

Лаборатори №2 (Дундын хэсэг)

Ц. Цолмон

ХШУИС-МКУТ, Компьютерийн Ухаан 3-р түвшний оюутан

tsoomoo446@gmail.com

Лаборатори №2 (Дундын хэсэг)	1
1. ОРШИЛ	2
2. ЗОРИЛГО	2
3. ОНОЛЫН СУДАЛГАА	2
3.1 Local Enhancement	2
3.2 Enhancement using Arithmetic / Logic Operations	3
3.3 Smoothing linear filters	3
4. Хэрэгжүүлэлт	5
4.1 Local Enhancement	5
4.2 Enhancement using Arithmetic / Logic Operations	7
4.3 Smoothing linear filters	9
4.4 Smoothing linear filters	11
5. Дүгнэлт	14
5. Эх сурвалж	14
6. Хавсралт	14

1. ОРШИЛ

Лекц дээр үзсэн Local enhancement олон арга техникийн талаар судалж ойлгох ,задгайгаар (өөрөө бичих) болон бэлэн функц ашиглан хэрэгжүүлэх.

2. ЗОРИЛГО

- Local Enhancement талаар судлах
- OpenCV2 ба numpy сангуйудын талаар судлах
- Логик операторуудыг судлах
- Python хэл дээр cv2, numpy, matplotlib ашиглан өгөгдсөн даалгавруудыг гүйцэтгэх

3. ОНОЛЫН СУДАЛГАА

3.1 Local Enhancement

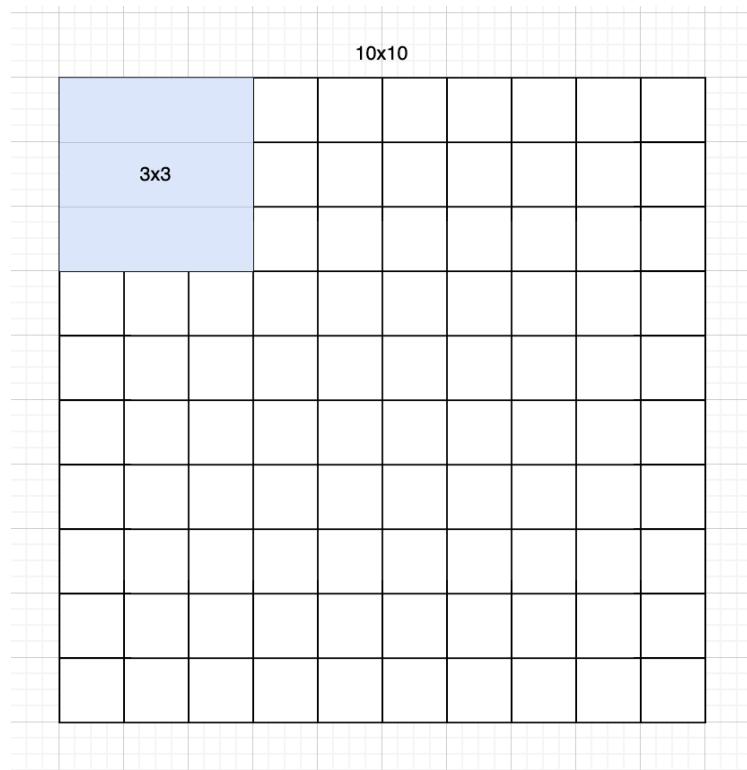
Global enhancement - нь зургийг глобал буюу бүтнээр нь сайжруулахийг хэлдэг бол ,

Local enhancement - нь зургийг зөвхөн глобал сайжруулах биш маск ашиглан зургийг маскийн тухайн хэсэг бүрт нь тохирсон сайжруулалт хийхийг хэлнэ. [1,4]

Зургийн жижиг деталиудыг сайжруулж, анализ хийхэд ашиглагддаг.

Жишээ нь:

10x10 зураг дээр histogram equalization шууд хийхэд энэ нь **Global Enhancement** болно.Харин 3x3 маск ашиглан зураг дээрээ гүйлгэж 3x3 хэсэг бүрт нь histogram equalization хийх үед энэ нь **Local enhancement** болно.



3.2 Enhancement using Arithmetic / Logic Operations

Арифметик ба Логик операторууд ашиглан зураг сайжруулах арга нь **масклах**, зургийн хэлбэр ба онцлогуудыг таньж тодорхойлоход ашиглагддаг. Зураг ба маскийн пиксел бүр дээр OR, AND, XOR гэх мэт олон төрлийн логик үйлдлүүдийг ашиглан онцлог хэсгийн пикселийн утгыг бодож олно. [5]

Жишээ нь:

AND оператор ба хар цагаан маск ашиглан маскийн зөвхөн цагаан хэсэгт байгаа зургийн онцлогийг гарган авч болно

		AND	&			OR				XOR	^			
*Image	1	1	0	0		1	1	0	0		1	1	0	0
Mask	1	0	1	0		1	0	1	0		1	0	1	0
Result	1	0	0	0		1	1	1	0		0	1	1	0

3.3 Smoothing linear filters

Smoothing spatial filter - зургийг бүдгэрүүлийх ба шуугианыг арилгахад ашигладаг.

Smoothing filter нь :

1. Шугаман
2. Шугаман бус гэсэн 2 төрөлтэй байдаг. [2]

1. Шугаман филтер нь ихэвчлэн маск дахь утгуудын дунджийг олж голын пикселийн утгад орлуулдаг ба үүнийг мөн **averaging filter** хэмээн нэрлэдэг. Шугаман филтер нь хийж буй зүйлийнхээ шаардлагаас хамаарч дотроо Standard average , Weighted average гэсэн 2 төрөлтэй байдаг..

1/9 X	1	1	1	1/16 X	1	2	1
	1	1	1		2	4	2
	1	1	1		1	2	1
	Standard Average				Weighted Average		

Standard Average - Маскийн пиксел бүрийн утгыг маскийн хэмжээнд хуваана.

Weighted Average - Урьдчилан тодорхойлсон маскийн утгуудаар пиксел бүрээ үржээд маск дахь утгуудын нийлбэрт хуваана. [3]

Ерөнхий томъёо :

$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)}$$

- $g(x, y)$ - үр дүн
- $f(x, y)$ - анхны зураг
- $w(s, t)$ - weighted averaging маск
- a, b - маскийн хэмжээ

2. Шугаман бус филтер нь маск дахь утгуудыг эрэмбэлэн тухайн эрэмбэ дээр үндэслэн голын пикселийн утгыг тодорхойлдог. Хамгийн түгээмэл жишээ нь **медиан филтер** юм.

Жишээ нь: 3x3 маскын бүх утгыг эрэмбэлж үзээд мэдийн нь 20 тул голын пиксeld 20 гэсэн утгыг өгнө.

	20	10	15
3x3	20	100	20
	20	20	25

10, 15, 20, 20, 20, 20, 20, 20, 25, 100

Маск ашиглахад захын пикселүүдийн (padding) утгыг олох боломжгүй байдаг учраас Zero-Padding , Replicating гэсэн 2 аргыг ашигладаг.

- Zero-Padding - Захын бүх пикселүүдийг 0 утгатай авна.
- Replicating - Эвклидийн зал нь хамгийн ойр цэгийн утгатай тэнцүү байхаар авна.

Zero Padding					Replicating				
0 0 0 0 0					1 1 2 3 3				
0	1	2	3	0	1	1	2	3	3
0	4	5	6	0	4	4	5	6	6
0	4	5	7	0	4	4	5	7	7
0 0 0 0 0					4	4	5	7	7

4. Хэрэгжүүлэлт

4.1 Local Enhancement

cv2.calcHist(images,channels,mask,intensity,range) - OpenCV2 histogram тооцох функц [5]

Параметрүүд:

- images - Гистограм тооцох зурагнууд
- channels - Гистограм тооцох зурагны channel
- range - intensity range

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

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

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)

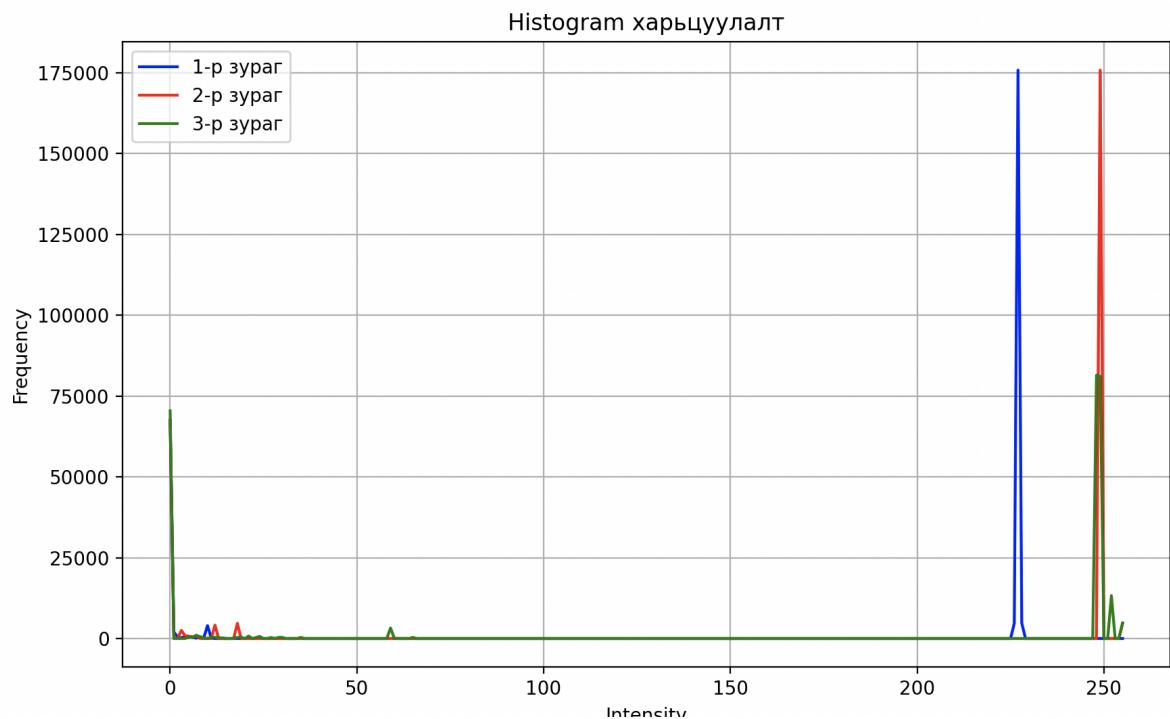
result = cv2.hconcat([image, img_function, output_image_3x3])
cv2.imwrite("result.png", result)

# 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])

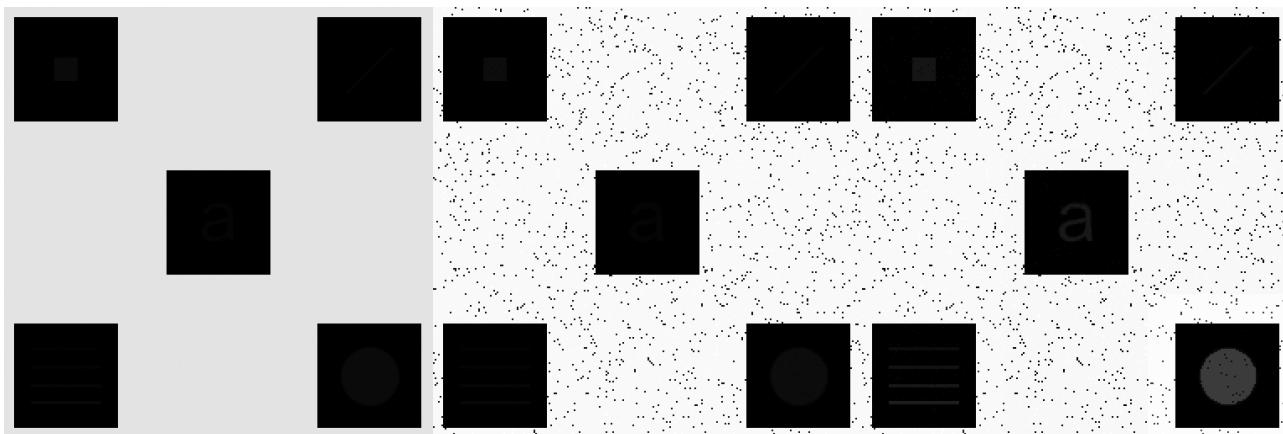
plt.figure(figsize=(10, 6))
plt.plot(hist1, color='blue', label='1-р зураг')
plt.plot(hist2, color='red', label='2-р зураг')
plt.plot(hist3, color='green', label='3-р зураг')

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

ҮР ДҮН :



ҮРДҮНХИЛСЭН ЗУРАГЫН ТОЛХОО :



4.2 Enhancement using Arithmetic / Logic Operations

& - Логик AND оператор

| - Логик OR оператор

`cv2.bitwise_and(image1, image2)` - 2 зураг дээр AND оператор ашиглах функц [5]

`cv2.bitwise_or(image1, image2)` - 2 зураг дээр OR оператор ашиглах функц [5]

```
Enhancement using Arithmetic / Logic Operations

import cv2
import numpy as np

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

# and mask - аа үүсгээд 255 тодорхой сегментийг цагаан болгоно
and_mask = np.zeros_like(image, dtype=np.uint8)
and_mask[10:300,250:490] = 255

# and image үүсгэнэ мөн задгайгаар & оператор ашиглаж болно.
and_image = cv2.bitwise_and(image, and_mask)
and_image_custom = image & and_mask

# or mask - аа үүсгээд 255 тодорхой сегментийг хар болгоно
or_mask = np.full(image.shape, 255, dtype=np.uint8)
or_mask[10:300,250:490] = 0

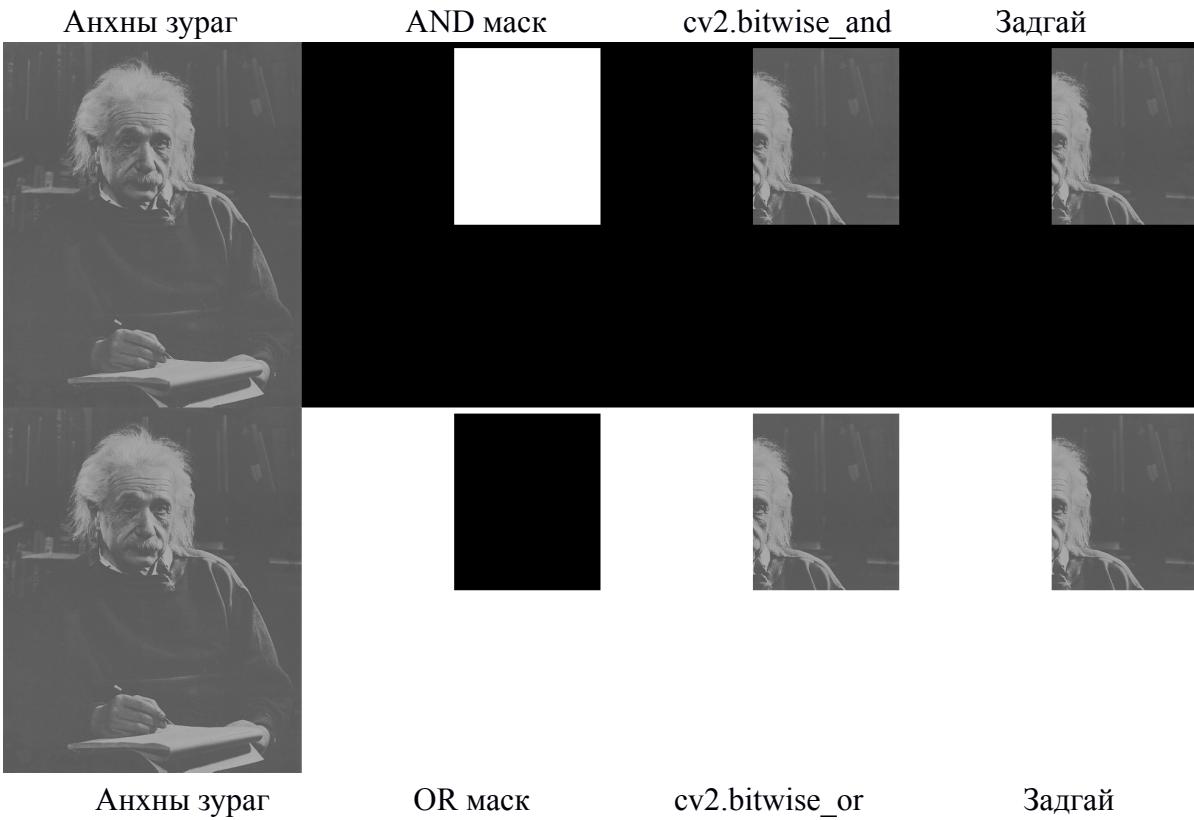
# or image үүсгэнэ мөн задгайгаар | оператор ашиглаж болно.
or_image = cv2.bitwise_or(image, or_mask)
or_image_custom = image | or_mask

row1 = cv2.hconcat([image, and_mask, and_image, and_image_custom])
row2 = cv2.hconcat([image, or_mask, or_image, or_image_custom])

result = cv2.vconcat([row1, row2])

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

ҮРДҮН:



4.3 Smoothing linear filters

cv2.filter2D(image, depth, kernel, borderType) - average filter хийхэд ашигладаг cv2 функц [5]

- image - зураг
- kernel - дундажлах маск
- borderType - Padding буюу захын цэгүүдийг тооцоолох төрөл

1. Задгайгаар **custom_filter2D** функц үүсгэх ба энэ нь зураг болон маскийг авж дундаж утгыг олон image буцаана.

2. **cv2.filter2D** функцийг ашиглан маск бүр дээр ажиллуулна.

```
Smoothing linear filters

import cv2
import numpy as np

# Load the input image
image = cv2.imread("../Images/a.tif", cv2.IMREAD_GRAYSCALE)

def custom_filter2D(image, kernel):
    img_height, img_width = image.shape
    kernel_height, kernel_width = kernel.shape

    # padding олно. Мөн Zero-padding аяра хэрэглэнэ.
    pad_height = kernel_height // 2
    pad_width = kernel_width // 2
    output_image = np.zeros_like(image)
    for y in range(pad_height, img_height - pad_height):
        for x in range(pad_width, img_width - pad_width):
            # Хөрш пиксэлүүдийн утгыг олоод кернелээрээ үржүүлнэ.
            neighborhood = image[y - pad_height:y + pad_height + 1, x - pad_width:x + pad_width + 1]
            result = np.sum(neighborhood * kernel)
            output_image[y, x] = result

    return output_image

kernel_sizes = [3, 5, 9, 15, 35]

results = [image]
cv2_result = [image]
for size in kernel_sizes:
    kernel = np.ones((size, size), np.float32) / (size * size)

    # Задгай
    custom_filtered_image = custom_filter2D(image, kernel)
    # CV2 filter2D фүнкц
    filtered_image = cv2.filter2D(image, -1, kernel)
    results.append(custom_filtered_image)
    cv2_result.append(filtered_image)

row1 = cv2.hconcat([results[0], results[1]])
row2 = cv2.hconcat([results[2], results[3]])
row3 = cv2.hconcat([results[4], results[5]])

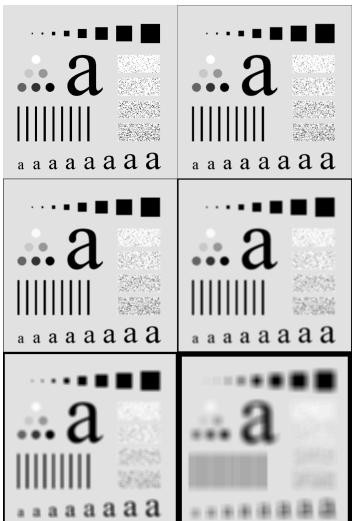
result = cv2.vconcat([row1, row2, row3])
cv2.imwrite("result.png", result)

cv_row1 = cv2.hconcat([cv2_result[0], cv2_result[1]])
cv_row2 = cv2.hconcat([cv2_result[2], cv2_result[3]])
cv_row3 = cv2.hconcat([cv2_result[4], cv2_result[5]])

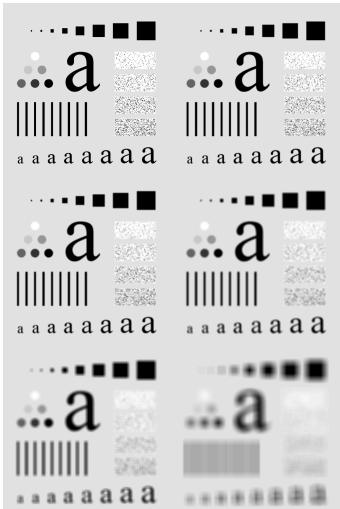
result = cv2.vconcat([cv_row1, cv_row2, cv_row3])
cv2.imwrite("cv_result.png", result)
```

ҮР ДҮН (Задгай):

Хуудас 10
2023/10/25



ҮР ДҮН (cv2) :



4.4 Smoothing linear filters

cv2.filter2D(image, depth, kernel, borderType) - average filter хийхэд ашигладаг cv2 функц

- image - зураг
- kernel - дундажлах маск
- borderType - Padding буюу захын цэгүүдийг тооцоолох төрөл

ҮР ДҮН :

```
Smoothing linear filters

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

# Load the input image
image = cv2.imread("../Images/galaxy.tif", cv2.IMREAD_GRAYSCALE)

results = [image]
kernel = np.ones((15, 15), np.float32) / (15 * 15)
# Зурсаа 15X15 маскаар averaging filter хийнэ
filtered_image = cv2.filter2D(image, -1, kernel)
results.append(filtered_image)
threshold = np.zeros_like(image, dtype=np.uint8)

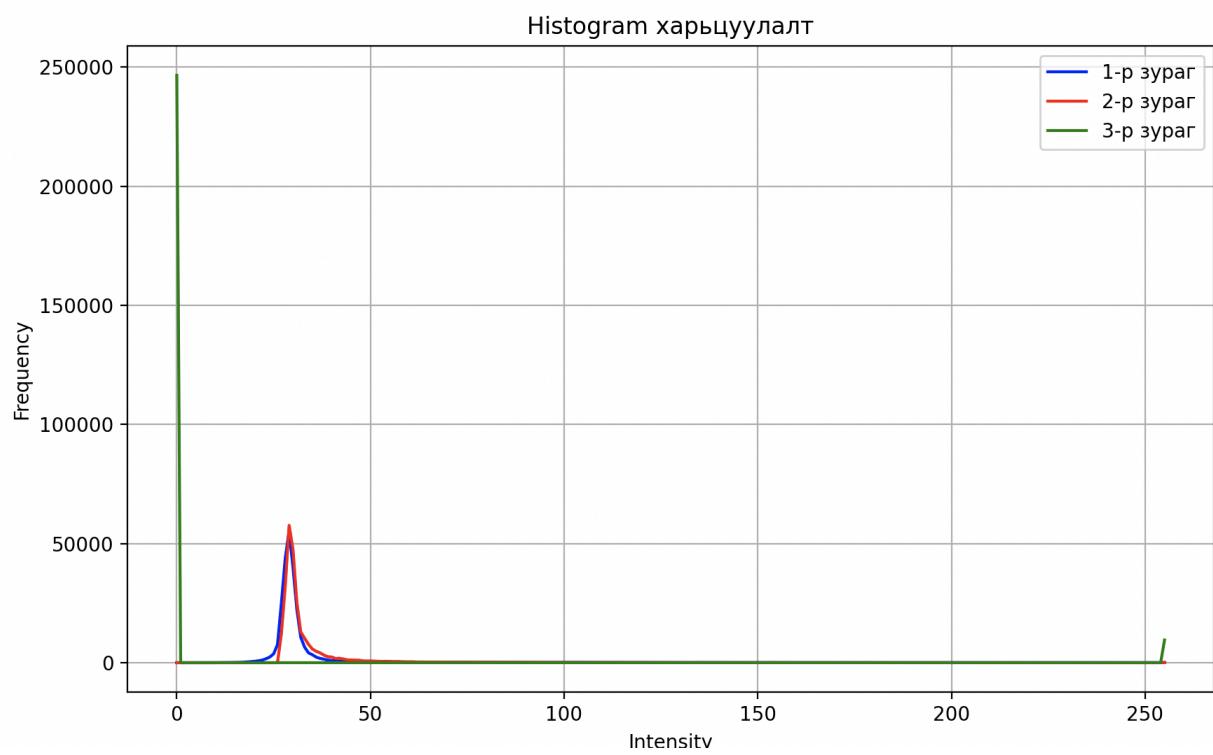
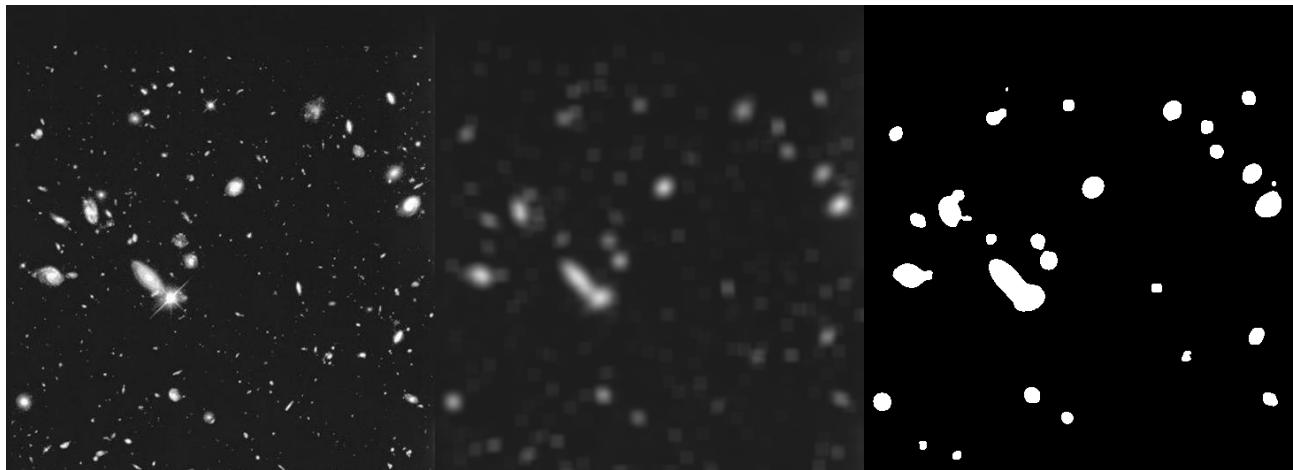
for i in range(filtered_image.shape[0]):
    for j in range(filtered_image.shape[1]):
        # 65 утгаар threshold хийнэ
        if filtered_image[i, j] > 65:
            threshold[i, j] = 255
        else:
            threshold[i, j] = 0
results.append(threshold)
result = cv2.hconcat(results)
cv2.imwrite("result.png", result)

# Histogram
hist1 = cv2.calcHist([results[0]], [0], None, [256], [0, 256])
hist2 = cv2.calcHist([results[1]], [0], None, [256], [0, 256])
hist3 = cv2.calcHist([results[2]], [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 :



5. Дүгнэлт

Local enhancement буюу дотоод сайжруулалтын олон арга техникийг өөрийн гараар болон OpenCV функцуудыг ашиглан python дээр хэрэгжүүлж, гарсан үр дүнгээ matplotlib сан ашиглан histogram-ыг нь гарган авч анализ хийж сурлаа.

5. Эх сурвалж

1. [https://web.ece.ucsb.edu/~manj/ece178-Fall2008/e178-L7\(2008\).pdf](https://web.ece.ucsb.edu/~manj/ece178-Fall2008/e178-L7(2008).pdf)
2. <https://www.geeksforgeeks.org/spatial-filtering-and-its-types/>
3. <https://medium.com/@rajilini/different-filters-for-image-processing-698e72924101>
4. Хэв танилтын үндэс - Лекц 2, Б.Сувдаа
5. <https://pyimagesearch.com/2021/01/19/opencv-bitwise-and-or-xor-and-not/>
6. https://docs.opencv.org/3.4/d4/d86/group_imgproc_filter.html

6. Хавсралт

Gitlab эх код :

1. Local Enhancement-
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-7?ref_type=heads
2. Enhancement using Arithmetic and Logical Operations -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-8?ref_type=heads
3. Smoothing linear filters -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-9?ref_type=heads
4. Smoothing linear filters -
https://gitlab.com/tsoomo/hev-tanilt/-/tree/main/lab2-10?ref_type=heads