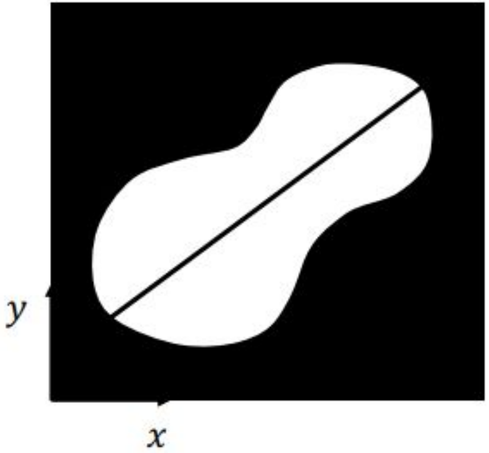
$$\overline{x} = \frac{1}{A}\iint_I x\,b(x,y)\,dx\,dy \quad , \quad \overline{y} = \frac{1}{A}\iint_I y\,b(x,y)\,dx\,dy$$

Reference:
https://fpcv.cs.columbia.edu/Monographs



Orientation

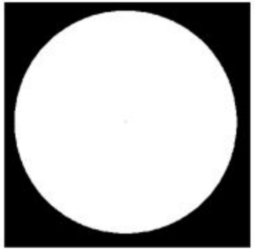
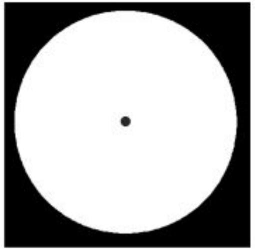
Difficult to define!



Use: Axis of Least Second Moment

Reference:
https://fpcv.cs.columbia.edu/Monographs

Examples

Gray Image	Binary Image	Orientation	Roundedness
			0.19
			0.49
			1.0

Reference:
https://fpcv.cs.columbia.edu/Monographs

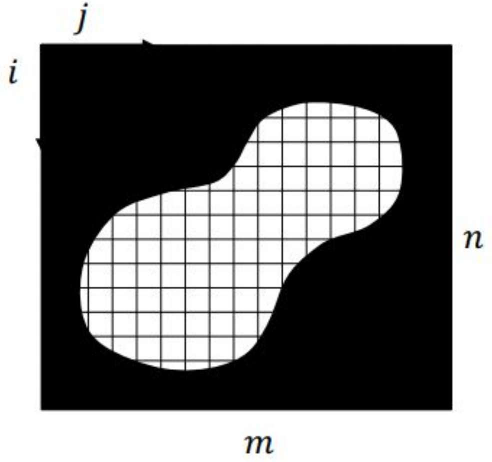
Given a binary image of an object, we can

- compute its area, location, and its maximum and minimum moments
- the area and the two moments are useful features because they are not affected by translation and rotation of the object
- use case: distinguish between a set of objects

After an object is recognized, the position and the orientation are used to enable a robot to pick up the object.

Discrete Binary Images

b_{ij}: Value at cell (pixel) in row i and column j.
Assume pixel area = 1.



Area: $$A = \sum_{i=1}^{n}\sum_{j=1}^{m} b_{ij}$$

Position: Center of Area (First Moment)

$$\bar{x} = \frac{1}{A}\sum_{i=1}^{n}\sum_{j=1}^{m} i b_{ij} \qquad \bar{y} = \frac{1}{A}\sum_{i=1}^{n}\sum_{j=1}^{m} j b_{ij}$$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Discrete Binary Images

Second Moments:

$$a' = \sum_{i=1}^{n}\sum_{j=1}^{m} i^2 b_{ij} \qquad b' = 2\sum_{i=1}^{n}\sum_{j=1}^{m} ij b_{ij} \qquad c' = \sum_{i=1}^{n}\sum_{j=1}^{m} j^2 b_{ij}$$

Note: a', b', c' are second moments w.r.t origin.
a, b, c (w.r.t. center) can be found from a', b', c', $\overline{x}$, $\overline{y}$, A

Hint: Expand $a = \sum_{i=1}^{n}\sum_{j=1}^{m}(i - \bar{x})^2 b_{ij}$ and represent in terms of a', $\overline{x}$, A.

Reference:
https://fpcv.cs.columbia.edu/Monographs

Multiple Objects

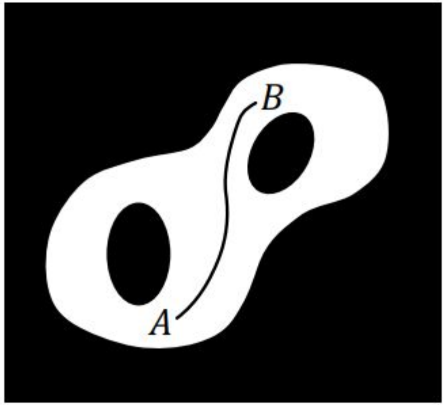
Need to Segment image into separate Components

Non-Trivial!

Reference:
https://fpcv.cs.columbia.edu/Monographs

Connected Component

Maximal Set of Connected Points



A and B are connected if path exists between A and B along which $b(x,y)$ is constant.

Reference:
https://fpcv.cs.columbia.edu/Monographs

Connected Component Labeling

Region Growing Algorithm

(a) Find Unlabeled "Seed" point with $b = 1$.
If not found, Terminate.

(b) Assign New Label to seed point

(c) Assign Same Label to its Neighbors with $b = 1$

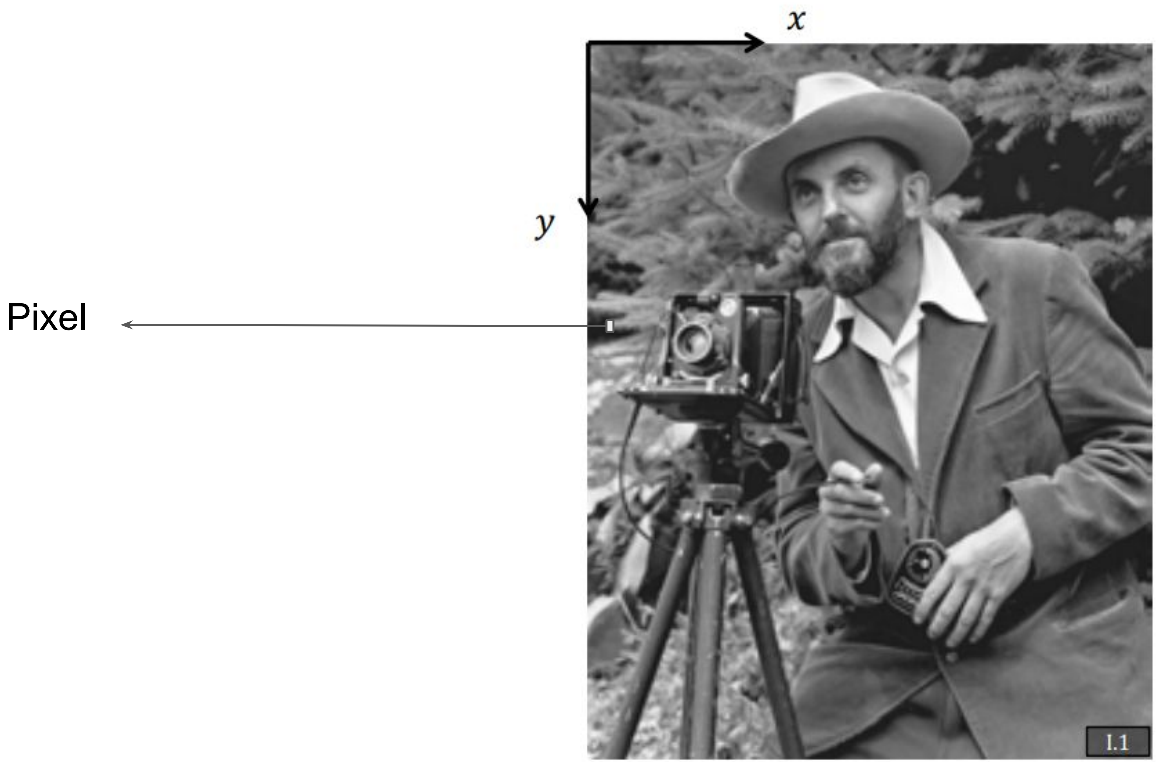
(d) Assign Same Label to Neighbors of Neighbors with $b = 1$. Repeat until no more Unlabeled Neighbors with b=1.

(e) Go to (a)

Reference:
https://fpcv.cs.columbia.edu/Monographs

Pixel Processing

Image as a Function



$f(x, y)$ is the image intensity at position (x, y)

Pixel (Point) Processing

Transformation T of intensity f at each pixel to intensity g:

$$g(x, y) = T\left(f(x, y)\right)$$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Point Processing

Original (f)

Darken $(f - 128)$

Lighten $(f + 128)$

Invert $(255 - f)$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Pixel Processing

Original (f)

Low Contrast $(f/2)$

High Contrast $(f * 2)$

Gray $(0.3f_R + 0.6f_G + 0.1f_B)$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Image Processing

Image as a Function

$f(x, y)$ is the image intensity at position (x, y)

Linear Shift Invariant System (LSIS)

$$f(x) \longrightarrow \boxed{\text{LSIS}} \longrightarrow g(x)$$

Study of Linear Shift Invariant Systems (LSIS) leads to useful image processing algorithms.

Reference:
https://fpcv.cs.columbia.edu/Monographs

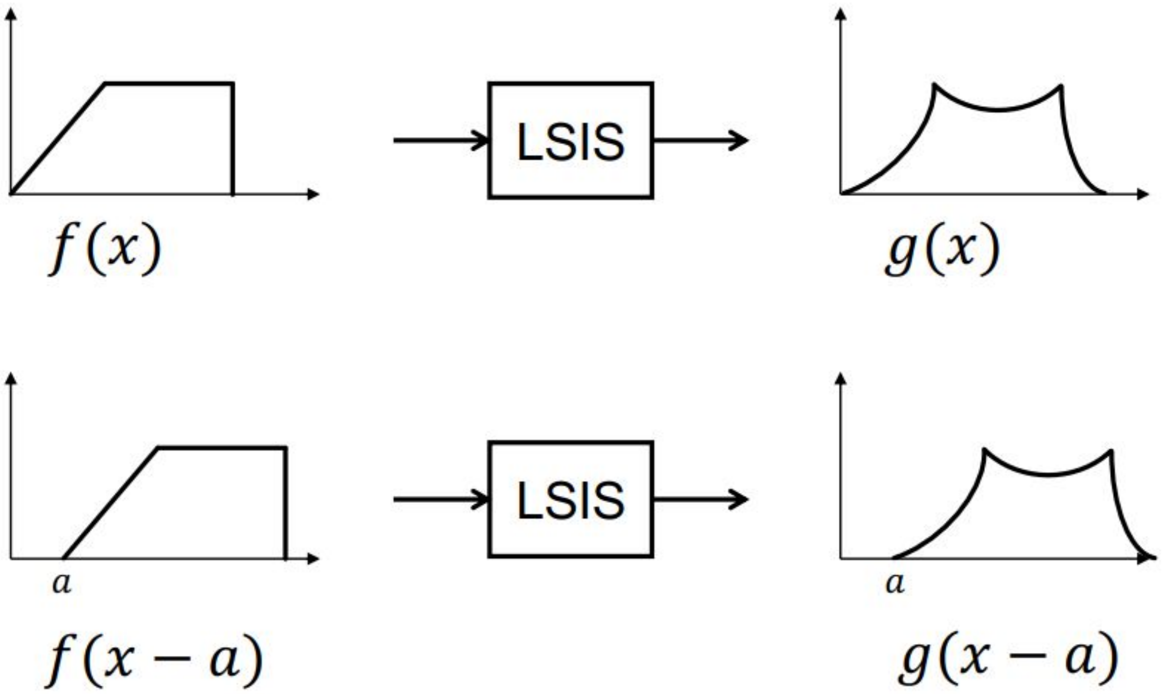
LSIS: Linearity

$$f_1 \longrightarrow \boxed{\text{LSIS}} \longrightarrow g_1 \qquad\qquad f_2 \longrightarrow \boxed{\text{LSIS}} \longrightarrow g_2$$

$$\alpha f_1 + \beta f_2 \longrightarrow \boxed{\text{LSIS}} \longrightarrow \alpha g_1 + \beta g_2$$

Reference:
https://fpcv.cs.columbia.edu/Monographs

LSIS: Shift Invariance

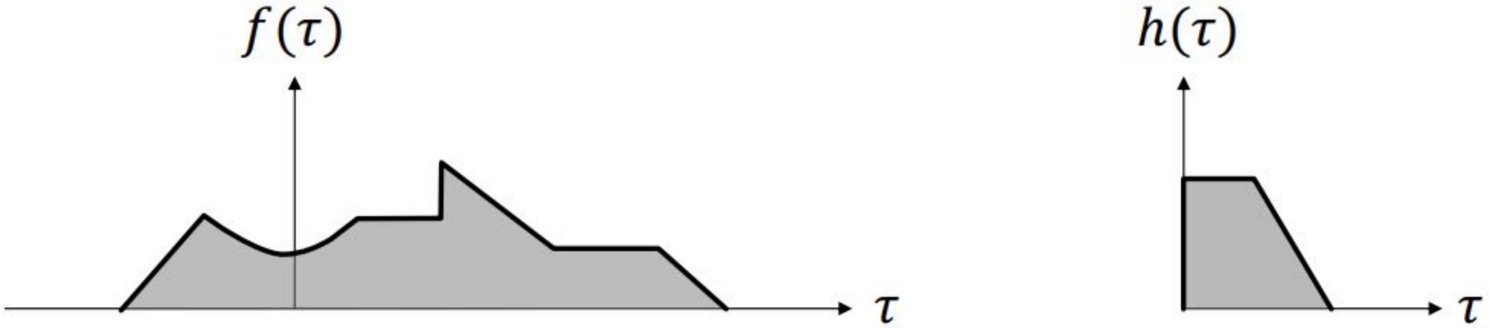


Reference:
https://fpcv.cs.columbia.edu/Monographs

Convolution

Convolution of two functions $f(x)$ and $h(x)$

$$\boxed{g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau}$$



Reference:
https://fpcv.cs.columbia.edu/Monographs

Convolution is LSIS

Linearity:

Let: $g_1(x) = \int_{-\infty}^{\infty} f_1(\tau)h(x-\tau)\,d\tau$ and $g_2(x) = \int_{-\infty}^{\infty} f_2(\tau)h(x-\tau)\,d\tau$

Then:

$$\int_{-\infty}^{\infty} (\alpha f_1(\tau) + \beta f_2(\tau))h(x-\tau)\,d\tau$$

$$= \alpha \int_{-\infty}^{\infty} f_1(\tau)h(x-\tau)\,d\tau + \beta \int_{-\infty}^{\infty} f_2(\tau)h(x-\tau)\,d\tau$$

$$= \alpha g_1(x) + \beta g_2(x)$$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Convolution is LSIS

Shift Invariance:

Let: $$g(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\,d\tau$$

Then:

$$\int_{-\infty}^{\infty} f(\tau - a)h(x-\tau)\,d\tau$$

$$= \int_{-\infty}^{\infty} f(\mu)h(x-a-\mu)\,d\mu \qquad \boxed{1}$$ (Substituting $\mu = \tau - a$)

$$= g(x-a)$$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Properties of Convolution

Commutative $a * b = b * a$

Associative $(a * b) * c = a * (b * c)$

Cascaded System

$$f \longrightarrow \boxed{h_1} \longrightarrow \boxed{h_2} \longrightarrow g$$

$$\equiv \quad f \longrightarrow \boxed{\quad h_1 * h_2 \quad} \longrightarrow g$$

$$\equiv \quad f \longrightarrow \boxed{\quad h_2 * h_1 \quad} \longrightarrow g$$

Reference:
https://fpcv.cs.columbia.edu/Monographs

2D Convolution

LSIS:

$$f(x,y) \longrightarrow \boxed{h(x,y)} \longrightarrow g(x,y)$$

Convolution:

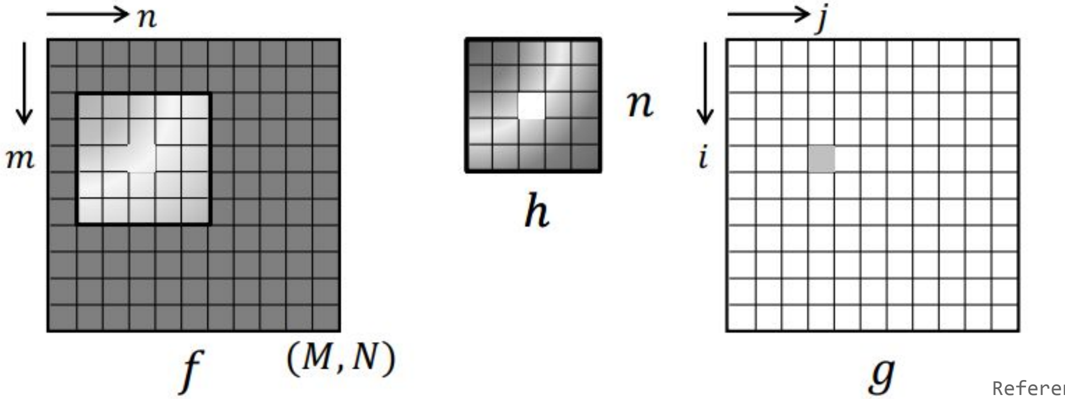
$$g(x,y) = \iint_{-\infty}^{\infty} f(\tau,\mu)h(x-\tau,y-\mu)\,d\tau d\mu$$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Convolution with Discrete Images

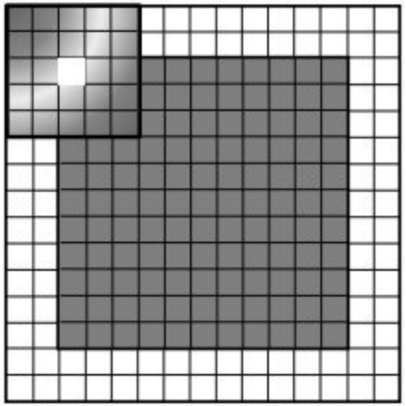
$$f[i,j] \longrightarrow \boxed{h[i,j]} \longrightarrow g[i,j]$$

$$g[i,j] = \sum_{m=1}^{M} \sum_{n=1}^{N} f[m,n]\underbrace{h[i-m, j-n]}_{\text{"Mask," "Kernel," "Filter"}}$$



Reference:
https://fpcv.cs.columbia.edu/Monographs

Border Problem



Solution:

- Ignore border
- Pad with constant value
- Pad with reflection

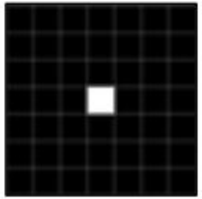
Reference:
https://fpcv.cs.columbia.edu/Monographs

Example: Impulse Filter

Input

$f(x,y)$

$*$



$\delta(x,y)$

$=$

Output

$f(x,y)$

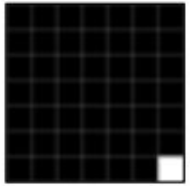
Reference:
https://fpcv.cs.columbia.edu/Monographs

Example: Image Shift

Input

Output

$*$

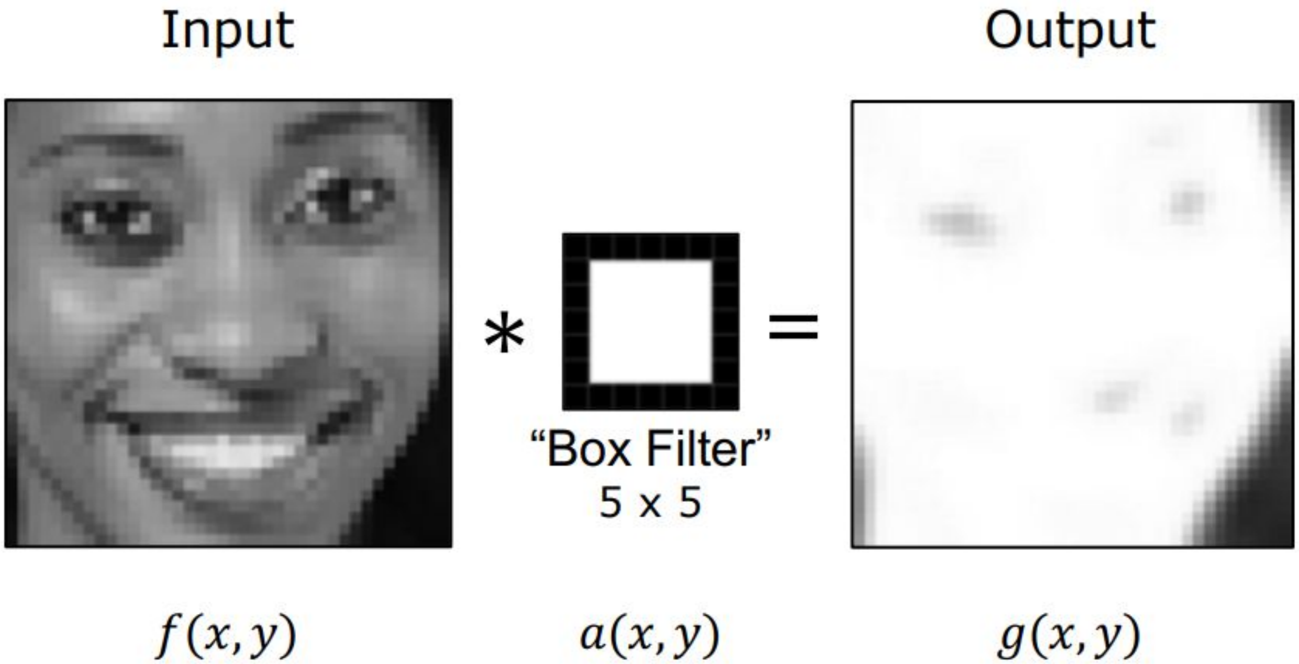


$=$

$f(x,y)$ $\qquad$ $\delta(x-u, y-v)$ $\qquad$ $f(x-u, y-v)$

Reference:
https://fpcv.cs.columbia.edu/Monographs

Example: Averaging



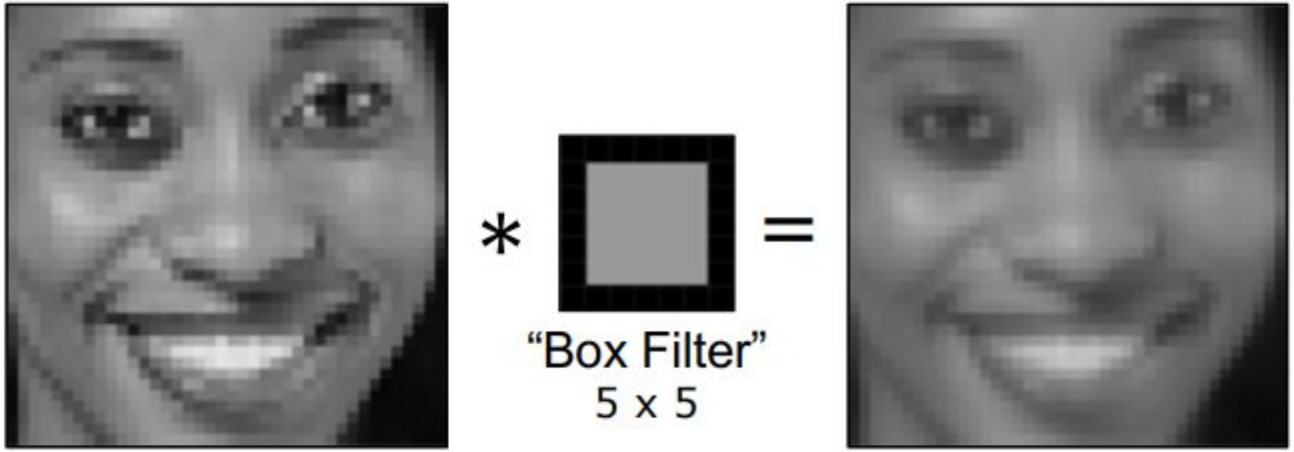
Result Image is saturated. Why?

Reference:
https://fpcv.cs.columbia.edu/Monographs

Example: Averaging

Input

Output



$f(x,y)$ $a(x,y)$ $g(x,y)$

Sum of all the filter (kernel) weights should be 1.

Reference: https://fpcv.cs.columbia.edu/Monographs