LAPORAN TEORI

PENGOLAHAN CITRA DIGITAL

NAMA : Rama Dinantiar

NIM : 202331044

KELAS : PCD C

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 21

ASISTEN : 1. Abdur Rasyid Ridho

2. Rizqy Amanda

3. Kashrina Masyid Azka

4. Izzat Islami Kagapi

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1. **Perbedaan antara operator deteksi tepi Sobel, Prewitt, dan Canny**
    - Sobel: Sobel itu operator deteksi tepi yang sederhana, biasanya digunakan untuk mendeteksi perubahan besar intensitas warna di citra. Bisa dibilang Sobel itu semacam solusi cepat untuk deteksi tepi.
        - Keunggulan: Cepat dan mudah diterapkan
        - Kelemahan: Cenderung sensitif terhadap noise, jadi kalau citranya berisik, hasilnya bisa kurang bagus.
    - Prewitt: Mirip dengan Sobel, cuma di Prewitt mask-nya sedikit berbeda. Prewitt lebih halus dibanding Sobel, tapi tetap bekerja dengan cara mendeteksi perubahan intensitas citra.
        - Keunggulan: Hasilnya lebih stabil dan halus.
        - Kelemahan: Masih rentan terhadap noise, dan agak kurang tajam dibanding Sobel.
    - Canny: Canny adalah deteksi tepi yang lebih canggih, dia melalui beberapa tahap, dari smoothing citra dengan filter Gaussian, mencari tepi, sampai menghilangkan tepi yang nggak penting.
        - Keunggulan: Hasilnya sangat tajam dan jelas, dan dia jauh lebih baik mengurangi noise dibanding Sobel dan Prewitt.
        - Kelemahan: Prosesnya lebih lama karena banyak tahap, jadi butuh lebih banyak komputasi.
2. **Perbedaan antara transformasi translasi, rotasi, dan skala dalam transformasi geometrik citra. Dan contohnya.**
    - Translasi (Pergeseran): Cuma memindahkan citra ke posisi yang berbeda, tanpa merubah ukuran atau orientasi.
      Contoh: Misalnya, kamu menggeser gambar ke kanan 50 piksel.
    - Rotasi: Memutar citra dalam arah tertentu, bisa 90°, 180°, atau sudut lainnya.
      Contoh: Citra diputar 90° searah jarum jam.
    - Skala (Penskalaan): Mengubah ukuran citra, bisa jadi lebih besar atau lebih kecil, tapi bentuknya tetap sama.
      Contoh: Kamu memperbesar gambar dari ukuran 200x200 piksel menjadi 400x400 piksel.
3. **Transformasi affine dalam transformasi geometrik. Dua contoh dalam aplikasi pemrosesan citra.**
   Transformasi Affine: Transformasi ini tetap mempertahankan garis lurus dan hubungan jarak antar titik meskipun posisi atau ukuran citra berubah. Artinya, objek nggak akan berubah bentuk atau kecenderungannya, cuma dipindah, diputar, atau diubah ukurannya.
   Contoh:
        - **Penskalaan**: Mengubah ukuran objek citra.
        - **Rotasi**: Memutar objek citra dalam ruang 2D.
4. **Tujuan utama antara operasi deteksi tepi dengan operasi transformasi geometrik (resize dan rotasi)**
    - Deteksi Tepi: Tujuannya mencari batas-batas antara objek dalam citra, jadi kamu bisa melihat kontur dengan jelas.

Contoh: Misalnya, untuk mengenali bentuk objek, kita perlu mendeteksi tepian objek tersebut.

- Transformasi Geometrik: Tujuannya lebih untuk mengubah ukuran atau orientasi citra tanpa mengubah informasi penting di dalam citra itu. Jadi, kamu mungkin ingin merubah citra agar pas dengan tampilan atau sudut tertentu.
  Contoh: Merotasi citra supaya objek yang ada di dalamnya menjadi lebih sesuai dengan kebutuhan kita.

5. **Proses rotasi citra bekerja dari sudut pandang pemetaan piksel**
   Proses Rotasi: Saat citra diputar, posisi setiap piksel dihitung berdasarkan matriks rotasi, tetapi citra itu sendiri nggak berubah ukurannya. Karena citra diputar, beberapa piksel akan keluar dari batas citra yang asli dan menciptakan area kosong.
   Kenapa Ada Area Kosong? Setelah citra diputar, ada beberapa area yang tidak terisi oleh piksel citra asli, jadi bagian ini diisi dengan warna hitam (biasanya disebut sebagai "background" atau "area kosong"). Jadi, bagian sudut citra yang terputar nggak ada data citranya, dan inilah yang membuat bagian tersebut menjadi hitam.

LAPORAN PRAKTIKUM

PENGOLAHAN CITRA DIGITAL

NAMA : Rama Dinantiar

NIM : 202331044

KELAS : PCD C

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 21

ASISTEN : 1. Abdur Rasyid Ridho

2. Rizqy Amanda

3. Kashrina Masyid Azka

4. Izzat Islami Kagapi

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

Deteksi Tepi dan Operasi Geometrik

IMPORT LIBRARY

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi: Sel ini mengimpor tiga pustaka utama yang sering digunakan dalam pengolahan citra di Python:

- cv2 (OpenCV): Ini adalah pustaka utama untuk fungsi-fungsi pengolahan citra, seperti membaca, memanipulasi, dan menampilkan gambar.
- numpy as np: NumPy adalah pustaka fundamental untuk komputasi numerik di Python, terutama untuk bekerja dengan array (larik) multi-dimensi, yang merupakan representasi umum untuk gambar.
- matplotlib.pyplot as plt: Matplotlib adalah pustaka untuk membuat plot dan visualisasi data. pyplot digunakan untuk menampilkan gambar dalam bentuk grafik atau jendela terpisah.

1. KONVERSI WARNA

```
image = cv2.imread('parkiran.jpg')
image.shape

# Rama Dinantiar (202331044)
```

```
(799, 1200, 3)
```

Tujuan dan Fungsi:

- image = cv2.imread('parkiran.jpg'): Baris ini membaca (memuat) sebuah gambar dari file bernama parkiran.jpg dan menyimpannya ke dalam variabel image. Fungsi cv2.imread() adalah bagian dari pustaka OpenCV yang berfungsi untuk tujuan ini.
- image.shape: Baris ini menampilkan bentuk (dimensi) dari gambar yang telah dimuat. Bentuk gambar biasanya direpresentasikan dalam format (tinggi, lebar, jumlah_kanal_warna). Misalnya, (799, 1200, 3) berarti gambar memiliki tinggi 799 piksel,

lebar 1200 piksel, dan 3 kanal warna (merah, hijau, biru), menandakan ini adalah gambar berwarna (RGB).

- Output (799, 1200, 3) mengonfirmasi dimensi gambar yang berhasil dimuat.

MEMBACA GAMBAR

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi: Sel ini melakukan konversi warna pada gambar:

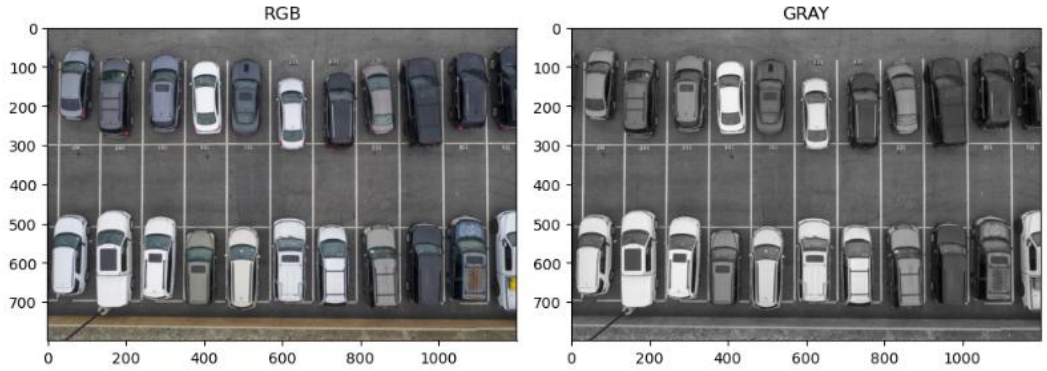
- image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB): OpenCV secara default membaca gambar dalam format BGR (Biru, Hijau, Merah). Baris ini mengonversi gambar image (yang semula BGR) menjadi format RGB yang umum digunakan oleh pustaka lain seperti Matplotlib untuk tampilan yang benar. Hasilnya disimpan dalam image_rgb.
- img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY): Baris ini mengonversi gambar image (yang semula BGR, namun konversi ke RGB tidak memengaruhi konversi ke grayscale dari image asli) menjadi gambar *grayscale* (skala abu-abu). Gambar *grayscale* hanya memiliki satu kanal intensitas piksel, yang seringkali diperlukan untuk algoritma deteksi tepi tertentu. Hasilnya disimpan dalam img_gray.

```
# Menampilkan gambar RGB dan Grayscale
fig, axs = plt.subplots(1,2, figsize=(10,10))

ax = axs.ravel()

ax[0].imshow(image_rgb)
ax[0].set_title("RGB")

ax[1].imshow(img_gray,cmap ='gray')
ax[1].set_title("GRAY")

plt.tight_layout()
plt.show()

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi: Sel ini bertanggung jawab untuk menampilkan gambar asli (dalam format RGB) dan gambar skala abu-abu yang telah dikonversi.

- fig, axs = plt.subplots(1,2, figsize=(10,10)): Membuat sebuah *figure* (area plot keseluruhan) dan set kumpulan *subplots* (plot individu) di dalamnya. 1,2 berarti akan ada 1 baris dan 2 kolom *subplot*. figsize=(10,10) mengatur ukuran total *figure*.
- ax = axs.ravel(): Mengubah array 2D axs (jika ada lebih dari 1x1 *subplot*) menjadi array 1D agar lebih mudah diakses menggunakan indeks tunggal.
- ax[0].imshow(image_rgb): Menampilkan gambar image_rgb (gambar asli dalam format RGB) pada *subplot* pertama (ax[0]).
- ax[0].set_title("RGB"): Memberikan judul "RGB" pada *subplot* pertama.
- ax[1].imshow(img_gray,cmap ='gray'): Menampilkan gambar img_gray (gambar skala abu-abu) pada *subplot* kedua (ax[1]). Parameter cmap='gray' memastikan gambar ditampilkan dalam skema warna *grayscale*.
- ax[1].set_title("GRAY"): Memberikan judul "GRAY" pada *subplot* kedua.
- plt.tight_layout(): Menyesuaikan parameter *subplot* agar *subplots* pas di dalam *figure area* dengan rapi, menghindari tumpang tindih.
- plt.show(): Menampilkan *figure* yang berisi kedua gambar (RGB dan Grayscale).

OUTPUT:



2. DETEKSI TEPI MENGGUNAKAN CANNY

MENDETEKSI AMBANG BATAS DARI TEPI

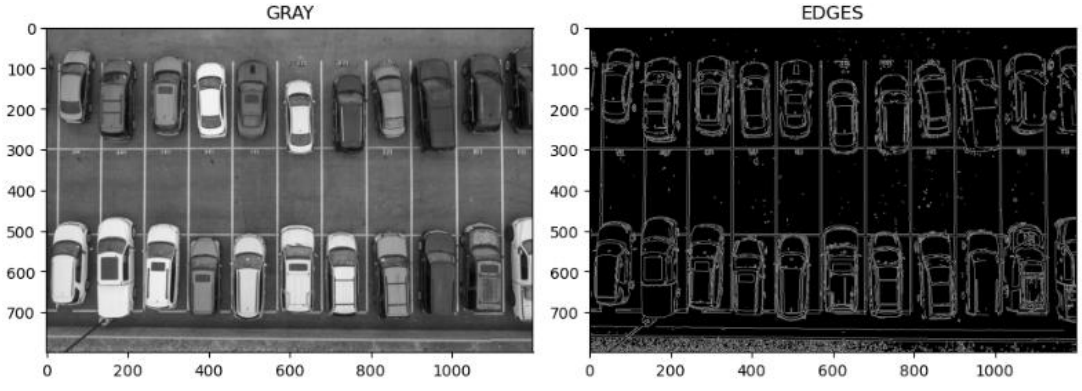
```
# Deteksi tepi menggunakan Canny
edges = cv2.Canny(image, 100,150)
# Rama Dinantiar (202331044)
```

```
fig, axs = plt.subplots(1,2, figsize=(10,10))

ax = axs.ravel()

ax[0].imshow(img_gray,cmap='gray')
ax[0].set_title("GRAY")

ax[1].imshow(edges,cmap ='gray')
ax[1].set_title("EDGES")

plt.tight_layout()
plt.show()

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi: Sel ini menerapkan algoritma deteksi tepi Canny pada gambar dan menampilkan hasil deteksi tepi menggunakan algoritma Canny.

- edges = cv2.Canny(image, 100,150): Fungsi cv2.Canny() digunakan untuk melakukan deteksi tepi.
- fig, axs = plt.subplots(1,1, figsize=(10,10)): Membuat sebuah figur dengan satu subplot (1 baris, 1 kolom).
- ax = axs.ravel(): Mengubah array 2D axs menjadi array 1D.
- ax[0].imshow(edges,cmap = 'gray'): Menampilkan gambar hasil deteksi tepi (edges) pada subplot. cmap='gray' digunakan karena edges adalah gambar biner (hitam putih) yang menunjukkan tepi.
- ax[0].set_title("Canny Edges"): Memberikan judul "Canny Edges" pada subplot.
- plt.tight_layout(): Menyesuaikan tata letak.
- plt.show(): Menampilkan figur.

OUTPUT:



```
# Deteksi garis menggunakan Hough Transform
lines = cv2.HoughLinesP(edges,1,np.pi/100,30, maxLineGap=150)

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi: Pada baris ini, digunakan fungsi cv2.HoughLinesP() dari OpenCV untuk mendeteksi garis pada gambar yang sudah diproses dengan deteksi tepi Canny (edges).

```
# Gambar salinan dari gambar asli untuk menggambar garis
img_line = image.copy()

# Menggambar garis-garis yang terdeteksi
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(img_line, (x1, y1), (x2, y2), (100, 80, 255), 1)

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi:

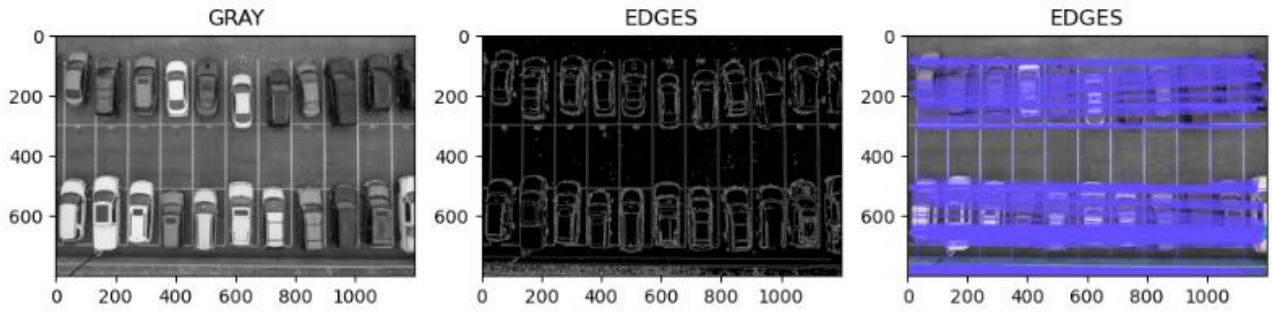
- img_line = image.copy(), gambar asli (image) disalin ke variabel img_line. Ini dilakukan agar gambar asli tidak berubah, dan salinan ini akan digunakan untuk menggambar garis yang terdeteksi.
- Lalu dilakukan iterasi pada setiap garis yang terdeteksi pada langkah sebelumnya. Setiap garis digambar di atas gambar salinan (img_line).
    - x1, y1, x2, y2 adalah koordinat titik awal dan akhir garis.
    - Fungsi cv2.line() digunakan untuk menggambar garis dengan warna (100, 80, 255) (RGB) dan ketebalan garis 1.

```
# Menampilkan gambar hasil deteksi garis
fig, axs = plt.subplots(1,3, figsize=(10,10))

ax = axs.ravel()

ax[0].imshow(img_gray,cmap='gray')
ax[0].set_title("GRAY")

ax[1].imshow(edges,cmap ='gray')
ax[1].set_title("EDGES")

ax[2].imshow(img_line,cmap ='gray')
ax[2].set_title("EDGES")

plt.tight_layout()
plt.show()

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi: Di sini, kita menggunakan Matplotlib untuk menampilkan tiga gambar dalam satu baris:

- ax[0]: Menampilkan gambar grayscale (img_gray).
- ax[1]: Menampilkan hasil deteksi tepi menggunakan Canny (edges).
- ax[2]: Menampilkan gambar dengan garis yang terdeteksi (img_line), yang telah diubah kembali ke format RGB menggunakan cv2.cvtColor() untuk ditampilkan dengan benar di Matplotlib.
- plt.tight_layout(): Fungsi ini digunakan untuk memastikan tampilan gambar tidak terpotong.

OUTPUT:



3. OPERASI GEOMETRIK

MEMBACA GAMBAR

```
img_padang = cv2.imread('gambar nasi padang.jpg')
rows, cols, _ = img_padang.shape
print('img shape: ', img_padang.shape)
# Rama Dinantiar (202331044)
```

```
img shape:  (1024, 1024, 3)
```

Tujuan dan Fungsi: Pada cell ini, gambar gambar nasi padang.jpg dibaca menggunakan cv2.imread(). Fungsi ini akan mengembalikan gambar dalam bentuk array NumPy.

- img_padang.shape digunakan untuk mengambil dimensi gambar (jumlah baris, kolom, dan saluran warna).
- Dimensi gambar dicetak dengan print() untuk memastikan ukuran gambar.
- Hasil dari img_padang.shape menunjukkan bahwa gambar tersebut memiliki dimensi 1024x1024 piksel dengan 3 saluran warna (merah, hijau, biru – RGB).

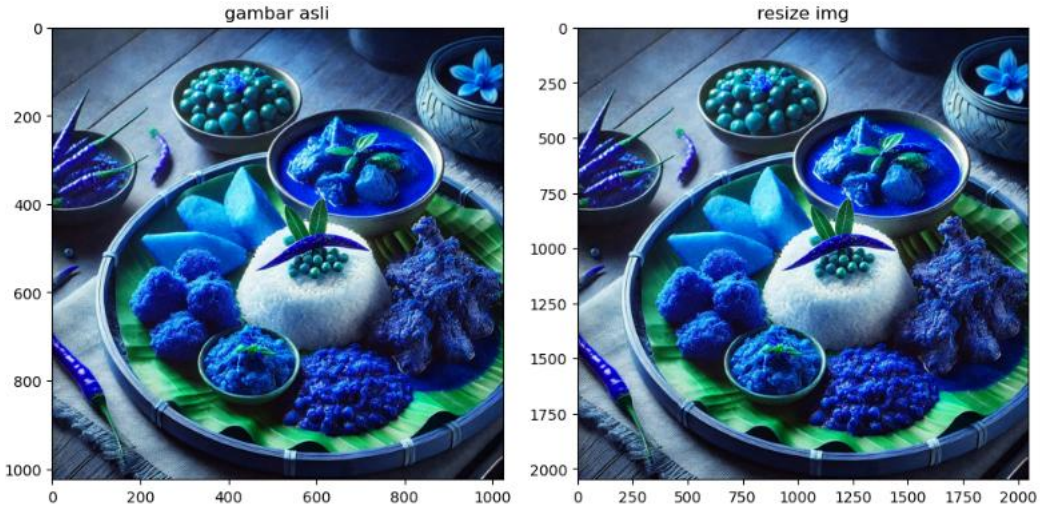
CARA 1

```
res = cv2.resize(img_padang, None, fx=2, fy=2,
                 interpolation=cv2.INTER_CUBIC)
fig, axs = plt.subplots(1,2, figsize=(10,5))
ax = axs.ravel()

ax[0].imshow(img_padang)
ax[0].set_title('gambar asli')

ax[1].imshow(res)
ax[1].set_title('resize img')

plt.tight_layout()
plt.show()

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi:

- Gambar diubah ukurannya dengan cv2.resize(). Di sini, fx=2 dan fy=2 menunjukkan bahwa gambar akan diperbesar dua kali lipat secara horizontal dan vertikal.
- Metode interpolasi yang digunakan adalah cv2.INTER_CUBIC, yang merupakan teknik interpolasi untuk menghasilkan gambar yang lebih halus saat diperbesar.
- Kedua gambar, yang asli dan yang sudah diubah ukurannya, ditampilkan berdampingan menggunakan Matplotlib (plt.subplots()), dengan masing-masing gambar diberi judul "gambar asli" dan "resize img".

OUTPUT:



CARA 2

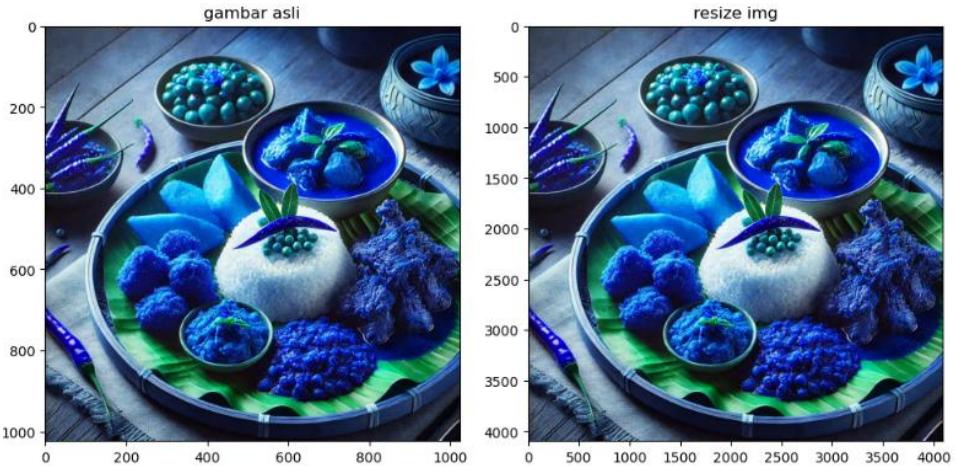
```
tinggi, lebar = img_padang.shape[:2]
res2 = cv2.resize(img_padang, (4*tinggi, 4*lebar),
                  interpolation=cv2.INTER_CUBIC)

fig, axs = plt.subplots(1,2, figsize=(10,5))
ax = axs.ravel()

ax[0].imshow(img_padang)
ax[0].set_title('gambar asli')

ax[1].imshow(res2)
ax[1].set_title('resize img')

plt.tight_layout()
plt.show()

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi:

- tinggi, lebar = img_padang.shape[:2]: Menyimpan dimensi gambar (tinggi dan lebar) pada variabel tinggi dan lebar menggunakan shape[:2], yang hanya mengambil dua dimensi pertama (baris dan kolom).
- res2 = cv2.resize(): Menggunakan fungsi cv2.resize() untuk mengubah ukuran gambar. Gambar akan diperbesar empat kali lipat pada kedua dimensi (tinggi dan lebar) dengan mengalikan tinggi dan lebar dengan 4.
- Metode interpolasi yang digunakan adalah cv2.INTER_CUBIC, yang membantu menghasilkan gambar yang lebih halus saat diperbesar.

- fig, axs = plt.subplots(1, 2, figsize=(10, 5)): Membuat dua subplot untuk menampilkan gambar asli (img_padang) di sebelah kiri dan gambar yang sudah diubah ukurannya (res2) di sebelah kanan.
- ax[0].imshow(img_padang): Menampilkan gambar asli pada subplot pertama.
- ax[1].imshow(res2): Menampilkan gambar yang sudah diubah ukurannya pada subplot kedua.
- plt.tight_layout(): Menyusun layout gambar dengan rapi agar tidak terpotong.
- plt.show(): Menampilkan hasil visualisasi gambar.

OUTPUT:



4. PENGAPLIKASIAN

PENGAPLIKASIAN

```
t = cv2.imread('plat.jpg')
plt.imshow(t)

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi:

- t = cv2.imread('plat.jpg'): Membaca gambar yang bernama 'plat.jpg' menggunakan OpenCV (cv2.imread()) dan menyimpannya dalam variabel t. Pastikan file gambar 'plat.jpg' berada di direktori yang sama dengan notebook ini atau sesuaikan path-nya.
- plt.imshow(t): Menampilkan gambar yang telah dibaca menggunakan matplotlib.pyplot.imshow(). Fungsi ini digunakan untuk menampilkan gambar yang telah dimuat oleh OpenCV di dalam notebook.

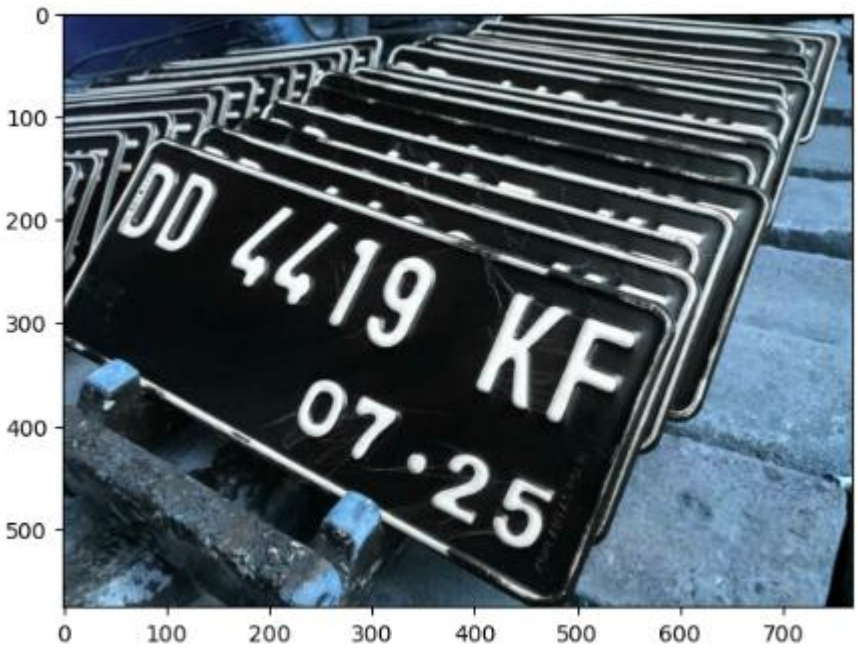
```
from skimage import transform

src = np.array([
    [0,0],
    [0,50],
    [300,50],
    [300,0]
])

crp = np.array([
    [89,137],
    [49,210],
    [525,422],
    [563,307]
])

crp2 = np.array([
    [246,346],
    [218,402],
    [456,532],
    [485,463]
])
```

Tujuan dan Fungsi:

- src adalah array yang berisi koordinat titik sumber (empat titik) dalam gambar asli, yang akan digunakan untuk mendefinisikan transformasi.
- crp dan crp2 adalah array yang berisi koordinat titik tujuan yang sesuai untuk masing-masing transformasi (proyeksi perspektif).
- crp digunakan untuk transformasi pertama, sementara crp2 digunakan untuk transformasi kedua.

OUTPUT:

```
<matplotlib.image.AxesImage at 0x267802f93a0>
```




```
tform = transform.ProjectiveTransform()
tform.estimate(src, crp)

tform2 = transform.ProjectiveTransform()
tform2.estimate(src, crp2)

warped = transform.warp(t, tform, output_shape=(50,300))
warped2 = transform.warp(t, tform2, output_shape=(50,300))

fig, axs = plt.subplots(1,3, figsize=(10,5))
ax = axs.ravel()

ax[0].imshow(warped)
ax[0].set_title('warped img 1')

ax[1].imshow(t)
ax[1].plot(crp[:,0], crp[:,1], '.r')
ax[1].plot(crp2[:,0], crp2[:,1], '.b')
ax[1].set_title('gambar ori')

ax[2].imshow(warped2)
ax[2].set_title('warped img 2')

for a in axs:
    a.axis('off')

plt.tight_layout()
plt.show()

# Rama Dinantiar (202331044)
```

Tujuan dan Fungsi:

- tform dan tform2 adalah objek dari kelas ProjectiveTransform yang digunakan untuk menghitung transformasi proyektif berdasarkan titik sumber (src) dan titik tujuan (crp dan crp2).
- tform.estimate(src, crp) dan tform2.estimate(src, crp2) menghitung parameter transformasi yang diperlukan untuk memetakan src ke crp dan crp2.

- transform.warp() digunakan untuk menerapkan transformasi yang telah dihitung (tform dan tform2) pada gambar t. Gambar t akan diubah sesuai dengan transformasi proyektif yang dihasilkan.
- output_shape=(50,300) menetapkan ukuran gambar hasil transformasi menjadi 50x300 piksel.
- Membuat tiga subplot untuk menampilkan gambar-gambar yang telah diproses:
    - Gambar 1 (warped): Gambar yang sudah ditransformasikan dengan tform (transformasi pertama).
    - Gambar 2 (t): Gambar asli dengan titik-titik crp (titik merah) dan crp2 (titik biru) yang dipetakan di atasnya.
    - Gambar 3 (warped2): Gambar yang sudah ditransformasikan dengan tform2 (transformasi kedua).
- plt.tight_layout() memastikan bahwa gambar-gambar ditata dengan rapi.
- a.axis('off') menghilangkan sumbu untuk memperjelas tampilan gambar.

OUTPUT: