Modul 3 – Operasi Citra Sederhana

A. TUJUAN

1. Mahasiswa dapat memahami dan mengimplementasikan Transformasi Linier Brightness
2. Mahasiswa dapat memahami dan mengimplementasikan Contrast Citra
3. Mahasiswa dapat memahami dan mengimplementasikan Inverse Citra
4. Mahasiswa dapat memahami dan mengimplementasikan Transformasi Logarithmic Brightness
5. Mahasiswa dapat memahami dan mengimplementasikan jenis-jenis operasi Grayscale
6. Mahasiswa dapat membuat aplikasi Gamma Correction
7. Mahasiswa dapat membuat simulasi Citra dengan image depth yang ditentukan
8. Mahasiswa dapat melakukan denoising dengan menggunakan Averaging
9. Mahasiswa dapat melakukan image masking menggunakan logical operator

B. ALAT DAN BAHAN

1. PC/LAPTOP
2. Github
3. *Google Colaborator*

C. ULASAN TEORI

C1. Operasi Citra Sederhana

a) Inverse Citra

Inverse Citra adalah proses yang digunakan untuk membalik nilai citra dengan cermin nilai pixel tengah. Hasil yang didapatkan dikenal dengan nama citra negative. Proses ini juga mudah sekali karena operasinya hanya mengurangkan nilai 255 dengan nilai pixel merah, hijau, dan biru.

$g(x) = 255 - f(x)$, dimana g(x,y) adalah citra negative, dan f(x,y) adalah citra asli

Jadi jika nilai pixel asli 255, dengan operasi ini hasilnya akan bernilai 0 (karena 255 – 255).

b) Transformasi Linear Brightness

Transformasi Linier Brightness merupakan operasi olah citra digital yang paling sederhana. Operasi ini melakukan penambahan nilai pixel jika nilai brightness dinaikkan, dan mengurangkan nilai pixel jika nilai brightness diturunkan.

$g(x,y) = f(x,y) + b$, dimana g(x,y) adalah nilai pixel setelah transformasi, f(x,y) adalah nilai pixel asli, dan *b* adalah nilai brightness.

Karena kesederhanaan operasi ini, maka komputasinya sangat cepat. Walaupun kecepatan komputasi tinggi, namun transformasi ini memiliki kelemahan. Transformasi linier brightness adalah transformasi satu arah, dimana citra setelah di transformasikan menggunakan operasi ini, tidak dapat dikembalikan ke nilai awalnya saat dilakukan reverse linier brightness. Sebagai contoh, jika pixel awal bernilai 240, kemudian ditransformasikan dengan penambahan nilai brightness 20, maka nilai pixel akhirnya adalah 260. Jika program dijalankan, maka akan terjadi error, karena nilai pixel hanya berada pada range 0 – 255. Begitu juga sebaliknya, jika dilakukan pengurangan yang hasilnya lebih kecil dari 0 akan terjadi error juga. Untuk mengatasi hal ini, biasanya dibuat fungsi Truncate yang tugasnya mencegah agar nilai pixel tidak lebih besar dari 255 atau lebih kecil dari 0. Jika nilai setelah operasi lebih besar dari 255 maka nilainya akan dipaksa (ditruncate) menjadi 255, dan jika nilai setelah operasi lebih kecil dari 0, maka nilainya akan dipaksa (ditruncate) menjadi 0. Dengan fungsi Truncate, nilai 240 dioperasikan brightness 20 akan menjadi 255 (bukan 260). Jika nilai 255 ini dioperasikan brightness -20 akan menjadi 235 (bukan 240), hal inilah yang membuat transformasi linier brightness tidak dapat direverse. Jadi fungsi truncate ini mencegah error karena nilai pixel tidak berada diluar range 0-255, tetapi akibatnya pixel yang sudah dibrightness tidak dapat kembali sempurna. Solusi untuk masalah ini dapat diselesaikan menggunakan transformasi Log Brightness yang dibahas juga pada modul ini.

Algoritma untuk truncate dapat dilihat pada tahapan berikut:

Steps	Process
1	Start
2	baca nilai pixel f1
3	Jika f1 < 0, maka nilai f1 diubah menjadi 0
4	Jika f1 > 255, maka nilai f1 diubah menjadi 255
5	Stop

c) Transformasi Contrast

Kontras adalah tingkat penyebaran pixel – pixel ke dalan intensitas warna. Ada tiga macam kontras, yaitu kontras rendah, kontras tinggi, dan kontras normal.

- Citra Kontras Rendah : Citra yang memiliki kontras rendah dapat terjdi karena kurangnya pecahayaan. Citra dengan kontras yang rendah mempunyai kurva histogram yang sempit, akibat penyebaran intensitas terang atau intensitas gelap yang tidak merata. Sehingga titik tergelap dari suatu citra tidak mencapai hitam paling pekat dan titik paling terang tidak mencapai putih paling cemerlang.
- Citra Kontras Tinggi : merupakan kebalikan dari citra kontras rendah karena memiliki kurva histogram yang lebar, sebaran intensitas gelap dan intensitas terang merata ke seluruh skala intensitas.
- Citra Kontras Normal : Terjadi bila lebar kurva histogram tidak terlalu lebar dan tidak terlalu sempit.

Operasi contrast dapat dilakukan dengan memodifikasi formula yang digunakan pada linier brightness, yaitu:

$g(x,y) = a * f(x,y) + b$, dimana g(x,y) adalah nilai pixel setelah transformasi, f(x,y) adalah nilai pixel asli, *a* adalah nilai contrast, dan *b* adalah nilai brightness.

Selain formula di atas, transformasi contrast dapat dilakukan dengan menghitung Contrast Correction Factor menggunakan rumus berikut:

$$F = \frac{259(C + 255)}{255(259 - C)}$$

Agar dapat bekerja, Nilai *F* disimpan dalam variable bertipe double dan bukan integer. Nilai C merepresentasikan level nilai contrast yang diinginkan.

Tahap berikutnya adalah dengan melakukan perhitungan nilai contrast pada tiap pixel R, G, dan B.

$R' = F(R - 128) + 128$, dimana R adalah nilai pixel Red dan R' adalah nilai pixel Red akhir setelah dilakukan operasi Contrast. Karena nilai hasil dari operasi ini dapat berada diluar range 0-255, maka fungsi truncate digunakan. Ingat kembali jika fungsi truncate dipakai, maka operasi ini tidak dapat direverse.

Berikut adalah pseudo-code dari rumus contrast diatas:

```
1  factor = (259 * (contrast + 255)) / (255 * (259 - contrast))
2  colour = GetPixelColour(x, y)
3  redBaru = Truncate(factor * (Red(colour) - 128) + 128)
```

4	`greenBaru = Truncate(factor * (Green(colour) - 128) + 128)`
5	`blueBaru  = Truncate(factor * (Blue(colour)  - 128) + 128)`
6	`SetPixelColour(x, y) = RGB(redBaru, greenBaru, blueBaru)`

d) Transformasi Logarithmic Brightness

Transformasi Log memetakan suatu citra dengan *range* warna sempit menjadi lebih lebar pada citra *output*. Transformasi Logaritma berguna dalam penggambaran grafik apabila pada deretan nilai terdapat selisih nilai yang sangat kecil maupun selisih nilai yang sangat besar, sehingga selisih nilai yang sangat kecil tersebut akan sulit dilihat. *Range input* yang sempit dari nilai *gray level* rendah dipetakan ke dalam *range* yang lebar pada *output gray level*.

Secara umum bentuk dari transfomasi log adalah:

$$s = c * log(1 + r)$$

dimana

c : konstanta

r : nilai grey-level citra input

s : nilai grey-level citra output

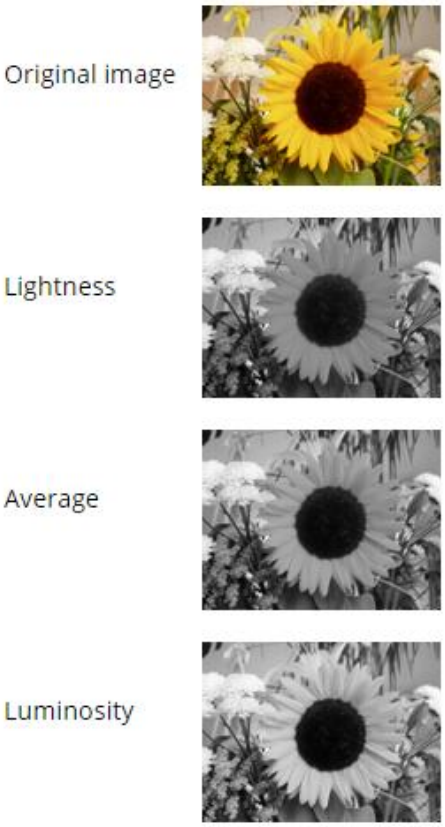
e) Grayscale

Grayscale adalah citra keabuan dengan nilai intensitas kedalaman pixel 8 bit. *Grayscale* biasanya dipakai untuk merepresentasikan intensitas cahaya pada jalur tunggal dari spectrum electromagnet (inframerah, cahaya visible, ultraviolet, dsb). Beberapa metode *grayscaling* yang sering dipakai adalah *Averaging*, *Lightness*, dan *Luminance*. *Averaging* ditentukan dengan merata-rata nilai dari tiap channel R, G, dan B. *Lightness* ditentukan dengan merata-rata nilai maksimum dan nilai minimum seluruh channel R, G, dan B. Sedangkan *Luminance* (Luminosity) ditentukan dengan memberikan pembobotan yang disesuaikan dengan spectrum visible yang paling banyak ada pada citra alami.

$$Grayscale_{avg} = \frac{(R + G + B)}{3}$$

$$Grayscale_{Lightness} = \frac{\max[R, G, B] + \min[R, G, B]}{2}$$

$$Grayscale_{Luminance} = 0.21R + 0.72G + 0.07B$$



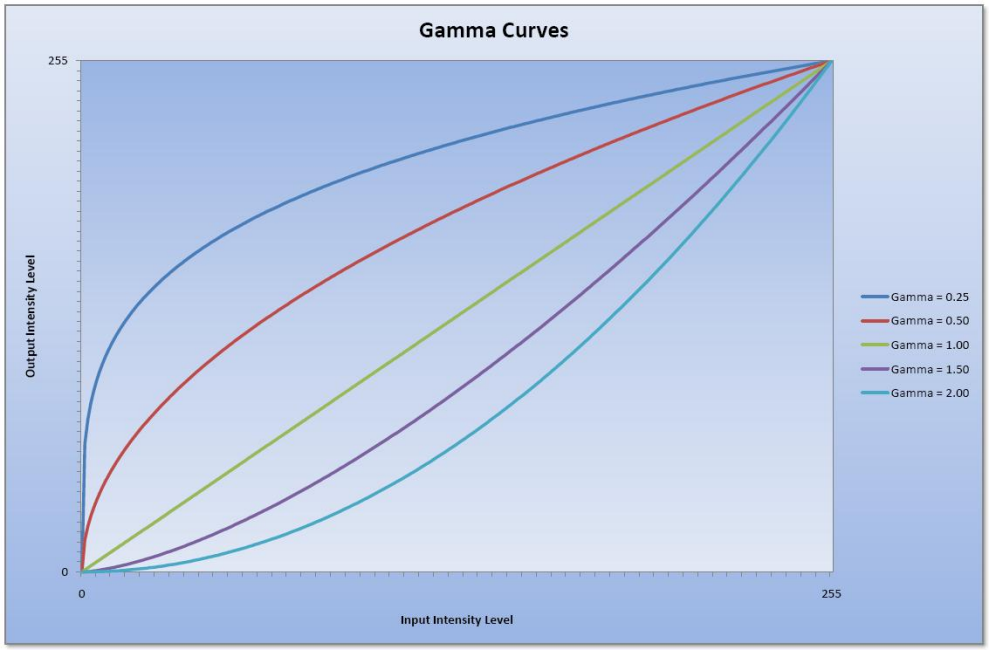
C2. Operasi Aritmatika dan Logika

a) Gamma Correction

Gamma (Power-Law Transformation), di simbolkan dengan huruf yunani γ, dijelaskan sebagai hubungan antara masukan dan keluaran yang dihasilkan. Masukan yang dimaksud adalah nilai intensitas RGB dari citra. Hubungan antara masukan dan keluaran yang dimaksud adalah keluarannya proporsional dengan masukan yang dipangkatkan dengan nilai Gamma. Rumus dari Gamma Correction adalah sebagai berikut:

$$I' = 255\ x\ \left(\frac{I}{255}\right)^{\gamma}$$

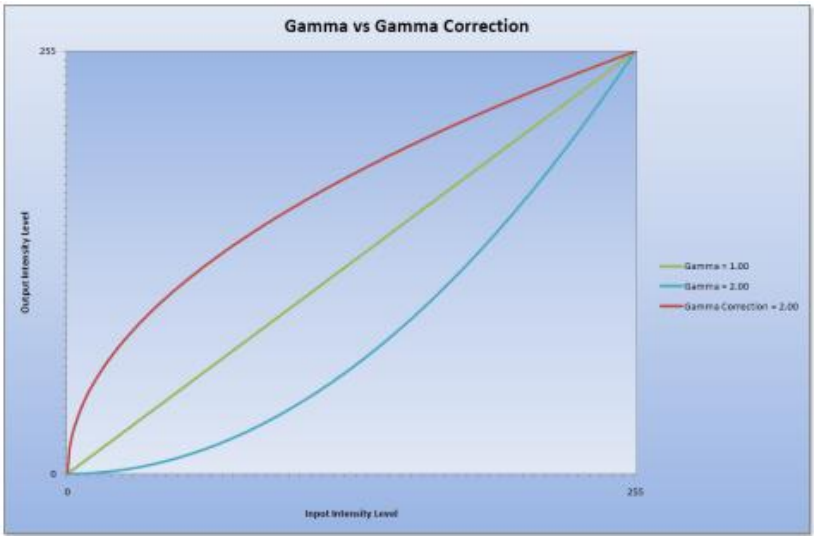
Berikut adalah ilustrasi hasil keluaran kurva Gamma pada beberapa nilai Gamma.



Perhatikan bahwa dengan Gamma bernilai 1, maka nilai masukan sama dengan nilai keluaran dan ditunjukkan dengan garis lurus. Untuk menghitung Gamma Correction, nilai masukan dipangkatkan dengan inverse nilai Gamma. Rumus untuk Gamma Correction adalah sebagai berikut:

$$I' = 255\ x\ \left(\frac{I}{255}\right)^{\frac{1}{\gamma}}$$

Berikut adalah grafik perbandingan antara kurva gamma dengan kurva gamma correction:



Nilai yang digunakan sebagai Gamma tergantung dari aplikasi yang digunakan. Berikut adalah beberapa contoh gambar RGB sebelum dan setelah diproses menggunakan Gamma Correction.

Gamma Correction dengan nilai Gamma 0.5

Gammar Correction dengan nilai Gamma 6

b) Bit Depth

Operasi ini digunakan untuk menunjukkan pada anda tentang kuantisasi citra. Kuantisasi yang dilakukan adalah pada nilai kedalaman warna 1 – 7bit. Transformasi ini sifatnya adalah simulasi saja, bit tidak benar benar berjumlah 1-7bit, hanya variasi warnanya saja sejumlah 1-7bit. Tentukan level kedalaman terlebih dahulu, kemudian nilai warna baru adalah hasil dari warna lama yang dioperasikan dengan level yang telah ditentukan. Berikut adalah rumus yang digunakan:

$$level = \frac{255}{2^{bit\_depth} - 1}$$

bit_dept adalah kedalaman warna yang diinginkan. Kedalaman sebelumnya 8 bit atau rentang gradasi dari nilai 0 sampai 255 (256 macam). Jika diinginkan rentang gradasi dari 0 sampai 15 (16 macam) maka pilih bit_dept = 4.

$$C' = round(\left(\frac{C}{level}\right) * level)$$

1bit depth rentang gradasi dari 0 sampai 1 (2 macam)

2bit depth rentang gradasi dari 0 sampai 3 (4 macam)

3bit depth rentang gradasi dari 0 sampai 7 (8 macam)

4bit depth rentang gradasi dari 0 sampai 15 (16 macam)

5bit depth rentang gradasi dari 0 sampai 31 (32 macam)

6bit depth rentang gradasi dari 0 sampai 63 (64 macam)

7bit depth rentang gradasi dari 0 sampai 127 (128 macam)

c) PSNR

PSNR (Peak Signal-to-Noise Ratio) adalah ratio antara nilai power maksimum dari citra dengan power dari citra yang terkena noise yang mempengaruhi kualitas dari citra ternoise. Untuk menghitung PSNR dari citra, yang perlu dilakukan adalah membandingkan image hasil denoising / terkompress dengan citra ideal yang memiliki power maksimum (dalam hal ini adalah citra asli).

PSNR diformulasikan dengan rumus sebagai berikut:

$$PSNR = 10log_{10}\left(\frac{(L-1)^2}{MSE}\right) = 20log_{10}\left(\frac{L-1}{RMSE}\right)$$

Dimana L adalah jumlah intensitas warna yang mungkin dari citra.

MSE adalah mean squared error dan dirumuskan sebagai berikut:

$$MSE = \frac{1}{MN}\sum_{1=0}^{m-1}\sum_{j=0}^{n-1}(Original(i,j) - Denoised(i,j))^2$$

RMSE (Root Mean-Square-Error) adalah akar dari MSE.

Berikut adalah fungsi dari PSNR pada Python:

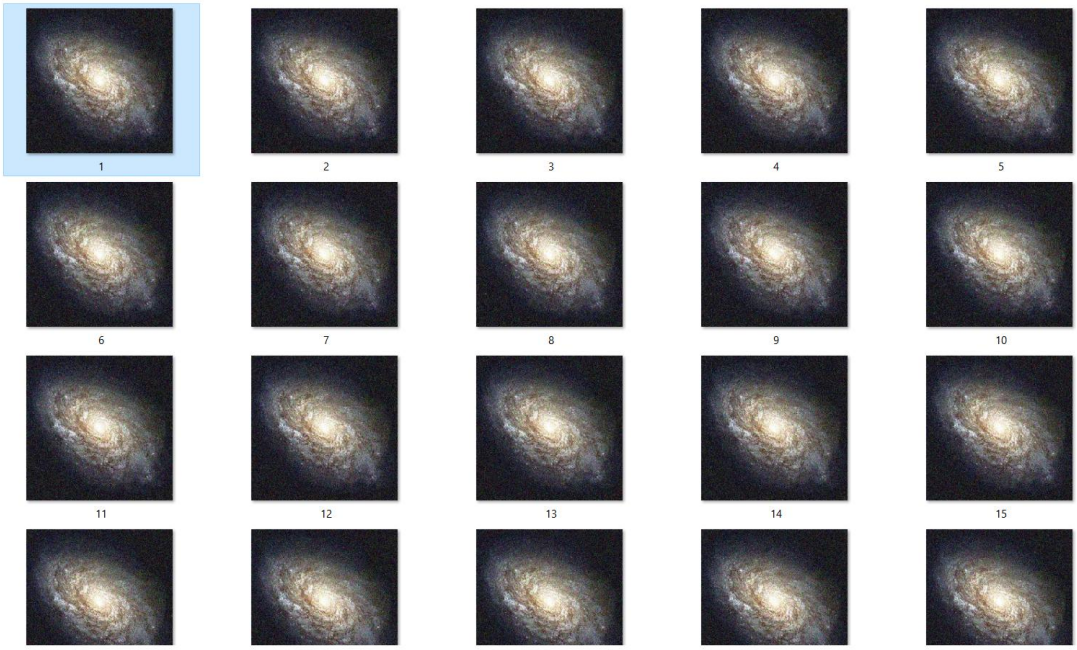
```
def PSNR(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
     if(mse == 0):  # MSE 0 maka tidak ada noise sama sekali, sehingga
      PSNR tidak memiliki arti
      return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr
```

d) Average Denoising

Average Denoising merupakan operasi aritmatika yang dikenakan pada citra untuk berbagai keperluan yang bermanfaat. Pada modul kali ini akan ditunjukkan operasi Averaging untuk denoising / mengurangi noise pada image. Operasi ini melakukan perhitungan nilai rata-rata tiap pixel yang berkoordinat sama dalam sebuah kumpulan citra. Operasi ini biasanya dilakukan untuk menghilangkan noise dari factor eksternal yang terjadi saat akuisisi citra. Average denoising biasa digunakan pada citra yang memiliki noise eksternal yang muncul akibat proses distribusi dari sumber ke tujuan.

Average image bekerja dengan menghitung nilai rata2 tiap pixel sesuai jumlah citra yang dirata-rata. Semakin banyak jumlah citra yang dirata2, maka akan semakin baik hasil denoised nya, semakin tinggi nilai PSNRnya. Perhatikan bahwa makin tinggi nilai PSNR, makin baik kualitas citra hasil denoise-nya.

Gambar berikut sengaja diberikan Gaussian noise secara acak dibuat sebanyak 100 kali:



Kumpulan Gambar yang diberikan Gaussian Noise secara acak

Salah satu gambar yang diperbesar dan telah diberikan Gaussian Noise

Berikut adalah hasil image yang telah didenoising menggunakan averaging pada 5 citra, 20 citra, 50 citra, dan 100 citra.

Average 5 citra

Average 20 citra

Average 50 citra

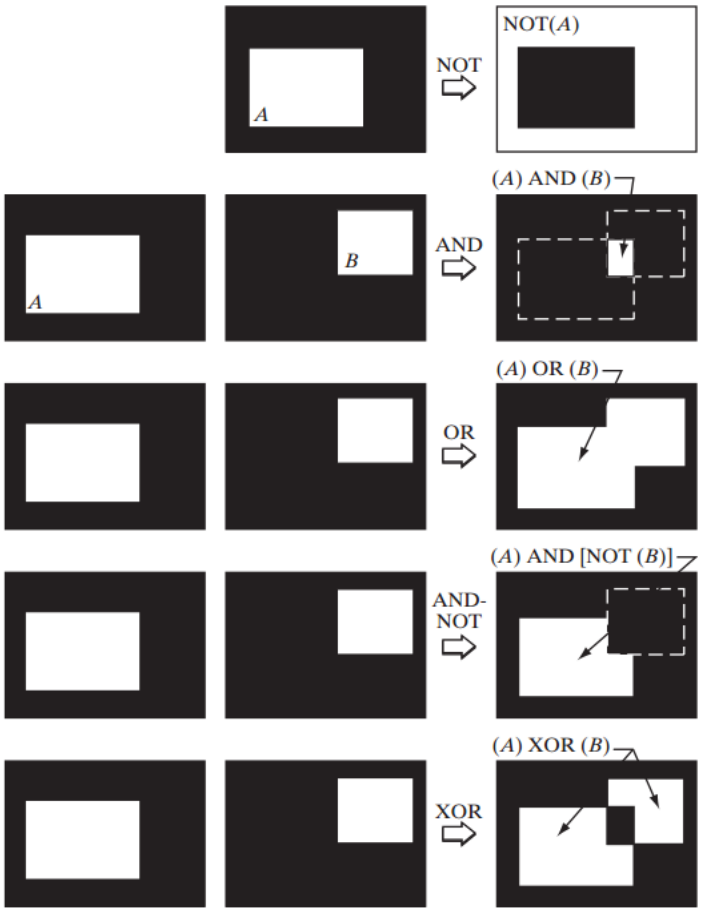
Average 100 citra

e) Image Masking

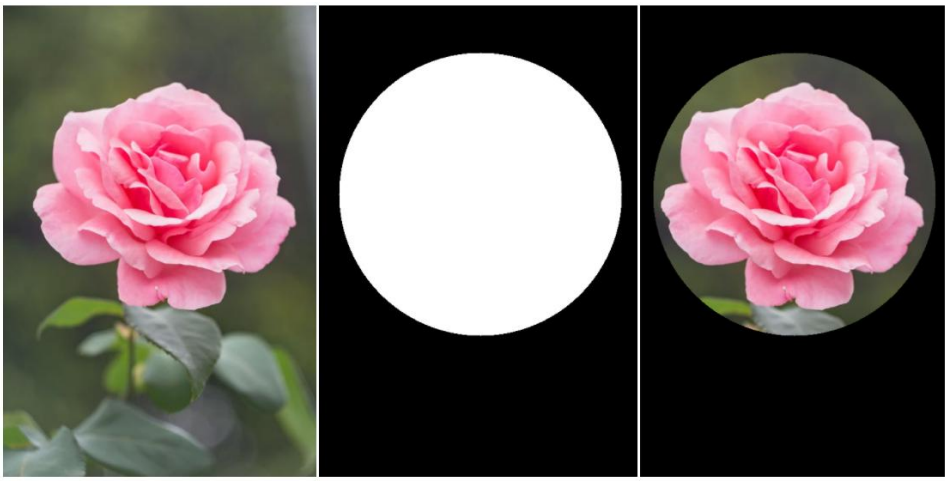
Image masking adalah contoh dari operasi Logika yang digunakan untuk mengolah citra. Buka kembali buku Gonzales section 2.6.4 untuk pembahasan lebih detail tentang operasi logika.

Beberapa operator logika yang sering digunakan pada pengolahan citra adalah operator OR, AND, dan NOT. Pada Image Masking, operator yang biasa digunakan adalah operator AND.

Jika diasumsikan dua region (sets) A dan B terdiri dari pixel foreground, maka operasi OR dari dua set ini adalah dapat berisi nilai dari A atau B. Gambar berikut mengilustrasikan operasi-operasi Logika pada citra.



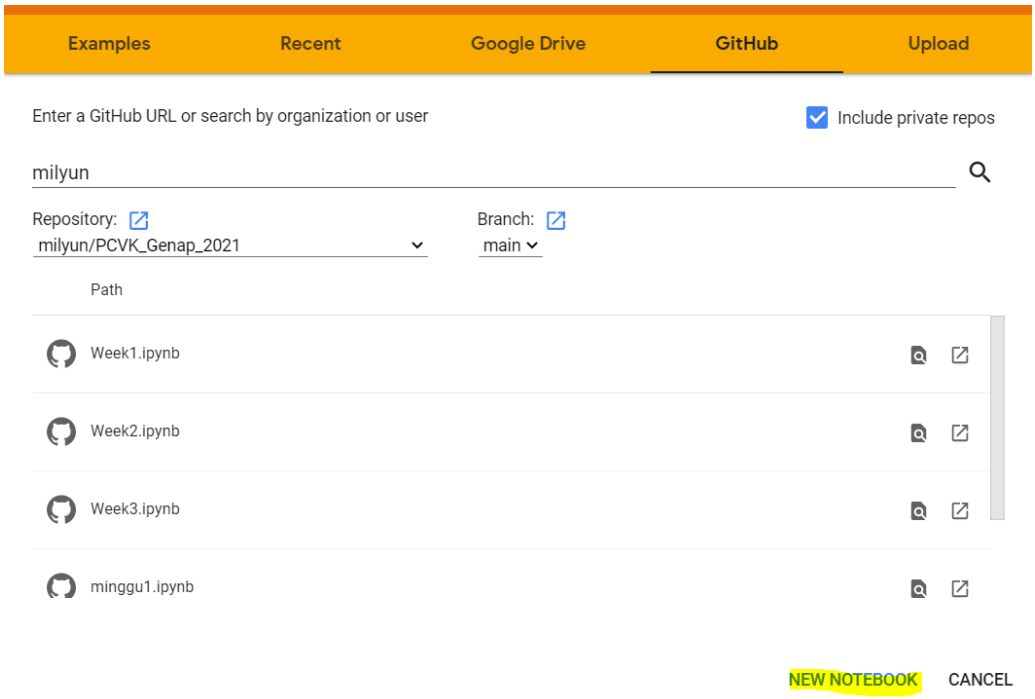
Gambar berikut menunjukkan hasil citra menggunakan image masking operator AND.



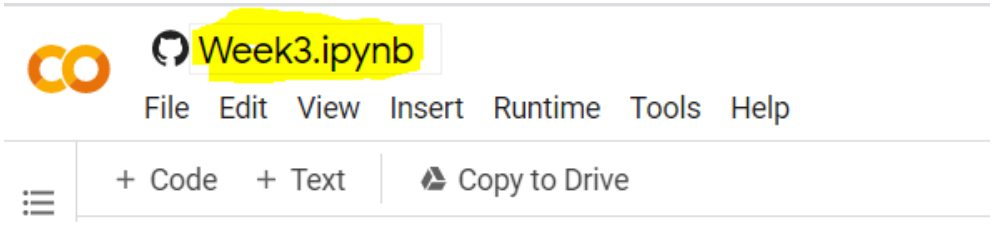
D. PRAKTIKUM

D1. Operasi Citra Sederhana

1. Buka https://colab.research.google.com/, pilih tab Github dan pastikan repository yang terpilih ada repository yang sama dengan praktikum pada minggu pertama dan kedua.



Lanjutkan dengan membuat notebook baru dan ubah nama file menjadi "Week3.ipynb". **Perhatian**: Jangan lupa untuk menyimpan salinan ke Github setelah melakukan perubahan / ketika Anda sudah selesai melakukan praktikum.



2. Akses folder images pada Google Drive Anda dengan kode berikut:

```
[ ]  from google.colab import drive

     drive.mount('/content/drive')

     Mounted at /content/drive
```

Ikuti alur autorisasinya hingga muncul pesan: "*Mounted at /content/drive*".

3. Melakukan transformasi linier brightness dengan memasukkan nilai konstanta tertentu dan menghasilkan warna. Seperti yang telah dibahas pada ulasan teori, formula untuk melakukan transformasi linier brightness adalah sebagai berikut:

$$g(x,y) = f(x,y) + b$$

dimana g(x,y) adalah nilai pixel setelah transformasi, f(x,y) adalah nilai pixel asli, dan *b* adalah nilai brightness.

Tuliskan potongan kode berikut untuk mengimplementasikan linier brightness pada google colab:

Transformasi Linier Brightness

Formula: *g(x,y)=f(x,y)+b*

g(x,y) adalah nilai pixel setelah transformasi, *f(x,y)* adalah nilai pixel asli, *b* adalah nilai brightness

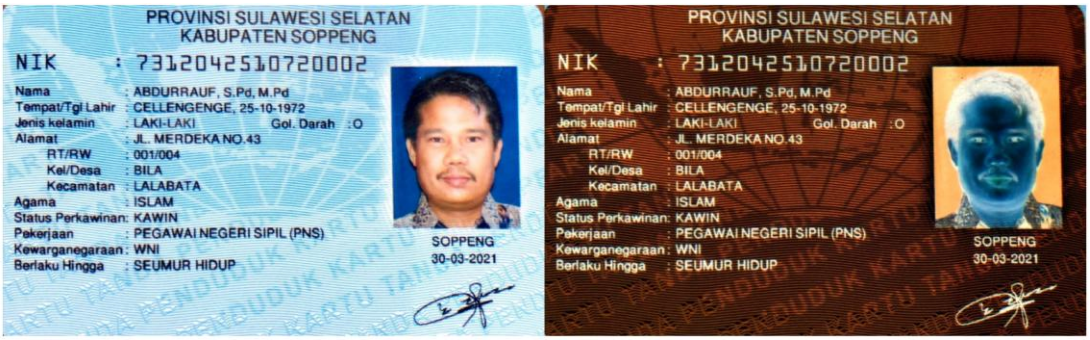
```
print(' Mengubah tingkat kecerahan citra ')
print('-----------------------------------')
try:
    brightness = int(input('Masukkan nilai kecerahan: '))
except ValueError:
    print('Error, not a number')

original = cv.imread('/content/drive/MyDrive/images/female.tiff')
brightness_image = np.zeros(original.shape, original.dtype)

#akses per piksel
for y in range(original.shape[0]):
    for x in range(original.shape[1]):
        for c in range(original.shape[2]):
            brightness_image[y,x,c] = np.clip(original[y,x,c] + brightness, 0, 255)

#cara simple tanpa for loop
#brightness_image = cv.convertScaleAbs(original, beta=brightness)

final_frame = cv.hconcat((original, brightness_image))
cv2_imshow(final_frame)
```

Dari potongan kode di atas dapat dilihat bahwa ketika dijalankan, system akan menampilkan *text field* untuk memasukkan konstanta nilai *brightness* yang diinginkan, dan akan disimpan pada variabel *brightness*. Kemudian ditentukan citra pada drive yang akan diolah dan ditampung di variabel original. Tahap selanjutnya adalah mengakses pixel citra masukan dengan 3 perulangan. Perulangan pertama dilakukan pada shape[0] untuk tinggi citra, perulangan kedua dilakukan pada shape[1] untuk lebar citra, dan perulangan ketiga dilakukan pada shape[2] yang merupakan channel warna pada citra. Setelah melakukan 3 perulangan, dilakukan transformasi linier brightness dengan cara menambahkan nilai brightness pada citra masukan. Contoh hasil dari kode program di atas adalah sebagai berikut.

```
Mengubah tingkat kecerahan citra
--------------------------------
Masukkan nilai kecerahan: 50
```

TUGAS PRAKTIKUM D1

1. Implementasikan inverse citra pada Google Colaboratory menggunakan formula yang terdapat pada bagian Ulasan Teori, sehingga menghasilkan keluaran seperti berikut:



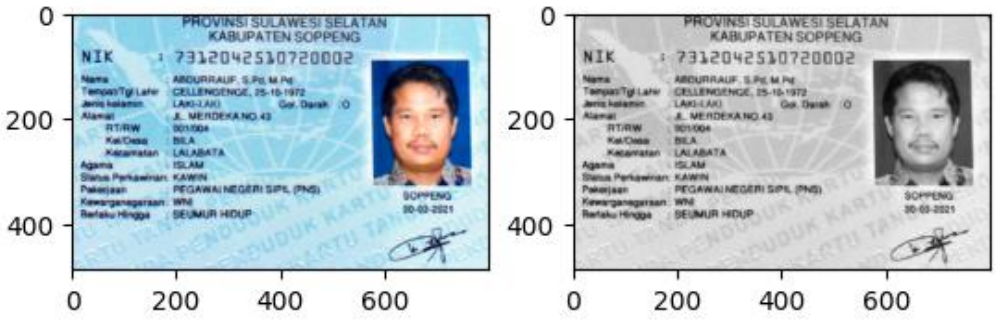
2. Implementasikan transformasi contrast pada Google Colaboratory menggunakan formula yang terdapat pada bagian Ulasan Teori untuk kontras, sehingga menghasilkan keluaran seperti berikut:

```
Mengubah kontras dan tingkat kecerahan citra
--------------------------------------------
Masukkan tingkat kecerahan [-255 - 255]: 50
Masukkan kontras [1.0 - 3.0]: 2
```

3. Implementasikan transformasi logarithmic brightness pada Google Colaboratory menggunakan formula yang terdapat pada bagian Ulasan Teori untuk transformasi log, sehingga menghasilkan keluaran seperti berikut:

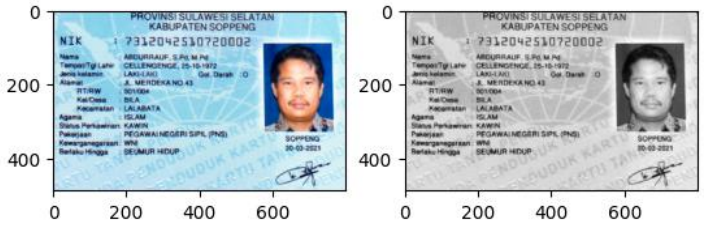
```
Mengubah tingkat kecerahan citra dengan Transformasi Log
----------------------------------------------------------
Masukkan nilai kecerahan: 50
```

4. Implementasikan transformasi grayscale menggunakan metode averaging, lightness, dan luminance pada Google Colaboratory menggunakan formula yang terdapat pada bagian Ulasan Teori, sehingga menghasilkan keluaran seperti berikut:
   a. Averaging



   b. Lightness

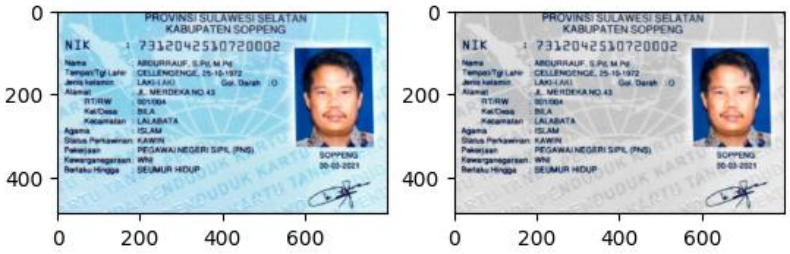
```
<matplotlib.image.AxesImage at 0x781d2577f7f0>
```



   c. Luminance

```
<matplotlib.image.AxesImage at 0x781d70a42fe0>
```

5. Tampilkanlah warna tertentu pada citra, dan ubah warna lain menjadi grayscale. Misal, tampilkan warna biru pada citra masukan dan ubah bagian lain yang tidak berwarna biru menjadi grayscale seperti pada contoh berikut:



D2. Operasi Aritmatika dan Logika

1. Buat Gamma Correction sesuai dengan petunjuk berikut

Percobaan ini akan meminta anda membuat Gamma Correction. Pada percobaan ini, nilai Gamma akan diset dengan meminta masukan dari pengguna. Berikut adalah kode untuk meminta masukan nilai dari pengguna. Lanjutkan kode tersebut dengan membuat image dengan gamma correction sesuai rumus yang telah diberikan.

```
print(' Gamma Correction pada citra ')
print('-----------------------------------')
try:
    gamma = int(input('Masukkan nilai Gamma: '))
except ValueError:
    print('Error, not a number')
```

```
 Gamma Correction pada citra
-----------------------------------
Masukkan nilai Gamma: 3
```

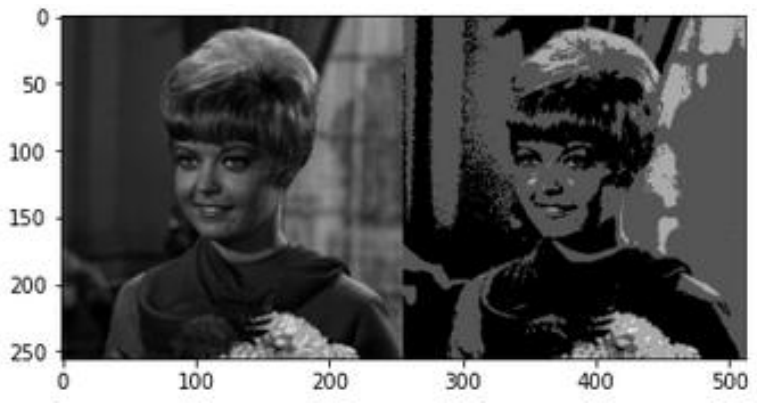
Gambar diatas adalah citra yang diproses menggunakan Gamma Correction nilai gamma 3.

2. Buat Simulasi Image Depth

Percobaan ini digunakan sebagai simulasi dari proses kuantisasi citra. Pada kuantisasi citra, pixel dapat direpresentasikan dengan n-bit kedalaman (default menggunakan 8-bit). Pada pixel 8-bit, warna yang memungkinkan adalah 256 warna, dari 0 (0000 0000) hingga 255(1111 1111). Pada pixel 7-bit, warna yang memungkinkan adalah 128 warna, dari 0 (000 0000) hingga 127 (111 1111). Kemungkinan warna didapat dari pangkat 2 jumlah bit. Jika 7bit, maka jumlah warnanya adalah 2^7 = 128, dst.

Berikut adalah kode untuk membaca citra masukan dan memberi nilai kedalaman citra, silahkan lanjutkan kode program berikut sehingga menghasilkan keluaran seperti contoh pada Gambar.

```
bit_depth=2
level = 255 / (pow(2,bit_depth)-1)
original = cv.imread('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/female.tiff', cv.IMREAD_GRAYSCALE)
```



3. Buat modul Average Denoising

Buat modul average denoising sesuai dengan rumus yang telah diberikan pada sub bab sebelumnya.

Citra asli sudah disediakan pada **/images/galaxy.jpg**.

100 Citra dengan Gaussian Noise sudah disediakan pada **/images/noises/*.jpg**

Anda dapat menggunakan code berikut untuk membaca semua image dalam satu folder , gunakan modul glob (**import glob**).

```
cv_img = []
for img in glob.glob('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/noises/*.jpg'):
    n= cv.imread(img)
    cv_img.append(n)
```

Dengan menggunakan code diatas, anda tinggal memanggil image difolder tersebut menggunakan cv_img[0], cv_img[1], dst.

Catat hasil PSNR pada tabel berikut. Dari hasil yang sudah anda catat, tuliskan kesimpulan anda:

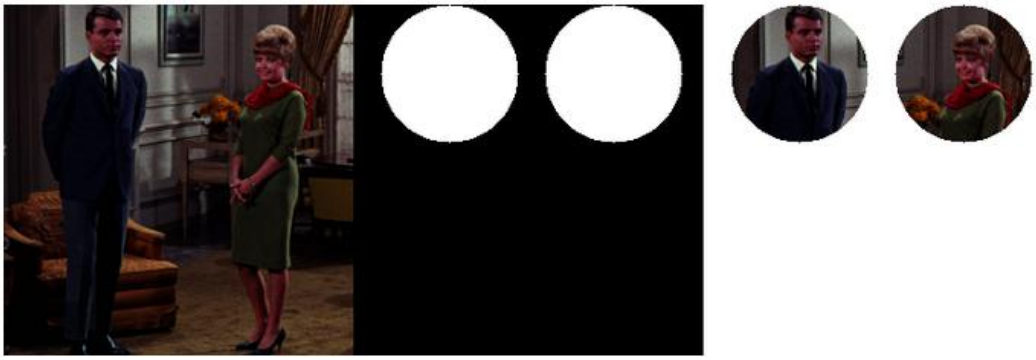
No	Jumlah Citra di Average	Image Hasil	Nilai PSNR (dB)
1.	10		
2.	20		
3.	40		
4.	80		
5.	100		

Dari hasil PSNR yang anda catat pada tabel diatas, kesimpulan yang dapat diambil adalah.

……
……
……
……

4. Buat image masking

Buat image masking untuk image berikut. Image kiri adalah image asli (images/couple.tiff), sedangkan image paling kanan adalah hasilnya:



Lakukan percobaan menggunakan operator lain dan tunjukkan hasilnya pada modul ini. Tuliskan hasil analisa anda kenapa citra keluarannya seperti itu.

No.	Operator	Image Input	Image Output
1.	**NOT (komplemen)**		
2.	**OR (Atau)**		
3.	**AND (Dan)**		
4	**NAND (Not And)**		

5.	**XOR (Exlusive Or)**		

Tuliskan hasil analisa anda:

……
……
……

……