

Лаборатори №2 (Эхний хэсэг)

Ц. Цолмон

ХШУИС-МКУТ, Компьютерийн Ухаан 3-р түвшний оюутан

tsoomoo446@gmail.com

1. ОРШИЛ	2
2. ЗОРИЛГО	2
3. ОНОЛЫН СУДАЛГАА	2
3.1 Spatial and Intensity Resolution	2
3.2 Image Interpolation	3
3.3 Contrast Stretching	5
3.4 Gray-Level Slicing	5
3.5 Bit-Plane Slicing	5
3.6 Histogram and Histogram equalization	6
4. Хэрэгжүүлэлт	7
4.1 Spatial and Intensity Resolution	7
4.2 Image Interpolation	9
4.3 Contrast Stretching	12
4.4 Gray-Level Slicing	14
4.5 Bit-Plane Slicing	17
4.6 Histogram and Histogram Equalization	18
5. Дүгнэлт	21
5. Эх сурвалж	22
6. Хавсралт	22

1. ОРШИЛ

Лекцээр үзсэн шүүлтүүрүүдийг задгайгаар (өөрөө бичих) болон бэлэн функц ашиглан хэрэгжүүлэх

2. ЗОРИЛГО

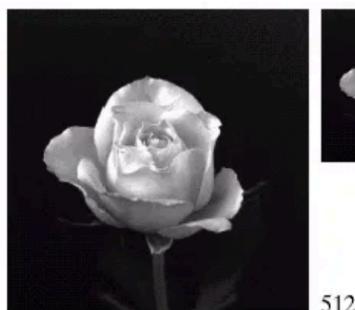
- OpenCV2 ба numpy сангудын талаар судлах
- Python хэл дээр cv2, numpy, matplotlib ашиглан өгөгдсөн даалгавруудыг гүйцэтгэх

3. ОНОЛЫН СУДАЛГАА

3.1 Spatial and Intensity Resolution

Spatial resolution - pixel гэж нэрлэгддэг зургийн хамгийн жижиг нарийвчлалын хэмжээг илэрхийлдэг.

Жишээ нь: 1024x1024 гэдэг нь 1024 pixel урттай, 1024 pixel өргөнтэй зураг гэсэн үг юм. Spatial Resolution өндөртэй байх тусмаа уг зураг нь илүү чанартай буюу олон pixel ээс бүрдсэн зураг гэсэн үг ба үүнийгээ дагаад зурагны хэмжээ мөн ихэснэ.



Intensity resolution - зургийг дүрсэлж буй ялгаралын түвшний тоог илэрхийлэх ба уг түвшин өндөр байх тусам зурагны өнгөний ялгарал сайн байна гэсэн үг юм.

Intensity resolution-г ихэнхдээ ялгаралын түвшингийн тоог хадгалах bit (2-тын тооллын) тоогоор илэрхийлдэг.

Жишээ нь: 8 битийн зургийн пикселийн 1 channel бүр 2^8 буюу 256 өөр утга авч болно гэсэн үг юм. Тэгэхээр 8 битийн RGB зургийн пиксел бүр нь 256^3 буюу 16.7 сая төрлийн өөр өнгөтэй байж чадна. [1,4]

Бит	Түвшингийн тоо	Жишээ
1	2	0, 1
2	4	00, 11, 10
3	16	111, 101, 000, 001
8	256	11010010, 00000000, 11111111
16	65536	1010101010101010

3.2 Image Interpolation

Image interpolation нь байгаа өгөгдлөө ашиглан зургийн хэмжээг томсгох, жижигсгэх үйл явц юм. Image Interpolation олон арга байдаг (Nearest neighbor, Bilinear, Bicubic гэх мэт)

Nearest neighbor - нь хамгийн ойр байгаа pixel-ын утгатай тэнцүүлэх замаар зургийн хэмжээг хувиргадаг.

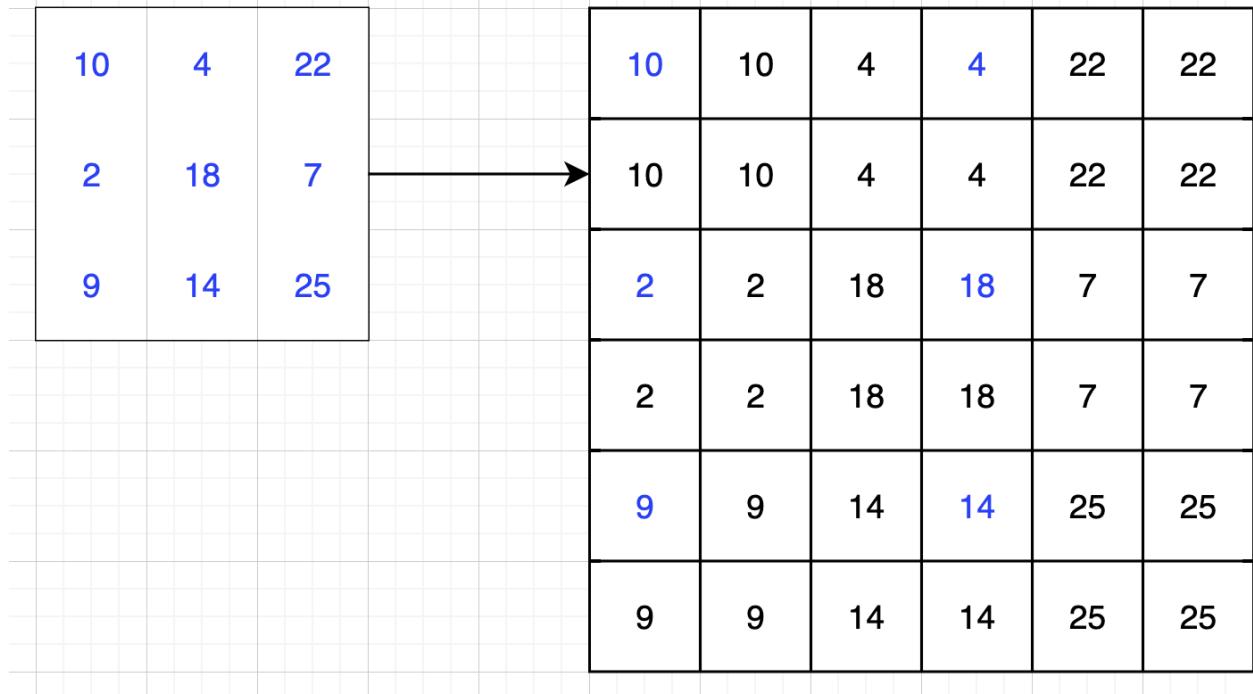
Жишээ нь:

3x3 зургийг 6x6 болгоходоо анх өгөгдсөн 9 пиксел бүрийн кординатаа 6x6 тай харьцуулж өргөн , урт нь 2 дахин томорч байгаа тул

3x3 кординат	6x6 кординат
(0,0)	(0,0)
(0,1)	(0,2)
(1,2)	(2,4)

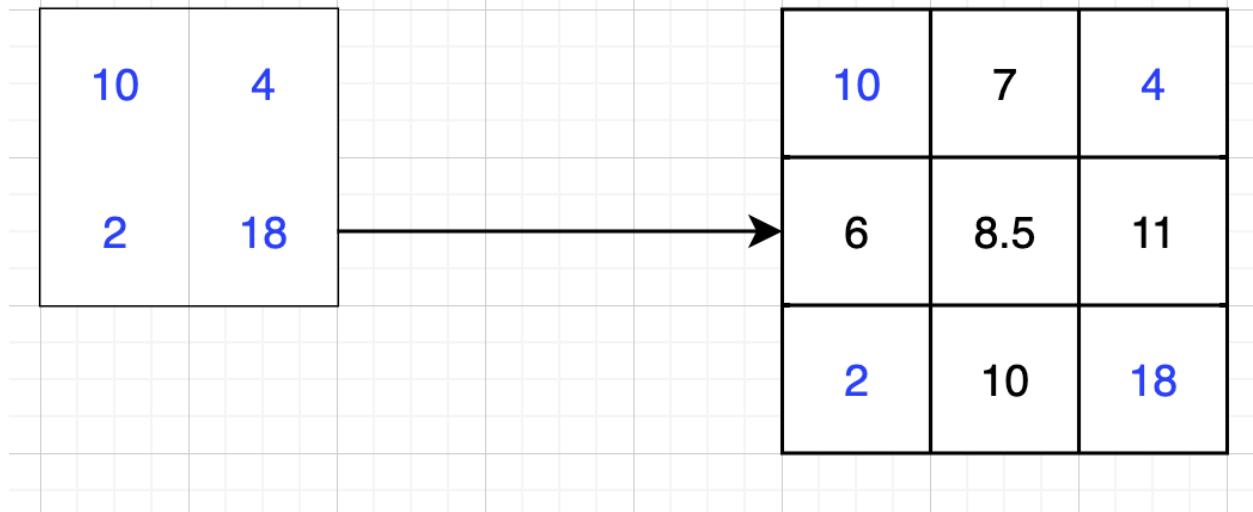
гэх мэт байдлаар анхны цэгүүдийн утгыг олоод. Үлдсэн кординатууд дээрх утгаа хамгийн ойрхон байгаа кординатын утгатай тэнцүүлэх замаар 6x6 зургаа гаргаж авна. [2,4]

Nearest neighbor interpolation



Bilinear interpolation - Nearest neighbor той адил анхны утгуудыг өгөх ба пиксел бүрийн хувьд хамгийн ойр хөрш пикселүүдийн дунджыг олох замаар зургаа үүсгэнэ.

Bilinear Interpolation



3.3 Contrast Stretching

Contrast Stretching - нь зургийн intensity -ын range буюу хамгийн бага ба хамгийн их утгыг өөрчлөн илүү ялгарал сайтай зураг гаргах техник юм. Ихэвчлэн хэт бараан эсвэл хэт гэрэлтэй зургийг янзлахад ашигладаг. [3,4]

Жишээ нь: Зурагны пиксел бүр нь [0-50] ын хооронд л утга авдаг байсан бол уг range -ыг [0-255] болгосноор зураг илүү ялгарал сайтай болдог.

$$\text{Томъёо : } s = 255 \times \frac{r - r(\min)}{r(\max) - r(\min)}$$

s - шинэ утга

r - өмнөх утга

r(min) - уг зурагны хамгийн бага intensity

r(max) - уг зурагны хамгийн их intensity

3.4 Gray-Level Slicing

Threshold - той төстэй бөгөөд энэ техник нь зураг дээрх объект, бүтэц, эсвэл тодорхой хэсгийн нарийн ширийн зүйлийг тодруулахад ашиглагддаг. [4]

Жишээ нь:

intensity > 150 , 255

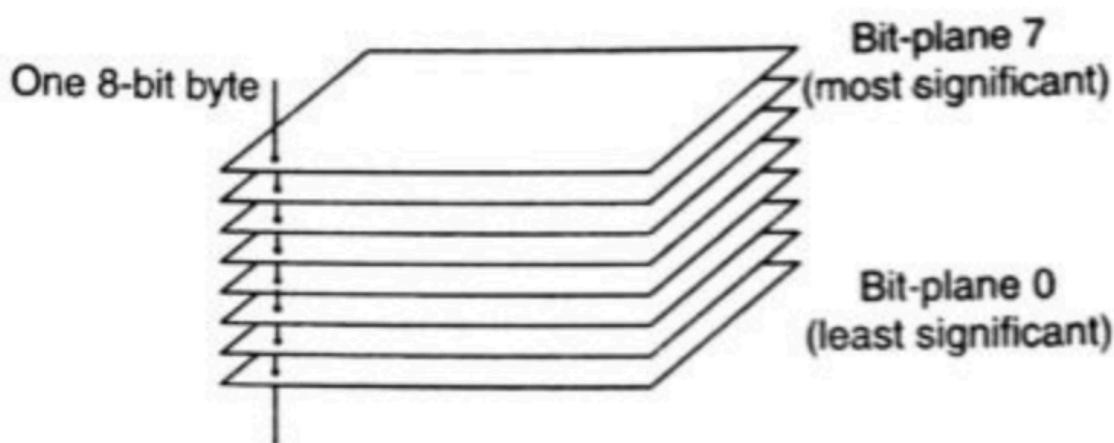
70 < intensity < 150, 140

intensity < 70 , 0

гэх мэт өөрт хэрэгтэй range - үүдээ сонгон авч гарах утгыг тохируулах боломжтой.

3.5 Bit-Plane Slicing

8-bit зурагны пиксел бүр нь 0 - 255 буюу 2-тын тооллоор “00000000” - “11111111” утга авч болно.

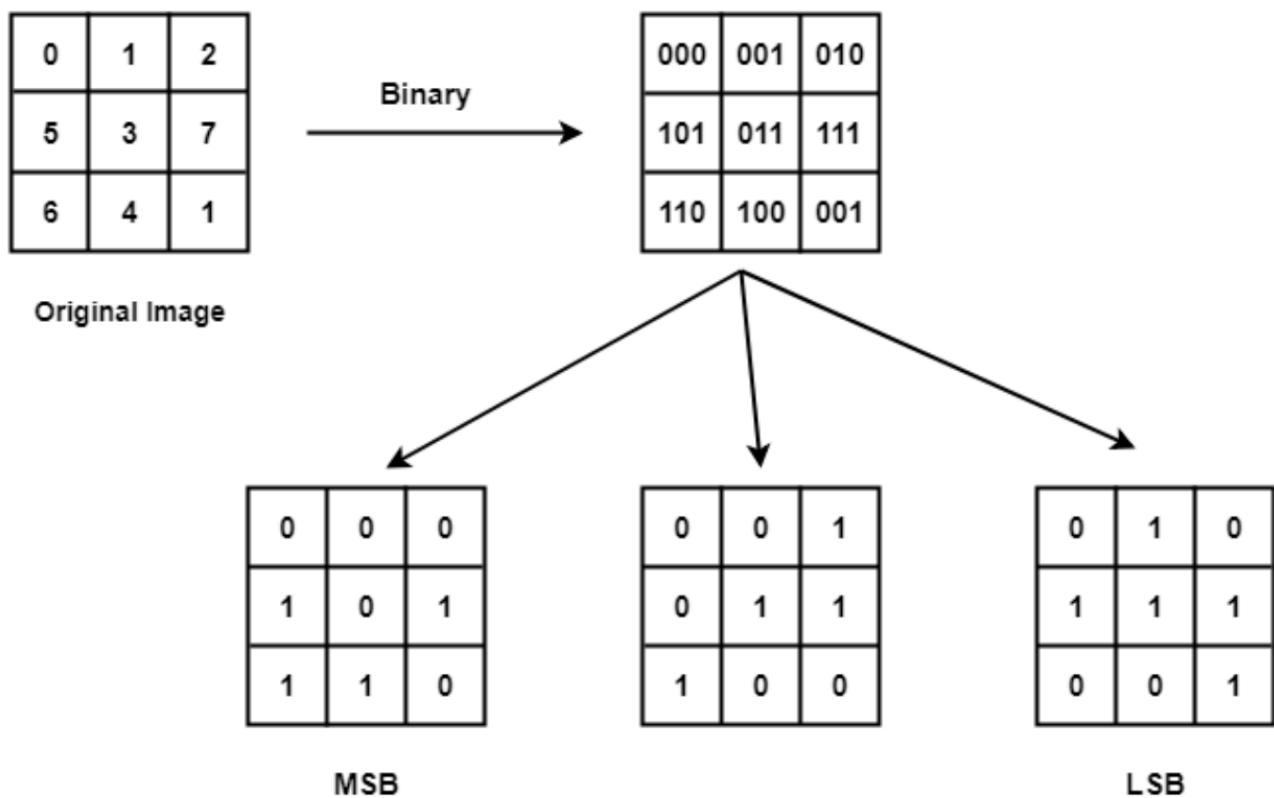


Хамгийн зүүн талын битүүд нь MSB (Most significant bit) харин хамгийн баруун талын LSM

Хуудас 5

2023/10/10

(Least significant bit) битүүд гэнэ.

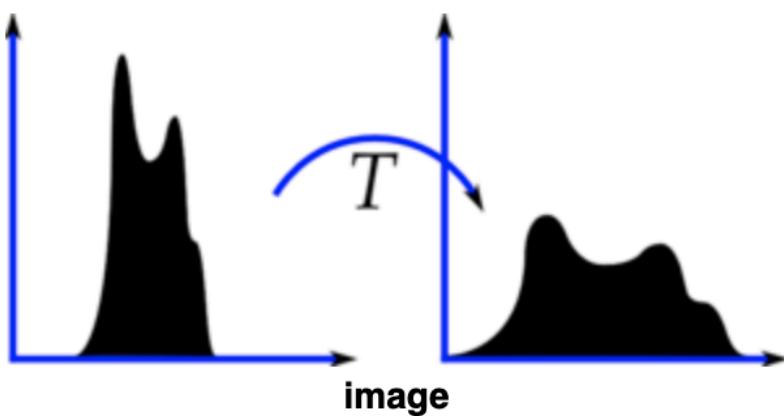


Bit-plane slicing нь зургийг жижигсгэх ба саарал зургийг binary зураг болгоход ашиглагддаг техник юм. [4,5]

3.6 Histogram and Histogram equalization

Histogram - нь зургийн intensity тархалтын график дүрслэл юм. Энгийнээр хэлбэл, энэ нь intensity утга бүрийн пикселийн тоо (y-axis) ба intensity (x-axis) илэрхийлдэг график.

Histogram equalization - нь зургийг илүү тод, ялгарал сайтай болгоход ашиглагддаг техник бөгөөд хамгийн их давтамжтай intensity утгыг үр дүнтэйгээр тарааж intensity мужийг сунгах замаар тод, ялгарал сайтай зураг гарган авдаг. [5,6]



4. Хэрэгжүүлэлт

4.1 Spatial and Intensity Resolution

cv2.imread(path, flags) - OpenCV зураг унших функц

Параметрүүд:

- path - зурагны замыг заах параметр
- flags - зургаа унших төрлийг заах параметр

np.zeros_like(img,dtype) - Numpy сангийн шинэ 0 массив үүсгэх функц

Параметрүүд:

- img - img-тэй адилхан урт, өргөнтэй массив үүсгэх параметр
- dtype - массивийн элементийн төрлийг зааж өгөх параметр

cv2.hconcat - OpenCV олон зурагнуудыг horizontal буюу хэвтээ тэнхлэгийн дагуу нийлүүлэх функц

cv2.vconcat - OpenCV олон зурагнуудыг vertical буюу босоо тэнхлэгийн дагуу нийлүүлэх функц

// оператор - Python-гын // оператор нь хуваагаад гарсан ногдворыг бүхэл тоо руу тоймлодог ба энгийн / оператор нь бутархай тоо бидэнд өгдөг тул // операторыг битийн зургаа гарган авахад ашигласан.

img.shape - нь Numpy-ын зурагны (урт, өргөн, channel) -ыг буцаадаг функц тул img.shape[0], img.shape[1] аар зурагын урт, өргөнийг авахад ашигласан.

```
● ● ● Spatial and Intensity Resolution

import cv2
import numpy as np

# skull.tif зургаа GRayscale төрлөөр унших
img = cv2.imread("../Images/skull.tif", cv2.IMREAD_GRAYSCALE)

# 2-7bit ялгаралын түвшинтэй зурагнуудаа үүсгэх
bit7 = (img // 2) * 2
bit6 = (img // 4) * 4
bit5 = (img // 8) * 8
bit4 = (img // 16) * 16
bit3 = (img // 32) * 32
bit2 = (img // 64) * 64

bit1 = np.zeros_like(img, dtype=np.uint8)

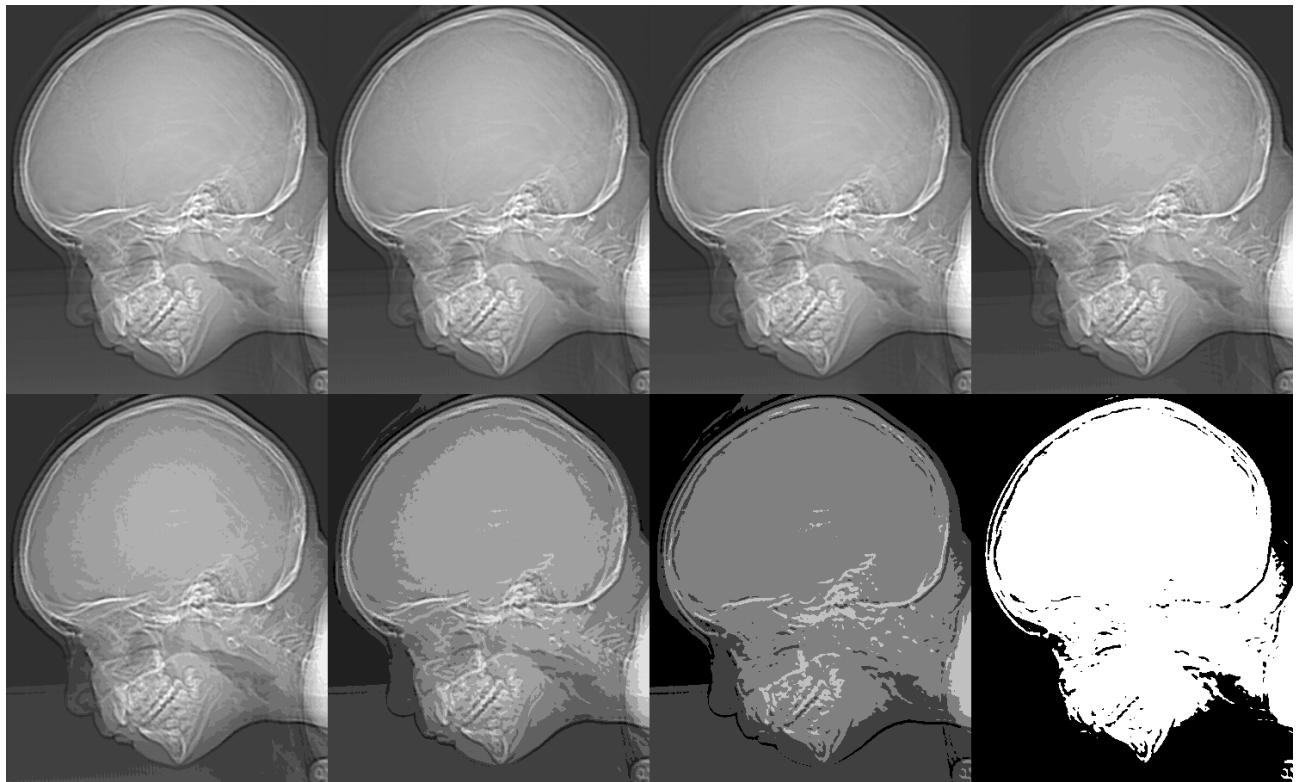
# 1bit ын буюу binary зурагаа 100 утгаар threshold хийн үүсгэх
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        if img[i, j] > 100:
            bit1[i, j] = 255
        else:
            bit1[i, j] = 0

# Үр дүнгээ нэгтгэн result гэсэн зураг үүсгээд result.png file-д хадгалах
column1 = cv2.hconcat([img, bit7, bit6, bit5])
column2 = cv2.hconcat([bit4, bit3, bit2, bit1])

result = cv2.vconcat([column1, column2])

cv2.imwrite("result.png", result)
```

ҮР ДҮН :



4.2 Image Interpolation

cv2.resize(image, dim, interpolation) - OpenCV зураг resize хийх функц

Параметрүүд :

- image - resize хийх зураг
- dim - resize хийх dimension буюу хэмжээ
- interpolation - resize хийх interpolation техник

```
● ● ● Image Interpolation

import numpy as np
import cv2

image = cv2.imread("../Images/rose.tif")

# Nearest interpolation ашиглан 4 дахин томруулах хэсэг
scale_factor = 4
height, width, channels = image.shape

new_width = width * scale_factor
new_height = height * scale_factor

upscaled_image = np.zeros((new_height, new_width, channels), dtype=np.uint8)

for y in range(new_height):
    for x in range(new_width):
        # Хамгийн ойр үтгатай координатыг олох
        src_x = int(x / scale_factor)
        src_y = int(y / scale_factor)
        upscaled_image[y, x] = image[src_y, src_x]

# OpenCV2 - ын resize функцыг ашиглан томруулах хэсэг
dim = (new_width, new_height)

function_linear = cv2.resize(image, dim, interpolation=cv2.INTER_LINEAR)
function_nearest = cv2.resize(image, dim, interpolation=cv2.INTER_NEAREST)

# Задгайгаар бичсэн зурагнаас хэр ялгаатай байгаа эсэхийг харах
sub_linear = function_linear - upscaled_image
sub_nearest = function_nearest - upscaled_image

# Томруулсан зургаа буцааж жижигсгээд алдааг нь харах хэсэг
function_linear_error = image - cv2.resize(function_linear, (height, width),
interpolation=cv2.INTER_LINEAR)
function_nearest_error = image - cv2.resize(function_nearest, (height, width),
interpolation=cv2.INTER_NEAREST)

result = cv2.hconcat([function_linear, function_nearest, sub_linear, sub_nearest])
error_result = cv2.hconcat([function_linear_error, function_nearest_error])

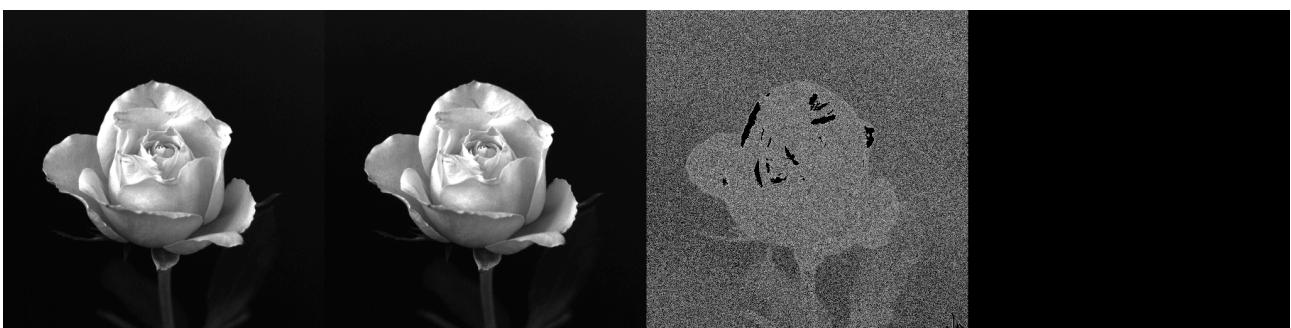
# Зурагнуудаа хадгалах
cv2.imwrite("upscaled.png", upscaled_image)
cv2.imwrite("result.png", result)
cv2.imwrite("error_result.png", error_result)
```

ҮРДҮН:

Хуудас 10
2023/10/10



Задгайгаар бичсэн nearest interpolation



cv2(linear)

cv2(nearest)

cv2(linear) зөрүү

cv2(nearest) зөрүү



cv2(linear) error

cv2(nearest) error

4.3 Contrast Stretching

np.min(image) - Numpy - ын image массивийн хамгийн бага утгыг олох функц

np.max(image) - Numpy - ын image массивийн хамгийн их утгыг олох функц

cv2.calcHist(images,channels,mask,intensity,range) - OpenCV2 histogram тооцох функц

Параметрүүд:

images - Гистограм тооцох зурагнууд

channels - Гистограм тооцох зурагны channel

range - intensity range

1. calcHist функцаа ашиглан тухайн маск бүр дээр гистограм тооцох функц
local_enhancement-ыг зарлана.

2. Маскаа гүйлгэн 3x3 пиксел бүр дээр local_enhancement функцаа ашиглан гистограм тэнцүүлнэ.

```

Local Enhancement

import cv2
from matplotlib import pyplot as plt
import numpy as np

image = cv2.imread("../Images/enhacement.tif", cv2.IMREAD_GRAYSCALE)

grid_size = (3,3)
image_height, image_width = image.shape

#local_enhancement тухайн region хэсэг дээр хийх функц
def local_enhancement(region):
    enhanced_region = cv2.equalizeHist(region)
    return enhanced_region

output_image_3x3 = np.zeros_like(image)
# 3x3 kernel бүрийн хувьд local_enhancement хийнэ
for i in range(grid_size[0]):
    for j in range(grid_size[1]):
        start_x = (i * image_height) // grid_size[0]
        end_x = ((i + 1) * image_height) // grid_size[0]
        start_y = (j * image_width) // grid_size[1]
        end_y = ((j + 1) * image_width) // grid_size[1]

        region = image[start_x:end_x, start_y:end_y]
        enhanced_region = local_enhancement(region)

        output_image_3x3[start_x:end_x, start_y:end_y] = enhanced_region

img_function = cv2.equalizeHist(image)

result = cv2.hconcat([image, img_function, output_image_3x3])
cv2.imwrite("result.png", result)

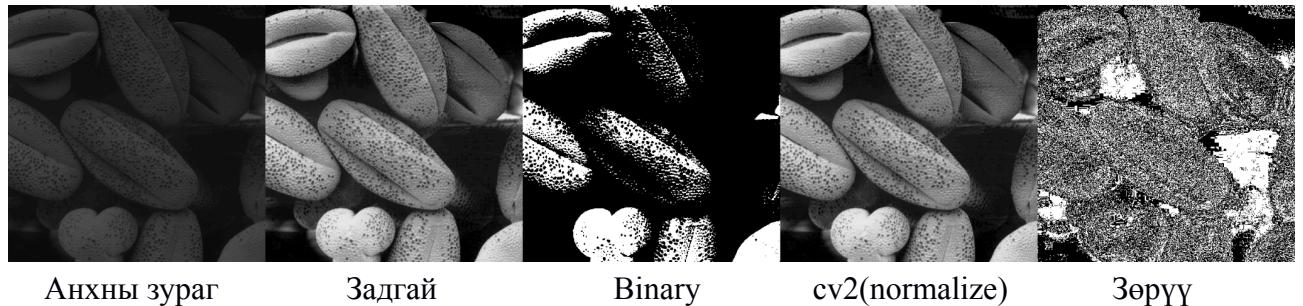
# Histogram харьцуулалт
hist1 = cv2.calcHist([image], [0], None, [256], [0, 256])
hist2 = cv2.calcHist([img_function], [0], None, [256], [0, 256])
hist3 = cv2.calcHist([output_image_3x3], [0], None, [256], [0, 256])

plt.figure(figsize=(10, 6))
plt.plot(hist1, color='blue', label='1-р зураг')
plt.plot(hist2, color='red', label='2-р зураг')
plt.plot(hist3, color='green', label='3-р зураг')

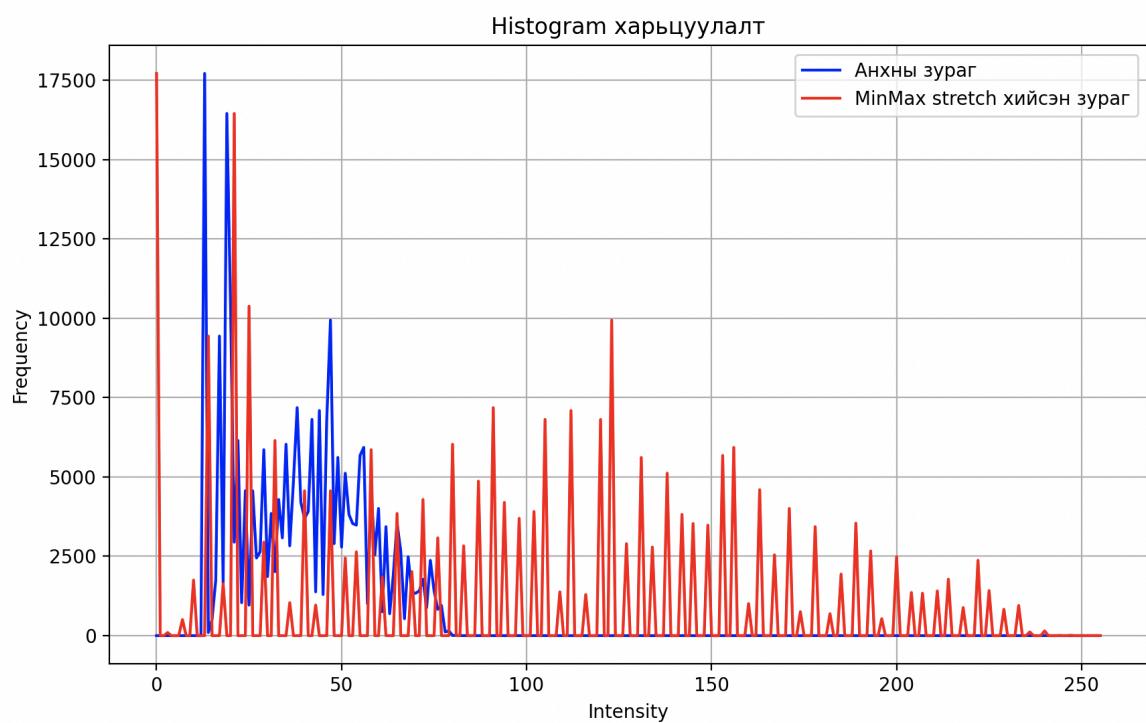
plt.title('Histogram харьцуулалт')
plt.xlabel('Intensity')
plt.ylabel('Frequency')
plt.legend()
plt.grid()
plt.show()

```

ҮР ДҮН :



Mathplot histogram харьцуулалт :



4.4 Gray-Level Slicing

Binary Slice : Intensity 150 аас их үед 255, эсрэг тохиолдолд 0 болгох замаар binary_img - ээ

Хуудас 14
2023/10/10

Үүсгэнэ.

3-Level Slice: Intensity 150 аас их үед 240, 70 аас 150 хооронд байвал 0 бусад тохиолдолд 142 болгож үүсгэнэ.

```

● ● ● Image Interpolation

from matplotlib import pyplot as plt
import numpy as np
import cv2

image = cv2.imread("../Images/kidney.tif", cv2.IMREAD_GRAYSCALE)

row, col = image.shape

j = np.zeros_like(image)
k = np.zeros_like(image)

# Binary Slice
for x in range(row):
    for y in range(col):
        if image[x, y] > 150:
            j[x, y] = 255
        else:
            j[x, y] = 0

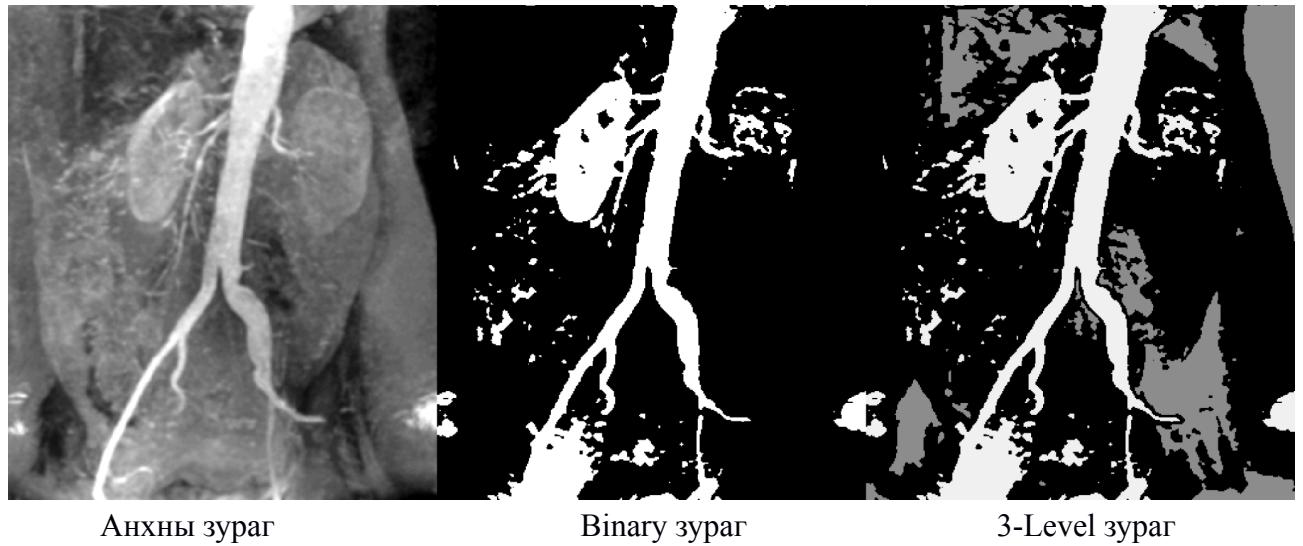
# 3 level Slice
for x in range(row):
    for y in range(col):
        if image[x, y] > 150:
            k[x, y] = 240
        elif 70 < image[x, y] and image[x, y] < 150:
            k[x, y] = 0
        else:
            k[x, y] = 142

result = cv2.hconcat([image, j, k])
cv2.imwrite("result.png", result)

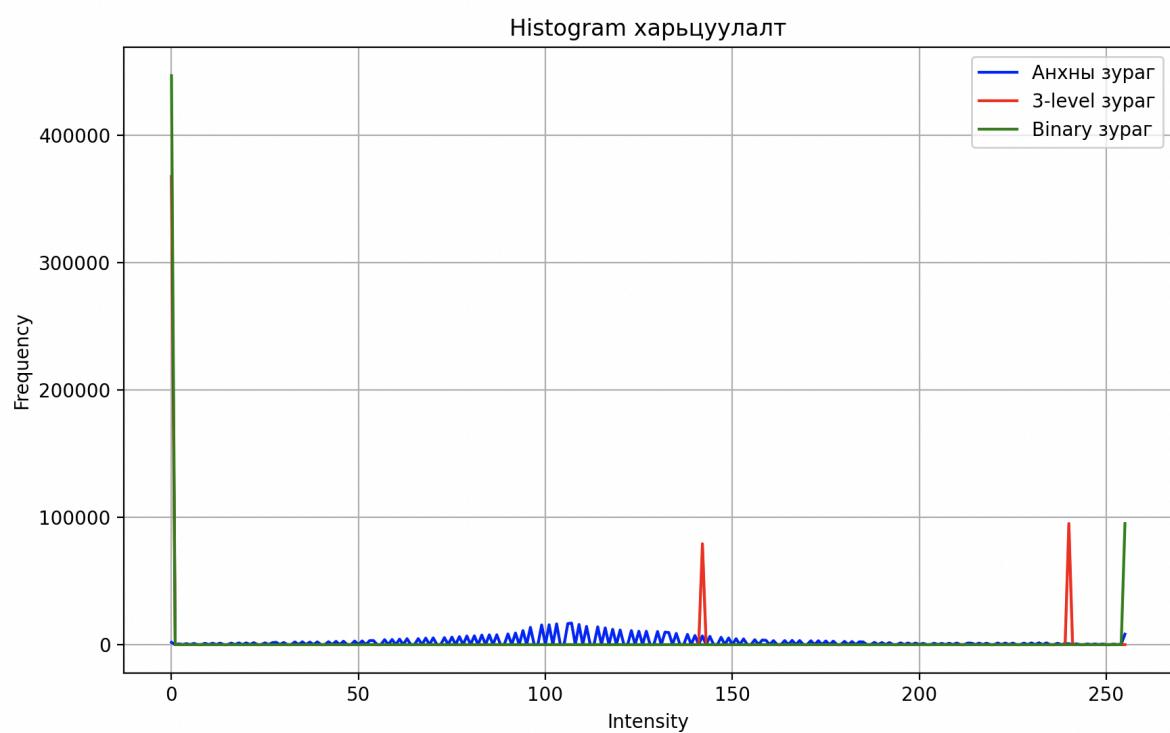
# Histogram харьцуулалт
hist1 = cv2.calcHist([image], [0], None, [256], [0, 256])
hist2 = cv2.calcHist([k], [0], None, [256], [0, 256])
hist3 = cv2.calcHist([j], [0], None, [256], [0, 256])
plt.figure(figsize=(10, 6))
plt.plot(hist1, color='blue', label='Анхны зураг')
plt.plot(hist2, color='red', label='3-level зураг')
plt.plot(hist3, color='green', label='Binary зураг')
plt.title('Histogram харьцуулалт')
plt.xlabel('Intensity')
plt.ylabel('Frequency')
plt.legend()
plt.grid()
plt.show()

```

ҮР ДҮН :



Mathplot histogram харьцуулалт :



4.5 Bit-Plane Slicing

<< оператор - нь 2-тын тоолын тоог 2 тын зэргээр үржүүлэх оператор юм. Жишээ нь: $1 << 1$ гэдэг нь 1 гэсэн 2-тын тоог 2^1 зэргээр үржүүлээд буцаад 2-тын тоолол руу хөрвүүлвэл 10 болох юм.

Зураг маань 8-bit ын зураг тул 1,10,100, ..., 10000000 гэсэн mask ашиглан бит болгон дээрх утгуудыг 1,0 гэсэн утгуудруу хөрвүүлээд бит болгон дээрх зургаа авна.

```
Bit Plane Slicing

import cv2
from matplotlib import pyplot as plt
import numpy as np

# Зургаа унших
image = cv2.imread('../Images/dollar.tif', cv2.IMREAD_GRAYSCALE)

bit_planes = []

for bit_index in range(8):
    # bit_index-ээр shift хийх
    bitmask = 1 << bit_index

    bit_plane = (image & bitmask) * 255

    bit_planes.append(bit_plane)

row1 = cv2.hconcat([image, bit_planes[0], bit_planes[1]])
row2 = cv2.hconcat([bit_planes[2], bit_planes[3], bit_planes[4]])
row3 = cv2.hconcat([bit_planes[5], bit_planes[6], bit_planes[7]])

result = cv2.vconcat([row1, row2, row3])
cv2.imwrite("result.png", result)
```

ҮР ДҮН :



4.6 Histogram and Histogram Equalization

cv2.equalizeHist(src) - OpenCV2 ын histogram тэнцүүлдэг функц

Параметрүүд:

src - функцаа хэрэгжүүлэх зураг

Histogram and Histogram Equalization

```
import cv2
from matplotlib import pyplot as plt
import numpy as np

img1 = cv2.imread("../Images/coffee-1.tif", cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread("../Images/coffee-2.tif", cv2.IMREAD_GRAYSCALE)
img3 = cv2.imread("../Images/coffee-3.tif", cv2.IMREAD_GRAYSCALE)
img4 = cv2.imread("../Images/coffee-4.tif", cv2.IMREAD_GRAYSCALE)

cv2.imwrite("first.png", cv2.hconcat([img1, img2, img3, img4]))


# Histogram харьцуулалт
hist1 = cv2.calcHist([img1], [0], None, [256], [0, 256])
hist2 = cv2.calcHist([img2], [0], None, [256], [0, 256])
hist3 = cv2.calcHist([img3], [0], None, [256], [0, 256])
hist4 = cv2.calcHist([img4], [0], None, [256], [0, 256])

eq1 = cv2.equalizeHist(img1)
eq2 = cv2.equalizeHist(img2)
eq3 = cv2.equalizeHist(img3)
eq4 = cv2.equalizeHist(img4)

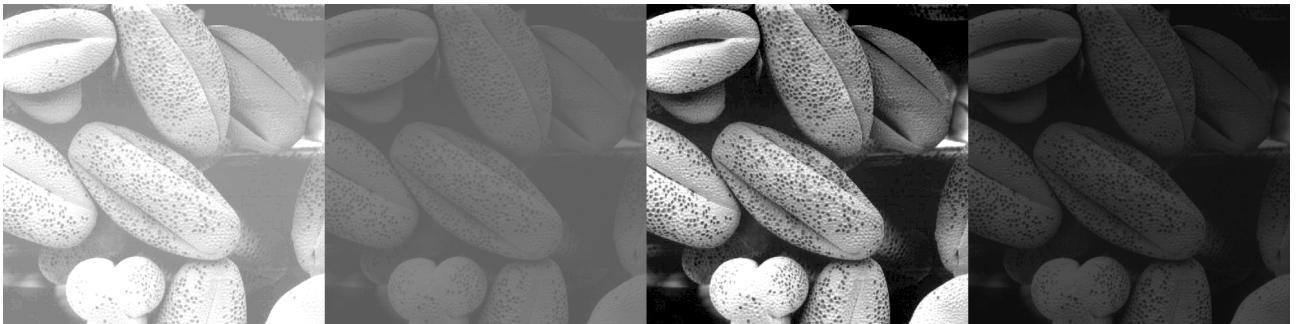
cv2.imwrite("hist_eq.png", cv2.hconcat([eq1, eq2, eq3, eq4]))


plt.figure(figsize=(10, 6))
plt.plot(hist1, color='blue', label='1-р зураг')
plt.plot(hist2, color='red', label='2-р зураг')
plt.plot(hist3, color='green', label='3-р зураг')
plt.plot(hist4, color='yellow', label='4-р зураг')


plt.title('Histogram харьцуулалт')
plt.xlabel('Intensity')
plt.ylabel('Frequency')
plt.legend()
plt.grid()
plt.show()
```

ҮР ДҮН:

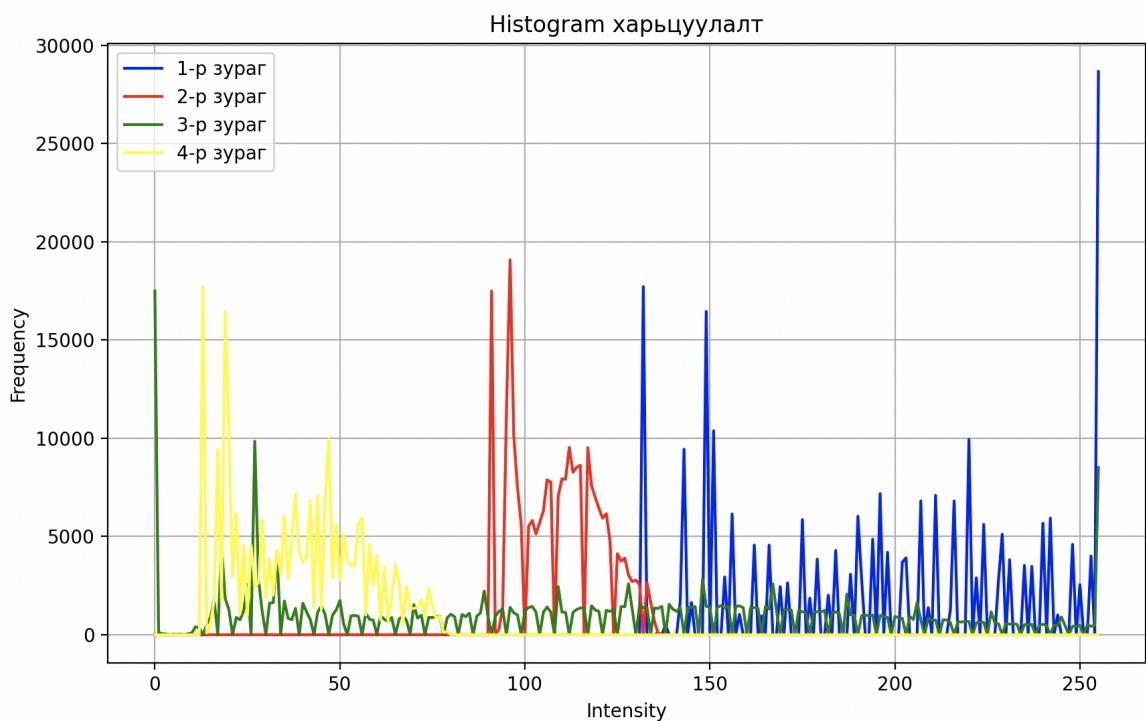
Анхны зурагнууд:



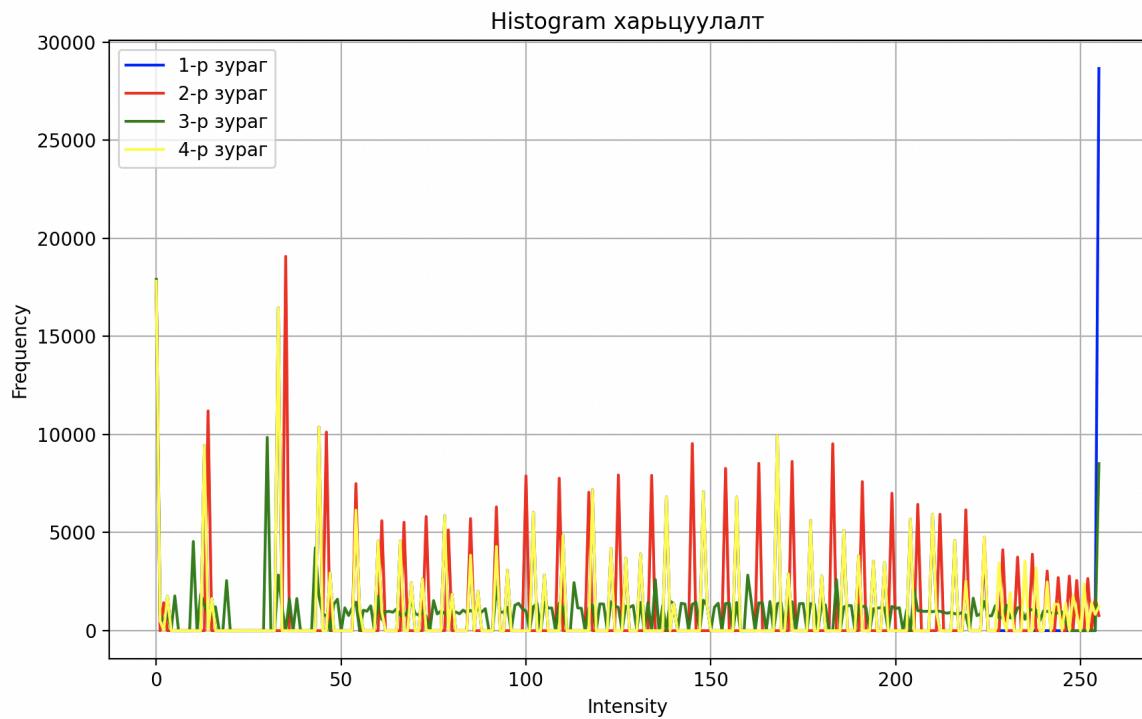
Histogram Equalization хэрэгжүүлсний дараах зурагнууд:



Анхны histogram-ууд:



Histogram Equalization хэрэгжүүлсний дараах histogram-үүд:



5. Дүгнэлт

Image Pre-Processing үед ашиглагддаг олон арга техникийдийг өөрийн гараар болон OpenCV функциудыг ашиглан python дээр хэрэгжүүлж, гарсан үр дүнгээ matplotlib сан ашиглан histogram-ыг нь гарган авч анализ хийж сурлаа.

5. Эх сурвалж

1. <https://samirkhanal35.medium.com/image-resolution-a325e7c4009e>
2. <https://www.cambridgeincolour.com/tutorials/image-interpolation.htm>
3. <https://samirkhanal35.medium.com/contrast-stretching-f25e7c4e8e33>
4. Хэв танилтын үндэс - Лекц 2, Б.Сувдаа
5. <https://theailearner.com/2019/01/25/bit-plane-slicing/>
6. <https://www.geeksforgeeks.org/histogram-equalization-in-digital-image-processing/>

6. Хавсралт

Gitlab эх код :

1. Spatial and Intensity Resolution -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-1?ref_type=heads
2. Image Interpolation -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-2?ref_type=heads
3. Contrast Stretching -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-3?ref_type=heads
4. Gray-Level Slicing -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-4?ref_type=heads
5. Bit-Plane Slicing -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-5?ref_type=heads
6. Histogram and Histogram Equalization
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-6?ref_type=heads