LAPORAN TEORI

PENGOLAHAN CITRA DIGITAL

NAMA : Moh Fahrul Gibran

NIM : 202331004

KELAS : C

DOSEN : Ir.Darma Rusjdi, M.Kom

NO.PC : 7

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, serta Keunggulan dan Kelemahan Masing-masing.

Operator Sobel dan Prewitt adalah metode deteksi tepi berbasis gradien yang relatif sederhana. Sobel menggunakan dua kernel 3x3 untuk menghitung gradien horizontal dan vertikal, memberikan bobot lebih pada piksel pusat, yang membuatnya cukup baik dalam mendeteksi tepi dan sensitif terhadap orientasi tepi, namun rentan terhadap noise dan menghasilkan tepi yang agak tebal. Prewitt serupa dengan Sobel tetapi semua piksel dalam kernel memiliki bobot yang sama, sehingga lebih sensitif terhadap noise dan menghasilkan tepi yang kurang akurat, meskipun komputasinya sederhana. Berbeda dengan keduanya, Operator Canny adalah metode multi-tahap yang lebih canggih, melibatkan penghalusan Gaussian, perhitungan gradien, penekanan non-maksimal, dan histeresis ambang batas. Keunggulan Canny terletak pada kemampuannya mendeteksi tepi yang kuat, tipis, dan kontinyu dengan sangat efektif serta ketahanannya terhadap noise, menjadikannya "gold standard" dalam deteksi tepi; namun, kompleksitasnya memerlukan komputasi yang lebih tinggi.

2. Perbedaan antara Transformasi Translasi, Rotasi, dan Skala dalam Transformasi Geometrik Citra, Beserta Contohnya.

Transformasi geometrik citra melibatkan perubahan posisi spasial piksel. Translasi, atau pergeseran, adalah proses memindahkan setiap piksel citra sejauh jarak tertentu dalam arah horizontal dan/atau vertikal tanpa mengubah orientasi atau ukurannya, contohnya memindahkan ikon dari pojok layar ke tengah. Rotasi adalah operasi memutar citra pada sudut tertentu di sekitar titik pusat, menjaga ukurannya tetap, seperti memutar foto 90 derajat. Sementara itu, transformasi skala adalah proses mengubah ukuran citra, baik dengan memperbesar atau memperkecilnya secara seragam atau tidak seragam, misalnya memperkecil gambar resolusi tinggi menjadi thumbnail.

3. Pengertian Transformasi Affine dalam Transformasi Geometrik dan Contoh Aplikasinya.

Transformasi affine adalah jenis transformasi geometrik yang mempertahankan sifat "kelurusan" dan "kesejajaran" garis, yang berarti garis lurus akan tetap lurus dan garis paralel akan tetap paralel setelah transformasi. Transformasi ini dapat berupa kombinasi translasi, rotasi, skala, dan geser (shear), dan dapat direpresentasikan dalam bentuk matriks. Dalam aplikasi pemrosesan citra, transformasi affine sangat berguna untuk koreksi distorsi geometrik, misalnya mengoreksi kemiringan gambar yang diambil dari sudut tertentu agar objek terlihat tegak lurus. Selain itu, transformasi affine juga esensial dalam pendaftaran citra (image registration), di mana ia digunakan untuk menyelaraskan beberapa citra agar objek yang sama berada pada posisi piksel yang identik, penting dalam proses penggabungan citra atau perbandingan citra dari waktu ke waktu.

4. Perbandingan Tujuan Utama Deteksi Tepi dan Transformasi Geometrik (Resize & Rotasi) dan Mengapa Hasilnya Berbeda.

Meskipun deteksi tepi dan transformasi geometrik seperti *resize* dan rotasi sama-sama memanipulasi piksel, tujuan utama dan hasil akhirnya sangat berbeda. Tujuan utama deteksi tepi adalah mengidentifikasi dan menyoroti perubahan intensitas signifikan yang menandakan batas objek atau fitur dalam citra, menghasilkan representasi abstrak dari struktur citra yang berfokus pada ekstraksi fitur. Sebaliknya, tujuan utama transformasi geometrik adalah mengubah posisi spasial dan/atau ukuran piksel secara keseluruhan, mengubah presentasi visual citra (misalnya, memperbesar untuk detail atau memutar untuk orientasi), sembari mempertahankan konten visual dasarnya. Perbedaan ini muncul karena deteksi tepi fokus pada informasi lokal dan perubahan intensitas untuk segmentasi, mengubah jenis informasi yang disajikan. Sementara itu, transformasi geometrik fokus pada posisi spasial piksel dan integritas visual global, mengubah cara informasi disajikan tanpa mengubah jenisnya, dengan prioritas pada kualitas visual citra.

5. Proses Rotasi Citra dari Sudut Pandang Pemetaan Piksel dan Penyebab Munculnya Area Kosong.

Proses rotasi citra melibatkan dua langkah utama dari sudut pandang pemetaan piksel. Pertama, transformasi koordinat dilakukan di mana untuk setiap piksel di citra output yang dirotasi, sistem akan mencari koordinat piksel *asli* di citra input menggunakan transformasi rotasi terbalik. Kedua, interpolasi intensitas diperlukan karena koordinat piksel asli yang ditemukan seringkali bukan bilangan bulat, sehingga nilai intensitas diperkirakan dari piksel-piksel tetangga terdekat di citra input menggunakan teknik seperti interpolasi bilinear. Area kosong yang seringkali berwarna hitam di sudut-sudut citra setelah rotasi dengan sudut selain kelipatan 90° atau 180° muncul karena kanvas citra output harus diperluas untuk menampung seluruh citra yang dirotasi. Akibatnya, ada piksel-piksel di kanvas output yang, setelah transformasi balik, tidak memiliki korespondensi yang valid di dalam batas citra input asli. Ketika sistem tidak menemukan data piksel yang valid untuk area ini, piksel-piksel tersebut diisi dengan nilai default 0, yang secara visual direpresentasikan sebagai warna hitam.

LAPORAN PRAKTIKUM

PENGOLAHAN CITRA DIGITAL

NAMA : Moh Fahrul Gibran

NIM : 202331004

KELAS : C

DOSEN : Ir.Darma Rusjdi, M.Kom

NO.PC : 7

ASISTEN : 1. Abdur Rasyid Ridho

2. Rizqy Amanda

3. Kashrina Masyid Azka

4. Izzat Islami Kagapi

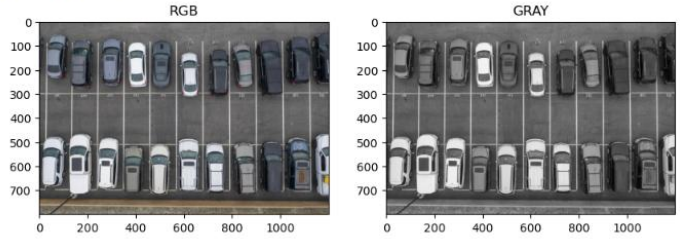
INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1.

```
[2]: #202331004 Moh Fahrul Gibran
import cv2
import numpy as np
import matplotlib.pyplot as plt

[6]: #202331004 Moh Fahrul Gibran
image = cv2.imread('parkiran.jpg')
image.shape

[6]: (799, 1200, 3)

[5]: #202331004 Moh Fahrul Gibran
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
fig, axs = plt.subplots(1,2, figsize=(10,10))

ax = axs.ravel()

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

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

[5]: Text(0.5, 1.0, 'GRAY')
```

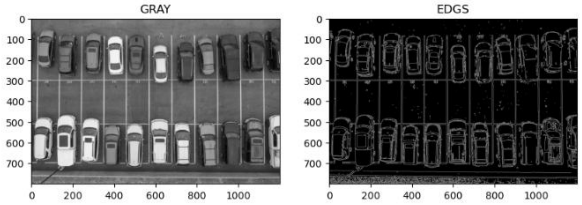


Pada bagian awal, kode mengimpor beberapa pustaka penting: cv2 (OpenCV) untuk operasi citra, numpy untuk komputasi numerik, dan matplotlib.pyplot untuk visualisasi. Selanjutnya, sebuah citra bernama 'parkiran.jpg' dimuat ke dalam program, dan dimensinya (799 baris, 1200 kolom, dan 3 kanal warna) ditampilkan, menunjukkan bahwa ini adalah citra berwarna.

Inti dari proses ini adalah konversi citra dari format warna (RGB) ke citra skala abu-abu (GRAY). Kode tersebut menggunakan fungsi cv2.cvtColor() untuk melakukan konversi ini. Setelah konversi, kedua citra, baik yang asli dalam format RGB maupun yang telah dikonversi ke skala abu-abu, ditampilkan secara berdampingan menggunakan matplotlib. Visualisasi akhir menunjukkan dua subplot: satu menampilkan citra asli 'parkiran.jpg' dalam warna RGB, dan yang lainnya menampilkan citra yang sama namun dalam skala abu-abu, menyoroti perbedaan antara representasi warna dan monokromatik dari citra yang sama.

2.

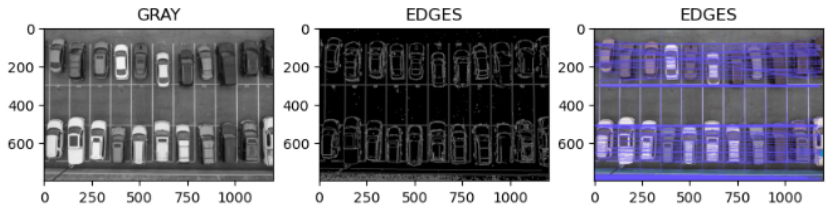
```
[7]: #202331004 Moh Fahrul Gibran
edges = cv2.Canny(image, 100, 150)

[8]: #202331004 Moh Fahrul Gibran
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("EDGS")

[8]: Text(0.5, 1.0, 'EDGS')
```



kode Python yang ditampilkan pertama-tama menerapkan algoritma Canny Edge Detection pada citra skala abu-abu (img_gray) yang kemungkinan besar dihasilkan dari proses sebelumnya. Fungsi cv2.Canny(image, 100, 150) digunakan untuk ini, di mana 100 dan 150 adalah nilai ambang batas (threshold) rendah dan tinggi untuk deteksi tepi. Hasil dari deteksi tepi ini disimpan dalam variabel edges.
Selanjutnya, kode tersebut menampilkan dua citra secara berdampingan untuk perbandingan. Subplot pertama (ax[0]) menampilkan citra asli dalam skala abu-abu (img_gray), dengan judul "GRAY". Subplot kedua (ax[1]) menampilkan hasil deteksi tepi (edges), dengan judul "EDGS". Kedua citra ini ditampilkan menggunakan matplotlib.pyplot.imshow dengan colormap gray. Visualisasi akhir menunjukkan bahwa algoritma Canny berhasil mengidentifikasi dan menyorot kontur serta batas-batas objek dalam citra parkiran, mengubah citra parkiran skala abu-abu menjadi citra biner yang hanya menampilkan tepi-tepi objek.

3.

```
[11]: #202331004 Moh Fahrul Gibran
lines = cv2.HoughLinesP(edges,1,np.pi/180,90, maxLineGap=150)
img_line = image.copy()

[12]: #202331004 Moh Fahrul Gibran
for line in lines :
        x1,y1,x2,y2 = line[0]
        cv2.line(img_line, (x1,y1), (x2,y2), (100,80,255),1)

[13]: #202331004 Moh Fahrul Gibran
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")

[13]: Text(0.5, 1.0, 'EDGES')
```



deteksi garis menggunakan Transformasi Hough Probabilistik. Pada bagian awal kode, fungsi cv2.HoughLinesP() diterapkan pada citra hasil deteksi tepi (edges). Parameter 1, np.pi/180, 90, dan maxLineGap=150 mengontrol sensitivitas dan kriteria deteksi garis. 1 menunjukkan resolusi jarak dalam piksel, np.pi/180 adalah resolusi sudut dalam radian, 90 adalah ambang batas minimum jumlah persimpangan untuk menganggapnya sebagai garis, dan maxLineGap=150 memungkinkan celah maksimum yang diizinkan antara segmen garis untuk menganggapnya sebagai satu garis tunggal. Hasil deteksi ini disimpan dalam variabel lines.
Selanjutnya, kode melakukan iterasi melalui setiap garis yang terdeteksi. Untuk setiap garis, koordinat titik awal (x1,y1) dan titik akhir (x2,y2) diekstrak. Kemudian, garis-garis ini digambar pada salinan citra asli (img_line) menggunakan fungsi cv2.line(), dengan warna biru (direpresentasikan oleh (100, 80, 255)) dan ketebalan 1 piksel.
Akhirnya, ada tiga subplot yang ditampilkan secara berdampingan:

1. GRAY: Menampilkan citra parkiran dalam skala abu-abu.
2. EDGES: Menampilkan citra hasil deteksi tepi Canny.
3. EDGES (dengan garis): Menampilkan citra hasil deteksi tepi yang sama, tetapi dengan garis-garis biru yang terdeteksi oleh Transformasi HoughProbabilistik ditambahkan di atasnya.
   Visualisasi ini secara jelas menunjukkan bagaimana algoritma berhasil mengidentifikasi dan menandai garis-garis lurus pada citra, seperti garis parkir atau batas-batas kendaraan, yang sangat berguna dalam aplikasi seperti analisis tata letak parkir atau penghitungan kendaraan.

4.

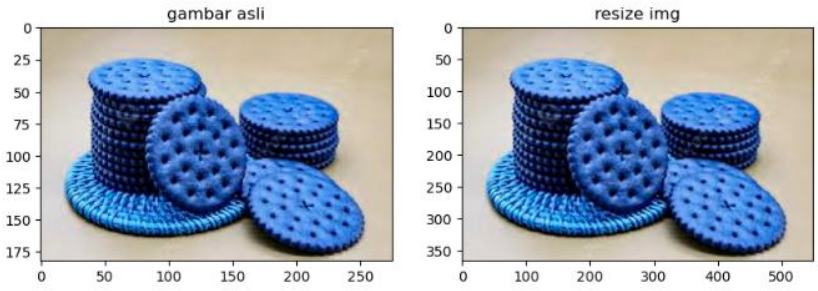
```
[15]: #202331004 Moh Fahrul Gibran
img_jet = cv2.imread('gambar makanan.png')
rows, cols, _=img_jet.shape
print('img shape: ', img_jet.shape)
```

```
img shape:  (183, 275, 3)
```

Cara 1

```
[16]: #202331004 Moh Fahrul Gibran
res = cv2.resize(img_jet, 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_jet)
ax[0].set_title('gambar asli')

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

```
[16]: Text(0.5, 1.0, 'resize img')
```



Pada bagian awal kode, sebuah citra bernama 'gambar makanan.png' dimuat ke dalam variabel img_jet. Informasi dimensi citra (183 baris, 275 kolom, dan 3 kanal warna) kemudian dicetak, menunjukkan bahwa ini adalah citra berwarna.

Selanjutnya, di bawah judul "Cara 1", kode melakukan operasi *resizing*. Fungsi cv2.resize() digunakan untuk mengubah ukuran img_jet. Parameter fx=2 dan fy=2 menunjukkan bahwa citra akan diperbesar dua kali lipat baik secara horizontal (fx) maupun vertikal (fy). Metode interpolasi yang digunakan adalah cv2.INTER_CUBIC, yang umumnya memberikan kualitas hasil yang lebih baik untuk pembesaran dibandingkan metode lain seperti INTER_LINEAR atau INTER_NEAREST.

Hasil dari operasi *resizing* ini kemudian ditampilkan berdampingan dengan citra asli. Subplot pertama (ax[0]) menampilkan "gambar asli" (citra img_jet), sedangkan subplot kedua (ax[1]) menampilkan "resize img" (citra hasil perubahan ukuran res). Visualisasi akhir secara jelas menunjukkan bahwa citra "gambar asli" yang lebih kecil diperbesar menjadi "resize img" yang ukurannya dua kali lipat lebih besar, mempertahankan detail visualnya dengan cukup baik berkat interpolasi kubik

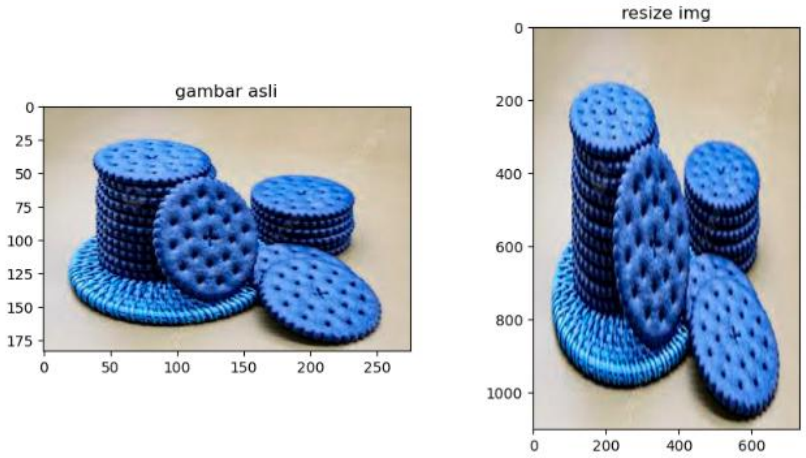
.

5.

Cara 2

```
[17]: #202331004 Moh Fahrul Gibran
tinggi, lebar = img_jet.shape[:2]
res2 = cv2.resize(img_jet, (4*tinggi, 4*lebar),
                  interpolation=cv2.INTER_CUBIC)
fig,axs = plt.subplots(1,2, figsize=(10,5))
ax = axs.ravel()

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

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

```
[17]: Text(0.5, 1.0, 'resize img')
```



pembahasan mengenai perubahan ukuran citra (resizing) menggunakan OpenCV, kali ini dengan "Cara 2" yang sedikit berbeda dalam menentukan dimensi target. Pada bagian awal kode, tinggi dan lebar citra asli (img_jet) diekstrak dari img_jet.shape[:2].

Kemudian, fungsi cv2.resize() digunakan untuk mengubah ukuran citra. Berbeda dengan cara sebelumnya yang menggunakan faktor skala (fx, fy), pada "Cara 2" ini, dimensi target secara eksplisit ditentukan sebagai (4*lebar, 4*tinggi). Ini berarti citra akan diperbesar empat kali lipat dari ukuran aslinya. Metode interpolasi yang digunakan masih cv2.INTER_CUBIC, yang umumnya menghasilkan kualitas yang baik untuk pembesaran citra. Hasil dari operasi *resizing* ini disimpan dalam variabel res2.

Seperti pada contoh sebelumnya, dua subplot ditampilkan untuk perbandingan. Subplot pertama (ax[0]) menampilkan "gambar asli" (img_jet), sedangkan subplot kedua (ax[1]) menampilkan "resize img" (res2). Visualisasi akhir secara jelas menunjukkan bahwa citra asli diperbesar secara signifikan (empat kali lipat) menjadi citra yang jauh lebih besar, menggambarkan bagaimana pengguna dapat secara langsung mengontrol dimensi output dari operasi *resizing* di OpenCV.

6.

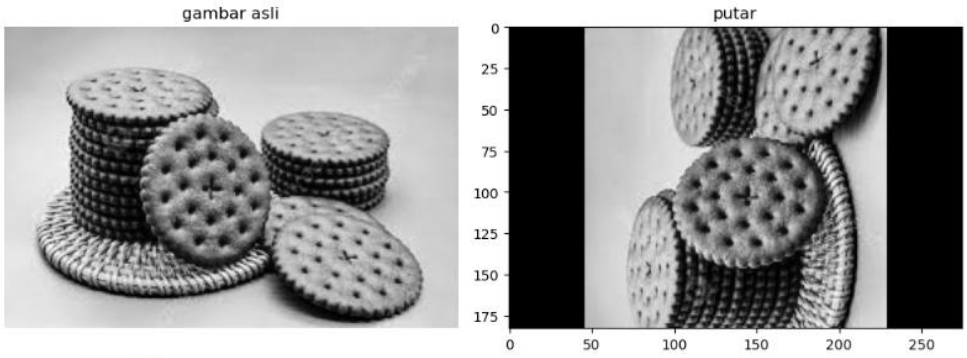
perputaran citra / rotasi gambar

```
[19]: #202331004 Moh Fahrul Gibran
img_ajg = cv2.imread('gambar makanan.png',0)
rows, cols=img_ajg.shape
print('img shape: ', img_ajg.shape)
```

```
img shape:  (183, 275)
```

Cara 1

```
[20]: #202331004 Moh Fahrul Gibran
M = cv2.getRotationMatrix2D(((cols-1)/2.0, (rows-1)/2.0),90,1)

img_putar = cv2.warpAffine(img_ajg, M, (cols, rows))

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

ax[0].imshow(img_ajg, cmap='gray')
ax[0].set_title('gambar asli')

ax[1].imshow(img_putar, cmap='gray')
ax[1].set_title('putar')

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

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




```
<Figure size 640x480 with 0 Axes>
```

Gambar di atas mendemonstrasikan proses rotasi atau perputaran citra menggunakan pustaka OpenCV di Python. Pada bagian awal kode, sebuah citra bernama 'gambar makanan.png' dimuat ke dalam variabel img_ajg. Perlu diperhatikan bahwa citra dimuat sebagai citra grayscale (parameter 0 pada cv2.imread()). Dimensi citra (183 baris, 275 kolom) kemudian dicetak.

Di bawah judul "Cara 1", proses rotasi dimulai. Pertama, sebuah matriks rotasi (M) dibuat menggunakan fungsi cv2.getRotationMatrix2D(). Parameter-parameter yang digunakan adalah:

- ((cols-1)/2.0, (rows-1)/2.0): Ini menentukan titik pusat rotasi, yaitu bagian tengah citra.
- 90: Sudut rotasi dalam derajat. Dalam kasus ini, citra akan diputar 90 derajat searah jarum jam.
- 1: Faktor skala, yang berarti citra tidak akan diperbesar atau diperkecil selama rotasi.

Setelah matriks rotasi diperoleh, fungsi cv2.warpAffine() digunakan untuk menerapkan transformasi rotasi pada citra asli (img_ajg) menggunakan matriks M. Hasil citra yang telah diputar disimpan dalam variabel img_putar.

Terakhir, dua subplot ditampilkan berdampingan:

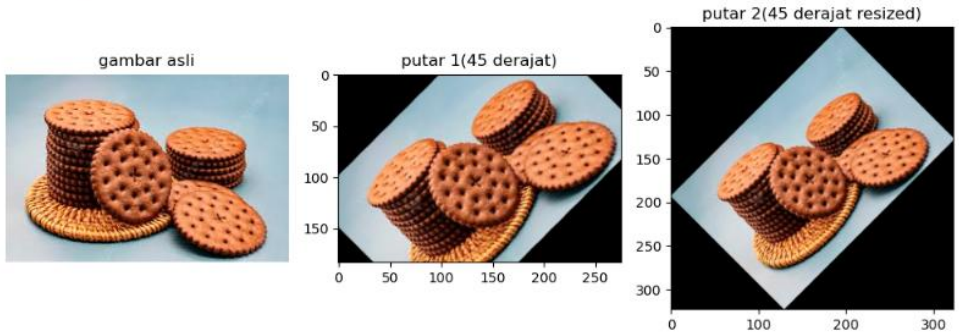
1. gambar asli: Menampilkan citra makanan asli dalam skala abu-abu.
2. putar: Menampilkan citra makanan yang telah diputar 90 derajat.

Visualisasi akhir secara jelas menunjukkan bahwa citra "gambar asli" telah dirotasi 90 derajat searah jarum jam, menghasilkan citra "putar" dengan orientasi yang berbeda. Area hitam di sekeliling citra yang diputar menunjukkan bagian yang tidak terisi karena rotasi.

7.

Cara 2

```
[22]: #202331004 Moh Fahrul Gibran
from skimage import io, transform

img_ajg2 = io.imread('gambar makanan.png')

rotated = transform.rotate(img_ajg2, 45, resize=False)
rotated2 = transform.rotate(img_ajg2, 45, resize=True)

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

ax[0].imshow(img_ajg2, cmap='gray')
ax[0].set_title('gambar asli')

ax[1].imshow(rotated, cmap='gray')
ax[1].set_title('putar 1(45 derajat)')

ax[2].imshow(rotated2, cmap='gray')
ax[2].set_title('putar 2(45 derajat resized)')

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

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

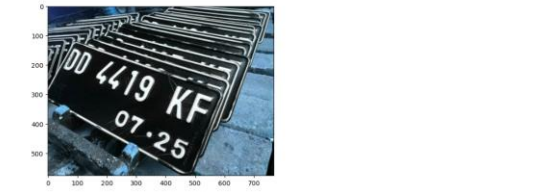



```
<Figure size 640x480 with 0 Axes>
<Figure size 640x480 with 0 Axes>
```

menunjukkan bagaimana sebuah foto makanan diputar menggunakan kode Python. Pertama, foto asli dimuat. Lalu, foto itu diputar 45 derajat dalam dua cara berbeda: satu yang ukurannya tetap sama (menyisakan area kosong), dan satu lagi yang ukurannya menyesuaikan agar seluruh foto yang diputar tetap terlihat penuh

8.

Pengaplikasian



Gambar di atas menunjukkan tahap awal dari sebuah "Pengaplikasian" (Application) dalam pengolahan citra. Pada bagian kode, sebuah citra bernama 'plat.jpg' dimuat menggunakan cv2.imread(), yang mengindikasikan bahwa ini adalah citra plat nomor kendaraan. Variabel output_shape = (80, 400) didefinisikan, meskipun dalam tangkapan layar ini belum digunakan untuk transformasi apa pun pada citra yang dimuat. Kemudian, citra yang telah

dimuat (t) langsung ditampilkan menggunakan plt.imshow(t). Hasil visualisasinya menunjukkan sebuah gambar plat nomor kendaraan berwarna hitam dengan tulisan putih yang jelas, lengkap dengan kode plat, angka, dan tanggal masa berlaku, yang menjadi subjek utama untuk pengolahan lebih lanjut dalam aplikasi ini.

9.

```
[51]: from skimage import io, transform
src = np.array([
    [0,0],
    [0,50],
    [300,50],
    [300,0]
])

crp = np.array([
    [92,136],
    [47,211],
    [524,419],
    [576,309]
])


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))
axs = axs.ravel()

axs[0].imshow(warped)
axs[0].set_title("Warped IMG 1")

axs[1].imshow(t)
axs[1].plot(crp[:,0], crp[:,1], '.r')
axs[1].plot(crp2[:,0], crp[:,1], '.b')
axs[1].set_title("Gambar Ori")


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

plt.tight_layout()
plt.show
[51]: <function matplotlib.pyplot.show(close=None, block=None)>
```

Kode tersebut bertujuan untuk mengisolasi dan meluruskan objek plat nomor dari sebuah gambar yang diambil dari sudut miring, sebuah proses yang dikenal sebagai **koreksi perspektif** atau *perspective warp*. Proses ini menggunakan pustaka scikit-image, khususnya modul transform. Langkah pertama adalah mendefinisikan dua set koordinat. Array src mendefinisikan sudut-sudut dari sebuah persegi panjang target yang ideal (sebagai "cetakan" output berukuran 300x50 piksel), sementara array crp berisi koordinat empat sudut yang bersesuaian dari plat nomor pada gambar asli yang miring.
Selanjutnya, program menghitung matriks transformasi yang diperlukan. Dengan menggunakan transform.ProjectiveTransform(), sebuah objek transformasi dibuat. Fungsi tform.estimate(src, crp) kemudian menghitung "peta" matematis yang dapat memetakan setiap titik dari cetakan persegi (src) ke posisi miringnya di gambar asli (crp). Peta transformasi inilah yang menjadi kunci untuk meluruskan gambar.
Langkah terakhir adalah menerapkan transformasi dan visualisasi. Fungsi transform.warp() mengambil gambar asli, menerapkan peta transformasi yang telah dihitung, dan "menggambar ulang" piksel-piksel dari plat nomor yang miring ke dalam sebuah gambar baru sesuai bentuk output_shape yang lurus. Hasilnya, seperti yang terlihat pada gambar "Warped IMG 1", adalah plat nomor yang tampak lurus seolah-olah difoto dari depan. Kode tersebut juga menggunakan matplotlib untuk menampilkan hasil akhir ini berdampingan dengan gambar original yang telah ditandai dengan titik-titik merah (crp) untuk menunjukkan sudut mana saja yang digunakan sebagai referensi.