TUTORIAL 1

Image Enhancement in Spatial Domain

Pengolahan Citra - Semester Gasal 2023/2024

1 *Point Processing*

Point processing merupakan operasi pemrosesan citra yang melibatkan satu piksel saja. Selanjutnya, akan dibahas beberapa teknik point processing yang akan didemonstrasikan menggunakan citra-citra *grayscale*.

1.1 Pendahuluan dan Tips *Debugging*

Sebuah gambar dua dimensi biasanya dapat dibayangkan seperti sebuah matriks tiga dimensi, di mana dua dimensi menunjukkan posisi sebuah piksel pada gambar dan dimensi ketiga merepresentasikan nilai dari tiap kanal untuk piksel tersebut.
Misalkan, sebuah citra berformat RGB berdimensi x kali y piksel dapat kita bayangkan sebagai sebuah matriks 3 dimensi berukuran x kali y kali 3 (untuk masing-masing kanal R, G, dan B). Meskipun demikian, representasi dari sebuah data citra yang dijalankan pada sebuah program bisa jadi berbeda-beda. Mengetahui seperti apa bentuk data yang anda sedang kerjakan bisa membantu anda debugging program anda.
Melakukan `print` data citra setelah di-load ke variabel bisa membantu anda menerka seperti apa format citra yang sedang anda kerjakan. Manfaatkan juga fungsi-fungsi konversi dari modul `color` karena terkadang, sebuah citra yang tampaknya grayscale bisa jadi memiliki format data RGB!
Selain dimensi, representasi value dari piksel-piksel sebuah citra bisa jadi berbeda. Terkadang, anda akan menemukan citra dengan piksel-piksel yang direpresentasikan sebagai angka desimal *float* dari 0 sampai 1 dan terkadang juga, anda akan menemukannya dalam bentuk unsigned byte integer 0 sampai 255. Manfaatkan fungsi `util.img_as_ubyte` dan fungsi sejenis untuk mengubah format value piksel citra. Terkadang, hasil pemrosesan citra anda bisa menjadi hitam seluruhnya akibat format yang tidak sesuai!

1.2 Image Negative

Secara matematis, image negative dapat diekspresikan sebagai berikut.

$$G_{baru} = 255 - G_{lama}$$

```
ori_img = util.img_as_ubyte(color.rgb2gray(
            io.imread('./gedung-fasilkom-ui.jpg')))
neg_img = 255 - ori_img

plt.figure(figsize=(10, 5))
plt.subplot(1,2,1); plt.imshow(ori_img)
plt.title('Original', fontsize=15); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(neg_img)
plt.title('Negative', fontsize=15); plt.axis("off")

plt.show()
```

Berikut ini adalah contoh citra hasil pemrosesan *image negative*,

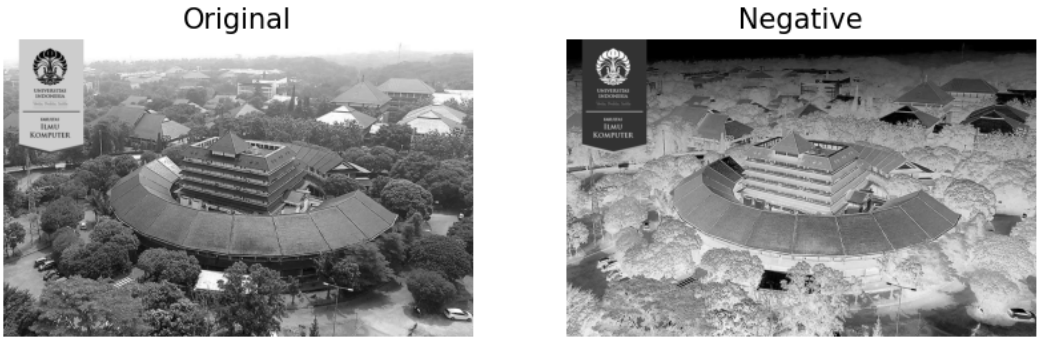

Figure 1: Gedung Fasilkom

1.3 Log Processing

Log Processing merupakan suatu proses pemetaan suatu citra dengan jangkauan warna yang sempit menjadi lebih lebar pada citra keluarannya. Tujuan dari implementasi log processing untuk mengekspansi nilai piksel gelap dan kompresi nilai piksel terang.

Secara matematis, log processing dapat diekspresikan sebagai berikut.

$$s = c \cdot log(1 + r)$$

di mana

- s = citra yang dihasilkan oleh *log processing*
- c = konstanta
- r = citra yang akan ditransformasi

```
a = ori_img / 255
c1 = 0.4
c2 = 1.7
l1 = c1 * np.log(1 + (a))
l2 = c2 * np.log(1 + (a))

plt.figure(figsize=(21, 14))

plt.subplot(1,3,1); plt.imshow(ori_img)
plt.title('Original', fontsize=15); plt.axis("off")
plt.subplot(1,3,2); plt.imshow(l1)
plt.title(f'C = {c1}', fontsize=15); plt.axis("off")
plt.subplot(1,3,3); plt.imshow(l2)
plt.title(f'C = {c2}', fontsize=15); plt.axis("off")
plt.show()
```

Berikut ini adalah contoh citra hasil pemrosesan *log processing*,

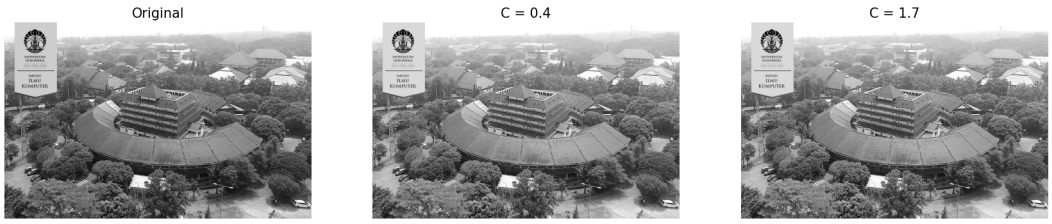

Figure 2: Gedung Fasilkom

1.4 Gamma Transformation

Secara matematis, *gamma transformation* dapat diekspresikan sebagai berikut,

$$s = cr^p$$

di mana

- s = citra hasil *gamma transformation*
- c = konstanta
- p = konstanta
- r = *input image*

```
a = ori_img / 255
c = 1
p1 = 0.4
p2 = 1.7
g1 = c * (a**p1)
g2 = c * (a**p2)

plt.figure(figsize=(21, 14))

plt.subplot(1,3,1); plt.imshow(ori_img)
plt.title('Original', fontsize=15); plt.axis("off")
plt.subplot(1,3,2); plt.imshow(g1)
plt.title(f'P = {p1}', fontsize=15); plt.axis("off")
plt.subplot(1,3,3); plt.imshow(g2)
plt.title(f'P = {p2}', fontsize=15); plt.axis("off")
plt.show()
```

Berikut ini adalah contoh citra hasil pemrosesan *gamma transformation*,

1.5 Image Histogram

Histogram merupakan suatu teknik penyajian data spasial (pixel) yang mampu memberikan informasi global dari sebuah citra. Pada umumnya, histogram menyajikan statistik persebaran nilai dari setiap piksel pada citra. Pada citra *grayscale*, nilai 0 sampai 255 (dalam

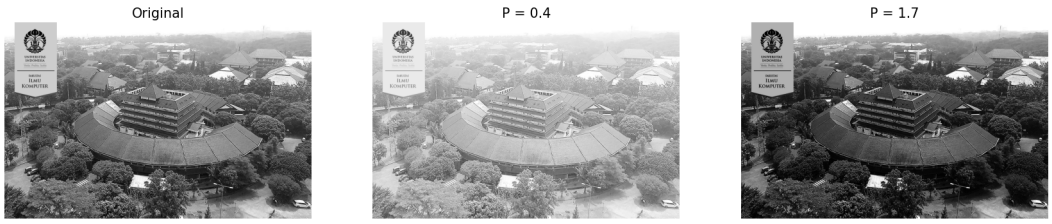

Figure 3: Gedung Fasilkom

integer) menunjukkan tingkat terang-gelap sebuah piksel. Semakin besar nilainya, maka semakin cerah piksel tersebut.

Sebuah histogram dapat menampilkan kontras dari citra tersebut. Apabila kebanyakan piksel terdistribusi dalam rentang yang sempit, citra yang dihasilkan akan samar-samar karena kumpulan piksel yang membentuknya bernilai mirip. Terdapat dua macam metode untuk melakukan manipulasi pada histogram, seperti

- *Contrast Stretching*
- *Histogram Equalization*

```
from skimage import util
plt.figure(figsize=(10, 5))

img_flat = util.img_as_ubyte(ori_img).flatten()
plt.subplot(1,1,1)
plt.hist(img_flat, 256, range=(0,256))
plt.title('Original', fontsize=15)
```

Berikut ini adalah contoh histogram pada gambar gedung fasilkom,

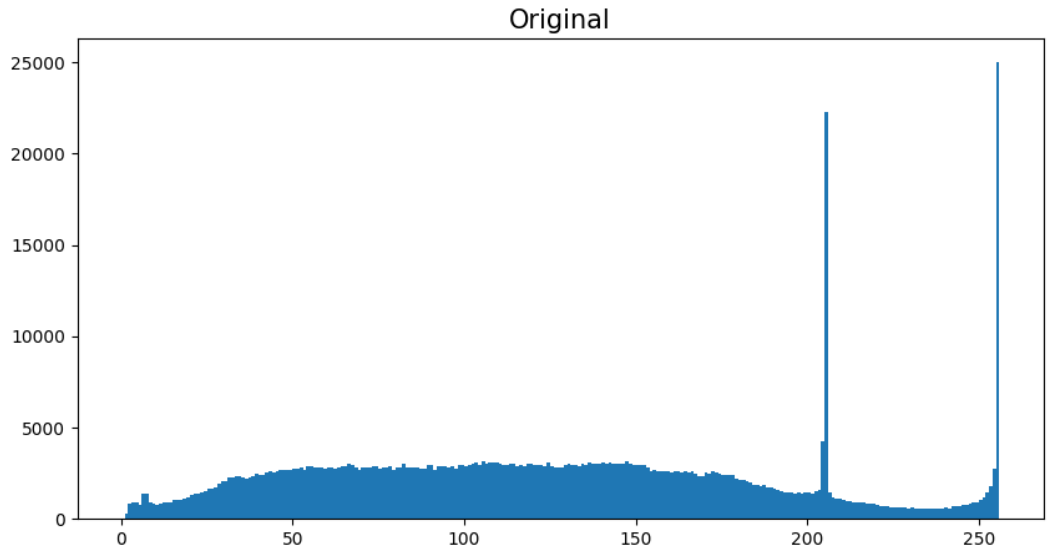

Figure 4: Gedung Fasilkom Histogram

1.6 Contrast Stretching

Contrast Stretching sesuai dengan namanya, yakni merentangkan piksel sedemikian sehingga tampilannya menjadi kontras. Secara umum, *contrast stretching* mengembangkan rentang distribusi nilai intesitas piksel pada histogram citra secara seragam. Teknik ini dapat digunakan untuk meningkatkan kontras suatu citra. Hal ini dikarenakan citra yang dihasilkan oleh teknik ini memiliki persebaran *gray value* yang lebih luas.

Berikut demonstrasi perbedaan citra original dengan hasil contrast stretching citra.

```
animal = util.img_as_ubyte(color.rgb2gray(
            io.imread('./animal-fog.jpg')))
mn = min(animal.flatten())
mx = max(animal.flatten())
b = int(np.floor(255 / (mx - mn)))
animal_cs = (animal - mn) * b

plt.figure(figsize=(12, 7))
plt.subplot(1,2,1); plt.imshow(animal)
plt.title('Original', fontsize=15); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(animal_cs)
plt.title('Contrast Stretching', fontsize=15); plt.axis("off")
plt.show()
```

Berikut ini adalah contoh citra yang telah diterapkan proses *contrast stretching*,

Figure 5: Animal Picture Low Contrast

Berikut demonstrasi perbedaan histogram pada citra original dengan hasil contrast stretching.

```
from skimage import util
plt.figure(figsize=(10, 5))

animal_flat = util.img_as_ubyte(animal).flatten()
plt.subplot(2,1,1)
plt.hist(animal_flat, 256, range=(0,256))
plt.title('Original', fontsize=15)
```

```
8
9  animal_cs_flat = util.img_as_ubyte(animal_cs).flatten()
10 plt.subplot(2,1,2)
11 plt.hist(animal_cs_flat, 256, range=(0,256))
12 plt.title('Contrast Stretching', fontsize=15)
13
14 plt.subplots_adjust(hspace=0.5)
15 plt.show()
```

Berikut ini adalah contoh histogram dari citra yang telah diterapkan proses *contrast stretching*,

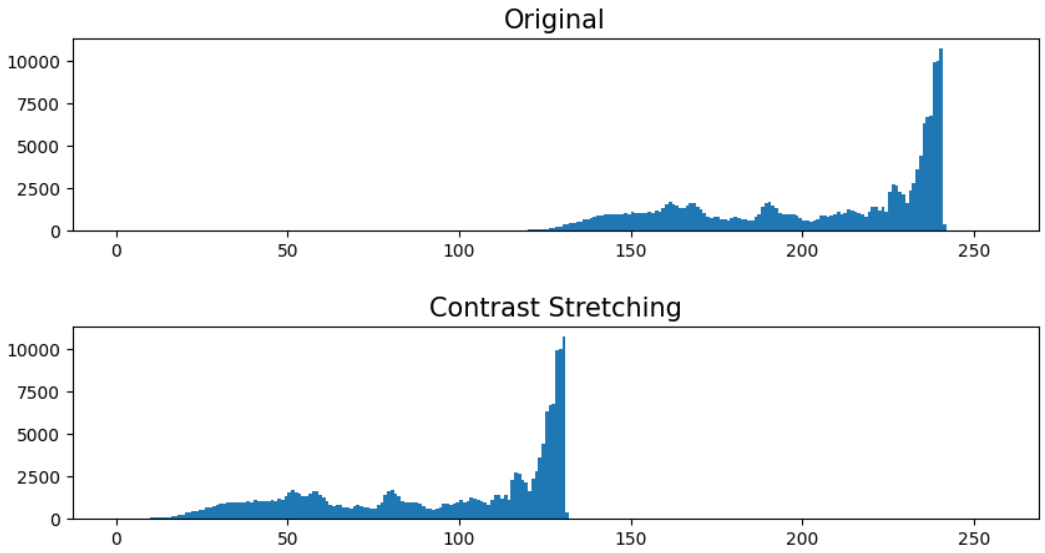

Figure 6: Histogram of Animal Picture Low Contrast

1.7 Histogram Equalization

Teknik *histogram equalization* sebenarnya mirip dengan *contrast stretching*. Perbedaan antara keduanya terletak pada frekuensi dari setiap *value* ikut diperhitungkan saat dipetakan ke nilai intensitas lain. Teknik *histogram equalization* ini dapat digunakan untuk meningkatkan kontras sebuah citra.

Berikut demonstrasi perbedaan citra original dengan hasil histogram equalization citra.

```
1 from skimage import exposure
2 animal_eq = exposure.equalize_hist(animal)
3
4 plt.figure(figsize=(12, 7))
5
6 plt.subplot(1,2,1); plt.imshow(animal)
7 plt.title('Original', fontsize=15); plt.axis("off")
8
9 plt.subplot(1,2,2); plt.imshow(animal_eq)
```

```
plt.title('Histogram Equalization', fontsize=15);
plt.axis("off")

plt.show()
```

Berikut ini adalah contoh citra yang telah diterapkan proses *histogram equalization*,

Figure 7: Animal Picture Histogram Equalization

Berikut demonstrasi perbedaan histogram pada citra original dengan hasil histogram equalization.

```
plt.figure(figsize=(12, 7))

animal_flat = util.img_as_ubyte(animal).flatten()
plt.subplot(2,1,1)
plt.hist(animal_flat, 256, range=(0,256))
plt.title('Original', fontsize=15)

animal_eq_flat = util.img_as_ubyte(animal_eq).flatten()
plt.subplot(2,1,2)
plt.hist(animal_eq_flat, 256, range=(0,256))
plt.title('Histogram Equalization', fontsize=15)

plt.subplots_adjust(hspace=0.5)
plt.show()
```

Berikut ini adalah contoh histogram dari citra yang telah diterapkan proses *histogram equalization*,

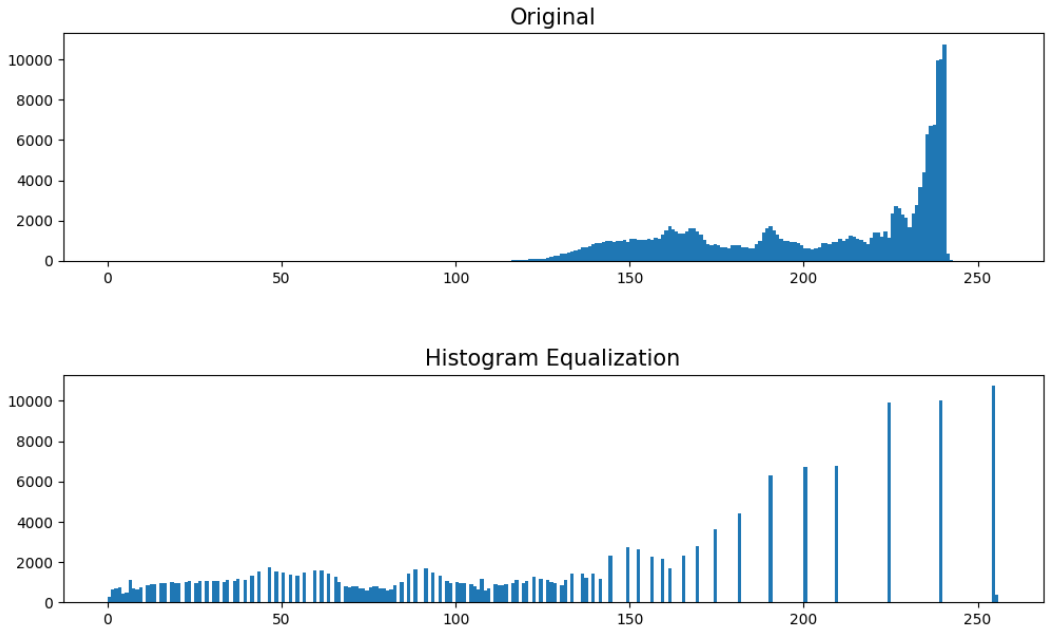

Figure 8: Histogram of Animal Picture Histogram Equalization

1.8 *Image Subtraction*

Image subtraction adalah proses pengurangan pixel nilai dari dua gambar digital. Ini digunakan untuk membandingkan dua gambar dan menyoroti perbedaan antara keduanya. Setiap pixel dalam gambar hasil merupakan selisih nilai pixel yang sesuai antara dua gambar awal. Teknik ini sering digunakan dalam aplikasi seperti deteksi perubahan pada gambar medis atau analisis perbedaan antara gambar sebelum dan sesudah peristiwa tertentu. Hasilnya adalah gambar yang mempertegas perbedaan visual antara dua gambar yang dibandingkan. *Image subtraction* dapat dilakukan jika kita ingin mengambil bagian tertentu saja dari citra.

```
i_mix = io.imread(home_dir + 'pancen.jpg')
i_back = io.imread(home_dir + 'bg.jpg')

# Konversi agar citra bisa diolah
i_mix = util.img_as_ubyte(color.rgb2gray(i_mix))
i_back = util.img_as_ubyte(color.rgb2gray(i_back))

i_subs = np.subtract(i_mix, i_back, dtype='int16')

plt.figure(figsize=(21, 14))
plt.subplot(1,3,1); plt.imshow(i_mix, cmap='gray')
plt.title('Original', fontsize=20); plt.axis("off")
plt.subplot(1,3,2); plt.imshow(i_back, cmap='gray')
plt.title('Background', fontsize=20); plt.axis("off")
plt.subplot(1,3,3); plt.imshow(i_subs, cmap='gray')
```

```
plt.title('Substracted', fontsize=20); plt.axis("off")
plt.show()
```

Figure 9: Visualisasi *Image substraction*

2 *Mask Processing*

- Mask Processing sering juga disebut dengan ***image filtering***.
- Operasi ini mempertimbangkan intensitas piksel dalam *neighborhood* dan koefisien filter yang berdimensi sama dengan *neighborhood*.
- Filter pada dasarnya adalah sebuah metode untuk meredam atau menghilangkan *noise* pada citra digital.
- Terdapat 2 jenis filter spasial; ***smoothing filter*** dan ***sharpening filter***.

2.1 *Smoothing Filter*

- *Smoothing* digunakan untuk mengaburkan citra dan mengurangi *noise*.
- Biasa dilakukan saat *pre-processing* citra.
- Pros: mereduksi *noise* dan detail yang tidak relevan dalam suatu citra. Cons: *edge* menjadi kabur.
- Bisa dilakukan dengan menggunakan linear filter ataupun non linear filter.

2.1.1 *Linear Filter*

Metode yang digunakan: korelasi atau konvolusi (perkalian setiap piksel di lingkungan dengan koefisien yang sesuai dan menjumlahkan hasil untuk mendapatkan respon pada setiap titik (x, y).)

Berikut beberapa macam filter linear yang dibahas pada lab ini.

1. **Average Filter**
   Mengganti intensitas tiap piksel dalam citra dengan rata-rata intensitas pada neighborhood.

```
from skimage import filters, morphology
i1 = io.imread(home_dir + 'palembang.jpg')
gray = color.rgb2gray(io.imread(home_dir +
    'palembang.jpg'))
fi = filters.rank.mean(gray,selem=morphology.square(4))

plt.figure(figsize=(15, 10))
plt.subplot(1,2,1)
plt.imshow(i1,cmap='gray',vmin=0,vmax=255)
plt.title('Original', fontsize=20); plt.axis("off")

plt.subplot(1,2,2)
plt.imshow(fi,cmap='gray',vmin=0,vmax=255)
plt.title('Average Filtered', fontsize=20); plt.axis("off")
plt.show()

plt.imshow(i1,cmap='gray',vmin=0,vmax=255)
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(fi,cmap='gray',vmin=0,vmax=255)
plt.title('Average Filtered'); plt.axis("off")
plt.show()
```

Figure 10: Visualisasi *Average filter*

2. **Gaussian Filter** Mengganti intensitas tiap piksel dalam citra dengan hasil *image fil-*

tering dari operasi konvolusi antara intensitas tiap piksel dalam citra dengan distribusi Gaussian. Berikut contoh langkah-langkah diterapkannya Gaussian Filter pada suatu gambar berwarna.

(a) Gambar diperkecil hingga 5 kali 5 piksel dan diekstraksi/direpresentasikan ke dalam matriks

$$\begin{bmatrix} 123 & 123 & 156 & 156 & 106 \\ 115 & 123 & 177 & 154 & 101 \\ 121 & 90 & 126 & 113 & 146 \\ 106 & 77 & 110 & 118 & 174 \\ 104 & 56 & 128 & 176 & 110 \end{bmatrix}$$

(b) Melakukan penentuan *filter/mask* yang ditentukan dengan distribusi Gaussian 2D. Contohnya menggunakan kernel berukuran 3 kali 3.

$$\begin{bmatrix} 0.075 & 0.124 & 0.075 \\ 0.124 & 0.204 & 0.124 \\ 0.075 & 0.124 & 0.075 \end{bmatrix}$$

(c) Melakukan image filtering dengan menerapkan operasi konvolusi sebagai berikut.

$$h(m,n) = (123*0.075)+(123*0.124)+(156*0.075)+(115*0.124)+(123*0.204)+$$
$$(177*0.124)+(121*0.075)+(90*0.124)+(126*0.075)$$

Sehingga dihasilkan matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 127 & 142 & 139 & 0 \\ 0 & 112 & 122 & 133 & 0 \\ 0 & 96 & 111 & 133 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
fi_gauss = util.img_as_ubyte(filters.gaussian(gray,
    sigma=1))
plt.figure(figsize=(15, 10))
plt.subplot(1,2,1)
plt.imshow(i1,cmap='gray',vmin=0,vmax=255)
plt.title('Original', fontsize=20); plt.axis("off")
```

2.1.2 *Non Linear Filter*

- Responnya didasarkan pada pengurutan (*ranking*) dari intensitas piksel-piksel yang dilingkup oleh filter.
- Selanjutnya, intensitas piksel pada pusat filter diganti dengan intensitas hasil pengurutan.

Figure 11: Visualisasi *Gaussian filter*

- Filter non-linear memiliki lebih banyak keunggulan dibandingkan filter linear dengan ukuran filter yang sama.

Sebelum membahas filter non linear, akan diperkenalkan suatu konsep *noise*, yaitu *salt-pepper noise* :

Figure 12: *Salt and Pepper Noise*

Pada gambar 12 ditampilkan contoh *salt and pepper noise*. Gambar kedua merupakan suatu kasus dimana *salt noise* mendominasi gambar sehingga gambar terlihat putih, sedangkan gambar ketiga merupakan kasus *pepper noise* dimana bintik hitam *pepper* mendominansi gambar.

Berikut beberapa macam filter non linear yang dibahas pada lab ini.

1. **Median Filter**

   - Mengganti intensitas piksel pada pusat filter dengan median dari intensitas *neighborhood*.
   - Efektif untuk menghilangkan *impulse noise/salt-pepper noise*.
   - Intensitas piksel dalam *neighborhood* diurutkan → menentukan nilai median → mengganti intensitas piksel pada pusat *neighborhood* dengan median.

```
gray_img = color.rgb2gray(input_image)
noise = util.img_as_ubyte(util.random_noise(gray_img ,mode
    = "s&p" , salt_vs_pepper =0.02))
fi_median = filters.rank.median( noise , selem =
    morphology.square(3))
```

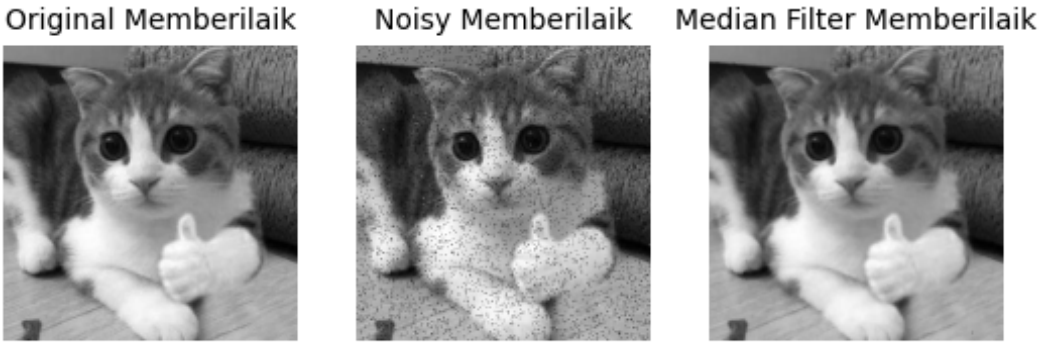

Figure 13: Median Filtering

2. Minimum Filter

   - Mengganti intensitas piksel pada pusat filter dengan minimum dari intensitas *neighborhood*.
   - Biasa digunakan untuk menghilangkan *positive outlier noise*.

```
fi_mini = filters.rank.minimum(gray_img,
    selem=morphology.square(4))
```

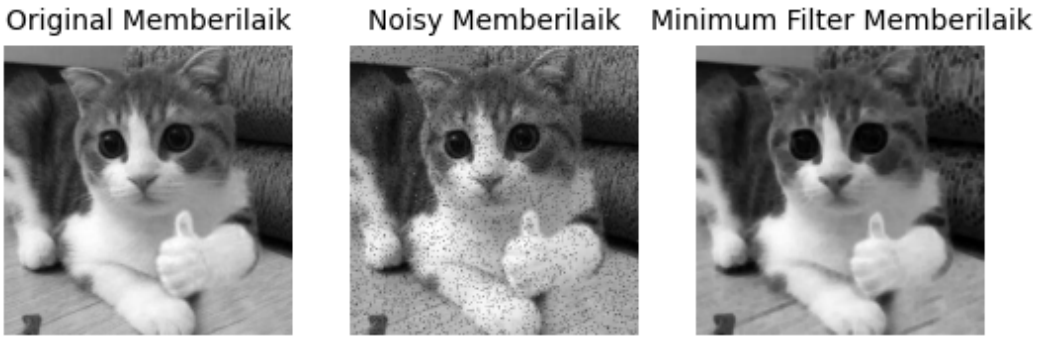

Figure 14: Minimum Filtering

3. Maximum Filter

   - Mengganti intensitas piksel pada pusat filter dengan maksimum dari intensitas *neighborhood*.
   - Biasa digunakan untuk menghilangkan *negative outlier noise*.

```
fi_maxi = filters.rank.maximum(gray_img,
   selem=morphology.square(4))
```

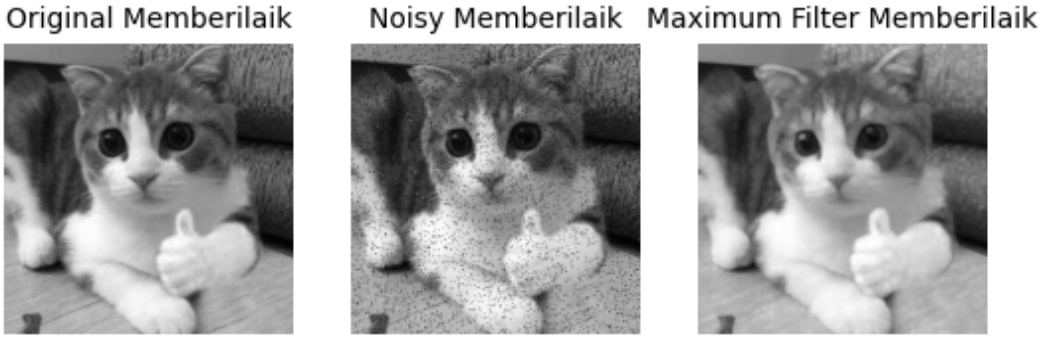

Figure 15: Maximum Filtering

2.2 *Sharpening Filter*

- *Sharpening* digunakan untuk menonjolkan detail dalam citra atau untuk mempertajam detail dalam citra atau untuk mempertajam detail yang kabur.
- Biasa dilakukan saat *pre-processing* citra.
- Kebalikan dari *smoothing* yang dilakukan dengan rata-rata (dengan integrasi), sharpening dapat dilakukan dengan derivatif (penurunan).

Robert, Prewitt, Sobel (*Edge Detection*)

Edge detection menghasilkan tepi-tepi dari objek citra. Suatu titik (x, y) dikatakan sebagai tepi (*edge*) dari suatu citra apabila mempunyai perbedaan yang tinggi dengan tetangganya.

- **Roberts**: metode ini memanfaatkan diferensial pada arah horizontal dan vertikal. Kernel filter yang digunakan dalam metode Roberts:

$$H = \begin{bmatrix} -1 & 1 \end{bmatrix}, \text{dan} \quad H = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

- **Prewitt**: Pengembangan metode Robert dengan menggunakan filter HPF (*High Pass Filter*) yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF (*High Pass Filter*). Kernel filter yang digunakan dalam metode Prewitt:

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \text{dan} \quad H = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Sobel**: Pengembangan metode Sobel dengan menggunakan filter HPF (*High Pass Filter*) yang diberi satu angka nol penyangga. Kelebihannya adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi. Kernel filter yang digunakan dalam metode Sobel:

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \text{dan} \quad H = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

```
1 kocheng = io.imread("kocheng.jpg")
2 gray_kocheng = color.rgb2gray(kocheng)
3 kocheng_roberts =
     util.img_as_ubyte(filters.roberts(gray_kocheng))
4 kocheng_prewitt =
     util.img_as_ubyte(filters.prewitt(gray_kocheng))
5 kocheng_sobel = util.img_as_ubyte(filters.sobel(gray_kocheng))
```

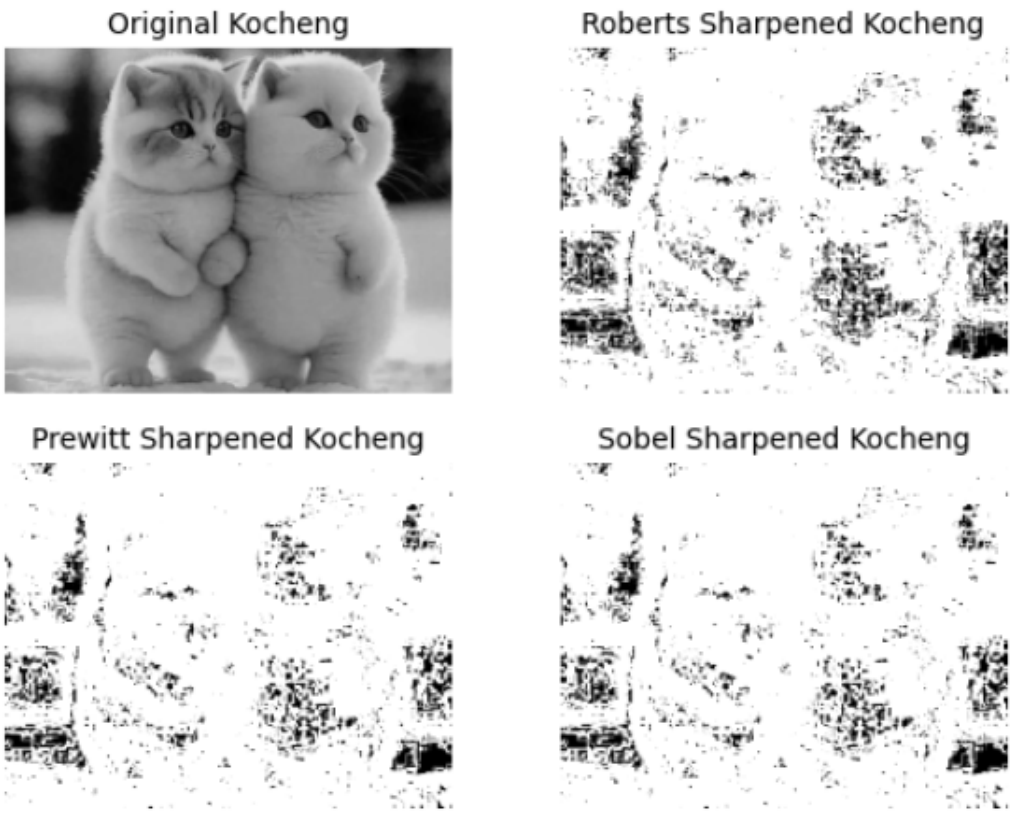

Figure 16: Sharpening