

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Хэв танилт хичээлийн лабораторийн тайлан
(Laboratory report for the pattern recognition)

Програм хангамж(D061302)
Тайлан

Удирдагч: _____ Б. Сувдаа

Хамтран удирдагч: _____

Гүйцэтгэсэн: _____ Д. Балжинням (20B1NUM0563)

Улаанбаатар

2023 оны 9 сар

ГАРЧИГ

УДИРТГАЛ	1
1. ДҮРСТЭЙ ТАНИЛЦАХ	2
1.1 Зурагны хэмжээ боловсруулах	2
1.2 Дундаж зураг тооцоолох	2
1.3 Зургийн ялгааг олох	4
1.4 Нэмэлт мэдээлэл	4
1.5 Кодны тайлбар	4
1.6 Дүгнэлт	5
2. ДҮРС БОЛОВСРУУЛАХ /ЭХНИЙ ХЭСЭГ/	6
2.1 Spatial and Intensity Resolution	6
2.2 Image interpolation	7
2.3 Contrast Stretching	12
2.4 Gray-Level Slicing	13
2.5 Bit-plane slicing	15
2.6 Дүгнэлт	16
3. ДҮРС БОЛОВСРУУЛАХ /ДУНД ХЭСЭГ/	18
3.1 Histogram Processing	18
3.2 Local Enhancement	21
3.3 OR/AND	22
3.4 SLF	23
ХАВСРАЛТ	26
А. ЖИШЭЭ ЗУРАГNUУД	26

ЗУРГИЙН ЖАГСААЛТ

2.1	Spatial and Intensity Resolution Зураг	7
2.2	Image interpolation Зураг	9
2.3	Ялгааг харуулсан зураг	11
2.4	Contrast Stretching	13
2.5	Gray-Level Slicing	15
2.6	Bit-plane slicing	17
3.1	Histogram	19
3.2	Histogram equalized	20
3.3	Local Enhancement	22
3.4	Arithmetic / Logic Operations	23
3.5	Smoothing linear filters	24
3.6	Smoothing Linear Filters	25
A.1	Жишээ-1	26
A.2	Жишээ-2	27

Кодын жагсаалт

1.1	Дундаж зураг тооцоолох код	2
1.2	Дундаж зураг тооцоолох код	4
1.3	Дундаж зураг тооцоолох код	4
2.1	Spatial and Intensity Resolution	6
2.2	Image interpolation	7
2.3	Mean squared error code	10
2.4	Contrast Stretching	12
2.5	Gray-Level Slicing	13
2.6	Bit-plane slicing	15
3.1	Local Enhancement	21

УДИРТГАЛ

Энэхүү тайланд дүрсийг таних талбарт олон тооны зургуудаас дундаж зургийг тооцоолж, зураг боловсруулах техникийн гүнзгий дүн шинжилгээг өгдөг. Энэ ажлын хүрээнд зураг унших, зургийн хэмжээг өөрчилж матрицыг нь адил болгох, дундаж зураг тооцоолох, тухайн зургийн дундаж зургаас хэр ялгаатайг олох зэргийг үзсэн.

Хэв танилт нь машин сургалт, зураг боловсруулалт зэрэг янз бүрийн салбарт чухал үүрэг гүйцэтгэдэг. Энэ нь өгөгдлийн хэв маяг, зүй тогтлыг олоход тусладаг. Энэхүү тайлангийн гол зорилго нь зургийн бусад зургаас хэрхэн өөр байгааг, дундаж зураг яаж олох, дижитал байдлаар зураг хэрхэн хадгалагддаг зэргийг судлах зорилготой.

1. ДҮРСТЭЙ ТАНИЛЦАХ

1.1 Зурагны хэмжээ боловсруулах

Энэ үйл явцын эхний алхам нь авч үзэж буй зураг бүр ижил хэмжээтэй байхыг шалгах явдал юм. Энэ нь өөр өөр зургуудыг харьцуулах боломжийг олгох тул, маш чухал юм. Хэмжээний аливаа ялгаа нь алдаатай үр дүн эсвэл буруу үр дүнд хүргэх, эсвэл бүр дундаж утгыг тооцоолох боломжгүй байдалд хүргэнэ. Тиймээс бид дараагийн алхам руу шилжихээсээ өмнө өгөгдлийн дэх зураг бүрийн хэмжээг стандарт хэмжээ болгон өөрчлөх хэрэгтэй.

1.2 Дундаж зураг тооцоолох

Бүх зургийн хэмжээг стандартчилсны дараа бид дундаж зургийг тооцоолж болно. Бүх зураг дээрх харгалзах пикселийн байрлал бүрийн дундаж утгыг тооцоолох замаар дундаж зургийг олж авна. Энэ процессын үр дүнд өгөгдлийн багц дахь бүх зургийн ”дундаж”-ыг илэрхийлэх нэг зураг гарч, бүх зургийн нийтлэг шинж чанаруудыг үр дүнтэйгээр гарган авах боломжтой.

```
1 import os
2
3
4 current_directory = os.getcwd()
5
6 items = os.listdir(current_directory + "/images")
7 jpgs = [item for item in items if "jpg" in item]
8 jpg_len = len(jpgs)
9 avg_image = 0
10
```

```

11 gray_img_list = []
12
13 for item in items:
14     if "jpg" in item:
15         try:
16             image = cv2.imread(current_directory + "/images/" + item)
17             print(item, "is read")
18             resized_image = cv2.resize(image, [500, 500])
19             gray_image = cv2.cvtColor(resized_image, cv2.COLOR_RGB2GRAY)
20         )
21         gray_img_list.append(gray_image)
22         avg_image += gray_image / jpg_len
23     except Exception as e:
24         print(e)
25
26 for gray_img, jpg in zip(gray_img_list, jpgs):
27     result = avg_image - gray_img
28     if not os.path.exists(current_directory + "/result"):
29         os.makedirs(current_directory + "/result")
30     cv2.imwrite(
31         current_directory + "/result/" + str(jpg) + "_gray_avg" + ".jpg"
32         ,
33         result,
34     )

```

Код 1.1: Дундаж зураг тооцоолох код

1.3 Зургийн ялгааг олох

Дараагийн шат нь өгөгдлийн багц дахь зураг бүрээс дундаж зургийг хасах явдал юм. Энэ алхмын зорилго нь зураг тус бүр болон дундаж зураг хоорондын ялгааг тодруулах явдал юм. Үүний үр дүнд тус бүр нь дунджаас ялгараах зургийн өвөрмөц онцлогийг илэрхийлэх ”ялгаатай зураг”-ийн багц үүснэ. Ялгаатай зургуудыг дараа нь сонирхолтой байж болох өвөрмөц хэв эсвэл гажигийг тодорхойлоход ашиглаж болно.

1.4 Нэмэлт мэдээлэл

Компьютерт зургийг массив байдлаар дүрсэлдэг энэ нь RGB буюу Red, Green, Blue гэсэн турван өнгийг илэрхийлэх ба зарим тохиолдолд Alpha channel нэмэгдэх ба энэ нь массивын уртыг нэгээр нэмэх бөгөөд өнгөний тодролыг илгтэнэ.

Зургийг матрицаар илэрхийлж болж байгаа учир адилхан хэмжээтэй зурагнууд дээр математик үйлдлүүд хийх боломжтой болох юм. Тухайлвал олон зурагны дунджийг олж болох ба, энэ нь матриц дээр математик үйлдлүүд цаанаа хийгдэх хэдий ч хэрэглэгчид харагдахаа зурагнуудыг хооронд нь нийлүүлсэн мэт мэдрэмжийг төрүүлнэ.

1.5 Кодны тайлбар

Python хэл ашигласан бөгөөд ‘OS’ санг ашиглан фолдер дотор байгаа бүх зургийн замыг авч чадна, дараагаар нь зураг бүрийг тус тусад нь уншиж тэр бүрдээ дундаж зургийг өөрчилнө.

```
1 items = os.listdir(current_directory + "/images")
```

Код 1.2: Дундаж зураг тооцоолох код

```
1 for item in items:
2     if "jpg" in item:
3         try:
4             image = cv2.imread(current_directory + "/images/" + item)
```

```

5     print(item, "isread")
6
7     resized_image = cv2.resize(image, [500, 500])
8
9     gray_image = cv2.cvtColor(resized_image, cv2.COLOR_RGB2GRAY
10
11    )
12
13    gray_img_list.append(gray_image)
14
15    avg_image += gray_image / jpg_len
16
17 except Exception as e:
18
19     print(e)

```

Код 1.3: Дундаж зураг тооцоолох код

Зурагны ялгааг олохдоо шууд ‘-‘ эсвэл ‘+‘ тэмдэг ашиглан математик үйлдэл хийж болно.

1.6 Дүгнэлт

Энэ тайланда авч үзсэн зургийн хэмжээсийг хэвийн болгох, дундаж зургийн тооцоолол, ялгааг тооцоолох арга нь хэв танилтын салбарт хүчирхэг хэрэгсэл юм. Энэ нь өгөгдлийн багц дахь нийтлэг болон өвөрмөц шинж чанаруудыг тодорхойлох боломжийг олгодог бөгөөд ингэснээр өгөгдөлд байгаа хэв маягийн талаар цогц ойлголтыг өгдөг. Энэ нь нүүр царай таних, гажиг илрүүлэх гэх мэт төрөл бүрийн хэрэглээтэй билээ.

2. ДҮРС БОЛОВСРУУЛАХ /ЭХНИЙ ХЭСЭГ/

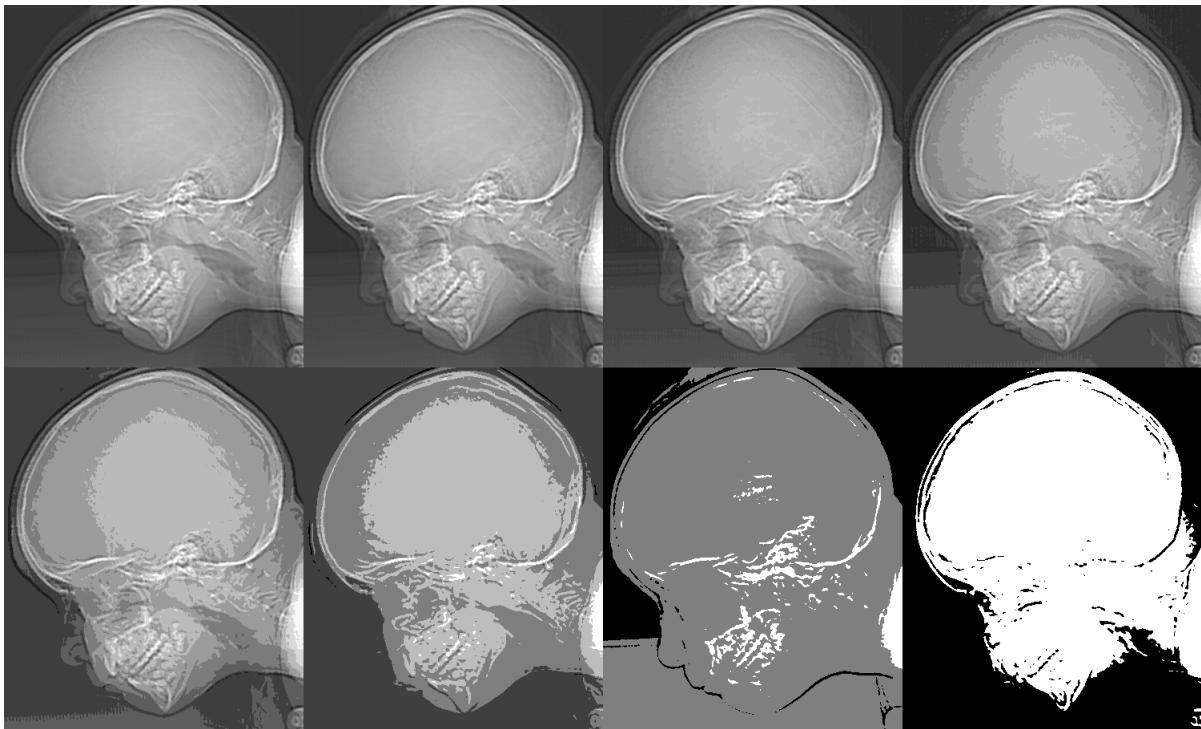
2.1 Spatial and Intensity Resolution

Spatial нь зураг бүтээхэд ашигласан пикселийн тоог илэрхийлдэг. Орон зайн өндөр нарийвчлалтай зургууд нь илүү их мэдээллийг агуулдаг. Intensity Resolution нь зургийн пиксел бүрийг илэрхийлэхэд ашигласан саарал түвшний тоог илэрхийлдэг. Энэ тоо нь ихсэх тусам илүү нарийвчлалтай хар цагаан зураг байна.

Би python дээр list-д хадгалсан зургийн пиксел бүрийг яг өөр дээр нь шууд хувиргалт хийж явсан. Үүний үр дүнд хоёрийн 1-ээс 8 зэрэг хүртэл тооноос нэгийг хасч хязгаар болгон авч энэхүү муждаа зургийн бит-г багтаасан. Үүний үр дүнд 8,7,6...2 бит-н зураг үүссэн.

```
1 current_directory = os.getcwd()
2
3 image = cv2.imread("skull.tif", -1)
4
5 for x in range(1, 9):
6     divider = 2**x - 1
7     print(x)
8     current_image = image.copy()
9     for idx_i, i in enumerate(current_image):
10        for idx_j, z in enumerate(i):
11            current_image[idx_i][idx_j] = round(z / divider) * divider
12
13 cv2.imwrite(f"{current_directory}/result/skull_{divider}.tif",
14             current_image)
```

Код 2.1: Spatial and Intensity Resolution



Зураг 2.1: Spatial and Intensity Resolution Зураг

2.2 Image interpolation

Зургийн интерполяци нь зургийн нийт пикселийн тоог нэмэгдүүлэх, багасгахад ашигладаг арга юм. Энэ нь илүү өндөр нарийвчлалтай, хамгийн сайн байдлаар зурагт гарах пикселийн утгыг тооцоолдог. Хамгийн ойрын хөрш, хоёр шугаман, бикуб зэрэг дүрсийг интерполяци хийх хэд хэдэн арга байдаг. Бас заримдаа нүдэнд нягтрал сайтай харагддаг ч яг үнэндээ тооцоолох хийхэд муу утгууд байдаг ийм учраас ялгааг харуулсан зургийг хавсаргалаа.

2.2.1 openCV

Python хэлний OpenCV сан нь Inter Nearest, Linear, Cubic гэх мэт interpolation хийх боломж бүхий бэлэн функцуудтай ба, энэхүү функцуудыг ашиглан зурагтай ажиллах боломжтой. [2]

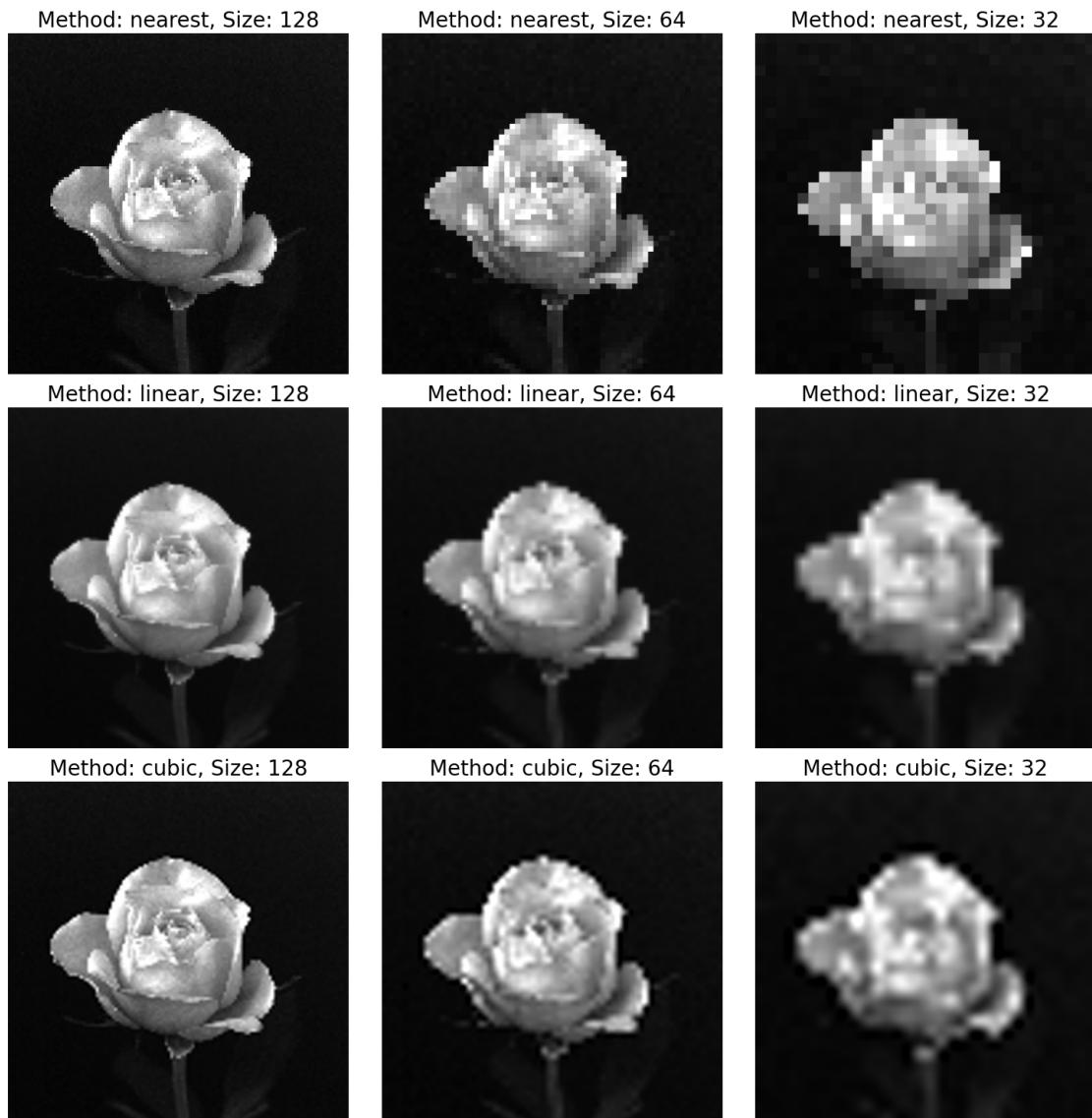
```
1 image = cv2.imread("rose.tif", -1)  
2
```

2.2. IMAGE INTERPOLATION БҮЛЭГ 2. ДҮРС БОЛОВСРУУЛАХ/ЭХНИЙ ХЭСЭГ/

```
3  interpolation = {
4      "nearest": cv2.INTER_NEAREST,
5      # cv2.IMREAD_UNCHANGED,
6      "linear": cv2.INTER_LINEAR,
7      "cubic": cv2.INTER_CUBIC,
8  }
9
10 sizes = [128, 64, 32]
11 print(image.shape)
12 for key, _intr in interpolation.items():
13     for size in sizes:
14         current_image = cv2.resize(image, (size, size), interpolation=
15             _intr)
16         current_image = cv2.resize(current_image, (1024, 1024),
17             interpolation=_intr)
18         cv2.imwrite(
19             f"{current_directory}/result/rose_{_intr}_{size}_{key}.tif"
20             , current_image
21         )
22         difference_image = image - current_image
23         cv2.imwrite(
24             f"{current_directory}/result/rose_{_intr}_{size}_{key}_difference.tif",
25             difference_image,
26         )
```

Код 2.2: Image interpolation

2.2. IMAGE INTERPOLATION БҮЛЭГ 2. ДҮРС БОЛОВСРУУЛАХ/ЭХНИЙ ХЭСЭГ/



Зураг 2.2: Image interpolation Зураг

2.2.2 Ялгааг олох

Зураг хоорондын ялгааг нүдээр харж аль зураг нь эх зурагтайгаа хамгийн төстэйг олж болохоос гадна. Дундаж квадрат алдааны (Mean Square Error) аргыг ашиглан олох боломжтой.[3] Үр дүнд хамгийн адил зураг нь хэмжээ бүр дээр linear арга ашигласан зураг гарсан. Яг үнэндээ нүдээр харахад энэ зураг хамгийн мую нь байлаа. Миний мую гэж хэлсэн шалтгаан нь sharpness багатай санагдсан.

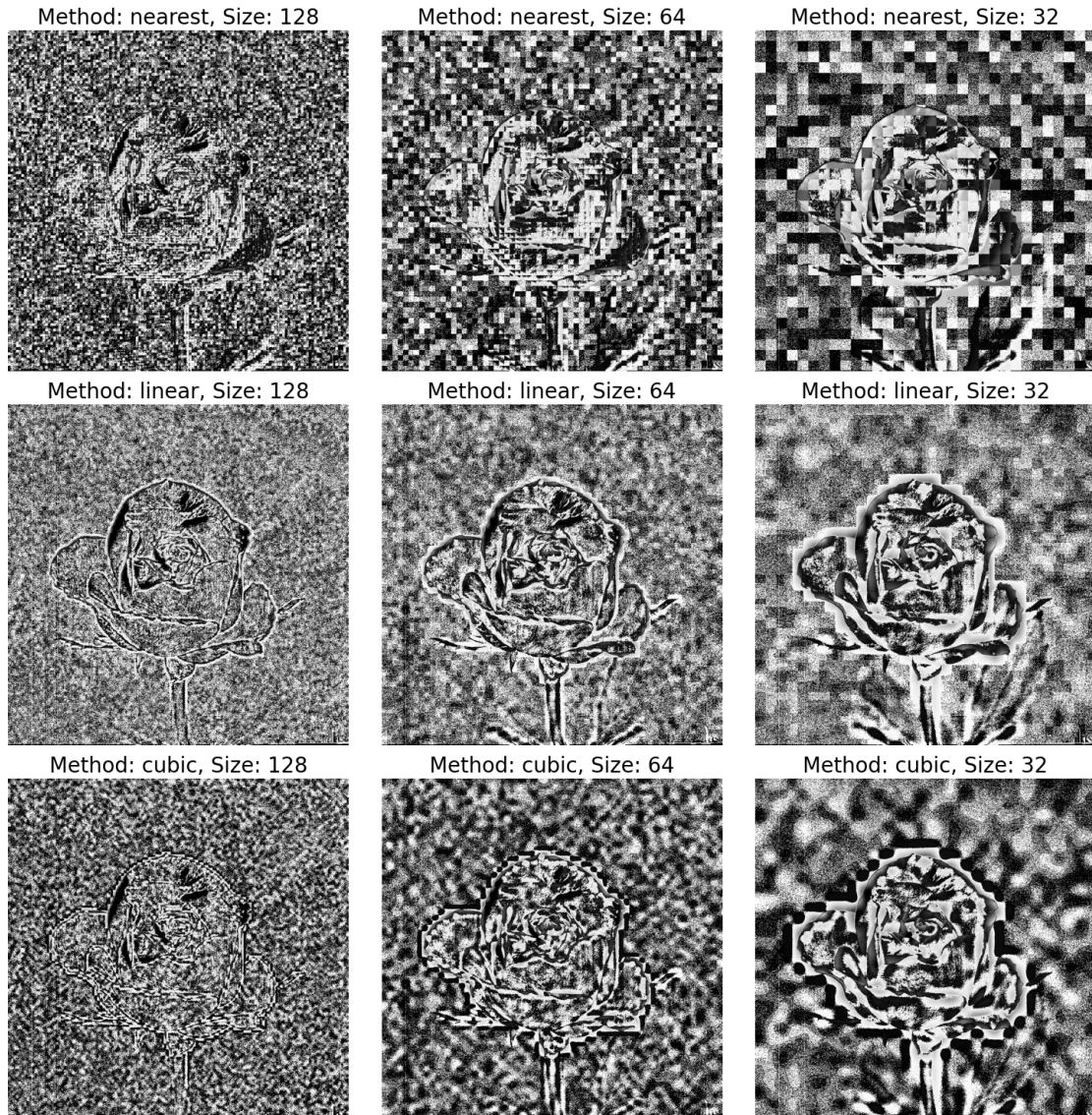
```

1 for key, value in sizes_dict.items():
2     difference_flat = [img[0].flatten() for img in value]
3     errors = [mean_squared_error(original_flat, diff) for diff in
4               difference_flat]
5     closest_index = np.argmin(errors)
6     print(f"Size:{key}")
7     print(value[closest_index][1])
8
9     most_different_index = np.argmax(errors)
10    print(value[most_different_index][1])
11    print(f"Most_different_index:{most_different_index}")

```

Код 2.3: Mean squared error code

2.2. IMAGE INTERPOLATION БҮЛЭГ 2. ДҮРС БОЛОВСРУУЛАХ/ЭХНИЙ ХЭСЭГ/



Зураг 2.3: Ялгааг харуулсан зураг

2.3 Contrast Stretching

Contrast Stretching нь зургийн төрөл нь дүрслэх боломжтой пикселийн эрчмийн бүрэн мужийг бүх (0-255) утгуудын хүрээг хамрахын тулд агуулж буй эрчмийн утгуудын мужийг сунгах замаар зургийн тодролыг сайжруулдаг.

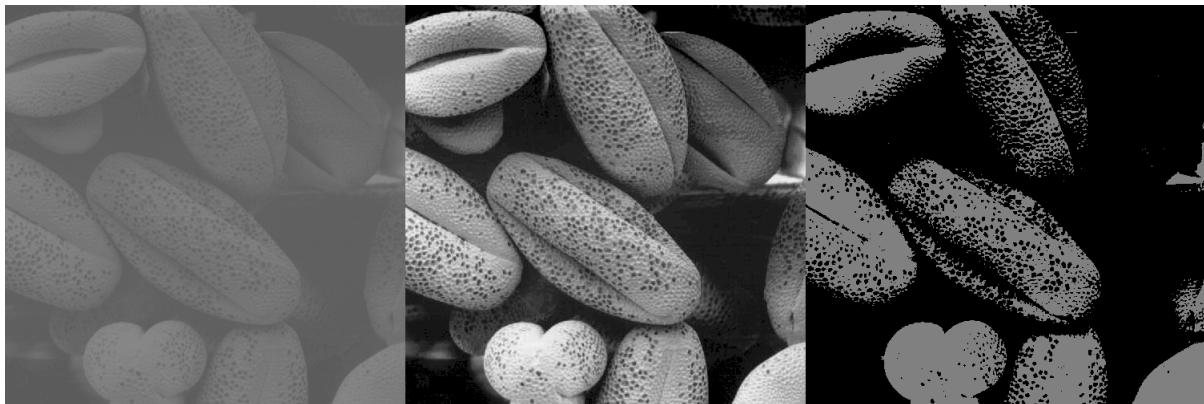
Contrast Stretching (ихэвчлэн normalization гэж нэрлэдэг) нь дүрсний тодосгогчийг хүссэн утгын хүрээг хамрахын тулд агуулж буй эрчмийн утгын мужийг ”сунгах” замаар дүрсийг сайжруулах энгийн арга юм. Энэ нь илүү боловсронгуй гистограмын тэгшитгэлээс ялгаатай нь зөвхөн зургийн пикселийн утгуудад шугаман scaling функцийг ашигладаг. Үүний үр дүнд ”сайжруулалт” нь илүү хатуу биш юм.[4]

```

1 if not os.path.exists(f"{current_directory}/result/contrast_stretching"
2     ):
3     os.makedirs(f"{current_directory}/result/contrast_stretching")
4
5 img = cv2.imread("input/lab3photo.tif")
6
7 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
8
9 min_val, max_val, _, _ = cv2.minMaxLoc(gray)
10
11 scale = (255 - 0) / (max_val - min_val)
12
13 shift = 0 - (scale * min_val)
14
15 stretched = cv2.convertScaleAbs(gray, alpha=scale, beta=shift)
16
17 bit1 = cv2.bitwise_and(stretched, 0x80)
18
19 cat = cv2.hconcat([gray, stretched, bit1])
20
21 cv2.imwrite(f"{current_directory}/result/contrast_stretching/lab3photo.
22 tif", cat)

```

Код 2.4: Contrast Stretching



Зураг 2.4: Contrast Stretching

2.4 Gray-Level Slicing

Gray-Level Slicing тусlamжтайгаар зураг дээрх саарал ёнгийн тодорхой мужийг тодруулж, бусдыг нь багасгадаг. Зорилго нь contrast нэмэгдүүлэх эсвэл тодорхой шинж чанаруудыг тодосгох явдал юм.

```

1 if not os.path.exists(f"{current_directory}/result/gray_level_slicing
2   "):
3
4     os.makedirs(f"{current_directory}/result/gray_level_slicing")
5
6
7 img = cv2.imread('input/task4.tif')
8 images = []
9
10 gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
11 bit2 = cv2.bitwise_and(gray, 0xCO)
12
13 gray = cv2.resize(gray, [400, 400])
14 bit1 = cv2.resize(gray, [400, 400])
15 bit2 = cv2.resize(gray, [400, 400])
16
17 for i in range(bit1.shape[0]):
```

```

14     for j in range(bit1.shape[1]):
15         if (bit1[i][j] < 150 and bit1[i][j] > 70):
16             bit1[i][j] = 37
17         else:
18             bit1[i][j] = 230
19
20     for i in range(bit2.shape[0]):
21         for j in range(bit2.shape[1]):
22             if (bit2[i][j] < 150 and bit2[i][j] > 70):
23                 bit2[i][j] = 70
24             else:
25                 bit2[i][j] = 180
26
27 images.append(gray)
28 images.append(bit1)
29 images.append(bit2)
30
31 cat = cv2.hconcat(images)
32 cv2.imwrite(f"{current_directory}/result/gray_level_slicing/task4.tif",
            cat)

```

Код 2.5: Gray-Level Slicing



Зураг 2.5: Gray-Level Slicing

2.5 Bit-plane slicing

Бит хавтгайн зүсэлт нь зургийг битийн хавтгайд хуваадаг бөгөөд дараа нь зургийн тодорхой шинж чанарыг тодруулах, эсвэл тодорхой битийн хавтгайг хаях замаар зургийг шахах зорилгоор тусад нь боловсруулж болно. Зургийн пиксел дэх бит бүрийг хамгийн багаас эхлээд хамгийн чухал бит хүртэл анхны зурагтай ижил хэмжээтэй хоёртын битийн хавтгай гэж үздэг.

```

1 if not os.path.exists(f"{current_directory}/result/bitplaneslicing"):
2     os.makedirs(f"{current_directory}/result/bitplaneslicing")
3
4 img = cv2.imread('input/Fig0314(a)(100-dollars).tif')
5 img = cv2.resize(img, (380, 250))
6 images = []
7
8 for i in range(8):
9     j = np.bitwise_and(img, 1 << i)
10    j = np.uint8(j * 255)
11    images.append(j)
12
13 row1 = images[:4]

```

```

14 row2 = images[4:]

15

16 row1_cat = cv2.hconcat(row1)
17 row2_cat = cv2.hconcat(row2)

18

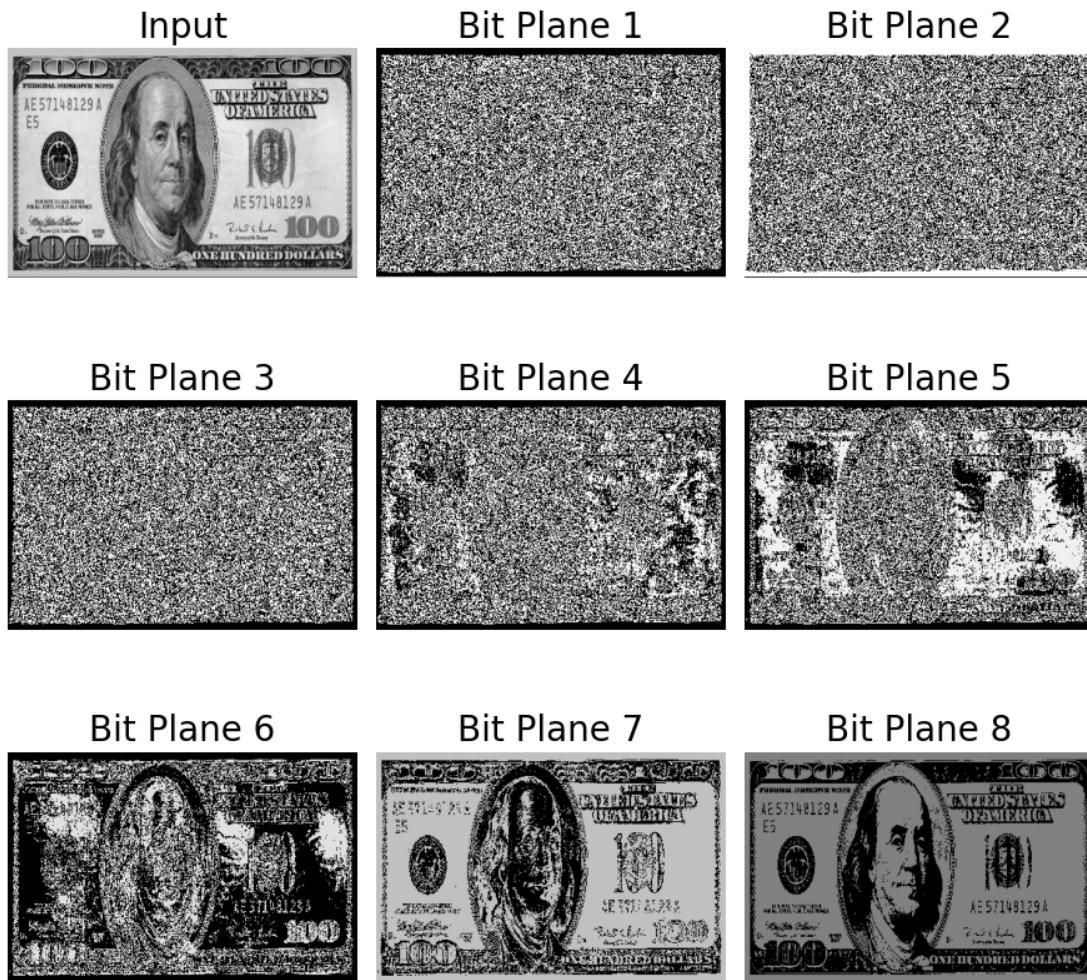
19 final_image = cv2.vconcat([row1_cat, row2_cat])
20 cv2.imwrite(f"{current_directory}/result/bitplaneslicing/task5.tif",
    final_image)

```

Код 2.6: Bit-plane slicing

2.6 Дүгнэлт

Энэ тайланда дүрсийг 5 аргаар боловсруулах талаар судаллаа. Арга бүхэн нь өөрийн гэсэн давуу ба сул талтай. Spatial and Intensity Resolution нь дүрсийг depth буюу гүнийг тодорхойлоход ашиглагддаг. Image interpolation нь зургийн нарийвчлал сайжруулах аль эсвэл compress хийхэд ашиглаж болдог. Contrast stretching нь зургийн pixel бүрийн утгын 0-255 ойртуулж хамрах хүрээг ихэсгэснээр тодосгодог. Гэх мэтчилэн маш олон хэрэглээтэй.



Зураг 2.6: Bit-plane slicing

3. ДҮРС БОЛОВСРУУЛАХ /ДУНД ХЭСЭГ/

3.1 Histogram Processing

3.1.1 Histogram Processing

Гистограмын боловсруулалт гэдэг нь түүний гистограммын тархалтыг тохируулах замаар зургийн тодосгогчийг (Contrast) сайжруулахын тулд дүрс боловсруулахад ашигладаг арга юм. X тэнхлэг нь боломжит эрчмийн түвшин бүрийг, Y тэнхлэг нь харгалзах эрчимтэй пикселийн тоог илэрхийлдэг.

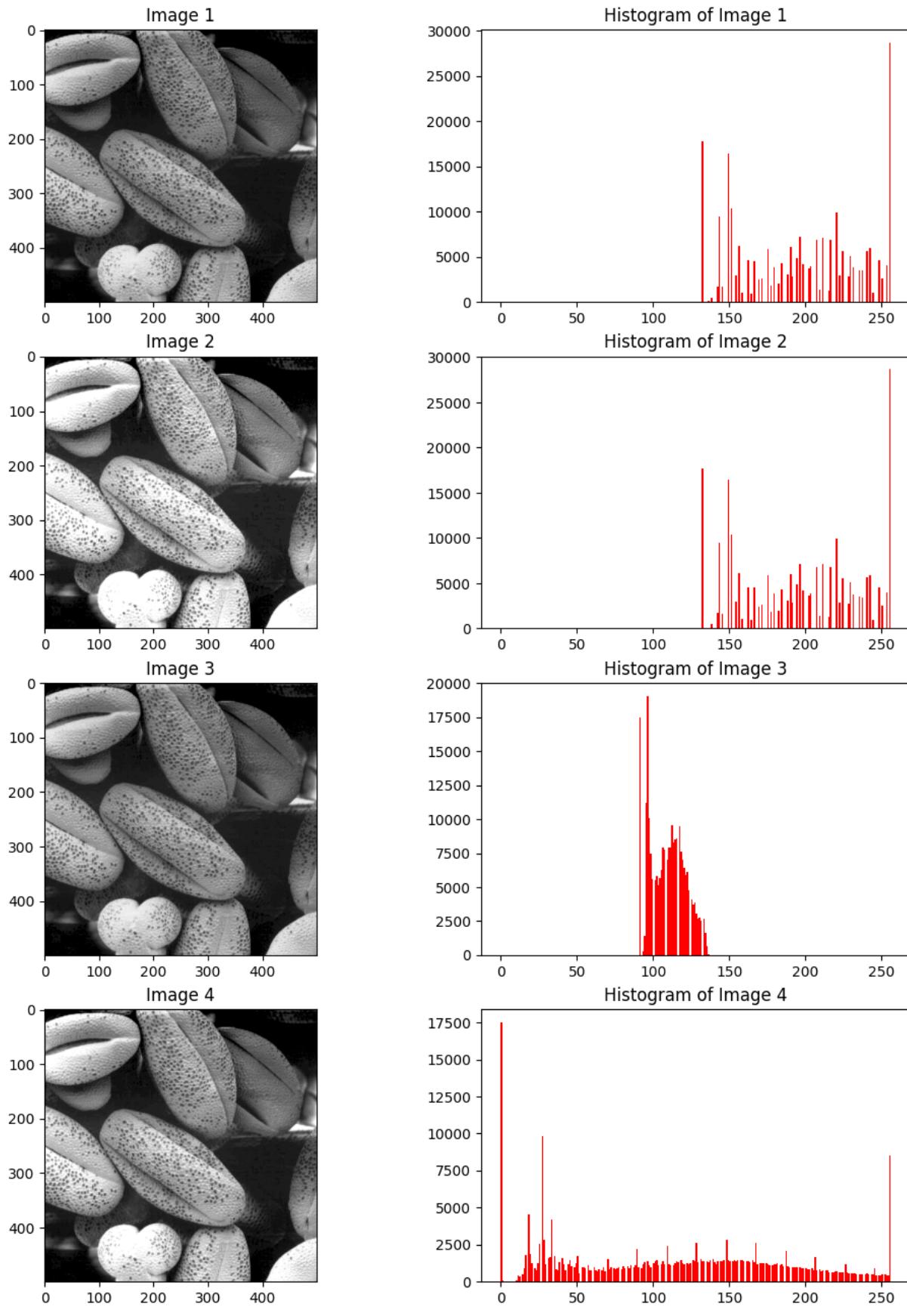
Гистограмм боловсруулах техникт гистограмм сунгах (Contrast stretching), гистограмыг гулсуулах (Brightness modification), гистограмм тэгшитгэх (Histogram equalization) зэрэг орно. Эдгээр техникиуд нь дүрсний харагдах байдлыг сайжруулахын тулд зургийн гистограммыг өөрчилдөг. Гистограммын боловсруулалтын зорилго нь ихэвчлэн жигд тархсан гистограммыг олж авах бөгөөд бүх эрчим хүчний түвшинг ижил хэмжээгээр ашигладаг[5].

3.1.2 Монотон функц

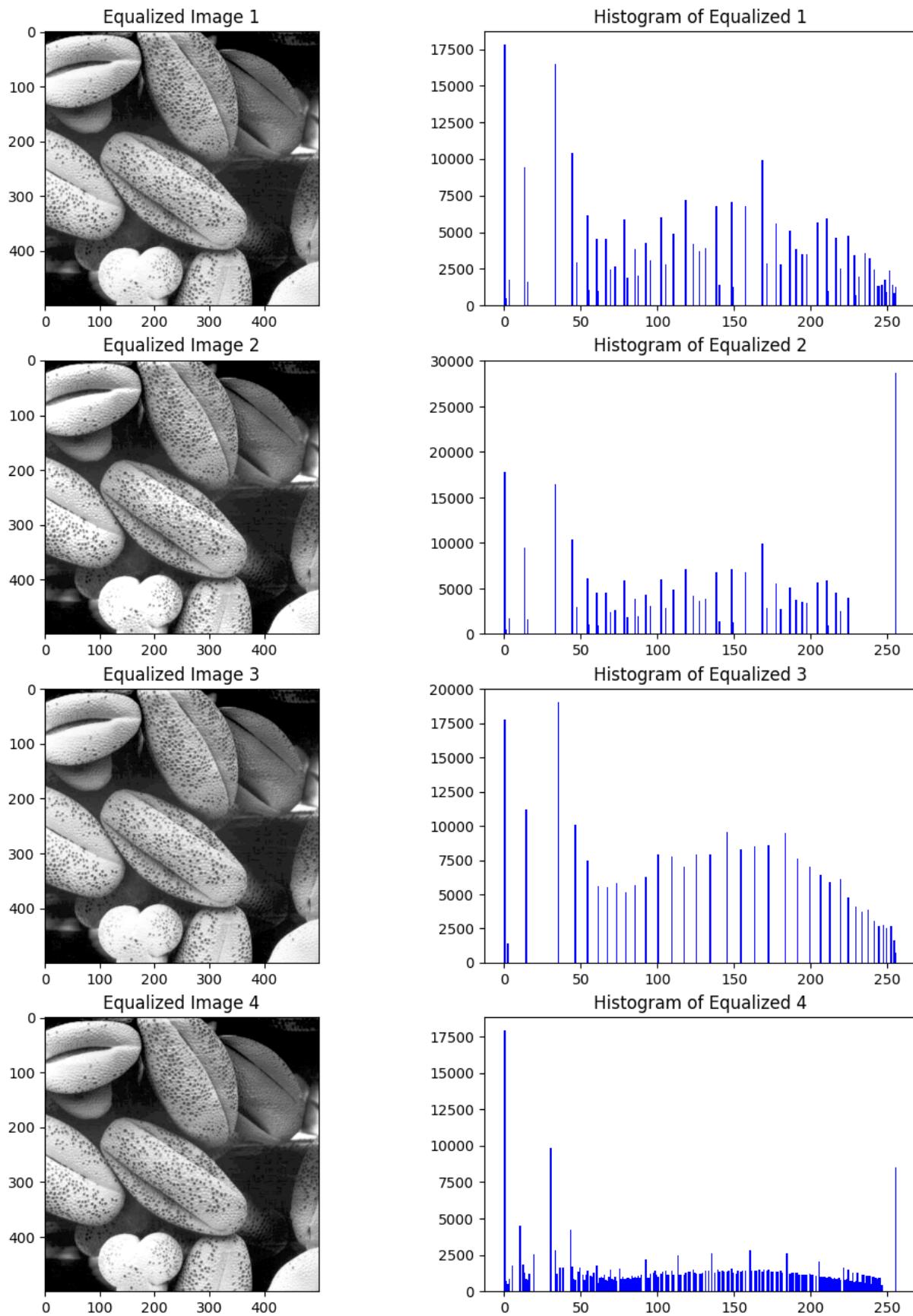
Математикт функцийг тухайн муж дахь бүх x ба y -ийн хувьд, хэрэв $x \leq y$ бол $f(x) \leq f(y)$ бол монотон өсөх (эсвэл энгийнээр хэлбэл монотон) гэж нэрлэдэг. Энэ нь функц нь хаана ч буурахгүй гэсэн үг юм. Өөрөөр хэлбэл, оролтын утгууд өсөхөд гаралтын утгууд буурахгүй.

Монотон функцууд нь тэдгээрийн графикууд нь локал максимум эсвэл минимумгүй байх шинж чанартай байдаг. Эдгээр нь энгийн, урьдчилан таамаглаж болохуйц тул математик болон янз бүрийн салбарт хэрэгтэй[6].

3.1. HISTOGRAM PROCESSING БҮЛЭГ 3. ДҮРС БОЛОВСРУУЛАХ /ДУНД ХЭСЭГ/



3.1. HISTOGRAM PROCESSING БҮЛЭГ 3. ДҮРС БОЛОВСРУУЛАХ /ДУНД ХЭСЭГ/



3.2 Local Enhancement

Local Enhancement нь зургийн чанарыг сайжруулахын тулд зураг боловсруулахад ашигладаг техник юм. Зургийг бүхэлд нь биш харин жижиг хэсгүүдийн пикселүүдийг өөрчилснөөр зургийн contrast-г сайжруулдаг тул үүнийг Local Enhancement гэж нэрлэдэг.

1. Зураг хуваах: Зургийг жижиг, давхцахгүй хавтан болгон хуваана.
 2. Гистограмыг тооцоолох: Хавтан тус бүрийн пикселийн эрчимжилтийн тархалтыг тооцоол.
 3. Ялгааг хязгаарлах: Noise нэмэгдүүлэхээс сэргийлэхийн тулд гистограммыг тодорхой утгаар таслана.
 4. Гистограммыг тэгшигтгэх: Ялгааг сайжруулахын тулд хавтан бүрт пикселийн эрчмийг тараана.
 5. Интерполяци: Хоёр шугаман интерполяцыг ашиглан хавтангийн хилийг тэгшлэнэ.
 6. Нэгтгэх: Хавтангуудыг хооронд нь холбож, эцсийн, сайжруулсан дурсийг бүрдүүлнэ.

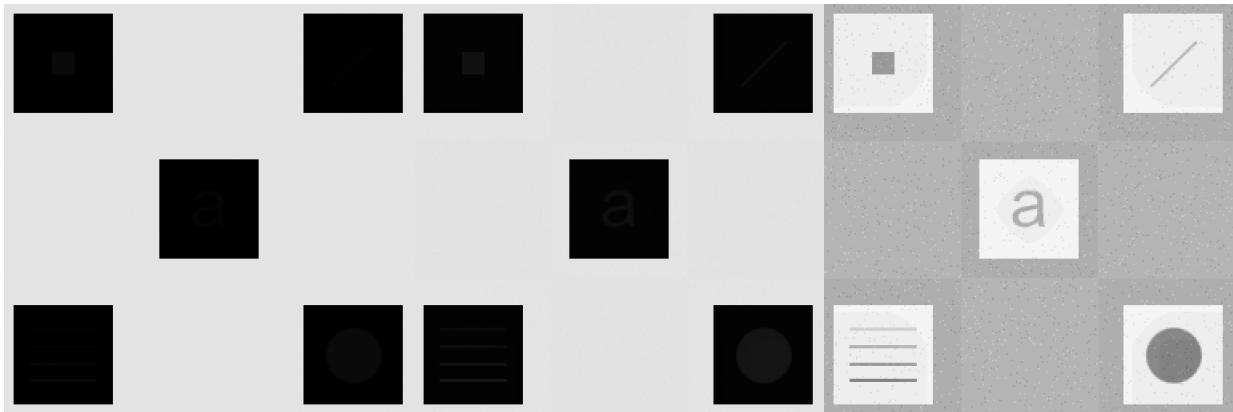
```
1 import cv2
2
3 import numpy as np
4
5 import matplotlib.pyplot as plt
6
7
8 img = cv2.imread('chap3/Fig0326(a)(embedded_square_noisy_512).tif', 0)
9
10 clahe: cv2.CLAHE = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(3,3))
11 enhanced = clahe.apply(img)
12 equalization = np.uint8(enhanced * 250) # type: ignore
13
14 fig, axs = plt.subplots(2, 2, figsize=(10, 10))
```

```

13  axs[0, 0].imshow(img, cmap='gray')
14  axs[0, 1].imshow(enhanced, cmap='gray')
15  axs[1, 0].imshow(equalization, cmap='gray')

```

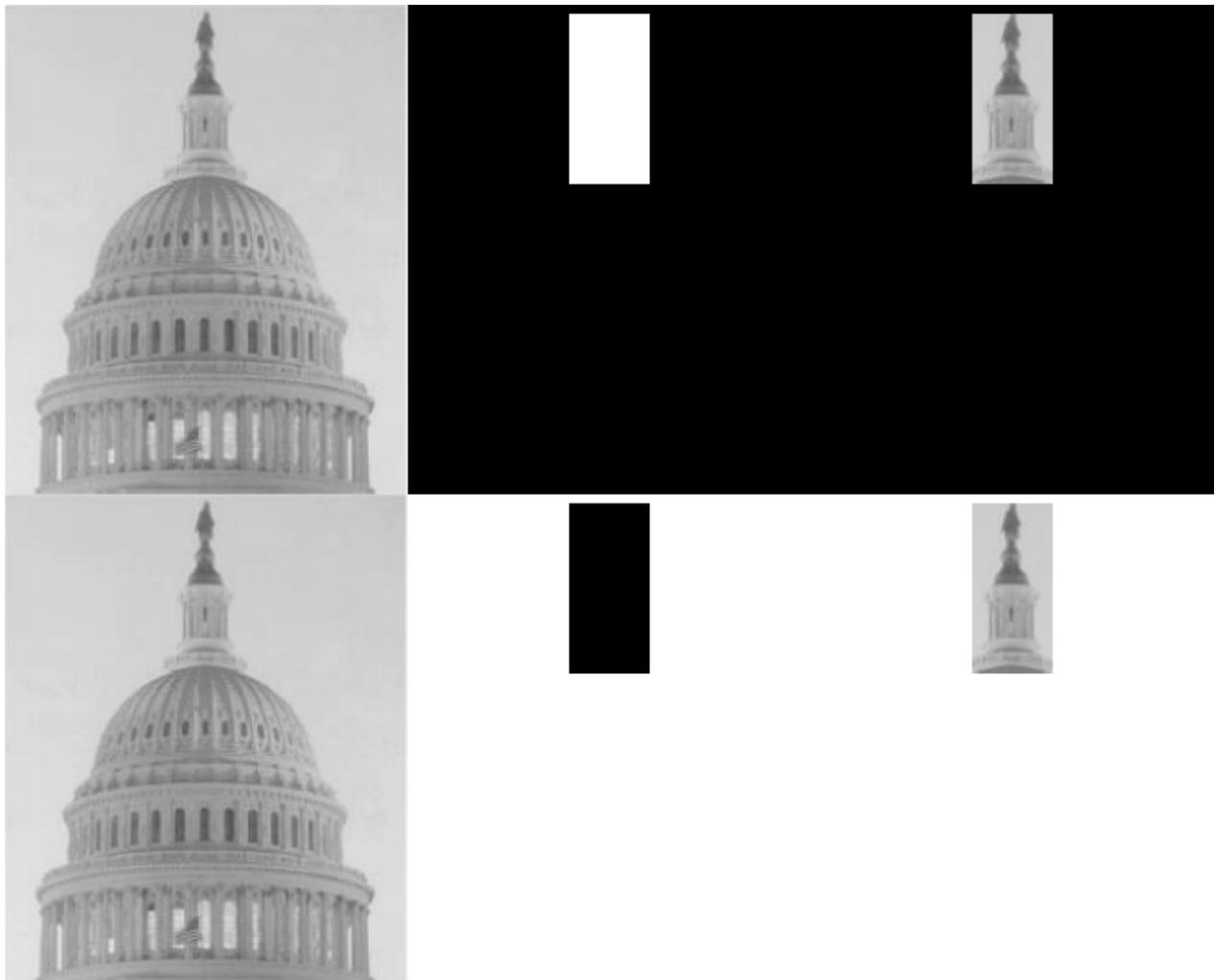
Код 3.1: Local Enhancement



Зураг 3.3: Local Enhancement

3.3 Enhancement using Arithmetic / Logic Operations

Арифметик болон логик үйлдлүүдийг ашиглан дүрсийг сайжруулах нь зургийн харагдах байдлыг сайжруулах эсвэл илүү сайн хувиргасан дүрслэлийг бий болгодог. Үүний тулд нэмэх, хасах, үргүүлэх, хуваах, битийн үйлдлүүд (AND, OR, NOT, XOR) зэрэг янз бүрийн арга техникийг хэрэглэдэг. Зураг өөрөө битүүдийн цуваагаар дүрслэгддэг тул бид хоёр зургийн хооронд бүхий л арифметик болон логик үйлдлүүдийг хэрэглэж болно. Хоёр зураг нь яг адлихан хэмжээтэй байх ёстой ба эдгээр аргуудыг ашигласнаар зургийн чанарын сайжруулах зарим хэсгийн илүү тодруулж харах гэх мэт олон боломж олгоно.



Зураг 3.4: Arithmetic / Logic Operations

3.4 Smoothing linear filters

Smoothing Linear Filters нь зураг боловсруулахад ашигладаг шүүлтүүрүүдийн ангилал бөгөөд noise багасгах, зургийг бүдгэрүүлэх, ирмэг илрүүлэх гэх мэт зургийг цаашдийн боловсруулалтад бэлтгэхэд тусалдаг.

Эдгээр шүүлтүүрүүд нь маскыг дүрстэй хослуулах замаар ажилладаг. Kernel гэж нэрлэгддэг маск нь бидний зурган дээрх утгыг тохируулахад ашиглагддаг жижиг матриц юм.

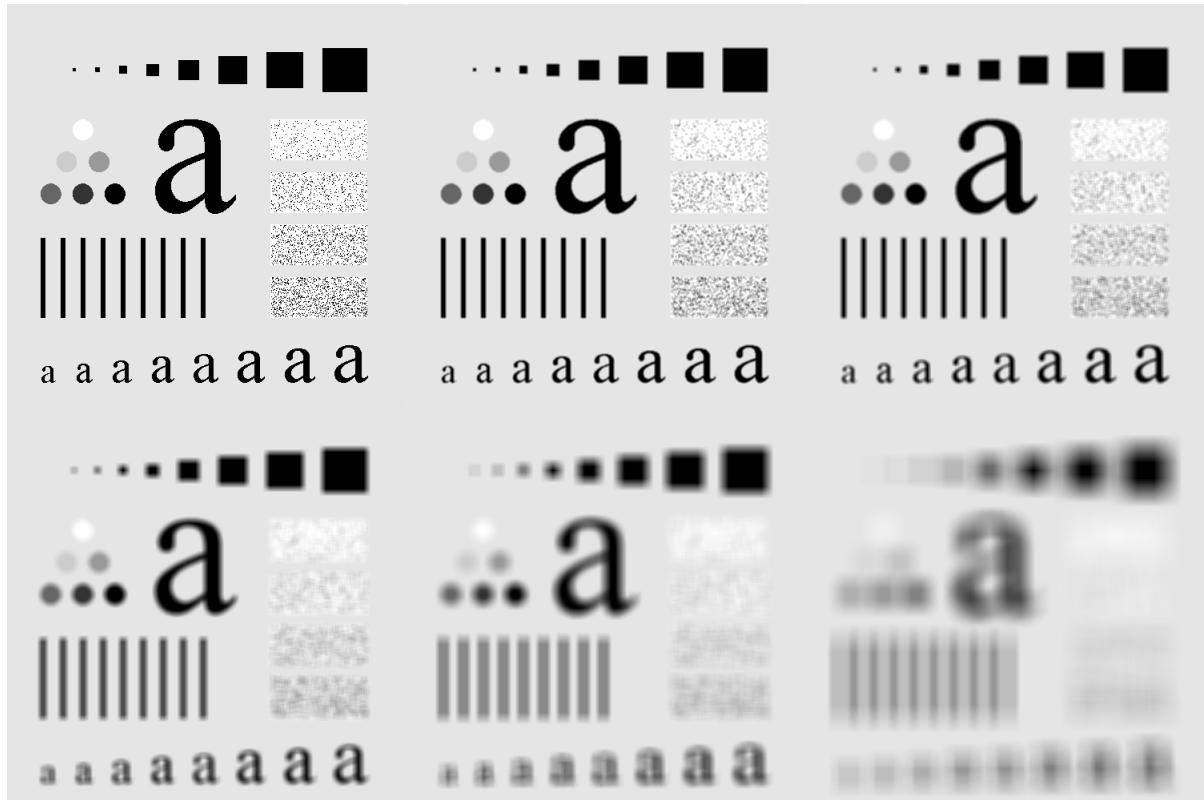
Box Filter (or Mean Filter)

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

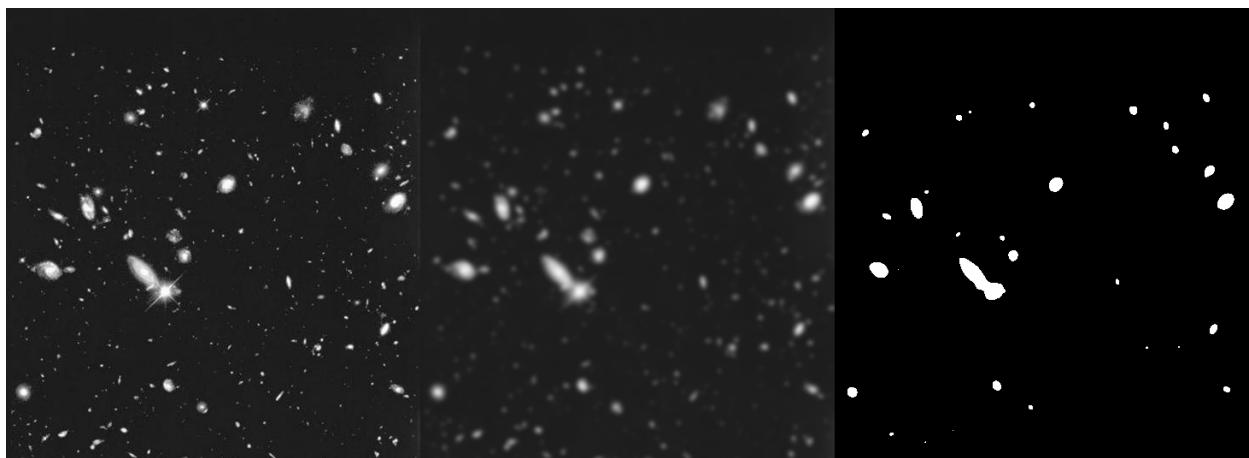
Gaussian Filter

Гауссын кернелд төвийн буюу тухайн утга тохируулагдаж байгаа pixel илүү өндөр жинтэй буюу илүү чухал байдаг.

$$G = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



Зураг 3.5: Smoothing linear filters



Зураг 3.6: Smoothing Linear Filters

A. ЖИШЭЭ ЗУРАГНУУД



Зураг А.1: Жишээ-1



Зураг А.2: Жишээ-2

Bibliography

- [1] Б. Сувдаа, (2021). Хэв танилтын үндэс лабораторийн гарын авлага. Улаанбаатар.
- [2] OpenCV, (2023). Documentation. Opencv https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.
- [3] Arturo Amor, Lucy Liu (2023). Documentation. scikit-learn https://scikit-learn.org/stable/modules/model_evaluation.html.
- [4] R. Fisher, S. Perkins, A. Walker, E. Wolfart (2023). Contrast Stretching <https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm>.
- [5] Gonzalez, R. C., & Woods, R. E. (2007). Digital Image Processing (3rd ed.). Pearson.
- [6] Rudin, W. (1976). Principles of Mathematical Analysis (3rd ed.). McGraw-Hill Education.
- [7] Хичээлийн веб хуудас <https://suvdaa.webs.com/class.htm>