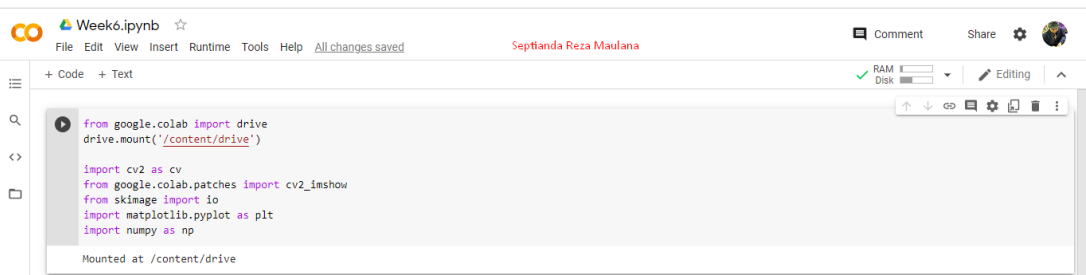
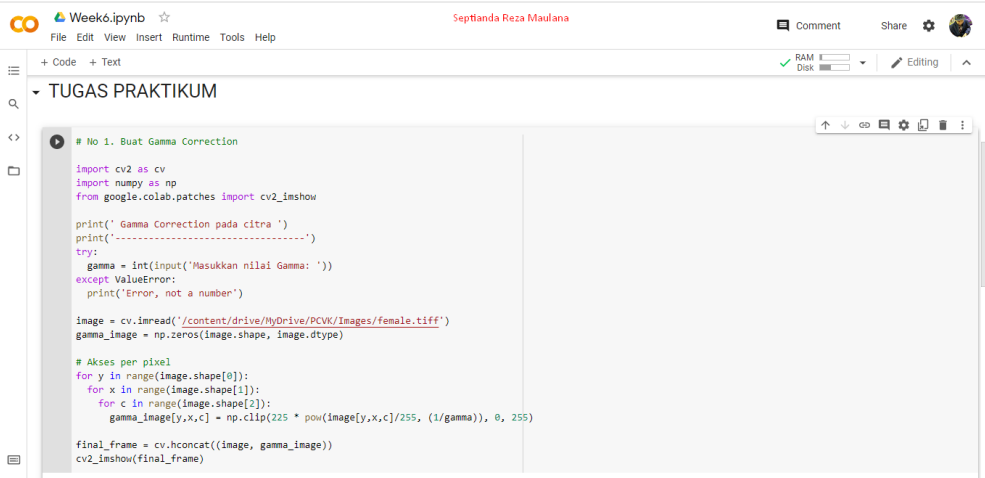


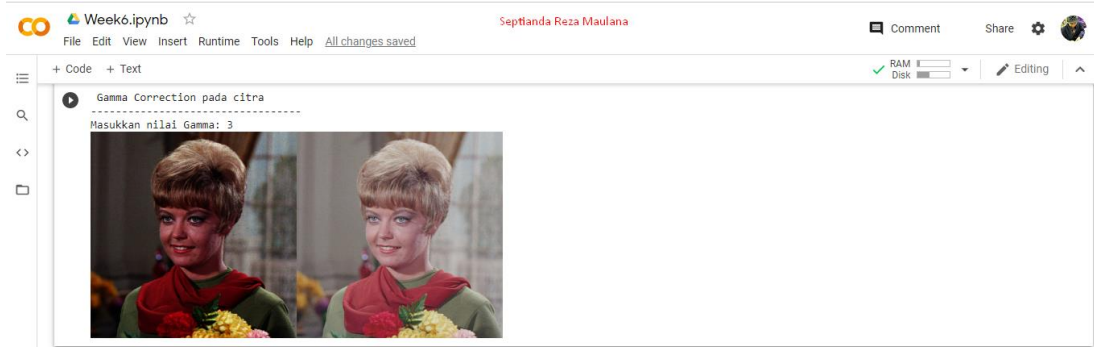


Tujuan

1. Mahasiswa dapat membuat aplikasi Gamma Correction.
2. Mahasiswa dapat membuat simulasi Citra dengan image depth yang ditentukan.
3. Mahasiswa dapat melakukan denoising dengan menggunakan Averaging.
4. Mahasiswa dapat melakukan image masking menggunakan logical operator.

Praktikum 1 : Buat Gamma Correction sesuai dengan petunjuk berikut

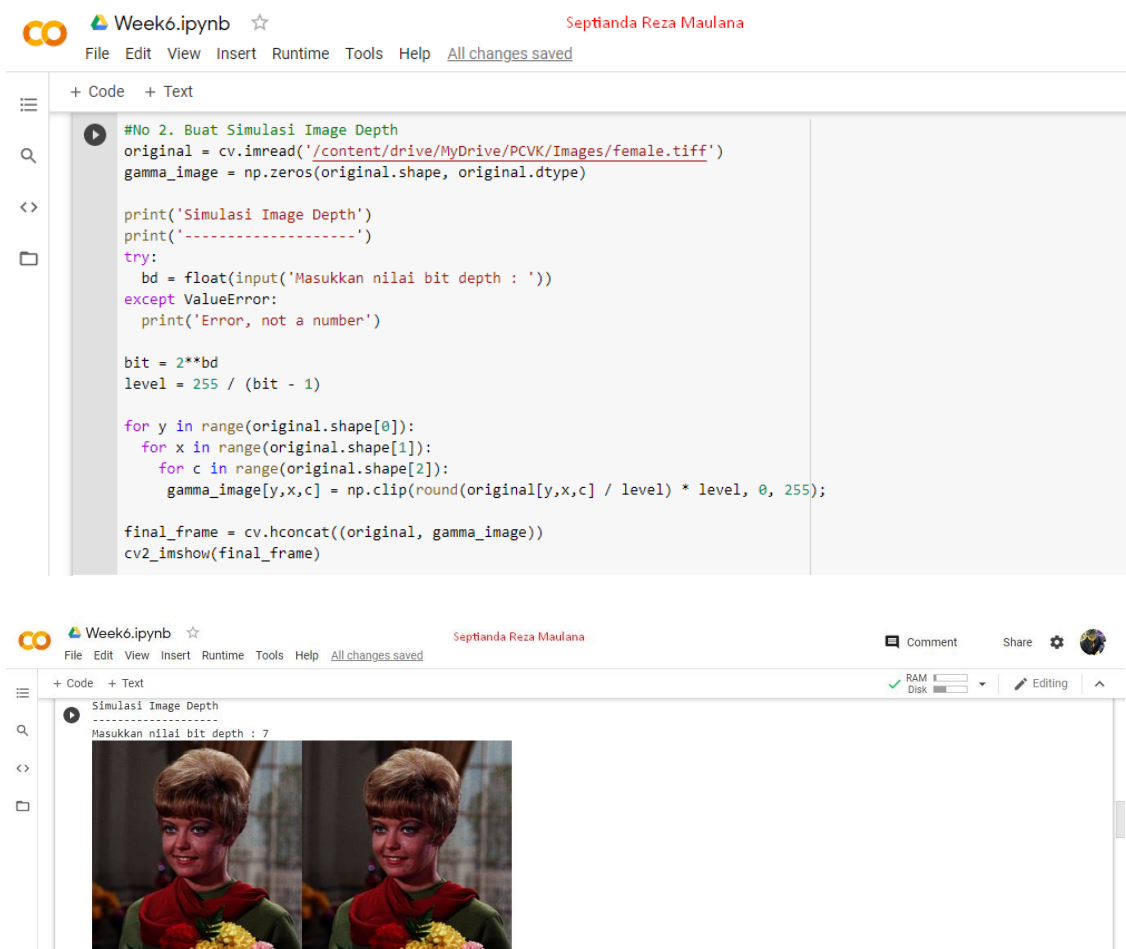
| Langkah | Keterangan |
|---------|--|
| 1 | <p>Percobaan ini akan meminta anda membuat Gamma Correction. Pada percobaan ini, nilai Gamma akan diset dengan meminta masukan dari pengguna. Berikut adalah kode untuk meminta masukan nilai dari pengguna. Lanjutkan kode tersebut dengan membuat image dengan gamma correction sesuai rumus yang telah diberikan.</p> <p>NB : Sebelum melakukan praktikum, perlu import folder yang ada di Drive Anda dengan cara sebagai berikut :</p>  <pre>from google.colab import drive drive.mount('/content/drive') import cv2 as cv from google.colab.patches import cv2_imshow from skimage import io import matplotlib.pyplot as plt import numpy as np</pre> <p>Mounted at /content/drive</p>  <pre># No 1. Buat Gamma Correction import cv2 as cv import numpy as np from google.colab.patches import cv2_imshow print('Gamma Correction pada citra ') print('-----') try: gamma = int(input('Masukkan nilai Gamma: ')) except ValueError: print('Error, not a number') image = cv.imread('/content/drive/MyDrive/PCVK/Images/female.tiff') gamma_image = np.zeros(image.shape, image.dtype) # Akses per pixel for y in range(image.shape[0]): for x in range(image.shape[1]): for c in range(image.shape[2]): gamma_image[y,x,c] = np.clip(255 * pow(image[y,x,c]/255, (1/gamma)), 0, 255) final_frame = cv.hconcat([image, gamma_image]) cv2_imshow(final_frame)</pre> |



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. Karena Visual Studio 2017 bekerja hanya pada 8 bit, maka percobaan ini hanya memanipulasi warna sehingga jumlah warnanya sesuai dengan kedalamannya. Untuk kasus 7-bit, maka dua warna 8-bit diwakili oleh satu warna 7-bit. Contoh pixel warna 0 dan 1 pada 8-bit, diwakili oleh warna 0 pada 7-bit. pixel warna 2 dan 3 pada 8-bit, diwakili oleh warna 1 pada 7-bit, dst

2



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).

3

```
Week6.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

# No 3. Buat modul Average Denoising

import glob
from matplotlib import pyplot as plt
import cv2 as cv
from math import log10, sqrt

def PSNR(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
    if(mse == 0):
        return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr

img_asli = cv.imread('/content/drive/MyDrive/PCVK/Images/galaxy.jpg')
cv_img = []
for img in glob.glob('/content/drive/MyDrive/PCVK/Noises/*.jpg'):
    n = cv.imread(img)
    cv_img.append(n)

denoised_image = np.zeros(cv_img[0].shape)
jumlahGambar = 100
for i in range(jumlahGambar):
    denoised_image = denoised_image + cv_img[i]
denoised_image = np.uint16(denoised_image/jumlahGambar)

psnr = PSNR(img_asli,denoised_image)

print('Nilai PSNR-nya adalah: ', psnr)
cv2_imshow(denoised_image)
```

- Hasil Citra di Average (5) dengan Nilai PSNR adalah **19.532551383853427**.

```
Week6.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

jumlahGambar = 5
for i in range(jumlahGambar):
    denoised_image = denoised_image + cv_img[i]
denoised_image = np.uint16(denoised_image/jumlahGambar)

psnr = PSNR(img_asli,denoised_image)

print('Nilai PSNR-nya adalah: ', psnr)
cv2_imshow(denoised_image)

Nilai PSNR-nya adalah: 19.532551383853427
```



- Hasil Citra di Average **(30)** dengan Nilai PSNR adalah **19.874270522646857**.

Week6.ipynb Septianda Reza Maulana

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```


jumlahGambar = 30
for i in range(jumlahGambar):
    denoised_image = denoised_image + cv_img[i]
    denoised_image = np.uint16(denoised_image/jumlahGambar)

psnr = PSNR(img_asli,denoised_image)

print('Nilai PSNR-nya adalah: ', psnr)
cv2_imshow(denoised_image)

```

Nilai PSNR-nya adalah: 19.874270522646857



- Hasil Citra di Average **(60)** dengan Nilai PSNR adalah **19.909911970473125**.

Week6.ipynb Septianda Reza Maulana

File Edit View Insert Runtime Tools Help

+ Code + Text

```


jumlahGambar = 60
for i in range(jumlahGambar):
    denoised_image = denoised_image + cv_img[i]
    denoised_image = np.uint16(denoised_image/jumlahGambar)

psnr = PSNR(img_asli,denoised_image)

print('Nilai PSNR-nya adalah: ', psnr)
cv2_imshow(denoised_image)

```

Nilai PSNR-nya adalah: 19.909911970473125



- Hasil Citra di Average **(80)** dengan Nilai PSNR adalah **19.91956608555111**.

Week6.ipynb Septianda Reza Maulana

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```


jumlahGambar = 80
for i in range(jumlahGambar):
    denoised_image = denoised_image + cv_img[i]
    denoised_image = np.uint16(denoised_image/jumlahGambar)

psnr = PSNR(img_asli,denoised_image)

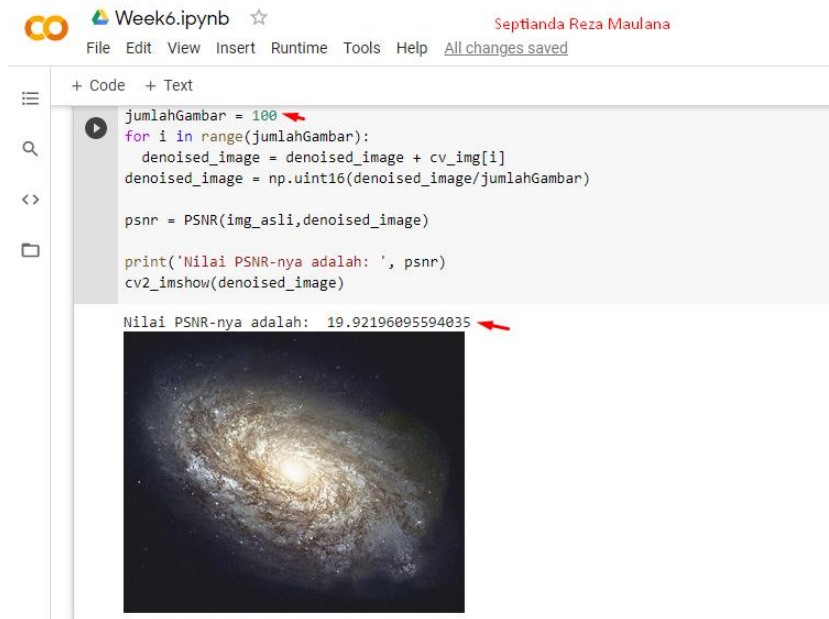
print('Nilai PSNR-nya adalah: ', psnr)
cv2_imshow(denoised_image)

```

Nilai PSNR-nya adalah: 19.91956608555111



- Hasil Citra di Average (**100**) dengan Nilai PSNR adalah **19.92196095594035**.



```

jumlahGambar = 100
for i in range(jumlahGambar):
    denoised_image = denoised_image + cv_img[i]
    denoised_image = np.uint16(denoised_image/jumlahGambar)

psnr = PSNR(img_asli,denoised_image)

print('Nilai PSNR-nya adalah: ', psnr)
cv2_imshow(denoised_image)

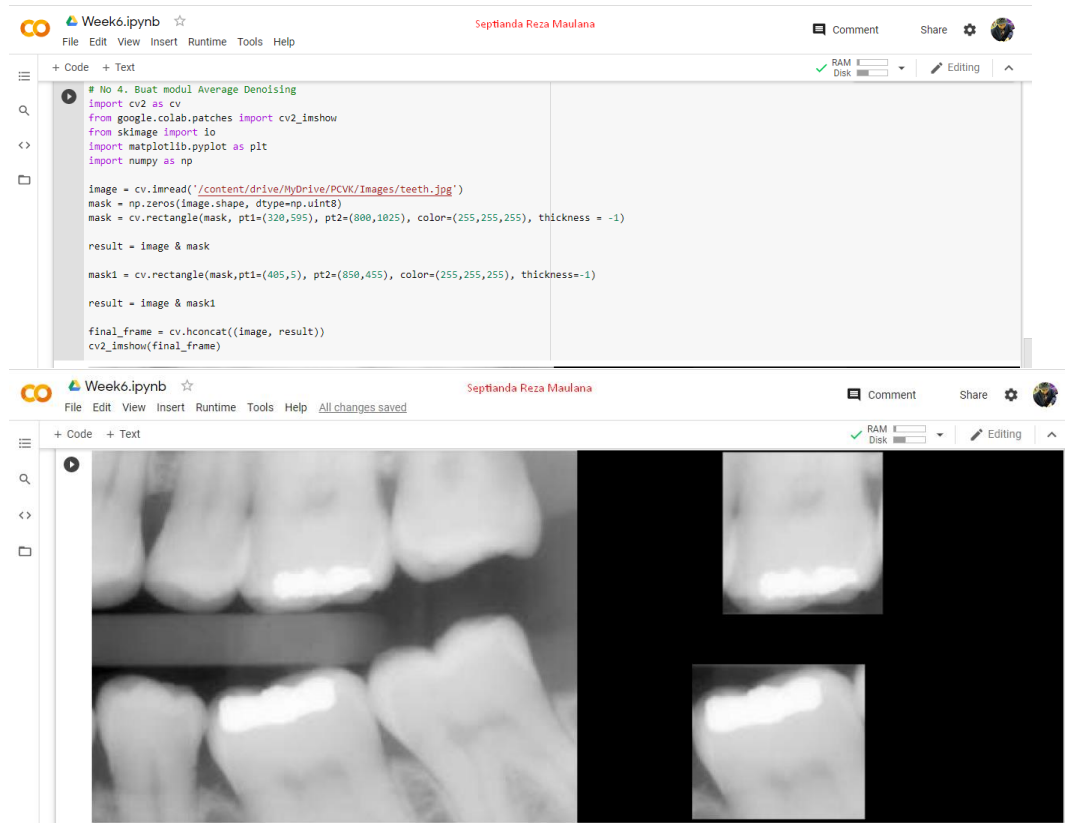
Nilai PSNR-nya adalah: 19.92196095594035

```

Dari hasil PSNR yang anda catat pada tabel diatas, kesimpulan yang dapat diambil adalah Pada saat kita mengganti jumlahGambar maka PSNR akan bertambah besar.

Buat image masking untuk image berikut. Image kiri adalah image asli (images/teeth.jpg), sedangkan image kanan adalah hasilnya:

4



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

- Hasil Menggunakan Operator **OR**

```
Week6.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

No 5. Lakukan percobaan menggunakan operator lain

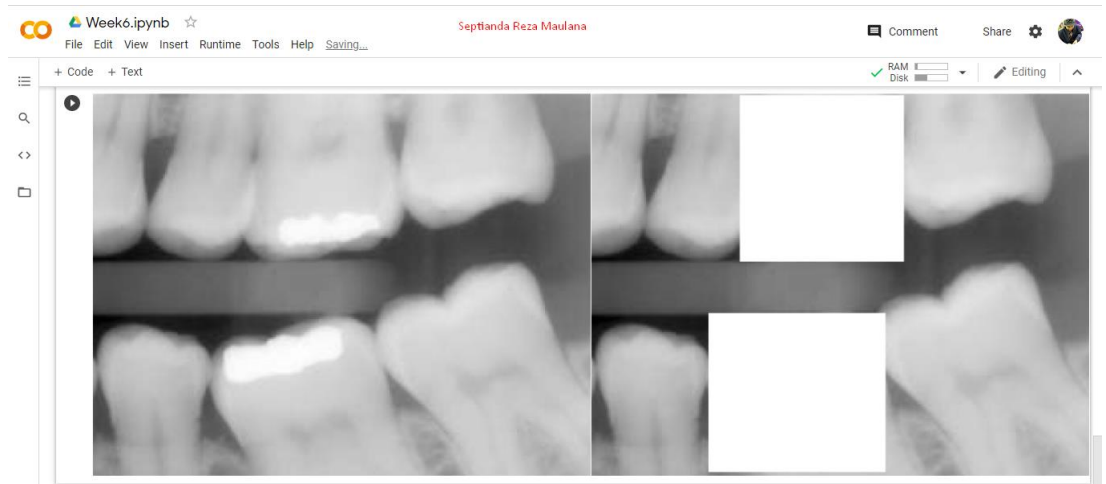
# Opetor OR
img = cv.imread('/content/drive/MyDrive/PCVK/Images/teeth.jpg')
mask_or = np.zeros(img.shape, dtype=np.uint8)
mask_or = cv.rectangle(mask_or, pt1=(320,595), pt2=(800,1025), color=(255,255,255), thickness=-1)

result = img | mask_or

mask_or1 = cv.rectangle(mask_or, pt1=(405,5), pt2=(850,455), color=(255,255,255), thickness=-1)

result_or1 = img | mask_or1

final_frame = cv.hconcat((img, result_or1))
cv2_imshow(final_frame)
```



- Hasil Menggunakan Operator **NOT**

```
Week6.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

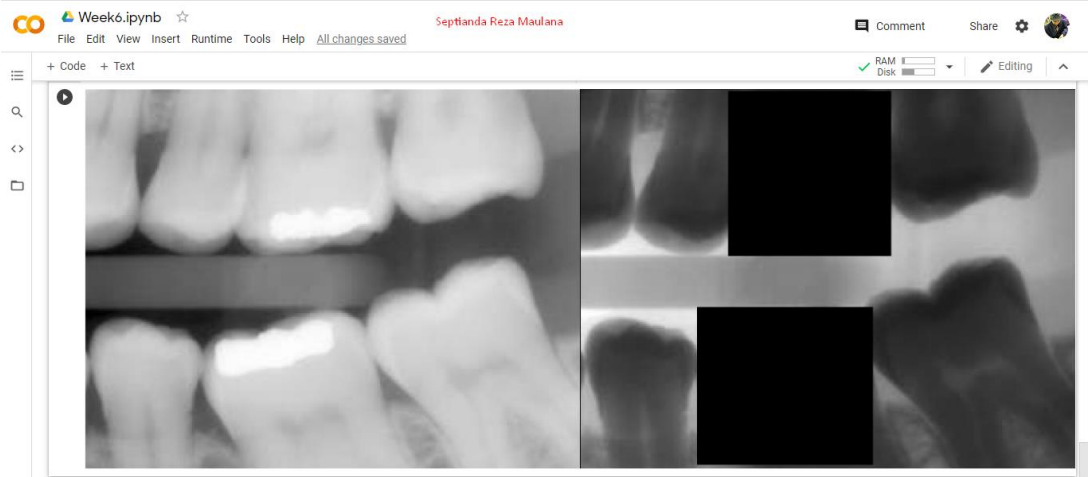
# Opetor NOT
img = cv.imread('/content/drive/MyDrive/PCVK/Images/teeth.jpg')
mask_not = np.zeros(img.shape, dtype=np.uint8)
mask_not = cv.rectangle(mask_not, pt1=(320,595), pt2=(800,1025), color=(255,255,255), thickness=-1)

result = ~img & ~mask_not

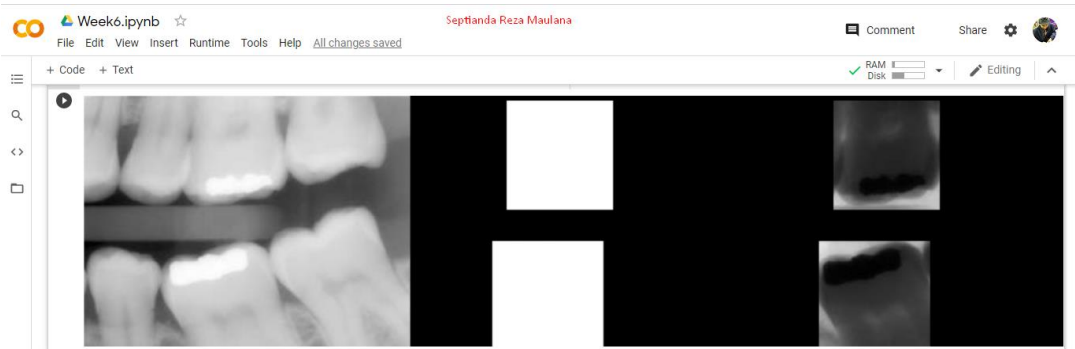
mask_not1 = cv.rectangle(mask_not, pt1=(405,5), pt2=(850,455), color=(255,255,255), thickness=-1)

result_not1 = ~img & ~mask_not1

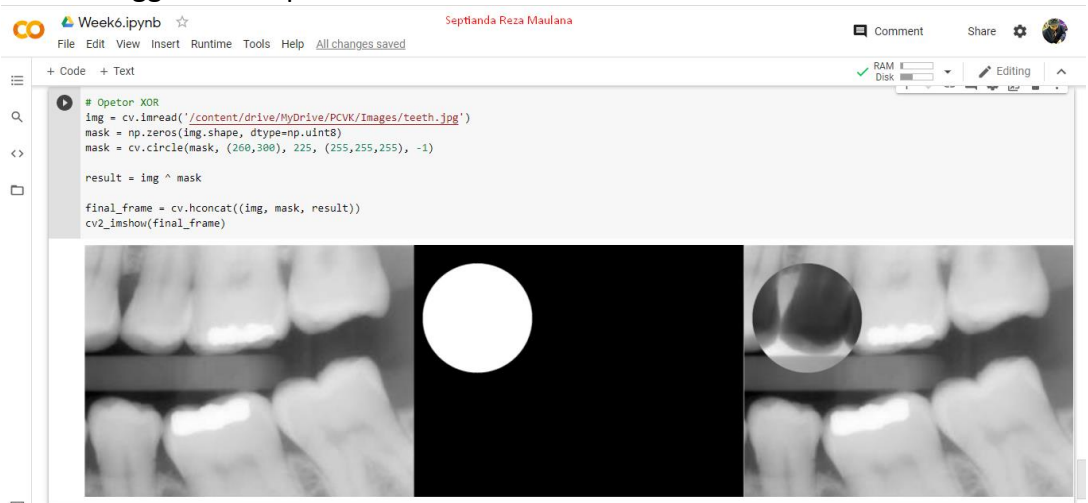
final_frame = cv.hconcat((img, result_not1))
cv2_imshow(final_frame)
```

- Hasil Menggunakan Operator **NAND**



- Hasil Menggunakan Operator **XOR**



Analisa :

1. Pada saat menggunakan Operator **OR**, **Citra A** akan tertumpuk oleh **Citra B** yang mana terdapat 2 bentuk kotak berwarna putih.
2. Pada saat menggunakan Operator **NOT**, hasilnya kebalikan dari Operator **OR** yang mana terdapat 2 bentuk kotak berwarna hitam.
3. Pada saat menggunakan Operator **NAND**, hasilnya akan menghasilkan keluaran Logika 0. Apabila semua masukan (Input) pada Logika 1 dan jika terdapat sebuah Input yang bernilai Logika 0 maka akan menghasilkan keluaran (Output) Logika 1.
4. Pada saat menggunakan Operator **XOR**, maka yang akan terjadi akan memunculkan 2 objek lingkaran dengan berwarna hitam dan putih.