TUGAS AKHIR COURSE MANDIRI - IMAGE PROCESSING

Disusun Oleh:

> Puteri Amelia Azli (Institut Teknologi Padang)

MSIB Kampus Merdeka Batch 6 - Bisa AI Academy

2 Mei 2024

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
```

```
# Membaca Gambar
img = cv2.imread('ty.jpg', 0)
rows, cols = img.shape
```

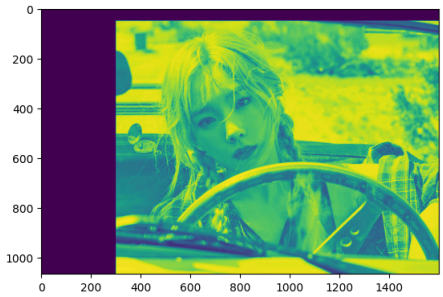
Teknik Geometric Transformation

Meliputi:

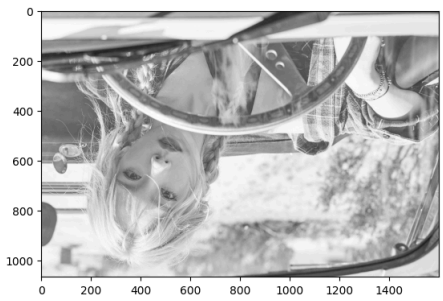
- Cropping
- Translation
- Reflection
- Rotation
- Scaling
- Shearing in X-Axis
- Shearing in Y-Axis

```
# Cropping
cropped_img = img[0:800, 100:1200]
cv.imwrite('cropped_out.jpg', cropped_img)
img2 = cv2.imread('cropped_out.jpg', 0)
plt.imshow(img2, cmap='gray')
cv.waitKey(0)
cv.destroyAllWindows()
```

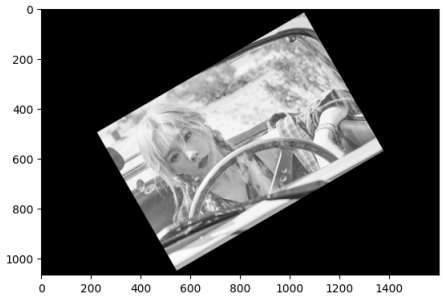

```
# Translation
M = np.float32([[1, 0, 300], [0, 1, 50]])
dst = cv.warpAffine(img, M, (cols, rows))
plt.imshow(dst)
cv.waitKey(0)
cv.destroyAllWindows()
```




```
# Reflection
M = np.float32([[1, 0, 0],
                [0, -1, rows],
                [0, 0, 1]])
reflected_img = cv.warpPerspective(img, M,
                                   (int(cols),
                                    int(rows)))
plt.imshow(reflected_img, cmap='gray')
cv.imwrite('reflection_out.jpg', reflected_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

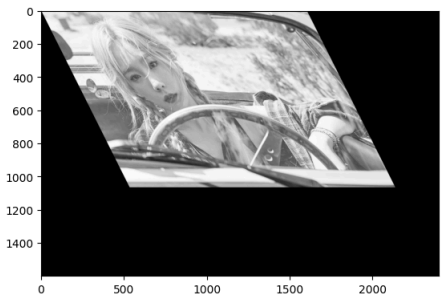


```
# Rotation
M = np.float32([[1, 0, 0], [0, -1, rows], [0, 0, 1]])
img_rotation = cv.warpAffine(img,
                             cv.getRotationMatrix2D((cols/2, rows/2),
                                                    30, 0.6),
                             (cols, rows))
plt.imshow(img_rotation, cmap='gray')
cv.imwrite('rotation_out.jpg', img_rotation)
cv.waitKey(0)
cv.destroyAllWindows()
```

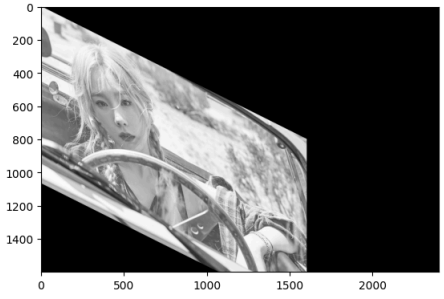



```
# Scaling
img_shrinked = cv.resize(img, (250, 200),
                         interpolation=cv.INTER_AREA)
plt.imshow(img_shrinked, cmap='gray')
img_enlarged = cv.resize(img_shrinked, None,
                         fx=1.5, fy=1.5,
                         interpolation=cv.INTER_CUBIC)
plt.imshow(img_enlarged, cmap='gray')
cv.waitKey(0)
cv.destroyAllWindows()
```


```
# Shearing in X-Axis
M = np.float32([[1, 0.5, 0], [0, 1, 0], [0, 0, 1]])
sheared_img = cv.warpPerspective(img, M, (int(cols*1.5), int(rows*1.5)))
plt.imshow(sheared_img, cmap='gray')
cv.waitKey(0)
cv.destroyAllWindows()
```




```
# Shearing in Y-Axis
M = np.float32([[1, 0, 0], [0.5, 1, 0], [0, 0, 1]])
sheared_img = cv.warpPerspective(img, M, (int(cols*1.5), int(rows*1.5)))
plt.imshow(sheared_img, cmap='gray')
cv.waitKey(0)
cv.destroyAllWindows()
```



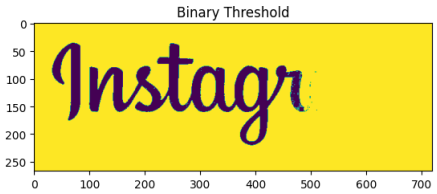
Teknik Thresholding

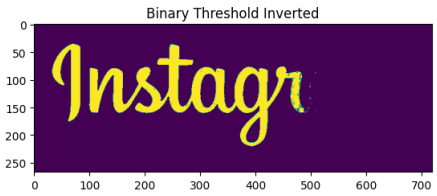
Meliputi:

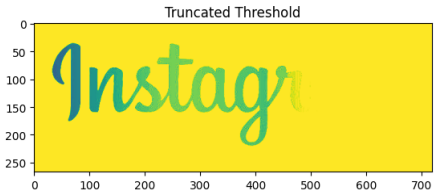
- Set-1 (Simple Thresholding)
- Set-2 (Adaptive Thresholding)
- Set-3 (Otsu Thresholding)

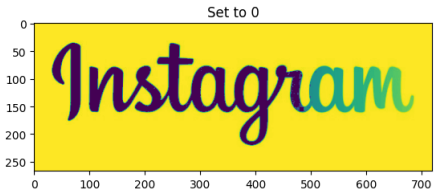
```
image1 = cv2.imread('ig.jpeg')
gambar_rgb = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
plt.imshow(gambar_rgb)
plt.axis('off')
plt.show()
```

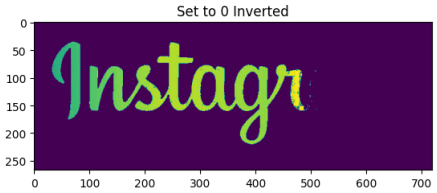
```
# Set-1 (Simple Thresholding)
# Python program to illustrate
# simple thresholding type on an image

# cv2.cvtColor is applied over the
# image input with applied parameters
# to convert the image in grayscale
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

# applying different thresholding
# techniques on the input image
# all pixels value above 120 will
# be set to 255
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 120, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO_INV)

# the window showing output images
# with the corresponding thresholding
# techniques applied to the input images
plt.imshow(thresh1, cmap=None)
plt.title('Binary Threshold')
plt.show()
plt.imshow(thresh2, cmap=None)
plt.title('Binary Threshold Inverted')
plt.show()
plt.imshow(thresh3, cmap=None)
plt.title('Truncated Threshold')
plt.show()
plt.imshow(thresh4, cmap=None)
plt.title('Set to 0')
plt.show()
plt.imshow(thresh5, cmap=None)
plt.title('Set to 0 Inverted')
plt.show()

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```



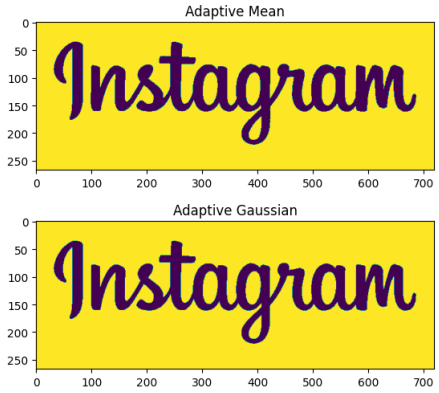








```
# Set-2 (Adaptive Thresholding)

# cv2.cvtColor is applied over the
# image input with applied parameters
# to convert the image in grayscale
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

# applying different thresholding
# techniques on the input image
thresh1 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                        cv2.THRESH_BINARY, 199, 5)

thresh2 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                        cv2.THRESH_BINARY, 199, 5)

# the window showing output images
# with the corresponding thresholding
# techniques applied to the input image
plt.imshow(thresh1, cmap=None)
plt.title('Adaptive Mean')
plt.show()
plt.imshow(thresh2, cmap=None)
plt.title('Adaptive Gaussian')
plt.show()


# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```




```
# Set-3 (Otsu Thresholding)
# Python program to illustrate
# Otsu thresholding type on an image

# cv2.cvtColor is applied over the
# image input with applied parameters
# to convert the image in grayscale
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

# applying Otsu thresholding
# as an extra flag in binary
# thresholding
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY +
                                        cv2.THRESH_OTSU)

# the window showing output image
# with the corresponding thresholding
# techniques applied to the input image
plt.imshow(thresh1)
plt.title('Otsu Threshold')

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```