13 February, 2019

→ labtest lab এর PC তে করবো

→ আজকে 4:10 এ theory class নেয়া হবে।

→ Anaconda3

```
import cv2 //open cv
img = cv2.imread('lena.jpg', cv2.IMREAD_GRAYSCALE)   ↖ Grayscale এ read করছি
cv2.imshow("input image", img)
cv2.waitKey(0)
cv2.destroyAllWindow() // সবগুলো window destroy করে দিবে।
print(img.shape) // কত pixel তা দিবে
```

প্রতিটা pixel access করার জন্য,

```
for i in range(img.shape[0]): // # rows
    for j in range(img.shape[1]): // # col
        a=img.item
// image হলো just একটা 2D array
        if a>155:
            img.itemset((i,j),255)
        else
            img.itemset((i,j),0)
```

binary image বানাই — 155 এর বেশি হলে 255 ও নীচে হলে 0 করে দিছি

→ negative করার জন্য (2D loop এর ভেতর)

```
img.itemset(i,j) = 255 - img.item(i,j)
or, img.itemset((i,j), 255 - img.item(i,j))
```

Scanned by CamScanner

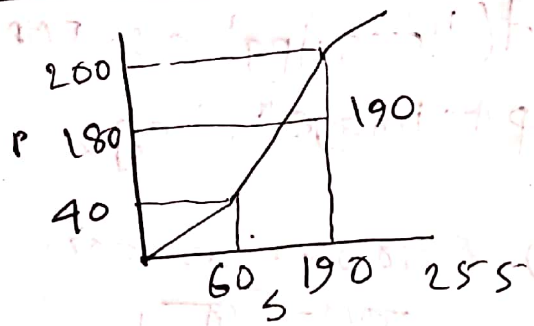
~~inverse~~ negative

0 . . . . . L−1

0 255

$r = (L-1) - s$

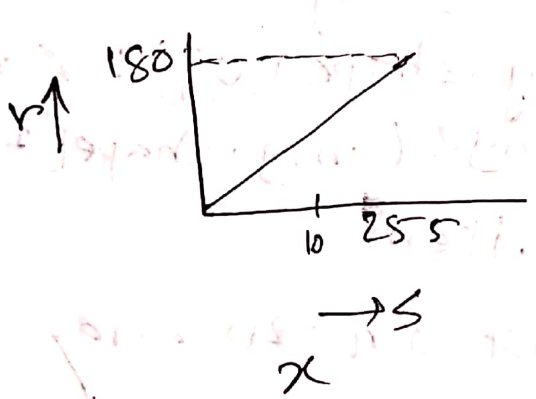
Contrast



$L = 256$

piece wise Linear

□ intensity changing



$m = \frac{180}{255}$

$y = mx$

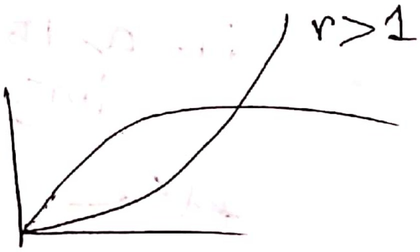
→ প্রত্যেক pixel কে গুণ দিয়ে গুণ করলে intensity সমান change হবে

□ $r = s^{\gamma}$

$\gamma = 0.5$

γ → gamma

gamma transformation



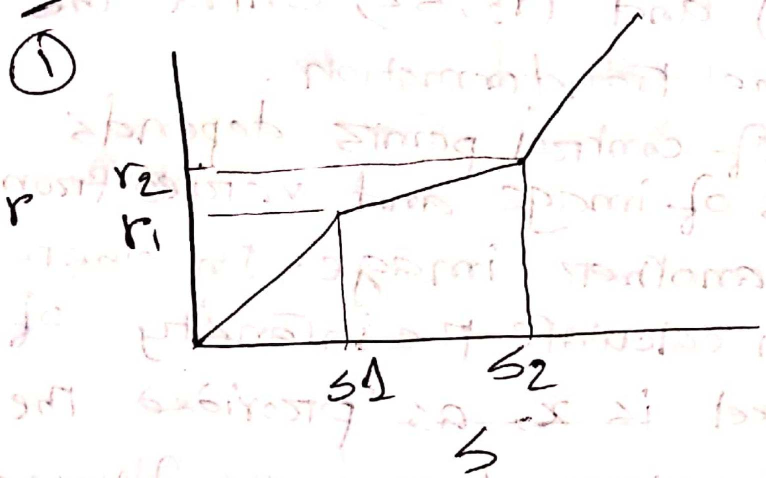
normalization (যখন $\gamma > 1$ হলে > 255 আসবে),

$$r = \left(\frac{s^{\gamma = 1.5}}{255^{1.5}}\right) \times 255$$

gamma 1.0 এর ~~বেশি~~ <1 দিলেও same কাজটা normalize করতে হবে। $r = \frac{(s^{\gamma = 0.5})}{255^{1.5}} \times 255$

HW

(1)



input: (s_1, r_1)
(s_2, r_2)
input দিয়ে → piecewise linear effect apply করো

(2) gamma transform
input : γ

→ in future gaussian function implement করবো হবে .... যদিও opencv তে builtin func. আছে

→ এগুলো আমরা নিজে করব ..... simple করে আনব হবে।

→ function নিজে লিখবো (আর নিজের built in টা call না করি।

→ আমাদের C দিয়ে pointer আমাদের direct mem access এর মাধ্যমে করতে হবে

→ কোনো বস্তুর intensity just এ কতটা উজ্জ্বল তার ~~…~~ বরং; ~~Background~~ intensity এরও উপরও impact ফেলে।

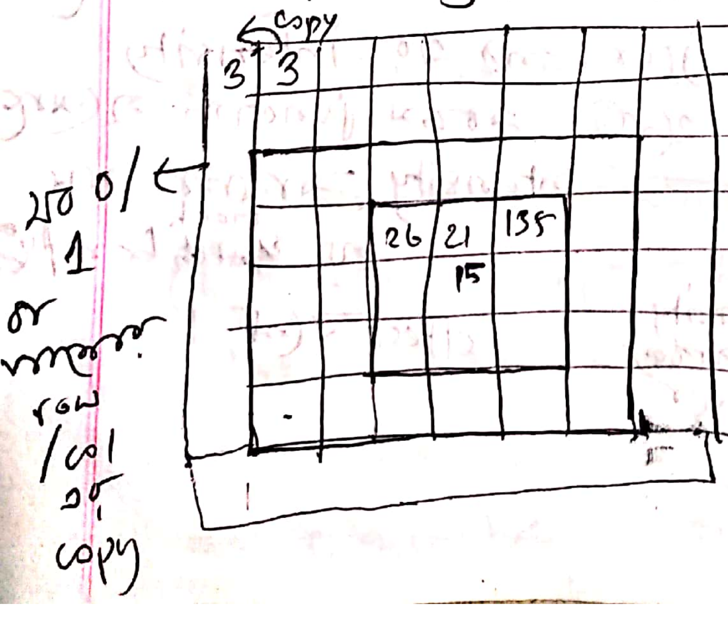
optical illusion:

February 27
Lab
Sunanda Sir

→ Filtering : image এর কোনো একটি property বাড়ানোর বা কমানোর।

→ filter করলে output pixel এর value নির্ভর করে ঐ pixel ও neighbour pixel এর value গুলোর উপর base করে হয়।

→ আর neighbourhood এর size = kernel size



এখন kernel এর যে element গুলো কোনো certain value / different value দিয়ে গুণ করে যোগ করা হয়। একে বলে weighted sum.

এই কাজ করার জন্য একটা kernel define করে নিবো,

1.2	0.5	1
0.1	0.2	.3
.3	.1	.2

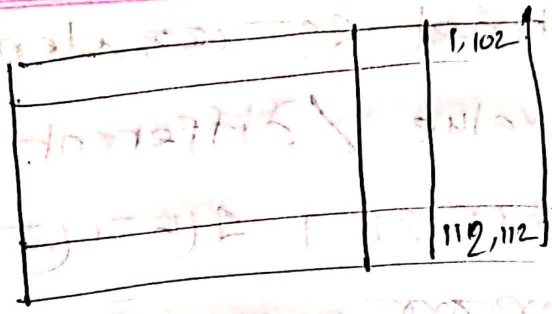
এই পুরো process হলো convoluation.

→ kernel size এর

convoul convoluation

অর্ধেক পরিমাণ এখানে $3/2 = 1$ সংখ্যক row, column বাড়াবো। যত এ row ও column এ সবগুলো 0 বা সবগুলো 255 বা পাশের যারা existing value গুলোই copy করে দিতে পারি।

→ এখন $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$ করলে এ pixel এর avg value আসবে ও image টি একটু smooth হয়ে যাবে।



corner এর দিকে smoothing হবে না

$(112\ ) + x\ ,\ 112 + (\ \ )$

COPYmakeborder(input, top, bottom, left, right, constantvalue, —)

(ছবি)

→ Task: say (5×5)

যেকোনো size এর filter design করো

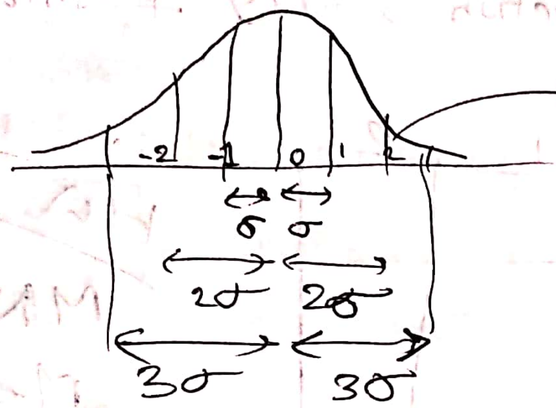
যেমন gaussian filter

$$G = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

↳ normalization const. (distribution কে [0,1] এর মধ্যে রাখার জন্য)

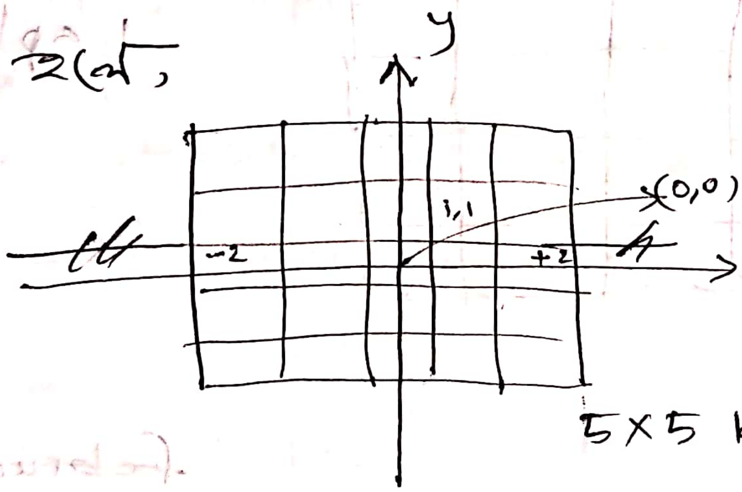
→ এটা normalize করলে এটার দরকার পড়বে না

→ basically e^{-x^2} এটাই gaussian function.



→ এ positions এ ই weight হিসেবে consider করবো

σ=1 হলে,



5×5 kernel

→ equation এ x,y vary করলে $e^{-\frac{x^2+y^2}{2\times1}}$ এ দিয়ে যা পাবো তাই kernel এর value

→ gaussian filter এ center এ high থাকে ও আশেপাশে কমে যায় value

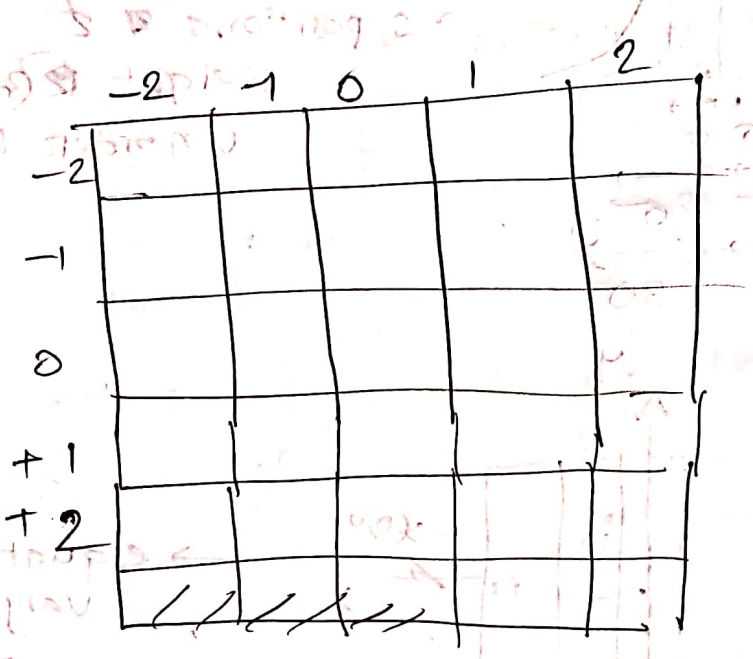
→ try করো যেন kernel size at least 5σ হয় (≈99% correct হবে আর ....)

$2\sigma^2$ দিয়ে line add করো

import matplotlib

same → obj, introduction, fig + conclusion → printed + label

min imum

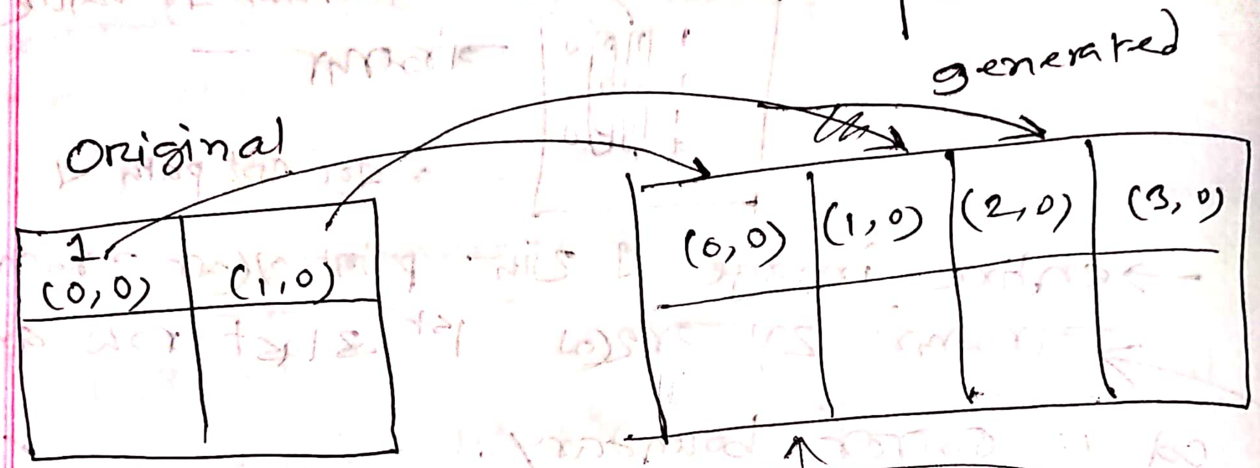


HW

MIN, MAX,
Median
Laplacian

~~february~~ 27
1

March 13, 2019
Lab



$x = \frac{x'}{2}$

$y = y'$

(0, 0) → (0, 0) mapping হবে
but (1, 0) এর জন্য
(0.5, 0) হয়।
তাই floor/ceil

ফাঁকা হবে। আবার Forward mapping এ
hole থাকে হলেও backward mapping
এ থাকবে না। কারণ সব value জানা
generate হয় but hole হবে না)

2D rotation matrix:

$$\begin{bmatrix} \cos 20^\circ & -\sin 20^\circ & 0 \\ \sin 20^\circ & \cos 20^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Scanned by CamScanner

$x' = 1.\cos 20^\circ - 2 \sin 20^\circ$

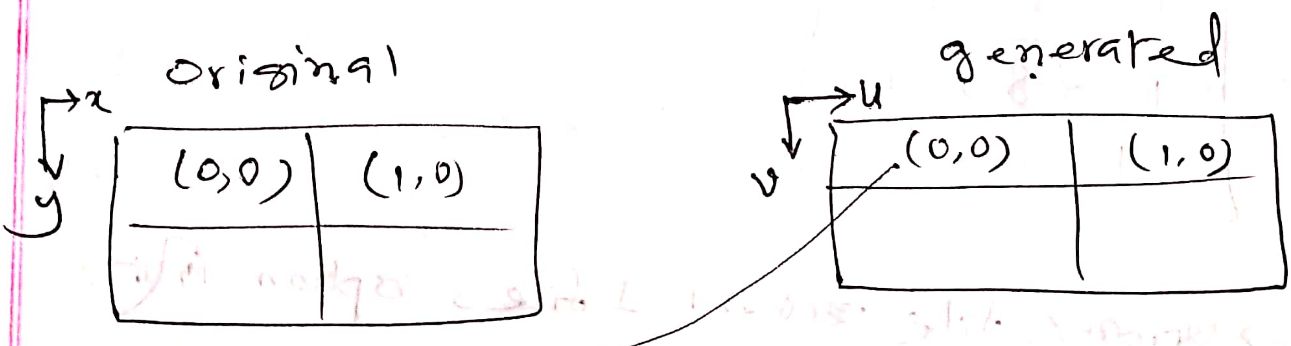
$y' = 1 \sin 20 + 2 \cos 20^\circ$

↳ এক্ষেত্রে rotation matrix use করবো but backward এর জন্য। অর্থাৎ co-ordinates নিয়ে এখান থেকে।

$$R^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\left.\begin{aligned} x &= u\cos\theta + v\sin\theta \\ y &= -u\sin\theta + v\cos\theta \end{aligned}\right]$$ এখানে যে image generate হবে তার co-ordinates

এর value হবে।



output img এর (0,0) pt. এর intensity এখন কত value নিয়ে আসবে? আর input variable এর কোথায় থেকে value আনতে হবে তা জানতে হবে।

[Forward mapping এ আমরা কোনো x থেকে y বের করি]

Backward mapping :

say u, v ($u = 10$, $v = 20$) এ θ = 43° হলো

x = 1.5 ও y = 2 (আমার ধরলাম)

input img এ থেকে (1.5, 2) নেই তাই

nearest neighbor (floor/round/ceil)

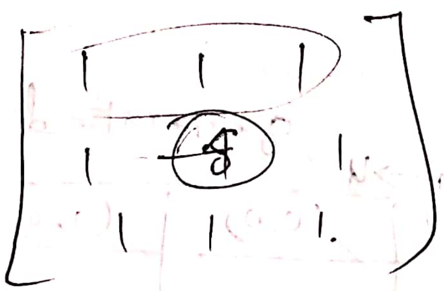
করে (2, 2) position এ intensity

নিয়ে output image এ (10, 20) এ

রাখলাম।

→ তারপর loop এ করবো.

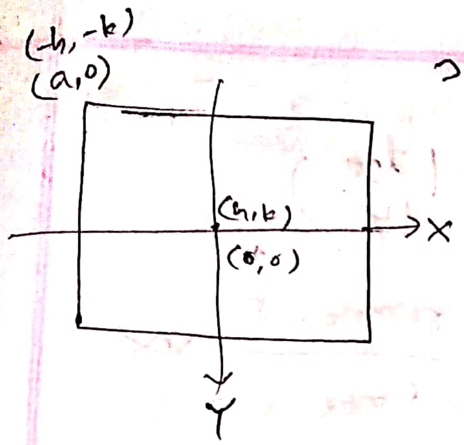
~~→ original img. থেকে~~



$$\frac{10\pi}{180}$$

→ আমার file হবেনা। 1 file, option দিবো

on loop



এখন (-h,-k) থেকে looping করবো .... এবং (0,0) পর্যন্ত

u,v → (-h+k) → ( ) হবে

→ output এর array করবো — ঠিকমতো

HW

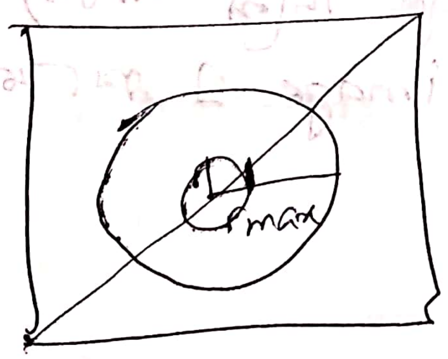
geometric operation → Twirl

→ এটি একটি non linear transformation

linear → rotation, translation, scalling

আমরা একটা নির্দিষ্ট pixel নিয়ে তার কোনো point হবে ও [illegible] হলে হবে

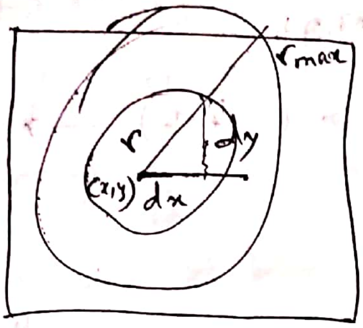
Rmax থেকে বড় কোনো মান নিয়ে ঘোরানো হবে



$$r = \sqrt{\underbrace{(x_1 - x_2)}_{dx}{}^2 + \underbrace{(y_1 - y_2)}_{dy}{}^2}$$

$(r < r_{max})$

CW এর percentage বেশি



$$\beta = \tan^{-1}\left(\frac{dx}{dy}\right) + \frac{r_{max} - r}{r_{max}} \times \alpha$$

March 14, 2019
Sunanda sir

Image Averaging:

say NASA এরা clear ছবি কিভাবে নেয়?

→ আমরা say 100টি ছবি তুলে then avg করলে পরে resolution বাড়বে ও noise reduction হবে।

→ Mask mode radiography: image subtraction এর example. $g(x,y) = f(x,y) - h(x,y)$

সাধারণত দেখা দুটি image এবং আগের এর তুলনা কোনটাও subtract করে দিলে পার্থক্য বোঝা যায়। say X-ray image এ ব্যবহার করা হয়।

gamma = 0.5

fwirl = 369400

contra =

LAB

Histogram Equation

$$s_k = T(r_k) = (L-1) \times \sum_{j=0}^{k} p_r(r_j)$$

(L−1) → 255

→ s_0 থেকে s_{255} পর্যন্ত বের করবো

n_k = say 0 intensity এর pixel আছে n_0

$$s_0 = T(r_0) = 255 \sum_{j=0}^{0} p_r(r_j) = 255\, p_r(r_0) = 1.33$$

$$s_1 = \dots \sum_{j=0}^{1} = \dots + \dots$$

( )

img.shape[0] × img.shape[1]

r_k	s_k
0	1
1	3
2	5
3	6
4	6
5	7
6	7

→ 6 এ conv. level যাইহোক ধরছে 3 ও 4 হয়েছে

→ pos6 (0, 70, ...

→ 6 (0, 70, ...

0-255

→ r_k = (illegible)

→ n_k = (illegible)

→ probability of r_k = (illegible)

$$p(r_k) = \frac{n_k}{\text{img}[0] \times [1]}$$

probability of

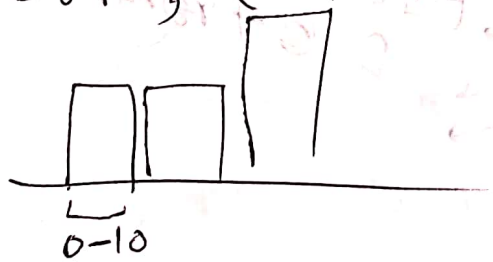
```
import cv
==
img = cv2.imread('hist' ——— )
cv2 ——— nahow
```

```
plt.hist(img.ravel(), 256, [0, 255])
```

256 → bin size

[0, 255] → gray level এর min ও max মান

→ say 0 থেকে 100 পর্যন্ত gray level আছে। তাহলে এর মধ্যে 10 টি gray level এ থাকবে তাহলে, (bin size = 10)



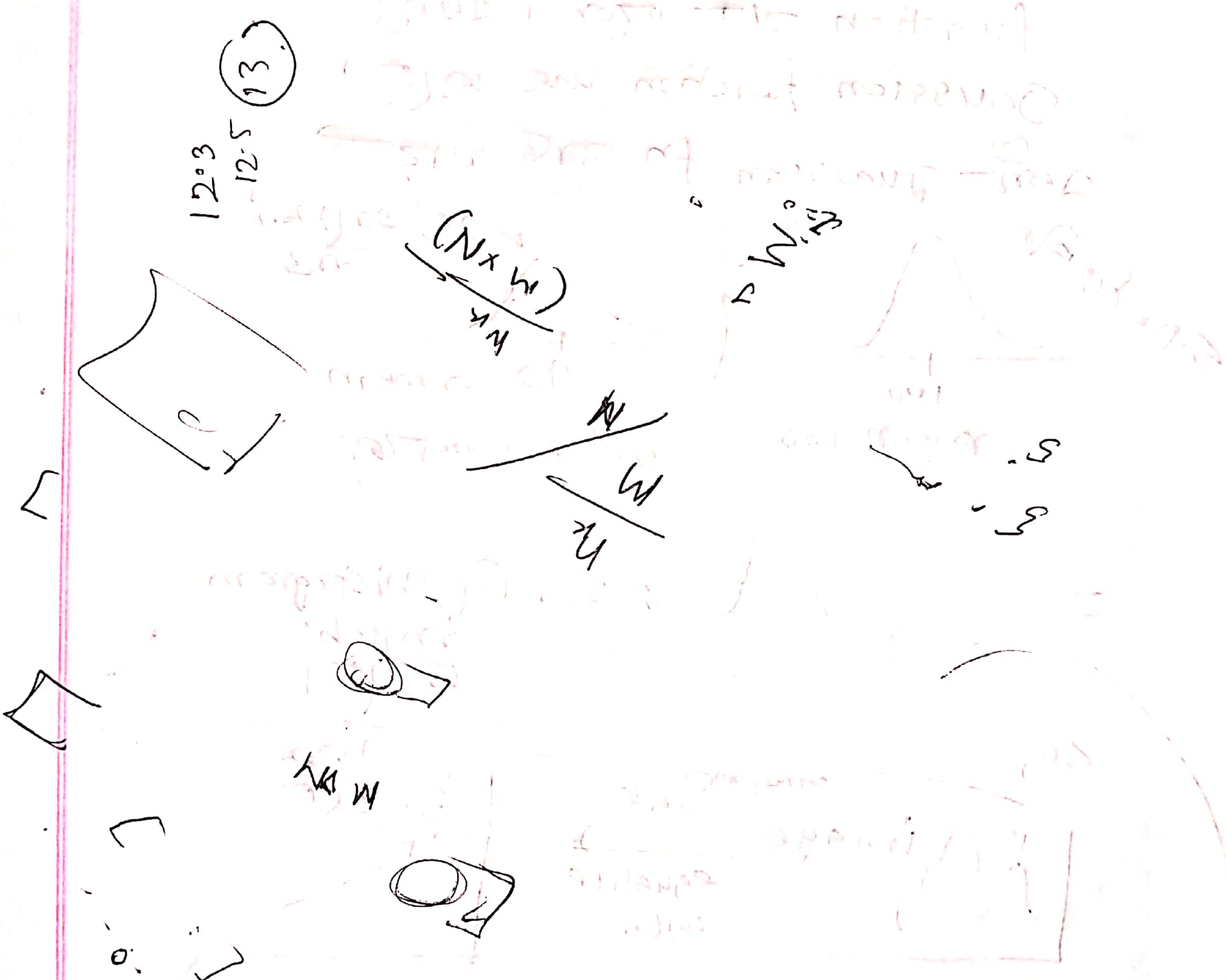
∴ Bin size = 10

তাহলে bin size 256 টি gray level এর জন্য আলাদা আলাদা bar থাকবে

→ 16 দিলে $\frac{256}{16}$ = 16 লেভেল হয়ে যাবে।

→ discrete histogram এর eqn কেন flat হবে না

// → slide এ content আছে sir এর দেওয়া for presentation

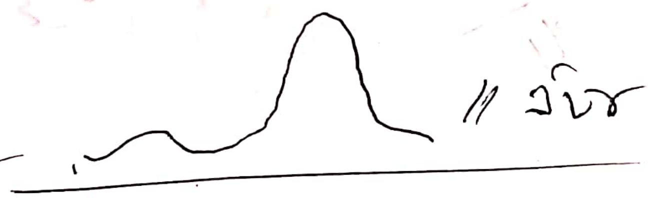


Histogram Matching

→ input image কিন্তু intensity span হচ্ছে automatically.

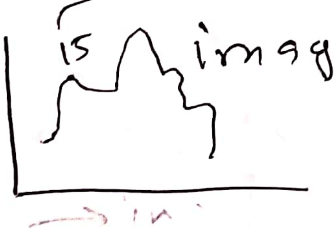
→ HW-তে একটা input intensity function বলে দিবে। একটা gaussian function use করে।

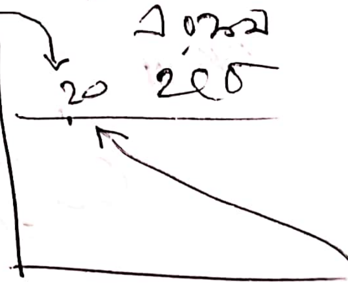
একটি guassian fn নিতে পারি

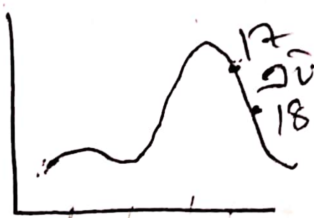


// এইটা দিয়ে histogram match করব।

So,



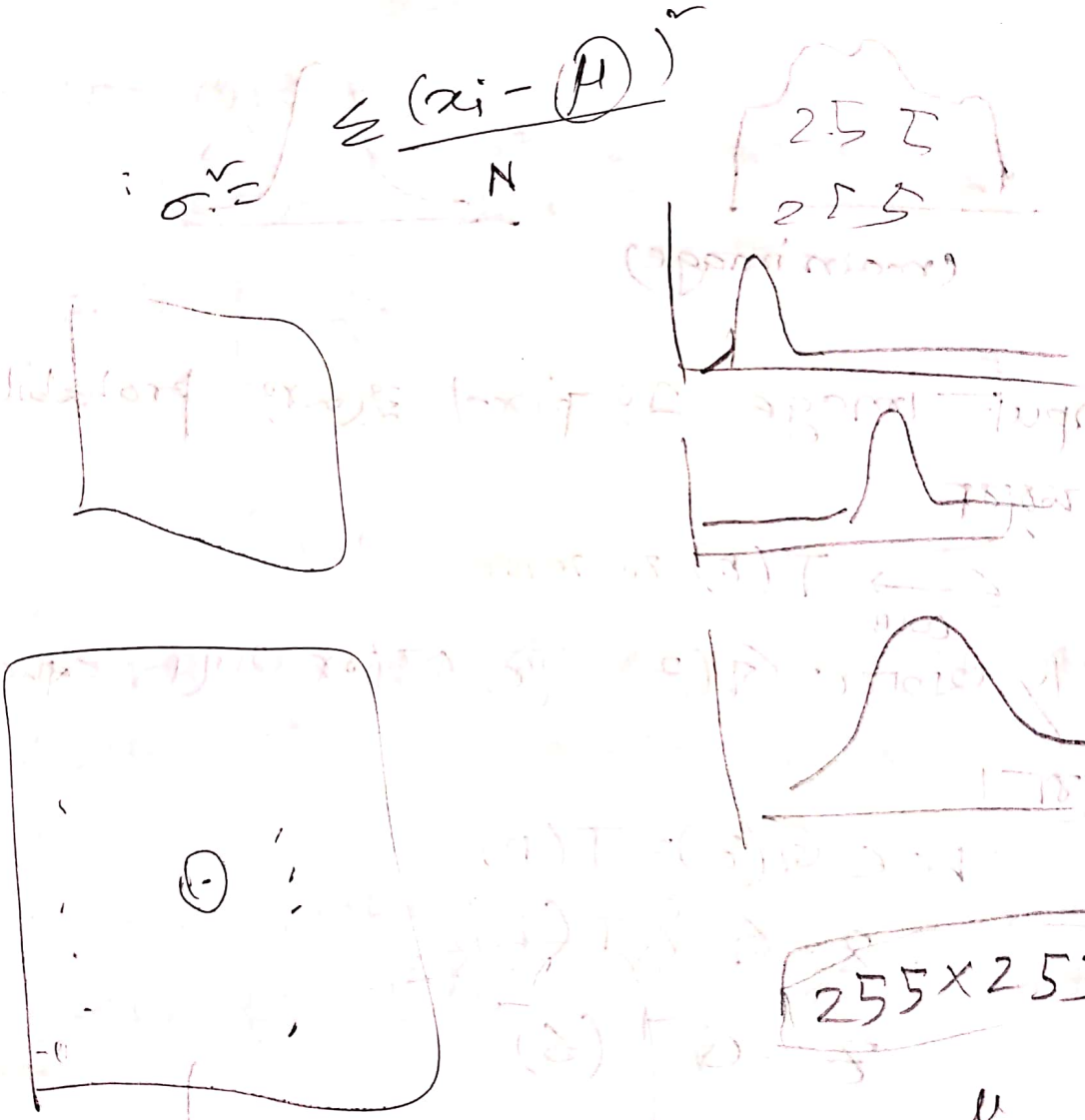




equalize করে একটা যেকোন function দিয়ে equalized করলে 20 24

মধ্যে মান 17 থেকে 18 কে equalize করলে 20 হবে। এখন input image টি নতুনভাবে 17 এর বেশি এবং 17 নিয়ে কে update করবে

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$$



$$\frac{x^2 - y^2}{20}$$

Lab
19 Apr, 2019

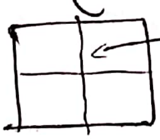
→ pattern noise removal ও ~~Fourier~~ freq. domain এ কাজ করব হবে

→ low freq গুলো কোণায় দিকে থাকে (low freq / high magnitude) (magnitude fig এ)

→ এখন corner এর low freq গুলো center এ নিয়ে আসলে সুবিধা হবে। এজন্য

→ আমরা চাই image এর low freq গুলো পাস দিব।

→ আমরা  এমন ideal low pass filter নিবো। তাহলে মাঝখানের গুলো pass হতে দিবে। বাকি গুলো ... এর low freq কে pass হতে দিবে।

→  এখানে vertical ও horizontal line গুলো (কালো দাগ গুলো) এগুলো হচ্ছে remove করবো

np.float32 // original img কে float এ নেয়ার
dft() ← cv2 এর fn
↳ এখানে pass করলে যে output আসবে তা complex output ... হলো দিচ্ছে।

→ এবার magnitude বের করব (dft[:, :, 0], dft[:, :, 1]))

→ origin এর value very high হওয়ায় এটি img এর avg [F(0,0)] এবার log এ আনা হচ্ছে যাতে 10 ও 100 এর difference কে 1 ও 2 তে আনা যায়।

→ এবার log transform এর পর value গুলো range এ আনতে cv2.normalize করি

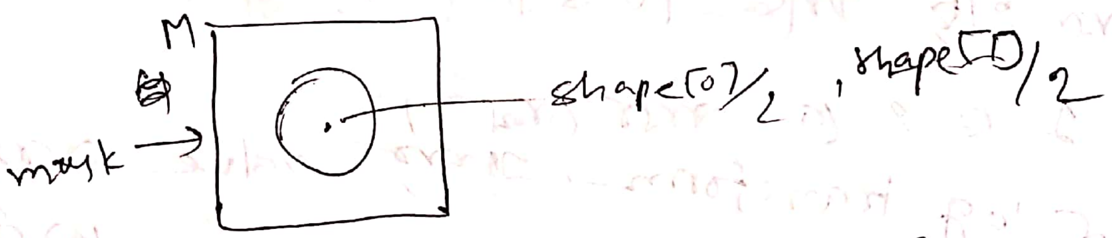
→ এবার চাচ্ছিলাম low freq কে মাঝে আনতে, np.fft.fftshift(dft) এটা দিয়ে function দিয়ে যা center এ চলে যাবে
— same কাজ ~~আগের~~ করে shifted এর আগে

f_ishift

→ inverse করবো original এ inverse shift করে ফেলতে হবে। তখন spacial domain এ ~~ifft~~ নিয়ে আসবো idft.

— original image এর same size এর img নেবো · img.shape

— এর low freq গুলো রাখবো।



$x^2+y^2=r^2$ // প্রতি point এ

$x^2+y^2<r^2$ হলে সেখানে 1 বসাবো

→ centre থেকে cut off freq input নিবো (r) say =50

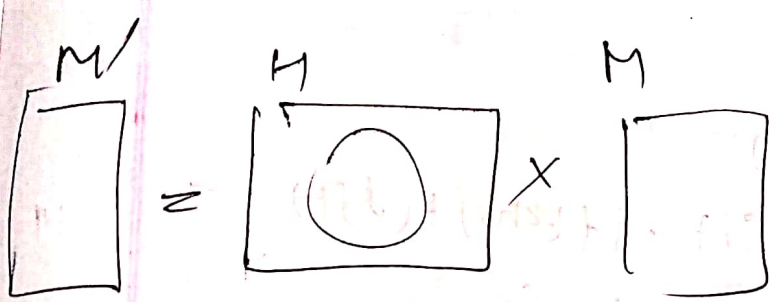
↳ এখন 50 radius এর 1

এ বাকিগুলো 0 দিয়ে visualize করবো।

→ original img $f(x,y)$ (ఈ conv. ఇవ్వు)

$f(x,y) \rightarrow F(u,v)$ (cartesian) → (polar) mag (M)

→ phase (Phase)



→ $F'(u,v)$ (Fourier ఇవ్వాలి)

↓ idft

$f'(x,y)$ (spatial dom)

→ ఇలా ఇవ్వాలి M' కి ఇది phase మార్చి back ఇవ్వు (ఈ error/or

→ shape information phase దగ్గర

→ forward transform ఇవ్వు, mask design ఇ multiply, inverse

means 0
SD 30
mean 186
SD 50

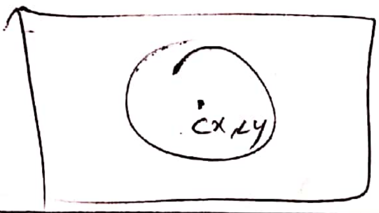
```
Ref = cv2.normalize(magnitude,
      magnitude, 0, 255, cv2.NORM_
        MINIMAL)

cv2.imshow(before)
dft.shift = np.fft.fftshift(dft)
magnitude-spectrum (np.log(cv2.
  magnitude(dft.shift[:,:,0],
                dft.shift[:::1]

f.shift = dftshift max
f.shift = np.fft.ifftshift(fshift)
img.back = cv2.magnitude(img.back
                [:,:,:],
              img_back[:,:,1])
img3 = cv2.normalize(img.base,
      0, 255, cv2.NORM_
        MINMAX
```

Cartesian → 2টা channel

polar → single channel ← convert করতে হবে

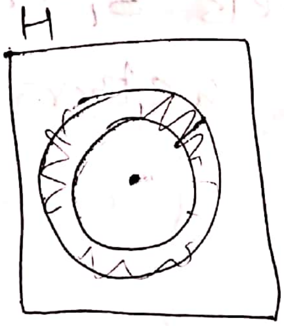
$x^2 + y^2 = r^2$

$x = r\cos\theta$ $y = r\sin\theta$	$x' = \frac{r}{2}\cos\theta = \frac{x}{2}$ $y' = \frac{r}{2}\sin\theta = \frac{y}{2}$	so you can do it in cartesian



→ assignment : Band reject filter design কর

— ideal, gaussian, butterworth



→ gonzalez এর ছবি 4র chapter এ: — original img [illegible]

→ ছবিটার band reject filter লাগাতে হয় ([illegible] ...) use করো / অন্য img use করো

Lab
Image proc
12 June

→ যে কোনো color image এ 3টি channel থাকে। যেকোনো color image যে 3 channel এ ভাগ করা যায়।

RGB

BGR ← opencv তে এই order এ থাকে

#split

```
red = img[:,:,2]
green = img[:,:,1]
blue = img[:,:,0]
cv2.imshow('red', red)
_____ ('green')
_____ ('blue')
```

এখন, input নাও

img = cv2.imread('lena.jpg', 1)

এরপরে add কর

———— numpy, cv2,

from matplotlib import pyplot as plt

→ matplotlib RGB হিসেবে নিবে তাই BGR কে RGB করলে, plot.figure(1) → করলে 1টি imge output আসবে

plt.imshow (img)

plt.figure (11)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

plt.figure(2)

plt.imshow(red, camp= 'Reds')

plt.figure(3)

plt.imshow(   )

→ Image representation এ ২ ধরনের model আছে
RGB model, HSI model

Hue, Saturation, Intensity

HUE: যে color আমরা দেখি (যেমন লাল or visual spectrum এ বিভিন্ন রঙের or দেখতে পাই। এক কথায় dominant color হলো hue

যে রং white length (?) কে বলে saturation

Task

HSV ≡ HSI (V = value)

RGB → HSI < H, S, I (channel আলাদা হবে

I → এখন শুধু I channel এ histogram equalization use করে

আবার 3টি channel কে আবার merge করে (original H & S . এবং I এর equalized form) → নিয়ে HSI কে আবার

→ পরে আবার BGR এ নিয়ে → opencv তে দেখাবো

→ RGB থেকে intensity বের possible না।

(opencv সবকিছু BGR আকারে নেয়)

```
cv2.imshow('origin', img) / read
HSV_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

cv2.imshow('HSV format image', HSV_img)

value = HSV_img[:, :, 2]
equ = cv2.equalizeHist(value)
new_img = cv2.merge((HSV_img[:, :, 0],
                     HSV_img[:, :, 1], equ))
RBB_img = cv2.cvtColor(new_img, cv2.COLOR_HSV2BGR)
```

(original H, S এবং new I)

cv2.imshow('out', RGB_img) (মিস্ট এয়ার)
cv2.waitKey(0)
cv2.destroyAllWindows()

Morphological operation

---

img2 = cv2.imread('operation2.tif')
cv2.imshow('input', img2)
img3 ——— closing
cv2 ——— inp 3, img3

Structuring element এ erosion এ এটা white eleminate করে দেবে যখন ste ele. use করবে সব white এ main image এর যেগুলো remove করবে তাদের ... match ... eleminate হবে, তাই যেগুলো eleminate করবে তার ...

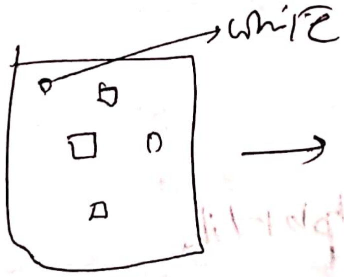
---

dialation এ 1টা point match করলেই ব্যবহার হবে।

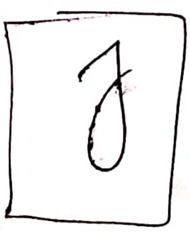
---

kernel = np.ones((5,5), np.uint

erosion = cv2.erode(img, kernel)



→ এখন opening এ ~~closing~~



→ ~~closing~~ dialation + erosion → closing

```
cv2.imshow('erosion', erosion)
dilation = cv2.dilate(img, kernel)
cv2.imshow('dilation', dilation)
opening = cv2.morphologyEx(img2, cv2.
                    MORPH_OPEN, kernel)
cv2.imshow('opening', opening)
kernel = np.ones((10,10), np.uint8)
erosion = cv2.erode(img, kernel, iteration=2)
closing = cv2.morph_Ex(img3, cv2.
                    MORPH_CLOSE, kernel)
cv2.imshow('closing', closing)
```

এর remove করার 2য় iteration.

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

→ color + non color img opencv ও matplotlib দিয়ে read করার ৩য় পর্ব।