LAPORAN TEORI

PENGOLAHAN CITRA DIGITAL

NAMA : Alvin Krisna Fratama

NIM : 202331238

KELAS : C

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 18

ASISTEN : 1. Abdur Rasyid Ridho

2. Rizqy Amanda

3. Kashrina Masyid Azka

4. Izzat Islami Kagapi

INSTITUT TEKNOLOGI PLN

SISTEM INFORMASI

2025

1. Perbedaan Operator Deteksi Tepi Sobel, Prewitt, dan Canny

Operator	Karakteristik	Keunggulan	Kelemahan
Sobel	Menggunakan operator gradien dengan bobot lebih besar di pusat arah horizontal dan vertikal.	Lebih sensitif terhadap perubahan tepi yang tajam; memperhalus noise secara ringan.	Tidak akurat jika terdapat noise tinggi; hanya mendeteksi tepi secara kasar.
Prewitt	Mirip Sobel, namun bobot mask-nya merata.	Lebih sederhana; efisien dalam perhitungan.	Kurang stabil dibanding Sobel dalam mendeteksi tepi diagonal.
Canny	Pendekatan multi-tahap: smoothing (Gaussian), gradien, non-maximum suppression, dan hysteresis thresholding.	Sangat akurat dan halus; mampu menangani noise; deteksi tepi yang lebih lengkap.	Kompleks dan lebih lambat; parameter (threshold) perlu disetel hati-hati.

2. Perbedaan Transformasi Translasi, Rotasi, dan Skala dalam Transformasi Geometrik

Transformasi	Deskripsi	Contoh
Translasi	Menggeser citra ke arah tertentu tanpa mengubah bentuk atau ukuran.	Menggeser gambar 30 piksel ke kanan dan 10 piksel ke bawah.
Rotasi	Memutar citra berdasarkan titik pusat tertentu (biasanya titik tengah).	Memutar gambar sebesar 45° searah jarum jam.
Skala	Mengubah ukuran (memperbesar atau memperkecil) citra.	Memperbesar gambar menjadi 150% dari ukuran aslinya.

3. Transformasi Afine dalam Transformasi Geometrik

Transformasi afine adalah jenis transformasi geometrik yang mempertahankan garis lurus dan rasio jarak paralel. Artinya, garis lurus tetap menjadi garis lurus setelah transformasi, dan garis-garis paralel tetap paralel. Namun, transformasi afine tidak harus mempertahankan sudut atau panjang. Ini adalah kombinasi dari translasi, rotasi, skala, dan shearing (pemotongan).

Contoh Aplikasi dalam Pemrosesan Citra:

1. **Koreksi perspektif (perspective correction):** digunakan untuk memperbaiki tampilan gambar yang diambil miring.
2. **Transformasi citra dalam augmentasi data**: digunakan dalam deep learning untuk menghasilkan variasi dari gambar asli.
3. **Registrasi Citra:** Menyelaraskan dua atau lebih citra yang diambil pada waktu atau dari sudut pandang yang berbeda. Misalnya, menyelaraskan citra satelit yang diambil dari berbagai orbit atau menyelaraskan citra medis (CT scan, MRI) dari pasien yang sama.

Transformasi afine digunakan untuk memetakan satu citra ke citra lainnya agar objek yang sama berada di lokasi piksel yang sama.

4. **Koreksi Distorsi Geometrik:** Mengoreksi distorsi yang terjadi pada citra akibat lensa kamera atau perspektif pengambilan gambar. Misalnya, mengoreksi citra yang diambil dari sudut miring agar terlihat seperti diambil secara tegak lurus, atau menghilangkan efek "barrel distortion" pada lensa kamera wide-angle.

4. Perbandingan Tujuan Deteksi Tepi vs Transformasi Geometrik

Aspek	**Deteksi Tepi**	**Transformasi Geometrik (Resize/Rotasi)**
Tujuan	**Menemukan struktur dan batas objek dalam citra.**	**Memanipulasi bentuk dan orientasi citra untuk tujuan visualisasi atau analisis.**
Hasil Akhir	**Gambar biner atau grayscale yang menampilkan garis batas.**	**Citra baru dengan ukuran atau orientasi berbeda.**
Fungsi Utama	**Analisis fitur citra (pengenalan objek, segmentasi).**	**Penyesuaian tampilan atau pemosisian citra.**

5. Bagaimana Proses Rotasi Citra Bekerja dari Sudut Pandang Pemetaan Piksel? Mengapa Sering Muncul Area Kosong?

Proses Rotasi Citra dari Sudut Pandang Pemetaan Piksel:Rotasi citra melibatkan pemindahan setiap piksel dari posisi aslinya ke posisi baru setelah diputar. Ada dua pendekatan utama:

1. **Transformasi Maju (Forward Mapping):**

   Setiap piksel (x,y) pada citra asli dipetakan ke posisi barunya (x′,y′) menggunakan rumus rotasi.Masalah: Seringkali, koordinat (x′,y′) yang dihitung bukan merupakan bilangan bulat. Ini berarti piksel asli akan mendarat di "antara" piksel-piksel pada grid citra tujuan. Selain itu, beberapa piksel tujuan mungkin tidak terisi (terlewatkan) karena pemetaan non-integer atau karena beberapa piksel asli memetakan ke piksel tujuan yang sama.

2. **Transformasi Balik (Backward Mapping) atau Interpolasi Balik:**

   Ini adalah metode yang lebih umum dan disukai.Untuk setiap piksel (x′,y′) pada citra tujuan yang kosong, kita menghitung koordinat piksel aslinya (x,y) menggunakan invers dari rumus rotasi.Rumus Invers Rotasi (di sekitar origin):

   $$x = x'\cos(\theta) + y'\sin(\theta)$$

   $$y = -x'\sin(\theta) + y'\cos(\theta)$$

   Interpolasi: Karena (x,y) yang dihasilkan dari rumus invers seringkali bukan bilangan bulat, kita tidak bisa langsung mengambil nilai intensitas dari piksel tersebut. Sebagai gantinya, kita menggunakan teknik interpolasi (misalnya, interpolasi tetangga terdekat, bilinear, atau bikubik) untuk memperkirakan nilai intensitas piksel (x′,y′) di citra tujuan berdasarkan nilai intensitas piksel-piksel tetangga terdekat di citra asli.

s

LAPORAN PRAKTIKUM

PENGOLAHAN CITRA DIGITAL

NAMA : Alvin Krisna Fratama

NIM : 202331238

KELAS : C

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 18

ASISTEN : 1. Abdur Rasyid Ridho

2. Rizqy Amanda

3. Kashrina Masyid Azka

4. Izzat Islami Kagapi

INSTITUT TEKNOLOGI PLN

SISTEM INFORMASI

2025

1.Metode Deteksi Tepi

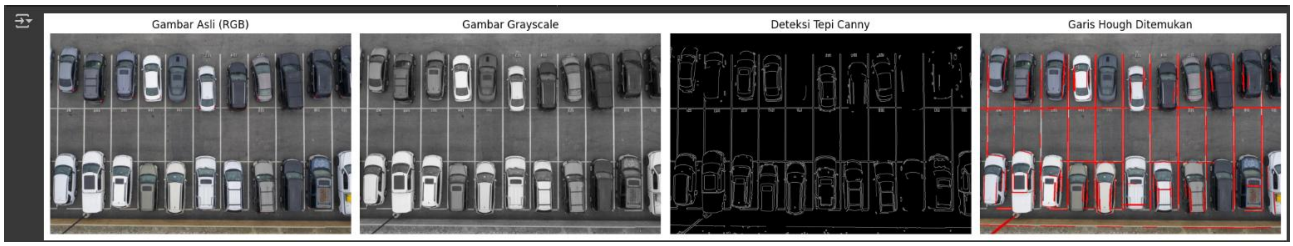
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
#202331238_Alvin Krisna F
# --- 1. Memuat Gambar ---

image_path = 'parkiran.jpg'
original_image = cv2.imread(image_path)

if original_image is None:
    print(f"Error: Gambar '{image_path}' tidak dapat dimuat. Pastikan file ada di direktori yang benar.")
else:
# --- 2. Pra-pemrosesan Gambar ---

    image_rgb = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
    gray_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)

# --- 3. Deteksi Tepi Menggunakan Metode Canny ---

    blurred_gray = cv2.GaussianBlur(gray_image, (5, 5), 0)
    # Deteksi tepi Canny
    canny_edges = cv2.Canny(blurred_gray, 100, 200) # Ambang batas disesuaikan sedikit

    # --- 4. Deteksi Garis Menggunakan Transformasi Hough (Probabilistic Hough Line Transform) ---
    hough_lines = cv2.HoughLinesP(canny_edges, 1, np.pi / 180,
                                  threshold=50,
                                  minLineLength=50,
                                  maxLineGap=10)

    # Buat salinan gambar asli untuk menggambar garis
    image_with_lines = image_rgb.copy()

    # Gambar garis-garis yang terdeteksi
```

- cv2: OpenCV, pustaka utama untuk pengolahan citra.
- numpy: Untuk operasi array dan matematika.
- matplotlib.pyplot: Untuk menampilkan grafik/gambar.
- cv2_imshow: Digunakan di Google Colab untuk menampilkan gambar dari OpenCV.
- image_path: Nama file gambar yang ingin diproses.
- cv2.imread: Membaca gambar dari file.
- if original_image is None: Mengecek apakah file berhasil dibaca.
- cv2.cvtColor: Mengubah format warna gambar.
- BGR2RGB: Untuk menampilkan gambar dengan benar di matplotlib.
- BGR2GRAY: Mengubah gambar warna menjadi hitam putih (grayscale).
- GaussianBlur: Mengurangi noise agar deteksi tepi lebih halus.
- Canny: Deteksi tepi berdasarkan perubahan intensitas piksel.100 dan 200: Ambang bawah dan atas deteksi tepi.
- cv2.HoughLinesP: Mendeteksi garis dari hasil deteksi tepi.

    - 1: Resolusi jarak (dalam piksel).
    - np.pi / 180: Resolusi sudut (dalam radian, yaitu 1°).
    - threshold=50: Minimal akumulasi agar dianggap sebagai garis.
    - minLineLength=50: Panjang minimal garis.
    - maxLineGap=10: Jarak antar titik pada satu garis.

- Menyalin gambar agar garis yang akan digambar tidak mengubah gambar asli.

```
if hough_lines is not None:
    for line in hough_lines:
        x1, y1, x2, y2 = line[0]
        cv2.line(image_with_lines, (x1, y1), (x2, y2), (255, 0, 0), 2)
else:
    print("Tidak ada garis yang terdeteksi menggunakan HoughLinesP dengan parameter yang diberikan.")
#202331238_Alvin Krisna F
# --- 5. Menampilkan Hasil ---
fig, axs = plt.subplots(1, 4, figsize=(20, 8))
axs = axs.ravel()

# Gambar Asli
axs[0].imshow(image_rgb)
axs[0].set_title('Gambar Asli (RGB)')
axs[0].axis('off') # Sembunyikan sumbu

# Gambar Grayscale
axs[1].imshow(gray_image, cmap='gray')
axs[1].set_title('Gambar Grayscale')
axs[1].axis('off')

# Hasil Deteksi Tepi Canny
axs[2].imshow(canny_edges, cmap='gray')
axs[2].set_title('Deteksi Tepi Canny')
axs[2].axis('off')

# Hasil Deteksi Garis Hough
axs[3].imshow(image_with_lines)
axs[3].set_title('Garis Hough Ditemukan')
axs[3].axis('off')

plt.tight_layout()
plt.show()

print("Pengolahan gambar berhasil diselesaikan dan ditampilkan.")
```



mplementasi pipeline pemrosesan citra untuk mendeteksi garis pada sebuah gambar menggunakan kombinasi metode **Canny Edge Detection** dan **Probabilistic Hough Transform**. Proses diawali dengan mengimpor beberapa pustaka penting seperti cv2 (OpenCV untuk pemrosesan citra), numpy (untuk operasi numerik), dan matplotlib.pyplot (untuk visualisasi). Setelah itu, gambar bernama parkiran.jpg dibaca dari direktori menggunakan cv2.imread(). Jika gambar tidak ditemukan, program akan menampilkan pesan kesalahan.

Setelah gambar berhasil dimuat, dilakukan tahap pra-pemrosesan. Gambar asli dikonversi ke format RGB untuk tampilan yang benar di matplotlib dan juga diubah ke skala abu-abu (grayscale) agar lebih mudah diproses pada tahap selanjutnya. Gambar grayscale kemudian diberikan efek blur menggunakan Gaussian Blur untuk mengurangi noise. Selanjutnya, dilakukan deteksi tepi dengan metode **Canny**, yang akan mendeteksi kontur tepi berdasarkan perubahan intensitas piksel.

Hasil deteksi tepi kemudian diproses menggunakan **Probabilistic Hough Line Transform**, yaitu teknik untuk mendeteksi garis lurus dari gambar biner (hasil Canny). Jika garis ditemukan, program akan menggambarkan garis-garis tersebut di atas salinan gambar asli dengan warna merah. Jika tidak ditemukan garis, maka akan ditampilkan pesan bahwa tidak ada garis yang terdeteksi.

Untuk menampilkan hasilnya, digunakan matplotlib untuk menyusun tampilan visual dalam empat panel: gambar asli dalam format RGB, gambar grayscale, hasil deteksi tepi menggunakan Canny, dan gambar akhir yang menunjukkan garis-garis yang terdeteksi. Masing-masing panel diberi judul dan sumbu koordinat disembunyikan untuk tampilan yang lebih bersih. Proses diakhiri dengan mencetak pesan bahwa pengolahan gambar telah berhasil diselesaikan dan ditampilkan ke layar. Kode ini sangat berguna untuk berbagai aplikasi komputer visi seperti deteksi marka jalan, parkir otomatis, atau sistem navigasi visual.

2.Resize

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
from skimage import io, transform

makanan_sulawesi_path = '/content/Makanan _Sulawesi.jpg'

# --- Bagian 1: Pengolahan Gambar Makanan Sulawesi (Resize) ---
#202331238_Alvin Krisna F

img_makanan_sulawesi = cv2.imread(makanan_sulawesi_path)


if img_makanan_sulawesi is None:
    print(f"Error: Gambar '{makanan_sulawesi_path}' tidak dapat dimuat. Pastikan file ada di direktori yang benar.")
else:

    img_makanan_sulawesi_rgb = cv2.cvtColor(img_makanan_sulawesi, cv2.COLOR_BGR2RGB)


    rows_makanan, cols_makanan = img_makanan_sulawesi.shape[:2]
    print(f'Dimensi gambar {makanan_sulawesi_path}: {img_makanan_sulawesi.shape}')

    # Resize Cara 1: Menggunakan faktor skala (fx, fy)
    resized_fx_fy = cv2.resize(img_makanan_sulawesi, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)
    resized_fx_fy_rgb = cv2.cvtColor(resized_fx_fy, cv2.COLOR_BGR2RGB)


    # Resize Cara 2: Menggunakan ukuran absolut
    new_width_abs = cols_makanan * 4
    new_height_abs = rows_makanan * 4
    resized_abs_size = cv2.resize(img_makanan_sulawesi, (new_width_abs, new_height_abs), interpolation=cv2.INTER_CUBIC)
    resized_abs_size_rgb = cv2.cvtColor(resized_abs_size, cv2.COLOR_BGR2RGB)
```

Untuk Melakukan resize (pengubahan ukuran) terhadap gambar makanan khas Sulawesi. Langkah pertama adalah melakukan **import pustaka** yang dibutuhkan seperti cv2 untuk pemrosesan citra, numpy untuk operasi numerik, matplotlib.pyplot untuk menampilkan gambar, dan skimage (khususnya modul io dan transform) yang juga sering digunakan dalam pemrosesan gambar. Gambar yang digunakan memiliki path './content/Makanan_Sulawesi.jpg'.

Proses selanjutnya dimulai dengan membaca gambar menggunakan cv2.imread(). Jika gambar gagal dimuat, maka akan muncul pesan kesalahan. Jika berhasil, gambar dikonversi dari format BGR ke RGB menggunakan cv2.cvtColor() agar warna tampil sesuai saat divisualisasikan. Setelah

itu, kode mengambil dimensi gambar (jumlah baris dan kolom piksel) dan mencetaknya ke konsol sebagai informasi.

Kemudian dilakukan dua metode **resize** atau pengubahan ukuran gambar:

1. **Metode pertama menggunakan faktor skala fx dan fy**, masing-masing bernilai 2. Ini berarti gambar akan diperbesar 2 kali lipat baik secara horizontal maupun vertikal menggunakan interpolasi bicubic (cv2.INTER_CUBIC) yang menghasilkan kualitas tinggi saat memperbesar gambar.
2. **Metode kedua menggunakan ukuran absolut**, di mana lebar dan tinggi baru dihitung dengan mengalikan dimensi asli sebanyak 4 kali. Resize dilakukan menggunakan fungsi cv2.resize() dengan parameter ukuran baru secara eksplisit dan juga menggunakan interpolasi bicubic.

Setelah proses resize dilakukan pada masing-masing metode, hasilnya kembali dikonversi ke format RGB agar nantinya bisa ditampilkan dengan benar menggunakan matplotlib atau cv2_imshow()

```
# Tampilkan hasil resize
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
axs[0].imshow(img_makanan_sulawesi_rgb)
axs[0].set_title('Gambar Asli')
axs[0].axis('off')

axs[1].imshow(resized_fx_fy_rgb)
axs[1].set_title('Resize (Faktor Skala 2x)')
axs[1].axis('off')
#202331238_Alvin Krisna F
axs[2].imshow(resized_abs_size_rgb)
axs[2].set_title(f'Resize (Ukuran Absolut {new_width_abs}x{new_height_abs})')
axs[2].axis('off')

plt.tight_layout()
plt.show()

# --- Bagian 2: Pengolahan Gambar Makanan Sulawesi (Rotasi dengan OpenCV) ---

rows_rot, cols_rot = img_makanan_sulawesi.shape[:2] # Menggunakan dimensi asli
print(f'Dimensi gambar untuk rotasi: {img_makanan_sulawesi.shape}')

# Rotasi Cara 1: Menggunakan OpenCV (Rotasi 90 derajat)
# Pusat rotasi di tengah gambar
M_rotate_cv = cv2.getRotationMatrix2D(((cols_rot-1)/2, (rows_rot-1)/2), 90, 1)
rotated_cv_image = cv2.warpAffine(img_makanan_sulawesi_rgb, M_rotate_cv, (cols_rot, rows_rot))

# Tampilkan hasil rotasi OpenCV
fig, axs = plt.subplots(1, 2, figsize=(12, 6))
axs[0].imshow(img_makanan_sulawesi_rgb)
axs[0].set_title('Gambar Asli')
axs[0].axis('off')

axs[1].imshow(rotated_cv_image)
axs[1].set_title('Gambar Diputar 90 Derajat (OpenCV)')
axs[1].axis('off')
```

Baris fig, axs = plt.subplots(1, 3, figsize=(18, 6)) membuat sebuah figure dan satu baris berisi tiga subplot (area gambar) dengan ukuran total 18x6 inci. Kemudian, baris axs[0].imshow(img_makanan_sulawesi_rgb) menampilkan gambar asli (kemungkinan bernama img_makanan_sulawesi_rgb) di subplot pertama. Setelah itu, axs[0].set_title('Gambar Asli') memberikan judul "Gambar Asli" pada subplot tersebut, dan axs[0].axis('off') menyembunyikan sumbu-sumbu koordinat agar gambar terlihat lebih bersih. Proses serupa diulang untuk subplot kedua: axs[1].imshow(resized_fx_fy_rgb) menampilkan gambar yang sudah diubah ukurannya berdasarkan faktor (resized_fx_fy_rgb), axs[1].set_title('Resize (Faktor Skala 2x)') memberikan judul "Resize (Faktor Skala 2x)", dan axs[1].axis('off') menghilangkan sumbunya. Subplot ketiga juga mengikuti pola yang sama: axs[2].imshow(resized_abs_size_rgb) menampilkan gambar yang diubah ukurannya ke dimensi absolut tertentu (resized_abs_size_rgb), axs[2].set_title(f'Resize (Ukuran Absolut {new_width_abs}x{new_height_abs})') memberikan judul yang menyertakan dimensi baru, dan axs[2].axis('off') menyembunyikan sumbunya. Setelah semua gambar ditampilkan, plt.tight_layout() secara otomatis menyesuaikan tata letak subplot agar tidak tumpang tindih, dan plt.show() menampilkan jendela berisi ketiga gambar tersebut.

Selanjutnya, kode masuk ke bagian pengolahan gambar untuk rotasi. Baris rows_rot, cols_rot = img_makanan_sulawesi.shape[:2] mengambil dimensi tinggi (rows_rot) dan lebar (cols_rot) dari gambar asli untuk digunakan dalam operasi rotasi. Kemudian, print(f'Dimensi gambar untuk rotasi: {img_makanan_sulawesi.shape}') mencetak dimensi tersebut ke konsol untuk informasi. Untuk melakukan rotasi 90 derajat, M_rotate_cv = cv2.getRotationMatrix2D(((cols_rot-1)/2, (rows_rot-1)/2), 90, 1) menghitung matriks transformasi rotasi. Parameter pertama adalah pusat rotasi (tengah gambar), parameter kedua adalah sudut rotasi 90 derajat, dan parameter ketiga adalah skala (1 berarti tidak ada perubahan skala). Setelah mendapatkan matriks rotasi, rotated_cv_image = cv2.warpAffine(img_makanan_sulawesi_rgb, M_rotate_cv, (cols_rot, rows_rot)) menerapkan transformasi rotasi ke gambar asli menggunakan matriks yang sudah dihitung, menghasilkan gambar yang telah diputar. Terakhir, untuk menampilkan hasil rotasi, fig, axs = plt.subplots(1, 2, figsize=(12, 6)) membuat figure baru dengan satu baris dan dua subplot. axs[0].imshow(img_makanan_sulawesi_rgb) menampilkan kembali gambar asli di subplot pertama dengan judul "Gambar Asli" dan sumbu dimatikan, sementara axs[1].imshow(rotated_cv_image) menampilkan gambar yang telah diputar di subplot kedua dengan judul "Gambar Diputar 90 Derajat (OpenCV)" dan sumbu juga dimatikan, sebelum akhirnya plt.show() menampilkan jendela dengan kedua gambar ini.

```
    plt.tight_layout()
    plt.show()

    # --- Bagian 3: Pengolahan Gambar Makanan Sulawesi (Rotasi dengan scikit-image) ---
    img_makanan_skimage = io.imread(makanan_sulawesi_path)

    # Rotasi Cara 2: Menggunakan scikit-image (Rotasi 45 derajat)
    # resize=False: Ukuran gambar keluaran sama dengan input, bagian yang diputar akan terpotong
    rotated_skimage_no_resize = transform.rotate(img_makanan_skimage, 45, resize=False)
    # resize=True: Ukuran gambar keluaran disesuaikan agar seluruh gambar hasil rotasi muat
    rotated_skimage_with_resize = transform.rotate(img_makanan_skimage, 45, resize=True)

    # Tampilkan hasil rotasi scikit-image
    fig, axs = plt.subplots(1, 3, figsize=(18, 6))
    axs[0].imshow(img_makanan_skimage)
    axs[0].set_title('Gambar Asli (skimage)')
    axs[0].axis('off')

    axs[1].imshow(rotated_skimage_no_resize)
    axs[1].set_title('Diputar 45 Derajat (Tanpa Resize)')
    axs[1].axis('off')
    #202331238_Alvin Krisna F
    axs[2].imshow(rotated_skimage_with_resize)
    axs[2].set_title('Diputar 45 Derajat (Dengan Resize)')
    axs[2].axis('off')

    plt.tight_layout()
    plt.show()

print("Semua proses pengolahan gambar (resize dan rotasi) telah diselesaikan dengan gambar Makanan_Sulawesi.jpg.")
```

pada proses rotasi gambar menggunakan pustaka scikit-image (skimage). Pertama, gambar dimuat dari path yang ditentukan (makanan_sulawesi_path) ke variabel img_makanan_skimage menggunakan io.imread().

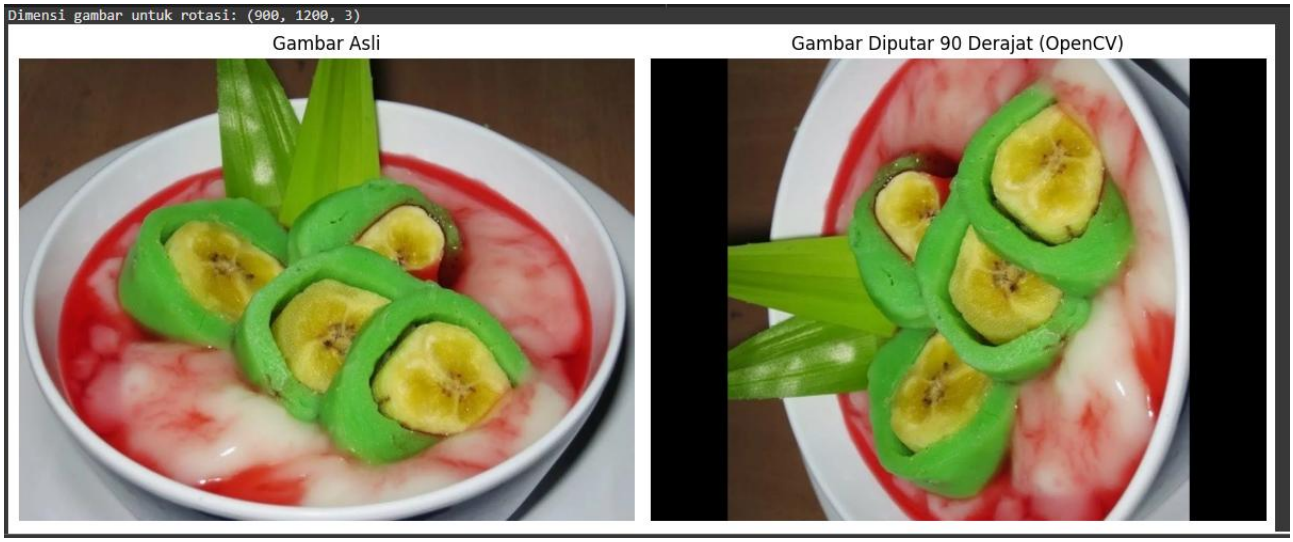
Selanjutnya, ada dua cara rotasi yang ditunjukkan:

1. **Rotasi Tanpa Penyesuaian Ukuran (resize=False)**: Baris rotated_skimage_no_resize = transform.rotate(img_makanan_skimage, 45, resize=False) melakukan rotasi gambar img_makanan_skimage sebesar 45 derajat. Parameter resize=False berarti ukuran gambar output akan tetap sama dengan gambar input, yang dapat menyebabkan bagian gambar terpotong jika setelah rotasi gambar melebihi batas dimensi awal.
2. **Rotasi Dengan Penyesuaian Ukuran (resize=True)**: Baris rotated_skimage_with_resize = transform.rotate(img_makanan_skimage, 45, resize=True) juga memutar gambar 45 derajat. Namun, parameter resize=True akan secara otomatis menyesuaikan dimensi (tinggi dan lebar) gambar output agar seluruh bagian gambar hasil rotasi dapat termuat tanpa terpotong.

Setelah proses rotasi, kode menampilkan ketiga gambar tersebut secara berdampingan:

- axs[0] menampilkan gambar asli (img_makanan_skimage) dengan judul "Gambar Asli (skimage)".
- axs[1] menampilkan gambar yang diputar 45 derajat tanpa penyesuaian ukuran (rotated_skimage_no_resize) dengan judul "Diputar 45 Derajat (Tanpa Resize)".
- axs[2] menampilkan gambar yang diputar 45 derajat dengan penyesuaian ukuran (rotated_skimage_with_resize) dengan judul "Diputar 45 Derajat (Dengan Resize)".

Setiap subplot diatur untuk tidak menampilkan sumbu (axis('off')) agar tampilan lebih bersih, dan plt.tight_layout() serta plt.show() digunakan untuk mengatur tata letak dan menampilkan gambar. Terakhir, print("Semua proses pengolahan gambar (resize dan rotasi) telah diselesaikan dengan Gambar Makanan_Sulawesi.jpg.") mencetak pesan konfirmasi bahwa semua proses pengolahan gambar telah selesai.



3.Olah Gambar Plat Nomor

```
t = cv2.imread('/content/plat.jpg')
t_gray = cv2.cvtColor(t, cv2.COLOR_BGR2GRAY)
fig, axs = plt.subplots(1, 2, figsize=(10, 10))
ax = axs.ravel()
ax[0].imshow(t)
ax[0].set_title('Original Image')
ax[1].imshow(t_gray, cmap='gray')
ax[1].set_title('Grayscale Image')
src = np.array([[0, 0], [0, 50], [300, 50], [300, 0]])
crp = np.array([[212, 493], [567, 303], [535, 256], [535, 256]])
crp2 = np.array([[165, 682], [199, 722], [529, 469], [496, 432]])
#202331238_Alvin Krisna F
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, 10))
ax = axs.ravel()
ax[0].imshow(warped)
ax[0].set_title('Warped Image 1')
ax[1].imshow(t)
ax[1].plot(crp[:, 0], crp[:, 1], '.r')
ax[1].plot(src[:, 0], src[:, 1], '.b')
ax[1].set_title('Original Image with Points')

ax[2].imshow(warped2)
ax[2].set_title('Warped Image 2')
for a in axs:
    a.axis('off')

plt.tight_layout()
plt.show()
```

Pertama, gambar bernama "plat.jpg" dimuat ke dalam variabel t menggunakan cv2.imread(). Kemudian, gambar berwarna ini diubah menjadi citra keabuan (grayscale) dan disimpan dalam variabel t_gray menggunakan cv2.cvtColor(t, cv2.COLOR_BGR2GRAY).

Selanjutnya, kode membuat dan menampilkan dua subplot. Subplot pertama (axs[0]) menampilkan gambar asli (t) dengan judul "Original Image". Subplot kedua (axs[1]) menampilkan gambar grayscale (t_gray) dengan peta warna 'gray' dan judul "Grayscale Image". Kedua sumbu subplot dimatikan untuk tampilan yang bersih.

Bagian inti dari kode ini adalah demonstrasi transformasi perspektif. Empat set titik koordinat didefinisikan dalam array NumPy:

- src: Ini adalah titik-titik sumber yang mendefinisikan wilayah yang akan ditransformasi.
- crp dan crp2: Ini adalah titik-titik tujuan yang berbeda, yang akan digunakan untuk mendefinisikan bagaimana wilayah dari src akan dipetakan setelah transformasi.

Dua objek ProjectiveTransform dibuat: tform dan tform2.

- tform.estimate(src, crp) menghitung matriks transformasi perspektif yang memetakan titik-titik dari src ke crp.
- tform2.estimate(src, crp2) melakukan hal yang sama, tetapi memetakan src ke crp2.

Kemudian, gambar asli (t) di-warp (diterapkan transformasi perspektif) menggunakan matriks transformasi yang telah dihitung:

- warped = transform.warp(t, tform, output_shape=(50, 300)) menerapkan tform ke gambar t. output_shape=(50, 300) menentukan dimensi tinggi dan lebar gambar hasil warping.
- warped2 = transform.warp(t, tform2, output_shape=(50, 300)) melakukan hal yang sama dengan tform2.

Terakhir, kode menampilkan tiga subplot baru.

- axs[0] menampilkan warped dengan judul "Warped Image 1".
- axs[1] menampilkan gambar asli (t) dan juga menandai titik-titik crp (dengan warna merah '.') dan src (dengan warna biru 'b') di atasnya untuk visualisasi bagaimana transformasi didefinisikan. Judul subplot ini adalah "Original Image with Points".
- axs[2] menampilkan warped2 dengan judul "Warped Image 2".

Sebuah loop for a in axs: a.axis('off') digunakan untuk mematikan sumbu pada semua subplot yang baru dibuat. plt.tight_layout() menyesuaikan tata letak subplot, dan plt.show() menampilkan jendela berisi semua gambar dan plot yang dihasilkan.