

**LAPORAN UJIAN AKHIR SEMESTER  
MATA KULIAH  
PENGOLAHAN GAMBAR**



**PENYUSUN LAPORAN**

<b>Nama Mahasiswa</b>	<b>NIM</b>	<b>Kelas</b>
(Ramanda Apriza)	(062340833200)	(1 MIN)



**PROGRAM STUDI MANAJEMEN INFORMATIKA  
JURUSAN MANAJEMEN INFORMATIKA  
POLITEKNIK NEGERI SRIWIJAYA  
2023**

Saya akan menjelaskan beberapa kode program terlebih dahulu, jika ingin langsung melihat hasil dari 5 citra ada bagian terakhir dari laporan ini.

Pada screenshot ini baris kode ini pip install berfungsi untuk menginstal paket yang bernama "rembg" dengan mendukung penggunaan GPU dan menyertakan antarmuka baris perintah, alasan saya memisahkan sel komentar dan juga syntax karena ketika saya gabung terjadi error. Program ini sangat penting agar kode-kode berikutnya dapat berjalan, terutama jika menggunakan google colab seperti saya ini. Jika di vs code tidak berjalan, saya sarankan menggunakan google colab.

```
[ ] # Pada baris kode ini pip install berfungsi untuk menginstal paket yang bernama "rembg" dengan mendukung penggunaan GPU dan menyertakan antarmuka baris perintah
pip install rembg[gpu,cli]
Requirement already satisfied: Jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (3.1.2)
Requirement already satisfied: MarkupSafe==2.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (2.1.3)
Requirement already satisfied: matplotlib==3.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (3.7.1)
Requirement already satisfied: orjson==3.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (3.9.10)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (23.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (1.5.3)
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (0.25.1)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (6.0.1)
Requirement already satisfied: semantic-version==2.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (2.10.0)
Requirement already satisfied: tomli==0.12.0 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (0.12.0)
Requirement already satisfied: typer[all]<1.0,>=0.9 in /usr/local/lib/python3.10/dist-packages (from gradio->rembg[cli,gpu]) (0.9.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.8.0->gradio->rembg[cli,gpu]) (2023.6.0)
Requirement already satisfied: websockets<12.0,>=10.0 in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.8.0->gradio->rembg[cli,gpu]) (11.0.3)
Requirement already satisfied: h11==0.8 in /usr/local/lib/python3.10/dist-packages (from uvicorn->rembg[cli,gpu]) (0.14.0)
Requirement already satisfied: jsonschema-specifications==2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema->rembg[cli,gpu]) (2023.12.1)
Requirement already satisfied: referencing==0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema->rembg[cli,gpu]) (0.32.1)
Requirement already satisfied: rpds-py==0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema->rembg[cli,gpu]) (0.16.2)
Requirement already satisfied: coloredlogs in /usr/local/lib/python3.10/dist-packages (from onnxruntime->rembg[cli,gpu]) (15.0.1)
Requirement already satisfied: flatbuffers in /usr/local/lib/python3.10/dist-packages (from onnxruntime->rembg[cli,gpu]) (23.5.26)
Requirement already satisfied: protobuf in /usr/local/lib/python3.10/dist-packages (from onnxruntime->rembg[cli,gpu]) (3.20.3)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from onnxruntime->rembg[cli,gpu]) (1.12)
Requirement already satisfied: platformdirs==2.5.0 in /usr/local/lib/python3.10/dist-packages (from pooch->rembg[cli,gpu]) (4.1.0)
Requirement already satisfied: requests==2.19.0 in /usr/local/lib/python3.10/dist-packages (from pooch->rembg[cli,gpu]) (2.31.0)
Requirement already satisfied: cycler==0.10.0 in /usr/local/lib/python3.10/dist-packages (from pyqt5->rembg[cli,gpu]) (0.58.1)
```

Kemudian ada

```
from google.colab import files
```

```
file = files.upload()
```

Pada bagian ini adalah program yang berisi perintah untuk menampilkan output upload files jika menggunakan google colab dan bisa juga menggunakan cara manual, karena saya menggunakan google colab maka saya menginput kode program ini

```
# Pada bagian ini adalah program yang berisi perintah untuk menampilkan output upload files jika menggunakan google colab dan bisa juga menggunakan cara manual
from google.colab import files
file = files.upload()

Choose Files  pisang.jpg
• pisang.jpg(image/jpeg) - 18997 bytes, last modified: 1/12/2024 - 100% done
Saving pisang.jpg to pisang.jpg
```

Kemudian ada

```
from google.colab import files
```

```
file = files.upload()
```

Pada bagian ini adalah program yang berisi perintah untuk menampilkan output upload files jika menggunakan google colab dan bisa juga menggunakan cara manual, karena saya menggunakan google colab maka saya menginput kode program ini

Kemudian berikut ini adalah kode program yang berfungsi mengimpor library dan juga modul yang akan dipakai dalam grayscale, deteksi tepi, dan segmentasi gambar.

```
[ ] # Pada baris kode ini berfungsi untuk mengimpor modul "rembg" untuk melakukan proses segmentasi menghapus background
from rembg import remove

# Kemudian pada baris kode ini berfungsi untuk mengimpor modul "PIL" untuk manipulasi gambar
from PIL import Image, ImageOps, ImageFilter

# Lalu pada baris kode ini berfungsi untuk mengimpor "io" untuk operasi input/output byte
import io

# Terakhir ada modul "matplotlib.pyplot" untuk menampilkan gambar
import matplotlib.pyplot as plt
```

Lalu pada kode program di bawah ini adalah proses di mana menentukan lokasi gambar yang akan diinput (nama file dan format harus sesuai dengan yang sudah kita upload tadi) dan juga lokasi gambar yang akan dioutput, dalam artian nanti hasil output otomatis terdownload dan tersimpan pada path yang sudah kita pilih lokasi penyimpanannya seperti di bawah ini saya memasukkan ke dalam content. Contoh saya menggunakan foto pisang.jpg

```
# Pada baris kode ini menentukan path gambar input dan path gambar output untuk citra grayscale, deteksi tepi, dan hasil segmentasi
input_path = '/content/pisang.jpg'
output_path_grayscale = '/content/grayscale.png'
output_path_edges = '/content/deteksi_tepi.png'
output_path_removed_bg = '/content/segmentasi.png'
```

Selanjutnya pada kode program di bawah ini bertujuan untuk membaca gambar atau citra asli yang sudah ditentukan, dapat dilihat di bawah ini.

```
# Pada bagian ini membaca gambar asli dari path yang telah ditentukan
with open(input_path, 'rb') as i: # Membuka file gambar dalam mode baca binary
    input_data = i.read() # Membaca data gambar sebagai byte
    original_image = Image.open(io.BytesIO(input_data)) # Membaca gambar dari data byte menggunakan modul "PIL"
```

Pada kode program di bawah ini nanti akan terjadi proses konversi dari gambar asli ke citra greyscale dan hasilnya akan disimpan sebagai file dengan path yang sudah ditentukan

```
# Kode ini mengonversi gambar asli ke citra grayscale menggunakan fungsi "ImageOps.grayscale"
# Hasilnya disimpan sebagai file gambar grayscale dengan path yang telah ditentukan
grayscale_image = ImageOps.grayscale(original_image)
grayscale_image.save(output_path_grayscale)
```

Lalu berikut ini adalah kode program yang berfungsi untuk merubah citra grayscale menjadi citra deteksi tepi dan akan tersimpan pada path yang sudah ditentukan.

```
# Pada kode ini metode deteksi tepi dengan menggunakan filter ImageFilter.FIND_EDGES pada citra grayscale
# Hasilnya nanti akan disimpan sebagai file gambar dengan deteksi tepi di output_path_edges
edges_image = grayscale_image.filter(ImageFilter.FIND_EDGES)
edges_image.save(output_path_edges)
```

Dan di bawah ini adalah proses terakhir yakni menghilangkan latar belakang atau mengubah citra deteksi tepi ke citra segmentasi. Di sini gambar sudah tidak mempunyai latar belakang lagi dan hanya menyisakan objek yang ada. Seperti biasa output akan tersimpan pada path yang sudah ditentukan.

```
# Pada baris kode ini saya menggunakan library "rembg" untuk menghapus latar belakang dari gambar asli
# Dan kemudian hasilnya akan disimpan sebagai file gambar dengan latar belakang yang sudah disegmentasi di output_path_removed_bg
output_data = remove(input_data)
removed_bg_image = Image.open(io.BytesIO(output_data))
removed_bg_image.save(output_path_removed_bg)
```

Pada baris kode ini terdapat kode yang berfungsi untuk menampilkan hasil gambar yang kita inginkan secara langsung

```
# Pada baris kode ini berfungsi agar menampilkan hasil gambar yang kita inginkan secara langsung
plt.figure(figsize=(12, 8))
```

Lalu kode di bawah ini adalah kode untuk menampilkan “Gambar Asli” citra yang belum diubah. Penjelasan lebih rinci tiap baris kode sudah saya buat pada sebelah kanan kode dan bisa dilihat pada gambar berikut ini. Tambahan pada plt axis off adalah untuk menyembunyikan sumbu pada setiap gambar.

```
# Ini adalah kode untuk menampilkan gambar asli dari yang diupload
# Kode ini juga menggunakan Matplotlib untuk menampilkan keempat gambar di bawah kode
plt.subplot(221) # Ini digunakan untuk menentukan layout gambar dan judulnya
plt.imshow(original_image) # Ini digunakan untuk menampilkan gambar
plt.title('Gambar Asli') # Ini digunakan untuk memberikan judul pada gambar
plt.axis('off') # Ini digunakan untuk menyembunyikan sumbu pada setiap gambar
```

Sama seperti tadi, namun kode program berikut ini untuk menampilkan gambar yang sudah menjadi citra grayscale dari yang awalnya gambar asli.

```
# Pada kode ini berfungsi untuk menampilkan gambar yang sudah di deteksi tepi dari yang sebelumnya citra grayscale
plt.subplot(223)
plt.imshow(edges_image, cmap='gray')
plt.title('Citra deteksi tepi')
plt.axis('off')
```

Kemudian ada kode program yang akan menampilkan citra deteksi tepi, sama seperti yang atas tadi namun ini bertujuan menampilkan citra deteksi tepi.

```
# Pada kode ini berfungsi untuk menampilkan gambar yang sudah di deteksi tepi dari yang sebelumnya citra grayscale
plt.subplot(223)
plt.imshow(edges_image, cmap='gray')
plt.title('Citra deteksi tepi')
plt.axis('off')
```

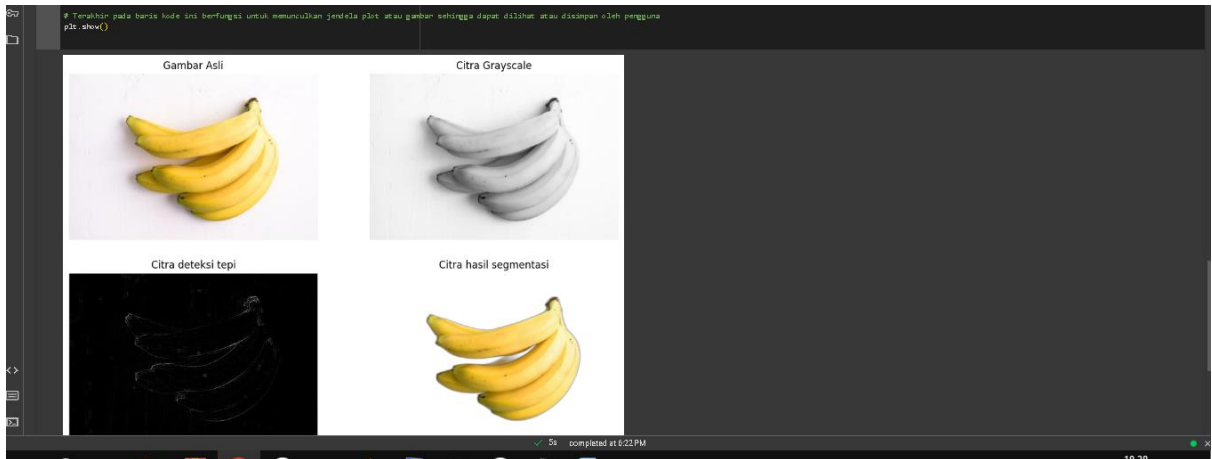
Sama seperti sebelumnya, berikut adalah program untuk menampilkan gambar segmentasi, berbeda dengan yang sebelumnya adalah menampilkan deteksi tepi sedangkan ini menampilkan citra segmentasi atau citra yang sudah tidak memiliki background.

```
# Pada kode ini berfungsi untuk menampilkan gambar yang sudah disegmentasi atau dihilangkan latar belakangnya
plt.subplot(224)
plt.imshow(removed_bg_image)
plt.title('Citra hasil segmentasi')
plt.axis('off')
```

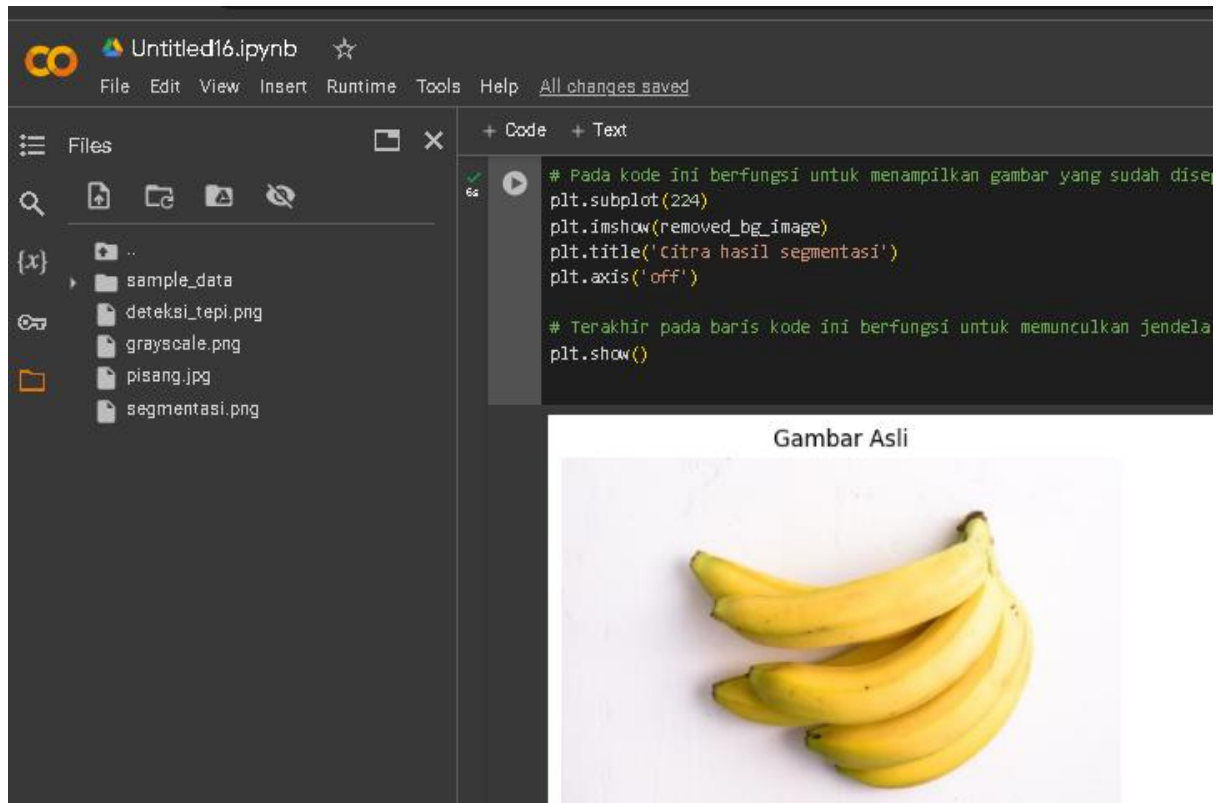
Kemudian terakhir adalah kode yang berfungsi untuk memunculkan jendela plot atau gambar sehingga dapat dilihat dan disimpan oleh pengguna.

```
# Terakhir pada baris kode ini berfungsi untuk memunculkan jendela plot atau gambar sehingga dapat dilihat atau disimpan oleh pengguna
plt.show()
```

**Berikut adalah output yang akan ditampilkan yakni empat gambar yaitu gambar asli, citra yang di grayscale kemudian citra deteksi tepi dan terakhir citra segmentasi**

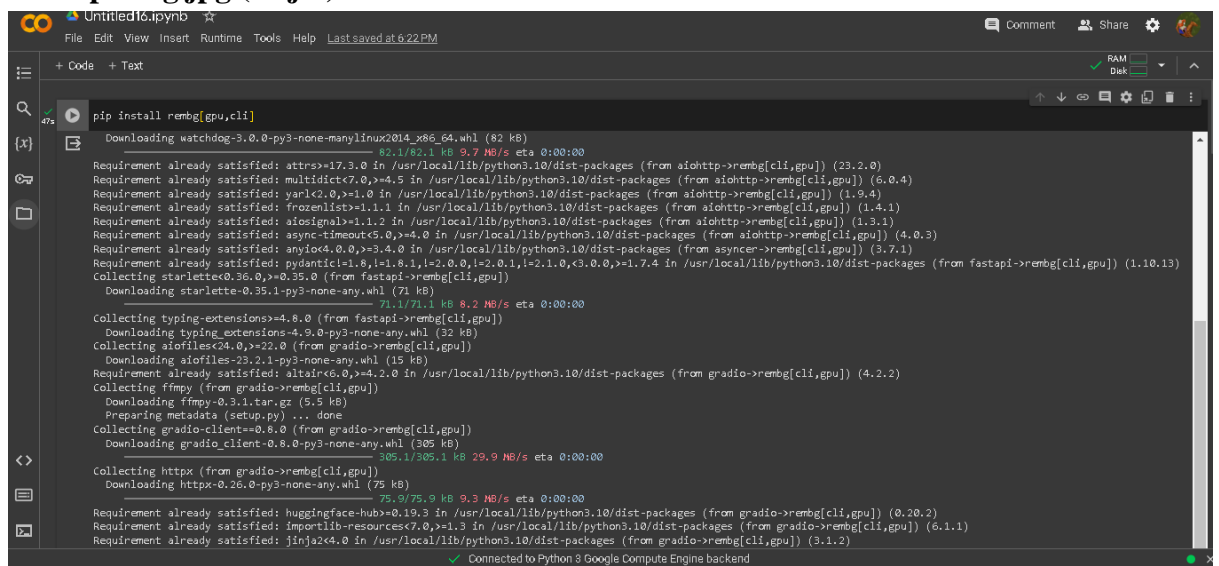


Dan bisa dilihat pada path sebelah kiri, ke tiga foto akan otomatis tersimpan pada lokasi yang sudah kita tentukan di awal tadi dan sesuai nama yang sudah kita siapkan



Berikut screenshot input dan output dari 5 citra yang saya kerjakan :

## 1. Citra pisang.jpg (wajib)



```
# Pada bagian ini adalah program yang berisi perintah untuk menampilkan output upload files jika menggunakan google colab dan bisa juga menggunakan cara manual
from google.colab import files
file = files.upload()

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

[6] # Pada baris kode ini berfungsi untuk mengimpor modul "rebg" untuk melakukan proses segmentasi menghapus background
from rebg import remove

# kemudian pada baris kode ini berfungsi untuk mengimpor modul "PIL" untuk manipulasi gambar
from PIL import Image, ImageOps, ImageFilter

# Lalu pada baris kode ini berfungsi untuk mengimpor "io" untuk operasi input/output byte
import io

# Terakhir ada modul "matplotlib.pyplot" untuk menampilkan gambar
import matplotlib.pyplot as plt

# Pada baris kode ini menentukan path gambar input dan path gambar output untuk citra grayscale, deteksi tepi, dan hasil segmentasi
input_path = '/content/pisang.jpg'
output_path_grayscale = '/content/grayscale.png'
output_path_edges = '/content/deteksi_tepi.png'
output_path_removed_bg = '/content/segmentasi.png'

# Pada bagian ini membaca gambar asli dari path yang telah ditentukan
with open(input_path, 'rb') as f: # membuka file gambar dalam mode baca binary
    input_data = f.read() # Membaca data gambar sebagai byte
    original_image = Image.open(io.BytesIO(input_data)) # Membaca gambar dari data byte menggunakan modul "PIL"

# Kode ini mengonversi gambar asli ke citra grayscale menggunakan fungsi "ImageOps.grayscale"
# Hasilnya disimpan sebagai file gambar grayscale dengan path yang telah ditentukan
grayscale_image = ImageOps.grayscale(original_image)
grayscale_image.save(output_path_grayscale)

# Pada kode ini metode deteksi tepi dengan menggunakan filter ImageFilter.FIND_EDGES pada citra grayscale
# Hasilnya nanti akan disimpan sebagai file gambar dengan deteksi tepi di output_path_edges
edges_image = grayscale_image.filter(ImageFilter.FIND_EDGES)
edges_image.save(output_path_edges)

# Pada baris kode ini saya menggunakan library "rebg" untuk menghapus latar belakang dari gambar asli
# Dan kemudian hasilnya akan disimpan sebagai file gambar dengan latar belakang yang sudah disegmentasi di output_path_removed_bg
output_data = remove(input_data)
removed_bg_image = Image.open(io.BytesIO(output_data))
removed_bg_image.save(output_path_removed_bg)

# Pada baris kode ini berfungsi agar menampilkan hasil gambar yang kita inginkan secara langsung
plt.figure(figsize=(12, 8))

# Ini adalah kode untuk menampilkan gambar asli dari yang diupload
# Kode ini juga menggunakan matplotlib untuk menampilkan keempat gambar di bawah
plt.subplot(221) # Ini digunakan untuk menentukan layout gambar dan judulnya
plt.imshow(original_image) # Ini digunakan untuk menampilkan gambar
plt.title('Gambar Asli') # Ini digunakan untuk memberikan judul pada gambar
plt.axis('off') # Ini digunakan untuk menyembunyikan sumbu pada setiap gambar

# Pada kode ini berfungsi untuk menampilkan gambar yang sudah menjadi citra grayscale
plt.subplot(222)
plt.imshow(grayscale_image, cmap='gray')
plt.title('Citra Grayscale')
plt.axis('off')

# Pada kode ini berfungsi untuk menampilkan gambar yang sudah di deteksi tepi dari yang sebelumnya citra grayscale
plt.subplot(223)
plt.imshow(edges_image, cmap='gray')
plt.title('Citra deteksi tepi')

# Pada kode ini berfungsi untuk menampilkan gambar yang sudah disegmentasi atau dihilangkan latar belakangnya
plt.subplot(224)
plt.imshow(removed_bg_image)
plt.title('Citra hasil segmentasi')
plt.axis('off')

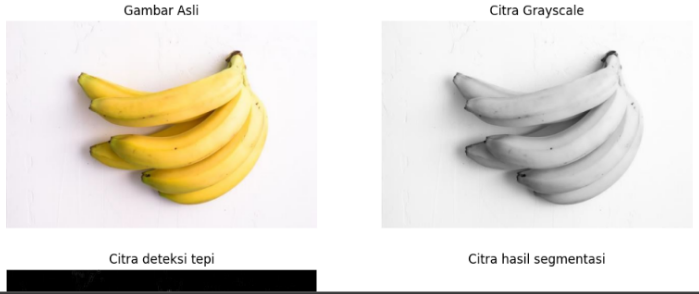
# Terakhir pada baris kode ini berfungsi untuk memunculkan jendela plot atau gambar sehingga dapat dilihat atau disimpan oleh pengguna
plt.show()
```

Gambar Asli

Citra Grayscale

Citra deteksi tepi

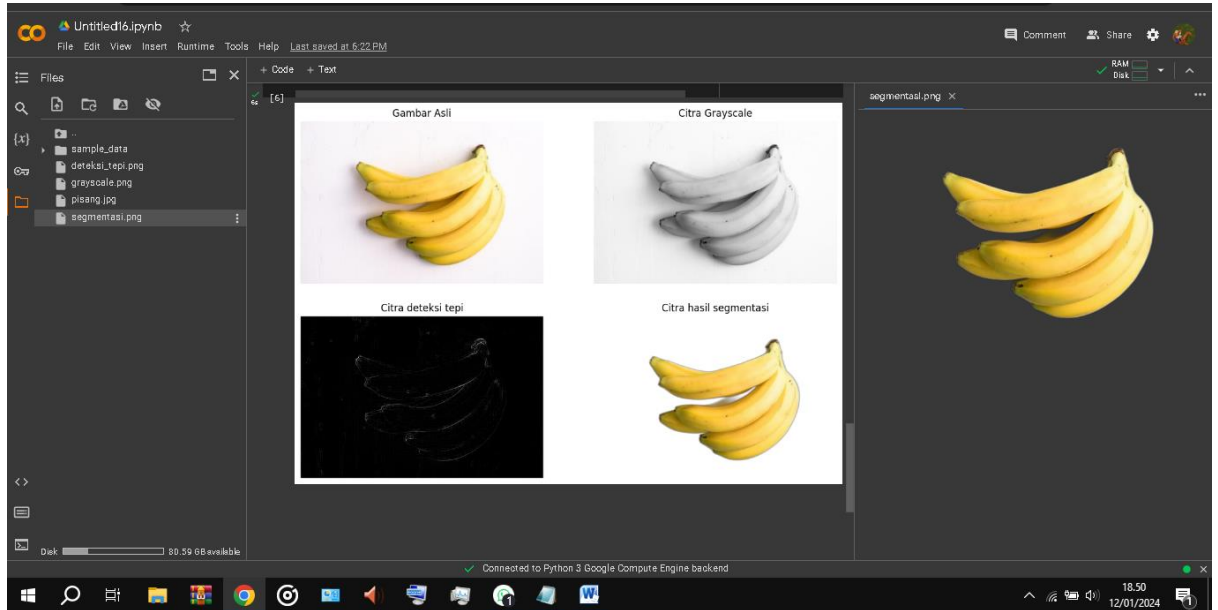
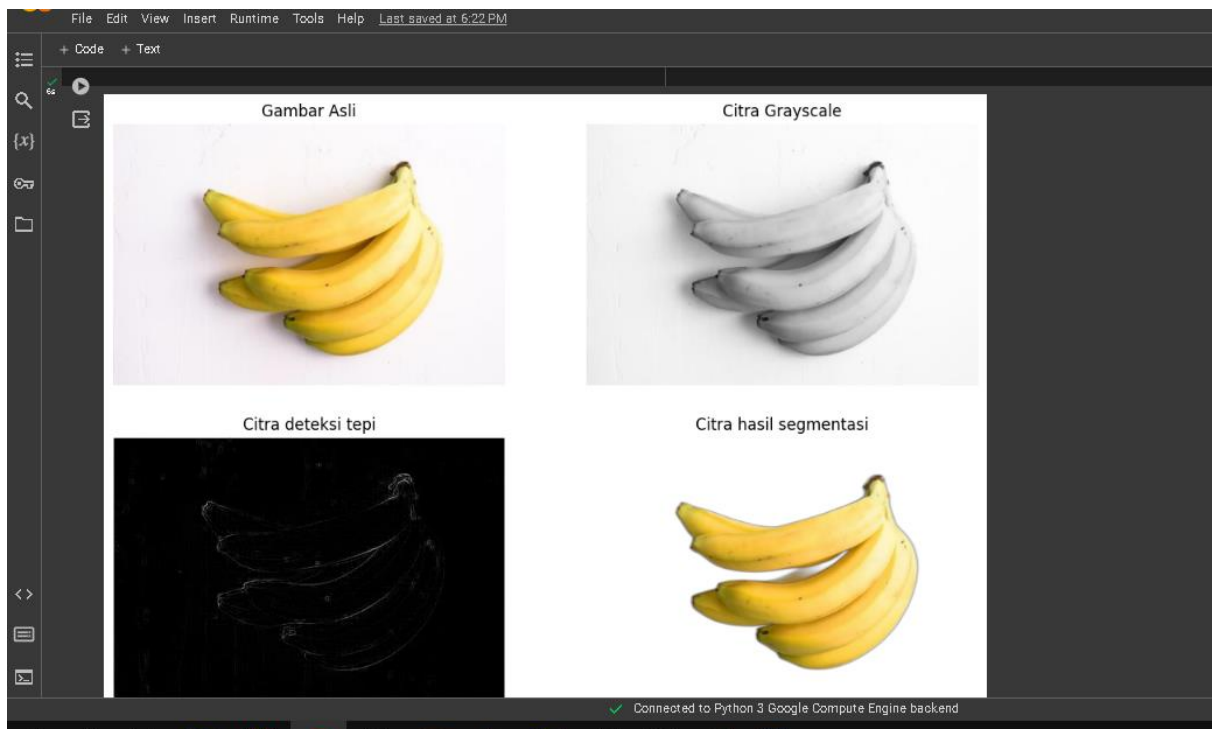
Citra hasil segmentasi



Connected to Python 3 Google Compute Engine backend

18:50  
12/01/2023





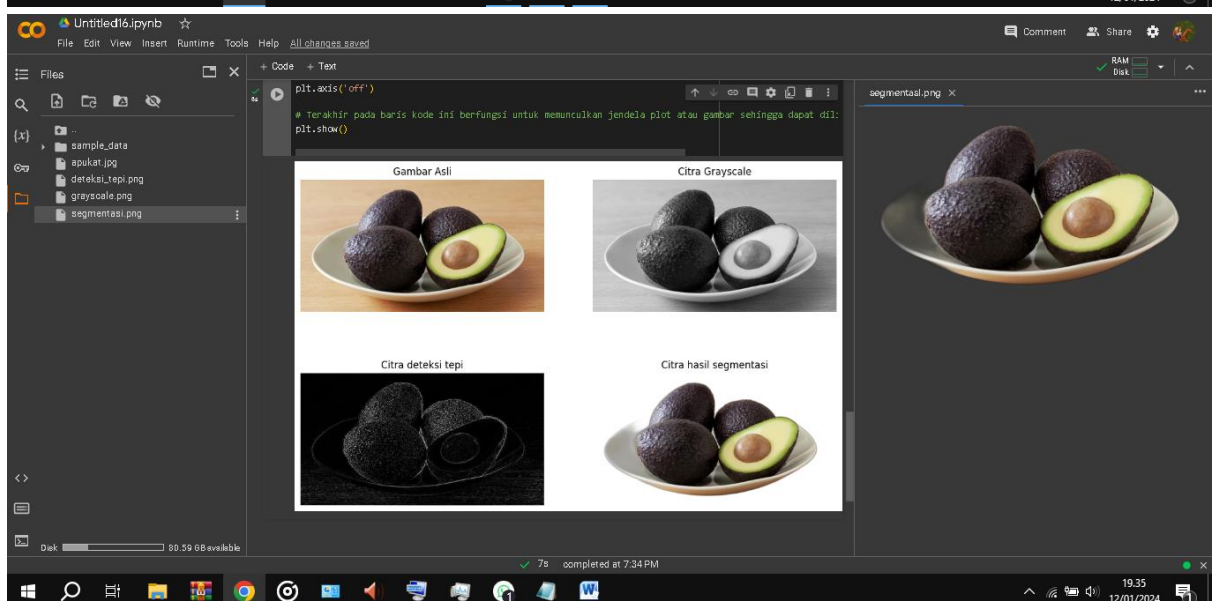
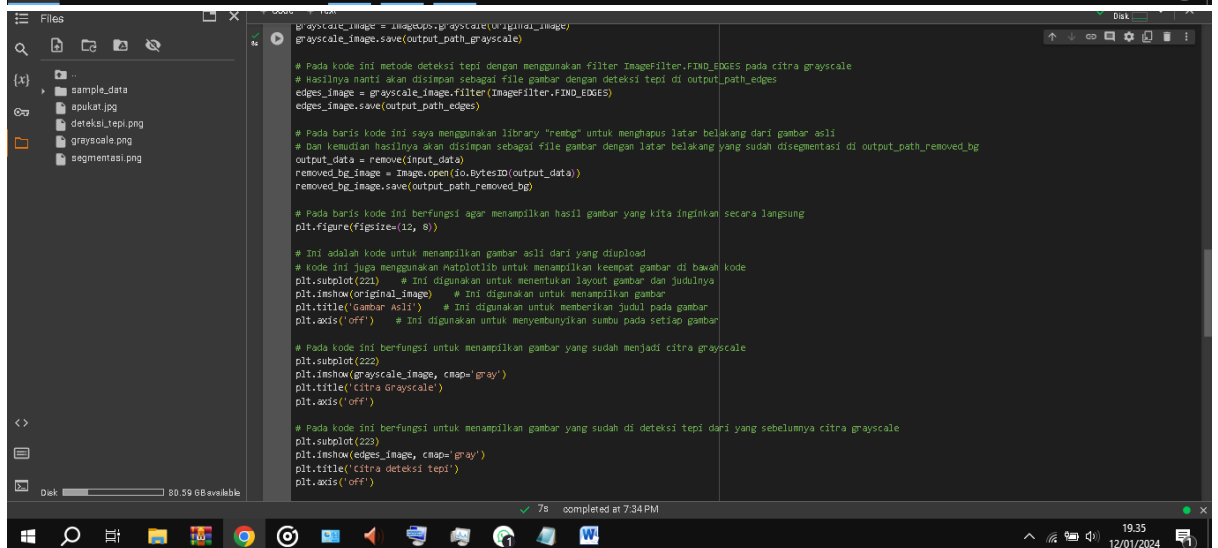
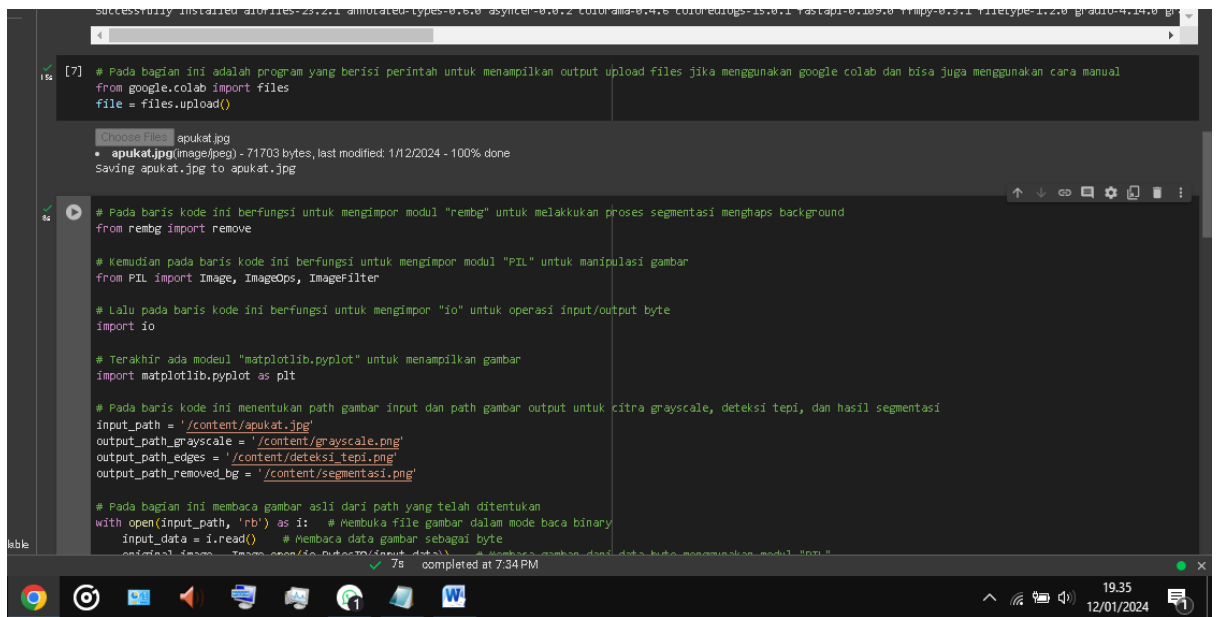


## 2. Citra apukat.jog (wajib)

```
Files + Code + Text
[x]
  sample_data
  apukat.jpg
  deteksi_tepi.png
  grayScale.png
  segmentasi.png
  <>
  80.59 GB available
  Disk

[2] pip install redbg[cuda,cli]

Downloading watchdog-3.0.0-py3-none-manylinux2014_x86_64.whl (62 kB)
62.1/62.1 kB 9.7 MB/s eta 0:00:00
Requirement already satisfied: attrs==17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->redbg[cuda,cli]) (23.2.0)
Requirement already satisfied: multidict==4.0, >=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->redbg[cuda,cli]) (6.0.4)
Requirement already satisfied: yarl==2.0, >=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->redbg[cuda,cli]) (1.9.4)
Requirement already satisfied: frozenlist==1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->redbg[cuda,cli]) (1.4.1)
Requirement already satisfied: aiosignal==1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->redbg[cuda,cli]) (1.3.1)
Requirement already satisfied: async-timeout==5.0, >=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->redbg[cuda,cli]) (4.0.3)
Requirement already satisfied: anyio==4.0.0, >=3.4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->redbg[cuda,cli]) (4.0.3)
Requirement already satisfied: pydantic==1.8, >=1.8.1, <=2.0.0, <=2.0.1, <=2.1.0, <=3.0.0, <=1.7.4 in /usr/local/lib/python3.10/dist-packages (from fastapi->redbg[cuda,cli]) (1.10.13)
Collecting starlette==0.36.0, >=0.35.0 (from fastapi->redbg[cuda,cli])
Downloading starlette-0.36.0-py3-none-any.whl (71 kB)
71.3/71.3 kB 8.2 MB/s eta 0:00:00
Collecting typing-extensions==4.3.0 (from fastapi->redbg[cuda,cli])
Downloading typing_extensions-4.3.0-py3-none-any.whl (32 kB)
Collecting aiofiles==24.0, >=22.0 (from gradio->redbg[cuda,cli])
Downloading aiofiles-24.0-py3-none-any.whl (12 kB)
Requirement already satisfied: altair==6.0, >=4.2.0 in /usr/local/lib/python3.10/dist-packages (from gradio->redbg[cuda,cli]) (4.2.2)
Collecting ffmpy (from gradio->redbg[cuda,cli])
Downloading ffmpy-0.3.1.tar.gz (5.5 kB)
Preparing metadata (setup.py) ... done
Collecting gradio-client==0.8.0 (from gradio->redbg[cuda,cli])
Downloading gradio_client-0.8.0-py3-none-any.whl (305 kB)
305.1/305.1 kB 25.9 MB/s eta 0:00:00
Collecting httpx (from gradio->redbg[cuda,cli])
Downloading httpx-0.26.0-py3-none-any.whl (75 kB)
75.0/75.0 kB 9.3 MB/s eta 0:00:00
Requirement already satisfied: huggingface-hub==0.19.3 in /usr/local/lib/python3.10/dist-packages (from gradio->redbg[cuda,cli]) (0.20.2)
Requirement already satisfied: importlib-resources==7.0, >=4.3 in /usr/local/lib/python3.10/dist-packages (from gradio->redbg[cuda,cli]) (6.1.1)
Requirement already satisfied: jinja2==3.0 in /usr/local/lib/python3.10/dist-packages (from gradio->redbg[cuda,cli]) (3.1.2)
Requirement already satisfied: MarkupSafe==2.0 in /usr/local/lib/python3.10/dist-packages (from gradio->redbg[cuda,cli]) (2.1.3)
Requirement already satisfied: matplotlib==3.0 in /usr/local/lib/python3.10/dist-packages (from gradio->redbg[cuda,cli]) (3.7.4)
Collecting orjson==3.0 (from gradio->redbg[cuda,cli])
Downloading orjson-3.9.10-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (138 kB)
138.7/138.7 kB 11.5 MB/s eta 0:00:00
7s completed at 7:34 PM
```





### 3. Citra apel2.jpg (wajib)

```
Files + Code + Text
[+]
sample_data
apel2.jpg
dataset1.jpg
grayoale.png
segmentasi.png
pip install rembg[cli,gpu]
Collecting rembg[cli,gpu]
  Downloading rembg-2.0.53-py3-none-any.whl (32 kB)
Requirement already satisfied: jsonschema in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (4.19.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (1.23.5)
Collecting onnxruntime (from rembg[cli,gpu])
  Downloading onnxruntime-1.16.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (6.4 MB)
    6.4/6.4 MB 23.4 MB/s eta 0:00:00
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (4.9.0.80)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (9.4.0)
Requirement already satisfied: pooch in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (1.8.0)
Collecting pywt (from rembg[cli,gpu])
  Downloading pywt-1.1.12-py3-none-any.whl (52 kB)
    52.0/52.0 KB 4.8 MB/s eta 0:00:00
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (0.19.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (1.11.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (4.66.1)
Collecting onnxruntime-gpu (from rembg[cli,gpu])
  Downloading onnxruntime-gpu-1.16.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (157.1 MB)
    157.1/157.1 MB 2.5 MB/s eta 0:00:00
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (3.9.1)
Collecting asyncer (from rembg[cli,gpu])
  Downloading asyncer-0.0.2-py3-none-any.whl (8.3 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from rembg[cli,gpu]) (8.1.7)
Collecting fastapi (from rembg[cli,gpu])
  Downloading fastapi-0.109.0-py3-none-any.whl (92 kB)
    92.0/92.0 KB 11.1 MB/s eta 0:00:00
Collecting filetype (from rembg[cli,gpu])
  Downloading filetype-1.2.0-py2.py3-none-any.whl (19 kB)
Collecting gradio (from rembg[cli,gpu])
  Downloading gradio-4.14.0-py3-none-any.whl (16.6 MB)
    16.6/16.6 MB 50.2 MB/s eta 0:00:00
Collecting python-multipart (from rembg[cli,gpu])
  Downloading python-multipart-0.0.6-py3-none-any.whl (45 kB)
    45.7/45.7 KB 5.1 MB/s eta 0:00:00
Collecting uvicorn (from rembg[cli,gpu])
  58 completed at 7:38 PM
```

```
Files + Code + Text
[2] Collecting pydantic==1.8.1, <=2.0.0, >=2.0.1, <=2.1.0, <=3.0.0, >=1.7.4 (from fastapi->rembg[cli,gru])
Downloading pydantic-2.5.3-py3-none-any.whl (381 kB)
Collecting numpy (from fastapi->rembg[cli,gru])
Downloading numpy-2.0.0-cp311-cp311-macosx_11_0_arm64.whl (27.2 MB)
[3] # Pada bagian ini adalah program yang berisi perintah untuk menampilkan output upload files jika menggunakan google colab dan bisa juga menggunakan cara manual
from google.colab import files
file = files.upload()

Choose File(s)
• apel2.jpg (image/pep) - 66456 bytes, last modified: 1/2/2024 - 100% done
Saving apel2.jpg to apel2.jpg

# Pada baris kode ini berfungsi untuk mengimpor modul "rembg" untuk melakukan proses segmentasi menghapus background
from rembg import remove

# kemudian pada baris kode ini berfungsi untuk mengimpor modul "PIL" untuk manipulasi gambar
from PIL import Image, ImageOps, ImageFilter

# Lalu pada baris kode ini berfungsi untuk mengimpor "io" untuk operasi input/output byte
import io

# Terakhir ada modul "matplotlib.pyplot" untuk menampilkan gambar
import matplotlib.pyplot as plt

# Pada baris kode ini menentukan path gambar input dan path gambar output untuk citra grayscale, deteksi tepi, dan hasil segmentasi
input_path = '/content/apel2.jpg'
output_path_grayscale = '/content/grayscale.png'
output_path_edges = '/content/deteksi_tepi.png'
output_path_removed_bg = '/content/segmentasi.png'

# Pada bagian ini membaca gambar asli dari path yang telah ditentukan
with open(input_path, 'rb') as f: # Membaca file gambar dalam mode baca binary
    input_data = f.read() # Membaca data gambar sebagai byte
    original_image = Image.open(io.BytesIO(input_data)) # Membaca gambar dari data byte menggunakan modul "PIL"

# Pada baris kode ini menentukan path gambar input dan path gambar output untuk citra grayscale, deteksi tepi, dan hasil segmentasi
input_path = '/content/apel2.jpg'
output_path_grayscale = '/content/grayscale.png'
output_path_edges = '/content/deteksi_tepi.png'
output_path_removed_bg = '/content/segmentasi.png'

# Pada bagian ini membaca gambar asli dari path yang telah ditentukan
with open(input_path, 'rb') as f: # Membaca file gambar dalam mode baca binary
    input_data = f.read() # Membaca data gambar sebagai byte
    original_image = Image.open(io.BytesIO(input_data)) # Membaca gambar dari data byte menggunakan modul "PIL"

# Kode ini mengonversi gambar asli ke citra grayscale menggunakan fungsi "ImageOps.grayscale"
grayscale_image = ImageOps.grayscale(original_image)
grayscale_image.save(output_path_grayscale)

# Pada kode ini metode deteksi tepi dengan menggunakan filter ImageFilter.FIND_EDGES pada citra grayscale
# Hasilnya nanti akan disimpan sebagai file gambar dengan deteksi tepi di output_path_edges
edges_image = grayscale_image.filter(ImageFilter.FIND_EDGES)
edges_image.save(output_path_edges)

# Pada baris kode ini saya menggunakan library "rembg" untuk menghapus latar belakang dari gambar asli
# Dan kemudian hasilnya akan disimpan sebagai file gambar dengan latar belakang yang sudah disegmentasi di output_path_removed_bg
output_data = remove(input_data)
removed_bg_image = Image.open(io.BytesIO(output_data))
removed_bg_image.save(output_path_removed_bg)

# Pada baris kode ini berfungsi agar menampilkan hasil gambar yang kita inginkan secara langsung
plt.figure(figsize=(12, 8))

# Ini adalah kode untuk menampilkan gambar asli dari yang diupload
# Kode ini juga menggunakan matplotlib untuk menampilkan keempat gambar di bawah kode
plt.subplot(221) # Ini digunakan untuk menentukan layout gambar dan judulnya
plt.imshow(original_image) # Ini digunakan untuk menampilkan gambar
plt.title('Gambar Asli') # Ini digunakan untuk memberikan judul pada gambar
plt.axis('off') # Ini digunakan untuk menyembunyikan sumbu pada setiap gambar
```

```
Files + Code + Text

# Pada baris kode ini menentukan path gambar input dan path gambar output untuk citra grayscale, deteksi tepi, dan hasil segmentasi
input_path = '/content/apel2.jpg'
output_path_grayscale = '/content/grayscale.png'
output_path_edges = '/content/deteksi_tepi.png'
output_path_removed_bg = '/content/segmentasi.png'

# Pada bagian ini membaca gambar asli dari path yang telah ditentukan
with open(input_path, 'rb') as f: # Membaca file gambar dalam mode baca binary
    input_data = f.read() # Membaca data gambar sebagai byte
    original_image = Image.open(io.BytesIO(input_data)) # Membaca gambar dari data byte menggunakan modul "PIL"

# Kode ini mengonversi gambar asli ke citra grayscale menggunakan fungsi "ImageOps.grayscale"
grayscale_image = ImageOps.grayscale(original_image)
grayscale_image.save(output_path_grayscale)

# Pada kode ini metode deteksi tepi dengan menggunakan filter ImageFilter.FIND_EDGES pada citra grayscale
# Hasilnya nanti akan disimpan sebagai file gambar dengan deteksi tepi di output_path_edges
edges_image = grayscale_image.filter(ImageFilter.FIND_EDGES)
edges_image.save(output_path_edges)

# Pada baris kode ini saya menggunakan library "rembg" untuk menghapus latar belakang dari gambar asli
# Dan kemudian hasilnya akan disimpan sebagai file gambar dengan latar belakang yang sudah disegmentasi di output_path_removed_bg
output_data = remove(input_data)
removed_bg_image = Image.open(io.BytesIO(output_data))
removed_bg_image.save(output_path_removed_bg)

# Pada baris kode ini berfungsi agar menampilkan hasil gambar yang kita inginkan secara langsung
plt.figure(figsize=(12, 8))

# Ini adalah kode untuk menampilkan gambar asli dari yang diupload
# Kode ini juga menggunakan matplotlib untuk menampilkan keempat gambar di bawah kode
plt.subplot(221) # Ini digunakan untuk menentukan layout gambar dan judulnya
plt.imshow(original_image) # Ini digunakan untuk menampilkan gambar
plt.title('Gambar Asli') # Ini digunakan untuk memberikan judul pada gambar
plt.axis('off') # Ini digunakan untuk menyembunyikan sumbu pada setiap gambar
```

```
Files + Code + Text

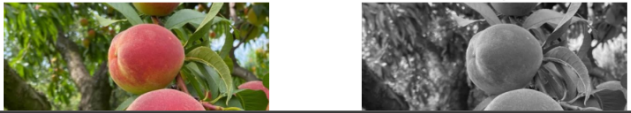
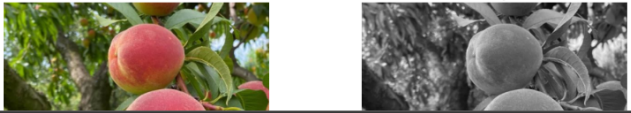
plt.imshow(original_image) # Ini digunakan untuk menampilkan gambar
plt.title('Gambar Asli') # Ini digunakan untuk memberikan judul pada gambar
plt.axis('off') # Ini digunakan untuk menyembunyikan sumbu pada setiap gambar

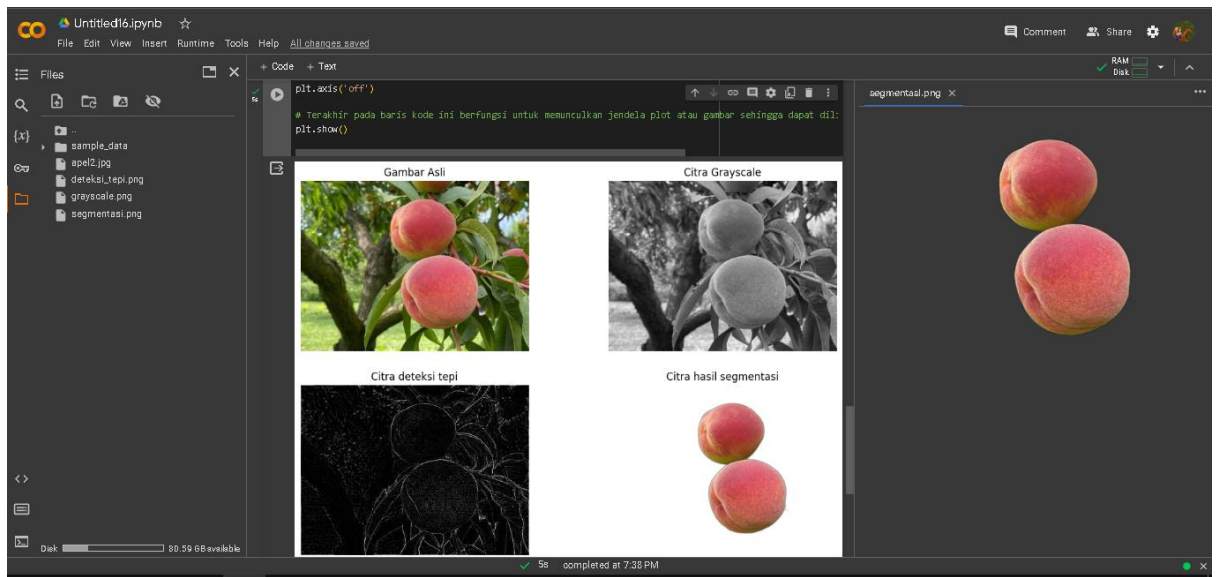
# Pada kode ini berfungsi untuk menampilkan gambar yang sudah menjadi citra grayscale
plt.subplot(222)
plt.imshow(grayscale_image, cmap='gray')
plt.title('Citra Grayscale')
plt.axis('off')

# Pada kode ini berfungsi untuk menampilkan gambar yang sudah di deteksi tepi dari yang sebelumnya citra grayscale
plt.subplot(223)
plt.imshow(edges_image, cmap='gray')
plt.title('Citra deteksi tepi')
plt.axis('off')

# Pada kode ini berfungsi untuk menampilkan gambar yang sudah disegmentasi atau dihilangkan latar belakangnya
plt.subplot(224)
plt.imshow(removed_bg_image)
plt.title('Citra hasil segmentasi')
plt.axis('off')

# Terakhir pada baris kode ini berfungsi untuk memunculkan jendela plot atau gambar sehingga dapat dilihat atau disimpan oleh pengguna
plt.show()

Gambar Asli Citra Grayscale


```




#### 4. Citra Kopi (hasil kamera handphone)



```
Untitled16.py - 1
File Edit View Insert Runtime Tools Help Saving...
Files
sample_data
IMG_20230706_191240.jpg
deteksi_tepi.png
graysoale.png
segmentasi.png
+ Code + Text
[2] pip install rembg[cli]
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<4.0.0,>=3.4.0->asyncio->rembg[cli,gpu]) (3.6)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<4.0.0,>=3.4.0->asyncio->rembg[cli,gpu]) (1.3.0)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<4.0.0,>=3.4.0->asyncio->rembg[cli,gpu]) (1.2.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.3->gradio->rembg[cli,gpu]) (3.15.1)
Requirement already satisfied: contourpy>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->gradio->rembg[cli,gpu]) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->gradio->rembg[cli,gpu]) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->gradio->rembg[cli,gpu]) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->gradio->rembg[cli,gpu]) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->gradio->rembg[cli,gpu]) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->gradio->rembg[cli,gpu]) (2.8.2)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev in /usr/local/lib/python3.10/dist-packages (from numba>=0.49.0->matplotlib->rembg[cli,gpu]) (0.41.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>3.0,>=1.0->gradio->rembg[cli,gpu]) (2023.11.1)
Collecting annotated-types>=0.4.0 (from pydantic<1.8,>=1.8.1,>=2.0.1,>=2.1.0,(3.0.0,>=1.7.4->fastapi->rembg[cli,gpu])
  Downloading annotated_types-0.6.0-py3-none-any.whl (12 kB)
Collecting pydantic-core==2.14.6 (from pydantic<1.8,>=1.8.1,>=2.0.1,>=2.1.0,(3.0.0,>=1.7.4->fastapi->rembg[cli,gpu])
  Downloading pydantic_core-2.14.6-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pochoir->rembg[cli,gpu]) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pochoir->rembg[cli,gpu]) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pochoir->rembg[cli,gpu]) (2023.11.17)
Collecting colorama>=0.5.0,<0.4.3 (from typer[all]<1.0,>=0.9->gradio->rembg[cli,gpu])
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting shellingham>=0.0.0,<1.0.0 (from typer[all]<1.0,>=0.9->gradio->rembg[cli,gpu])
  Downloading shellingham-1.5.4-py2.py3-none-any.whl (9.8 kB)
Requirement already satisfied: rich<14.0.0,>=10.11.0 in /usr/local/lib/python3.10/dist-packages (from typer[all]<1.0,>=0.9->gradio->rembg[cli,gpu]) (13.7.0)
Collecting humanfriendly>=5.1 (from coloredlogs->oncoruntime->rembg[cli,gpu])
  Downloading humanfriendly-10.0-py2.py3-none-any.whl (86 kB)
Collecting httplib2>=1.* (from httpx->gradio->rembg[cli,gpu])
  Downloading httplib2-1.0.2-py3-none-any.whl (76 kB)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->oncoruntime->rembg[cli,gpu]) (1.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0->gradio->rembg[cli,gpu]) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio->rembg[cli,gpu]) (2.0.2)
Requirement already satisfied: pygments>=2.0.0,<2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio->rembg[cli,gpu]) (2.16.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from typing_extensions>=3.7.4.3->pydantic-core==2.14.6->pydantic<1.8,>=1.8.1,>=2.0.1,>=2.1.0,(3.0.0,>=1.7.4->fastapi->rembg[cli,gpu]) (4.10.0)
11s completed at 7:43 PM
```

```
111 # Pada bagian ini adalah program yang berisi perintah untuk menampilkan output upload files jika menggunakan google colab dan bisa juga menggunakan cara manual
from google.colab import files
file = files.upload()

Choose File(s)
IMG_20230706_191240.jpg

Saving IMG_20230706_191240.jpg to /content/IMG_20230706_191240.jpg

112 # Pada baris kode ini berfungsi untuk mengimport modul "rembg" untuk melakukan proses segmentasi menghias background
from rembg import remove

# kemudian pada baris kode ini berfungsi untuk mengimport modul "PIL" untuk manipulasi gambar
from PIL import Image, ImageOps, ImageFilter

# Lalu pada baris kode ini berfungsi untuk mengimport "io" untuk operasi input/output byte
import io

# Terakhir ada modul "matplotlib.pyplot" untuk menampilkan gambar
import matplotlib.pyplot as plt

# Pada baris kode ini menentukan path gambar input dan path gambar output untuk citra grayscale, deteksi tepi, dan hasil segmentasi
input_path = '/content/IMG_20230706_191240.jpg'
output_path_grayscale = '/content/graysoale.png'
output_path_edges = '/content/deteksi_tepi.png'
output_path_removed_bg = '/content/segmentasi.png'

# Pada bagian ini membaca gambar asli dari path yang telah ditentukan
with open(input_path, 'rb') as i: # Membuka file gambar dalam mode baca binary
    input_data = i.read() # membaca data gambar sebagai byte
    # Membaca data gambar sebagai byte
    # original_image = Image.open(io.BytesIO(input_data)) # Membaca gambar dari data byte menggunakan modul "PIL"

11s completed at 7:43 PM
```

```
113 with open(input_path, 'rb') as i: # Membuka file gambar dalam mode baca binary
    input_data = i.read() # membaca data gambar sebagai byte
    original_image = Image.open(io.BytesIO(input_data)) # Membaca gambar dari data byte menggunakan modul "PIL"

# Kode ini mengonversi gambar asli ke citra grayscale menggunakan fungsi "ImageOps.grayscale"
# Hasilnya disimpan sebagai file gambar grayscale dengan path yang telah ditentukan
grayscale_image = ImageOps.grayscale(original_image)
grayscale_image.save(output_path_grayscale)

# Pada baris kode ini metode deteksi tepi dengan menggunakan filter ImageFilter.FIND_EDGES pada citra grayscale
# Hasilnya nanti akan disimpan sebagai file gambar dengan deteksi tepi di output_path_edges
edges_image = grayscale_image.filter(ImageFilter.FIND_EDGES)
edges_image.save(output_path_edges)

# Pada baris kode ini saya menggunakan library "rembg" untuk menghapus latar belakang dari gambar asli
# Dan kemudian hasilnya akan disimpan sebagai file gambar dengan latar belakang yang sudah di segmentasi di output_path_removed_bg
output_data = remove(input_data)
removed_bg_image = Image.open(io.BytesIO(output_data))
removed_bg_image.save(output_path_removed_bg)

# Pada baris kode ini berfungsi agar menampilkan hasil gambar yang kita inginkan secara langsung
plt.figure(figsize=(12, 9))

# Ini adalah kode untuk menampilkan gambar asli dari yang diupload
# Kode ini juga menggunakan matplotlib untuk menampilkan keempat gambar di bawah kode
plt.subplot(221) # Ini digunakan untuk menentukan layout gambar dan judulnya
plt.imshow(original_image) # Ini digunakan untuk menampilkan gambar
plt.title('Gambar Asli') # Ini digunakan untuk memberikan judul pada gambar
plt.axis('off') # Ini digunakan untuk menyembunyikan sumbu pada setiap gambar

# Pada baris kode ini berfungsi untuk menampilkan gambar yang sudah menjadi citra grayscale
plt.subplot(222)
plt.imshow(grayscale_image, cmap='gray')
plt.title('Citra Grayscale')
plt.axis('off')
```

Files

sample\_data

IMG\_20230706\_191240.jpg

deteksi\_tepi.png

grayscale.png

segmentasi.png

+ Code + Text

```
plt.title('Citra Grayscale')
plt.axis('off')



# Pada kode ini berfungsi untuk menampilkan gambar yang sudah di deteksi tepi dari yang sebelumnya citra grayscale
plt.subplot(223)
plt.imshow(edges_image, cmap='gray')
plt.title('Citra deteksi tepi')
plt.axis('off')

# Pada kode ini berfungsi untuk menampilkan gambar yang sudah disegmentasi atau dihilangkan latar belakangnya
plt.subplot(224)
plt.imshow(removed_bg_image)
plt.title('Citra hasil segmentasi')
plt.axis('off')

# Terakhir pada baris kode ini berfungsi untuk memunculkan jendela plot atau gambar sehingga dapat dilihat atau disimpan oleh pengguna
plt.show()
```

Gambar Asli

Citra Grayscale



11s completed at 7:43 PM

Untitled1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

sample\_data

IMG\_20230706\_191240.jpg

deteksi\_tepi.png

grayscale.png

segmentasi.png

+ Code + Text

```
plt.title('Citra Grayscale')
plt.axis('off')

# Pada kode ini berfungsi untuk menampilkan gambar yang sudah di deteksi tepi dari yang sebelumnya citra grayscale
plt.subplot(223)
plt.imshow(edges_image, cmap='gray')
plt.title('Citra deteksi tepi')
plt.axis('off')





# Pada kode ini berfungsi untuk menampilkan gambar yang sudah disegmentasi atau dihilangkan latar belakangnya
plt.subplot(224)
plt.imshow(removed_bg_image)
plt.title('Citra hasil segmentasi')
plt.axis('off')

# Terakhir pada baris kode ini berfungsi untuk memunculkan jendela plot atau gambar sehingga dapat dilihat atau disimpan oleh pengguna
plt.show()
```

Gambar Asli

Citra deteksi tepi

Citra hasil segmentasi



11s completed at 7:43 PM





**5. Citra Mainan Minion (hasil kamera handphone)**



