

Machine Problem No. 2: Applying Image Processing Techniques

Step 1: Install OpenCV

```
!pip install opencv-python-headless
```

```
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-packages (4.10.0.84)  
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python-headless) (1.26.4)
```

Step 2: Import Necessary Libraries

```
import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
from google.colab import files  
from io import BytesIO  
from PIL import Image  
  
# Function to display an image using matplotlib  
def display_image(img, title="Image"):  
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  
    plt.title(title)  
    plt.axis('off')  
    plt.show()  
  
# Function to display two images side by side  
def display_images(img1, img2, title1="Image 1", title2="Image 2"):  
    plt.subplot(1, 2, 1)  
    plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))  
    plt.title(title1)  
    plt.axis('off')  
  
    plt.subplot(1, 2, 2)  
    plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))  
    plt.title(title2)  
    plt.axis('off')  
  
    plt.show()
```

Step 3: Load an Image

```
# Upload an image  
uploaded = files.upload()  
  
# Convert to OpenCV format  
image_path = next(iter(uploaded)) # Get the image file name  
image = Image.open(BytesIO(uploaded[image_path]))  
image = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)  
  
display_image(image, "Original Image")
```

```
Choose Files | cat.jpg  
* cat.jpg(image/jpeg) - 40670 bytes, last modified: 9/19/2024 - 100% done  
Saving cat.jpg to cat.jpg
```



Implementing Image Transformations and Filtering:

Scaling and Rotation

```
# Scaling  
def scale_image(img, scale_factor):  
    height, width = img.shape[:2]  
    scaled_img = cv2.resize(img,  
(int(width * scale_factor), int(height * scale_factor)), interpolation=cv2.INTER_LINEAR)  
    return scaled_img  
  
# Rotate  
def rotate_image(img, angle):  
    height, width = img.shape[:2]  
    center = (width // 2, height // 2)  
    matrix = cv2.getRotationMatrix2D(center, angle, 1.0)  
    rotated_img = cv2.warpAffine(img, matrix, (width, height))  
    return rotated_img  
  
# Scale image by 0.5  
scaled_image = scale_image(image, 0.5)  
display_image(scaled_image, "Scaled Image (50%)")
```

```
# Rotate image by 45 degrees
rotated_image = rotate_image(image, 45)
display_image(rotated_image, "Rotated Image (45°)")
```



Scaled Image (50%)



Rotated Image (45°)



Blurring Techniques

```
# Gaussian Blur
gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)
display_image(gaussian_blur, "Gaussian Blur (5x5)")
```

```
# Median Blur
median_blur = cv2.medianBlur(image, 5)
display_image(median_blur, "Median Blur (5x5)")
```



Gaussian Blur (5x5)



Median Blur (5x5)



Edge Detection using Canny

```
# Canny Edge Detection
edges = cv2.Canny(image, 100, 200)
display_image(edges, "Canny Edge Detection (100, 200)")
```



Canny Edge Detection (100, 200)

