LAPORAN TEORI

PENGOLAHAN CITRA DIGITAL

NAMA : Fawwaz Murfid Muttaqin

NIM 202331107

KELAS : B

DOSEN : Ir. Darma Rusjdi, M.kom

NO.PC : 22

ASISTEN : 1. Davina Najwa Ermawan

2. Fakhrul Fauzi Nugraha Tarigan

3. Viana Salsabila Fairuz Syahla

4. Muhammad Hanief Febriansyah

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1. Transformasi Afine

Transformasi Afine adalah jenis transformasi geometrik yang mempertahankan garis lurus dan hubungan paralel antar garis, namun tidak mempertahankan panjang atau sudut. Contohnya meliputi translasi (pergeseran), rotasi, skala, dan shearing. Secara matematis, dinyatakan dalam bentuk matriks:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

2. **Deteksi Tepi vs Transformasi Geometrik**

Tujuan:

- Deteksi tepi bertujuan mengekstrak kontur atau batas objek.
- Transformasi geometrik bertujuan mengubah posisi, ukuran, atau orientasi gambar.

Cara Kerja:

- Deteksi tepi memakai filter seperti Sobel, Canny untuk mendeteksi perubahan intensitas piksel.
- Transformasi geometrik menggunakan matriks (Affine atau rotasi) untuk mengubah koordinat piksel.

Hasil:

- Deteksi tepi menghasilkan gambar hitam-putih dengan garis kontur.
- Transformasi menghasilkan gambar yang ukurannya berubah (resize) atau diputar (rotasi).

3. Canny vs HoughLinesP

- Canny: Deteksi tepi berbasis gradasi intensitas. Hasilnya berupa garis tepi tipis dan halus.
- HoughLinesP: Digunakan untuk mendeteksi garis lurus dari hasil deteksi tepi (misalnya dari Canny). Mengembalikan koordinat awal dan akhir garis. Perbedaan utama: Canny menghasilkan tepi, HoughLinesP menghasilkan garis lurus eksplisit.

4. Interpolasi Cubic & Perbedaan fx/fy

Interpolasi cubic menggunakan polinomial derajat 3 untuk memperhalus citra saat resize. Lebih halus dari nearest-neighbor dan bilinear.

- Jika menggunakan fx dan fy, berarti memperbesar/mengecilkan gambar berdasarkan faktor skala (contoh: fx=2 artinya 2x lebih besar).
- Jika menggunakan (width, height), maka ukuran absolut yang diinginkan ditentukan langsung.

5. Rotasi warpAffine dan Mencegah Pemotongan

warpAffine digunakan untuk menerapkan rotasi menggunakan matriks transformasi afine. Untuk mencegah gambar terpotong saat diputar:

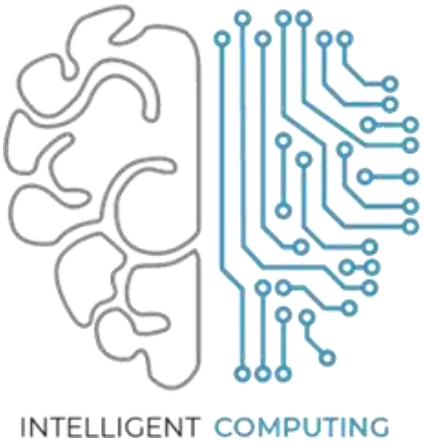
- Hitung bounding box hasil rotasi.
- Gunakan padding atau ubah ukuran canvas agar mencakup seluruh hasil rotasi.

Contoh:

```
M = cv2.getRotationMatrix2D(center, angle, scale)
rotated = cv2.warpAffine(image, M, (new_width, new_height))
```

LAPORAN PRAKTIKUM

PENGOLAHAN CITRA DIGITAL



NAMA : Fawwaz Murfid Muttaqin

NIM 202331107

KELAS : B

DOSEN : Ir. Darma Rusjdi, M.kom

NO.PC : 22

ASISTEN : 1. Davina Najwa Ermawan

2. Fakhrul Fauzi Nugraha Tarigan

3. Viana Salsabila Fairuz Syahla

4. Muhammad Hanief Febriansyah

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1. Import Library

```
[1]: ### Fawwaz Murfid Muttaqin 202331107

[1]: import numpy as np
     import matplotlib.pyplot as plt
     import cv2 as cv
```

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

- NumPy untuk manipulasi array.
- Matplotlib untuk plotting gambar.
- OpenCV (cv2) digunakan untuk pemrosesan citra.

2. Membaca Gambar

```
[2]: image = cv.imread("kijang.jpeg")
```

image = cv.imread("kijang.jpeg")

- Membaca file gambar kijang.jpeg.
- Gambar dibaca dalam format BGR (Blue, Green, Red).

3. Menampilkan Gambar

```
[3]: cv.imshow("Gambar Kijang", image)
     cv.waitKey(0)
     cv.destroyAllWindows()
```

cv.imshow("Gambar Kijang", image)
cv.waitKey(0)
cv.destroyAllWindows()

- Menampilkan gambar asli dalam jendela baru.
- cv.waitKey(0) menunggu tombol ditekan sebelum melanjutkan.
- cv.destroyAllWindows() menutup jendela tampilan.

4. Konversi ke Grayscale & Deteksi Tepi (Canny)

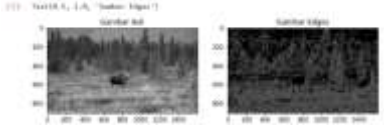
```
[4]: gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
     edges = cv.Canny(image, 75,150)

     cv.imshow("Gambar Kijang", gray)
     cv.waitKey(0)
     cv.destroyAllWindows()
```

gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
edges = cv.Canny(image, 75, 150)

- Gambar diubah ke grayscale agar proses deteksi tepi lebih efisien.
- cv.Canny() digunakan untuk deteksi tepi menggunakan threshold 75 dan 150.

5. Menampilkan Gambar Asli dan Tepi dengan Matplotlib



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

axs[0].imshow(gray, cmap="gray")
axs[0].set_title("Gambar Asli")

axs[1].imshow(edges, cmap="gray")
axs[1].set_title("Gambar Edges")

- Menampilkan dua subplot: satu gambar grayscale, satu gambar hasil deteksi tepi.

7. Menggambar Garis pada Gambar

```
lines = cv.HoughLinesP(edges,1,np.pi/180,30,maxLineGap=250)
image_line = image.copy()
```

```
for line in lines:
        x1,y1,x2,y2 = line [0]
        cv.line(image_line, (x1,y1),(x2,y2),(100,8,255),1)
```

for line in lines:

x1, y1, x2, y2 = line[0]

cv.line(image_line, (x1, y1), (x2, y2), (100, 8, 255), 1)

- Loop semua garis yang terdeteksi dan menggambarkannya dengan warna magenta (100, 8, 255) dan ketebalan 1.

8. Menampilkan Hasil Deteksi Garis

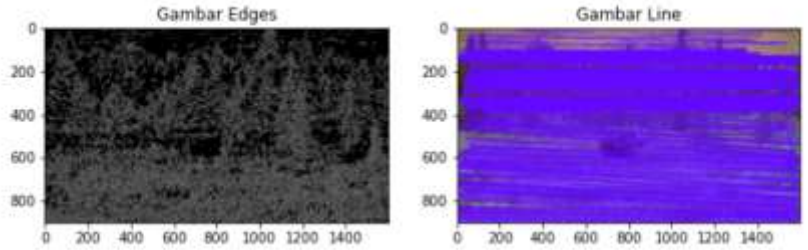
```
### Fawwaz Murfid Muttaqin 202331107
```

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

axs[0].imshow(edges, cmap="gray")
axs[0].set_title("Gambar Edges")

axs[1].imshow(image_line, cmap="gray")
axs[1].set_title("Gambar Line")
```

```
Text(0.5, 1.0, 'Gambar Line')
```



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

axs = axs.ravel()

axs[0].imshow(edges, cmap="gray")

axs[0].set_title("Gambar Edges")

axs[1].imshow(image_line, cmap="gray")

axs[1].set_title("Gambar Line")

- Menampilkan perbandingan antara hasil tepi dan hasil garis.

1. Import Library

```
[1]: ### Fawwaz Murfid Muttaqin

[1]: import numpy as np
     import matplotlib.pyplot as plt
     import cv2
```

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

- Import standar untuk manipulasi array, plotting, dan pemrosesan citra.

2. Membaca dan Menampilkan Gambar Grayscale

```
[2]: img_jalan = cv2.imread('jalan.jpeg', 0)
     rows, cols = img_jalan.shape
     print("IMG SHAPE: ", img_jalan.shape)

     IMG SHAPE:  (450, 678)
```

img_jalan = cv2.imread('jalan.jpeg', 0)
rows, cols = img_jalan.shape
print("IMG SHAPE: ", img_jalan.shape)

- Membaca gambar jalan.jpeg dalam mode grayscale (0).
- Variabel rows dan cols menyimpan dimensi gambar.

3. Rotasi Gambar dengan OpenCV

```
[8]: M = cv2.getRotationMatrix2D(((cols-1)/2.0, (rows-1)/2.0), 40,1)
     img_putar = cv2.warpAffine(img_jalan, M, (cols,rows))
```

M = cv2.getRotationMatrix2D(((cols-1)/2.0, (rows-1)/2.0), 40, 1)
img_putar = cv2.warpAffine(img_jalan, M, (cols, rows))

- Membuat matriks rotasi 2D untuk rotasi gambar sebesar 40 derajat di pusat gambar.
- warpAffine menerapkan rotasi tersebut.

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

axs[0].imshow(img_jalan, cmap='gray')
axs[0].set_title("Gambar Asli")

axs[1].imshow(img_putar, cmap='gray')
axs[1].set_title("Gambar Putar")

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

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

fig, axs = plt.subplots(1,2,figsize=(10,5))
...
axs[1].imshow(img_putar, cmap='gray')

- Menampilkan gambar asli dan hasil rotasi.

4. Rotasi dengan Skimage

```
[3]: ### Fawwaz Murfid Muttaqin 202331107

[4]: from skimage import io, transform

     img_jalan2 = io.imread('jalan.jpeg')

     rotated = transform.rotate(img_jalan2, -12, resize=False)
     rotated2 = transform.rotate(img_jalan2, -12, resize=True)
```

from skimage import io, transform

img_jalan2 = io.imread('jalan.jpeg')

rotated = transform.rotate(img_jalan2, -12, resize=False)
rotated2 = transform.rotate(img_jalan2, -12, resize=True)

- Membaca gambar dengan skimage.
- Melakukan rotasi -12 derajat:
  - resize=False: mempertahankan ukuran asli → kemungkinan bagian gambar terpotong.
  - resize=True: menyesuaikan ukuran agar seluruh gambar tetap terlihat.

5. Menampilkan Rotasi Gambar

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

axs[0].imshow(img_jalan2)
axs[0].set_title("Gambar Asli")

axs[1].imshow(rotated)
axs[1].set_title("Gambar Putar 1")

axs[2].imshow(rotated2)
axs[2].set_title("Gambar Putar 2")

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

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

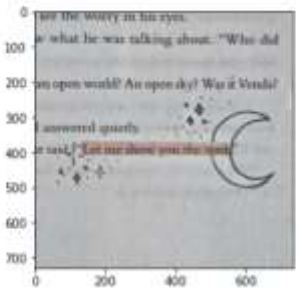
fig, axs = plt.subplots(1,3,figsize=(10,5))
axs[0].imshow(img_jalan2) # Asli
axs[1].imshow(rotated) # Rotasi (tanpa resize)
axs[2].imshow(rotated2) # Rotasi (dengan resize)

- Menampilkan perbandingan rotasi dengan dan tanpa penyesuaian ukuran.

6. Membaca Gambar untuk Transformasi Proyektif

```
[5]: t = cv2.imread('text.jpg')
     plt.imshow(t)
```

```
[5]: <matplotlib.image.AxesImage at 0x2d819905bb0>
```



t = cv2.imread('text.jpg')
plt.imshow(t)

- Membaca gambar text.jpg untuk digunakan pada transformasi perspektif.

7. Transformasi Proyeksi (Perspective Transform)

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

     crp = np.array ([
         [126,367],
         [129,409],
         [554,414],
         [559,371]
     ])
```

src = np.array([[0,0], [0,50], [300,50], [300,0]])
crp = np.array([[126,367], [129,409], [554,414], [559,371]])

- src: koordinat tujuan (output datar).
- crp: koordinat aktual dari teks miring di gambar input.

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

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

tform = transform.ProjectiveTransform()
tform.estimate(src, crp)
warped = transform.warp(t, tform, output_shape=(50,300))

- Mengestimasi dan menerapkan transformasi proyeksi untuk membuat tampilan seolah-olah teks telah diluruskan.

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

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

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

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

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

fig, axs = plt.subplots(1,3,figsize=(10,5))
axs[0].imshow(t)
axs[0].plot(crp[:,0], crp[:,1], '.r') # Titik sumber (merah)
axs[1].imshow(warped)

- Menampilkan gambar sebelum dan sesudah transformasi perspektif.