Photoshop ก็อปเกรด A

โดย

65010409	ธนธัส	พินธุ	กลุ่ม Algebra555
65010902	รชา	ตั้งตระกูล	กลุ่ม Algebra555
65010966	วลัญชน์	กิจจารักษ์	กลุ่ม Jojo Ara
65010983	วัศพล	สัมฤทธิ์	กลุ่ม Jojo Ara

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา
รายวิชา 01076032 ELEMENTARY DIFFERENTIAL EQUATIONS AND LINEAR ALGEBRA
ปีการศึกษา 2566

สารบัญ

เรื่อง **หน้า**

บทที่ 1

บทนำ

1.1 ที่มาของโครงงาน

เนื่องจากการทำงานด้านเอกสารหรือการนำเสนอในปัจจุบัน
มีความจำเป็นอย่างยิ่งในการใช้รูปสำหรับประกอบเนื้อหาเพื่อให้ผู้อ่านสามารถเข้าใจได้มากขึ้น
ซึ่งรูปที่มีนั้นอาจไม่เหมาะสมกับการนำมาประกอบเนื้อหา
จึงต้องมีการสร้างโปรแกรมสำหรับตัดแต่งรูปภาพให้ตรงตามความต้องการของผู้ใช้

1.2 จุดประสงค์โครงงาน

1.2.1 สร้างโปรแกรมที่สามารถปรับแต่งรูปภาพได้

1.2.2 สามารถปรับเปลี่ยนสีของบริเวณที่ต้องการของรูปภาพได้

1.2.3 สามารถตรวจจับเส้นขอบ และเปลี่ยนขอบของรูปภาพได้

1.2.3 สามารถตกแต่งเอฟเฟกต์ของแต่ละรูปภาพได้

1.2.4 สามารถนำความรู้ทางด้านคณิตศาสตร์เรื่อง เมตริกซ์รวมถึงเวกเตอร์
มาประยุกต์ใช้ในการเปลี่ยนแปลงเอฟเฟกต์ต่างๆ หรือการทำงานเกี่ยวกับรูปภาพได้

บทที่ 2

ขั้นตอนการทำงาน

2.1 การเตรียมข้อมูล

รวบรวมข้อมูลรูปภาพจากแหล่งรูปภาพจากเว็บไซต์ต่างๆ ดังนี้

https://www.kaggle.com/datasets/pavansanagapati/images-dataset
https://www.kaggle.com/datasets/kvpratama/pokemon-images-dataset
https://www.kaggle.com/datasets/alexteboul/english-premier-league-logo-detection-20k-images
https://www.classicfootballshirts.co.uk/
https://cocodataset.org/#download

2.1.1 สัดส่วนข้อมูล

ข้อมูลเป็นรูปภาพ จำนวน 150 รูป

2.2 การคลีนข้อมูล

คัดเลือกเฉพาะรูปภาพที่ไม่มีลายน้ำ มีคลื่นสัญญาณรบกวนในภาพต่ำ เป็นรูปที่มองเห็นเส้นขอบ และสีของวัตถุชัดเจน

ภาพที่มีสัญญาณรบกวนมาก

ภาพที่มีลายน้ำ

2.3 การนำข้อมูลเข้าสู่โครงงาน

นำข้อมูลรูปภาพมาแปลงเป็นข้อมูลในรูปแบบ Matrix 3 มิติ (ความสูง X ความกว้าง X Channl สี RGB)

2.4 ขั้นตอนการประมวลผล

ส่วนที่ 1 import library ที่จำเป็นต้องใช้ในโครงงานทั้งหมด

```
import cv2
from scipy import signal, spatial
import numpy as np
from scipy.signal import convolve2d
from matplotlib import pyplot as plt
```

ส่วนที่ 2 รับ input รูปภาพจาก User แล้วแปลงรูปภาพเป็นเมทริกซ์ขนาด 3 มิติ (ความสูง X ความกว้าง X Channel สี RGB)

```
img_name = input("Enter Image Name (Ex. image.jpg): ")
try:
    img = cv2.imread(f"imgin/{img_name}.jpg")
except cv2.error as e:
    pass

if img is None:
    raise Exception(f'Can\'t open/read file from (imgin/{img_name}.jpg): Please check file path/integrity.')

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
edge_op.new_image(img)
```

ส่วนที่ 3 ให้ User เลือกฟีเจอร์เพื่อตกแต่งรูปภาพ โดยมีทั้งหมด 6 ฟีเจอร์ ดังนี้ [1] Resize [2] Rotate [3] Crop [4] Color Threshold [5] Edge Enhance [6] Blur

```
24      mode_list = ["Resize", "Rotate", "Crop", "Color Threshold", "Edge Enhance", "Blur"]
25      print("Choose Features to Edit Image")
26      for i in range(len(mode_list)):
27          print(f"[{i + 1}]: {mode_list[i]}")
28
29      mode = int(input("Please Select Feature : "))
30      if mode == 1:
31          edited_image = resize_image(img)
32      elif mode == 2:
33          edited_image = rotate_image(img)
34      elif mode == 3:
35          edited_image = croping(img)
36      elif mode == 4:
37          edited_image = color_threshold(img)
38      elif mode == 5:
39          edge_op.get_params()
40          edited_image = edge_op.edge_enhance()
41      elif mode == 6:
42          edited_image = get_blurRGB(img)
43      else:
44          raise Exception("Incorrect Mode: Please Try Again")
```

ส่วนที่ 3.1 ฟีเจอร์ Resize เป็นการเปลี่ยนขนาดของรูปภาพให้เป็นไปตามที่ User ต้องการ โดยรับ Input ความกว้าง และความสูงของรูปภาพ

```python
def resize_image(img):
    w, h = input("Enter width and height (Ex.100 100) : ").split()
    resized_img = cv2.resize(img, (int(w), int(h)), interpolation=1)
    return resized_img
```

ส่วนที่ 3.2 ฟีเจอร์ Rotate จะรับ input เป็นค่ามุมที่ User ต้องการจะหมุนโดยหาก User input เป็นค่าบวก รูปจะหมุนทวนเข็มนาฬิกา หาก input ค่าลบ รูปจะหมุนตามเข็มนาฬิกา

```python
def rotate_image(img):
    angle = int(input("Enter rotate angle: (degrees) : "))
    h, w = img.shape[:2] # return (height, width, chanel)
    center_x, center_y = w // 2, h // 2

    # Generate Transformation Matrix
    rotation_matrix = cv2.getRotationMatrix2D((center_x, center_y), angle, 1.0)
    cos_of_rotation_matrix = np.abs(rotation_matrix[0][0])
    sin_of_rotation_matrix = np.abs(rotation_matrix[0][1])

    # Compute new size of an image
    new_w = int((h * sin_of_rotation_matrix) + (w * cos_of_rotation_matrix))
    new_h = int((h * cos_of_rotation_matrix) + (w * sin_of_rotation_matrix))
    new_center_x, new_center_y = new_w / 2, new_h / 2

    # Update Transformation Matrix New Center (x,y)
    rotation_matrix[0][2] += new_center_x - center_x
    rotation_matrix[1][2] += new_center_y - center_y
    rotated_img = cv2.warpAffine(img, rotation_matrix, (new_w, new_h))

    # Detect Image Contour and Crop an image
    gray = cv2.cvtColor(rotated_img, cv2.COLOR_BGR2GRAY)
    contours, _ = cv2.findContours(gray, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    x_min, y_min, x_max, y_max = float('inf'), float('inf'), -float('inf'), -float('inf')

    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        x_min = min(x_min, x)
        y_min = min(y_min, y)
        x_max = max(x_max, x + w)
        y_max = max(y_max, y + h)

    cropped_image = rotated_img[y_min:y_max, x_min:x_max]
    return cropped_image
```

ส่วนที่ 3.3 ฟีเจอร์ Crop โดยรับ input จาก User ดังนี้ x และ y คือการเลือกพิกัดที่จะ crop width และ height คือขนาดความกว้างและยาวที่ต้องการจะ Crop

```
def crop_tools(image, x, y, width, height):
    return image[y:y+height, x:x+width]

def croping(img):
    x, y, crop_width, crop_height = map(int, input("Enter X, Y, Width, and Height to crop (e.g., 100 100 400 300): ").split())
    cropped_img = crop_tools(img, x, y, crop_width, crop_height)
    return cropped_img
```

ส่วนที่ 3.4 ฟีเจอร์ Color Threshold เป็นการการเปลี่ยนสี ณ บริเวณที่ User ต้องการโดยกระบวนการที่ให้ User เลือกบริเวณที่ต้องการนั้นจะเป็นการทำ Color Threshold โดยรับ input จาก User เป็น Range ของค่าสี (HSV) หลังจากได้บริเวณที่ User ต้องการแล้ว ก็จะรับ input สี (RGB) และ ความเข้มข้น (Weight)ที่ User ต้องการ สุดท้ายเป็นการนำสีนั้นไปใส่ในบริเวณที่ User ต้องการตาม Weight ที่ กำหนด

```
def color_threshold(img_inp):

    # get input about user lower and upper thredhold  (HSV)
    hue_threshold = [int(x) for x in input("(HSV) Enter range HUE Threshold  < 0-359 deg > ex. 120-300 : ").split('-')]
    sat_threshold = [int(x) for x in input("(HSV) Enter range Saturation Threshold  < 0-100 % > ex. 30-80 : ").split('-')]
    value_threshold = [int(x) for x in input("(HSV) Enter range Value Threshold < 0-100 % > ex. 50-90 : ").split('-')]

    lower = (np.interp(hue_threshold[0], [0, 365], [0, 180]),
             np.interp(sat_threshold[0], [0, 100], [0, 255]),
             np.interp(value_threshold[0], [0, 100], [0, 255]))
    higher = (np.interp(hue_threshold[1], [0, 365], [0, 180]),
              np.interp(sat_threshold[1], [0, 100], [0, 255]),
              np.interp(value_threshold[1], [0, 100], [0, 255]))

    img_final = img_inp.copy()
    img_inp_hsv = cv2.cvtColor(img_inp, cv2.COLOR_RGB2HSV)

    # make mask from colot threshold
    mask = cv2.inRange(img_inp_hsv, lower, higher)

    # convert from GRAT TO RGB because it will be mask that have only black and wise
    mask = cv2.cvtColor(mask, cv2.COLOR_GRAY2RGB)

    color_layer = img_inp.copy()
    desire_color_RGB = [int(x) for x in input("input color that you want (RGB) xxx xxx xxx : ").split(' ')]
    beta = int(input("Weight (0 - 100%) ex. 10 : "))
    beta = beta / 100
    alpha = 1 - beta

    color_layer[(mask == 255).all(-1)] = desire_color_RGB

    cv2.addWeighted(img_final, alpha, color_layer, beta, 0, img_final)

    return img_final
```

ส่วนที่ 3.5 ฟีเจอร์ Edge enhance เป็นการเปลี่ยนสีของขอบวัตถุต่าง ๆ ในภาพโดยสามารถเลือกได้ว่าจะต้องการใช้ขอบที่เห็นไม่ชัดเจน (Soft edge) หรือขอบเห็นได้ชัดเจน (Hard edge) ซึ่งการกำหนดตำแหน่งของขอบในภาพจะขึ้นอยู่กับการ input กำหนดค่าใช้ในการคำนวนหาขอบ โดยจะสามารถเลือกใช้ค่าเริ่มต้น หรือเลือกกำหนดค่าเองก็ได้

3.5.1 Input เลือก กำหนดค่าเริ่มต้น ถ้า yes จะใช้ค่าเริ่มต้น ถ้า no จะกำหนดค่าเอง โดยการกำหนดค่าเองจะมีดังนี้

- Guassian kernel size เลือกเลขคี่ เพื่อกำหนดขนาด kernel กำจัดคลื่นรบกวน
- Guassian kernel intensity เลือกความเข้ม ใน kernel กำจัดคลื่นรบกวน
- Low threshold ratio เลือกอัตราส่วนเกณฑ์แบ่งเส้นขอบ Hard edge ที่ชัดน้อยสุด
- High threshold ratio เลือกอัตราส่วนเกณฑ์แบ่งเส้นขอบ Hard edge ที่ชัดมากสุด
- Edge magnitude เลือกความเข้มสีสุดท้าย ของเส้นขอบ Hard edge

```python
def get_params(self):
    print('Enter Edge Detection parameters')
    mode = input('Default mode (y/n):')
    if mode.lower() in ['n', 'no']:
        g_size = int(input('Guassian kernel size (3,5,7,9,...): '))
        if g_size % 2 != 1 or g_size < 3:
            g_size = max(g_size+1, 3)
            print(f'Warning: Kernel size is not an odd number. Using size of {g_size}')
        g_sigma = float(input('Guassian kernel intensity (default=1.6): '))

        ltr = float(input('Low threshold ratio (default=5): '))/100
        htr = float(input('High threshold ratio (default=9): '))/100

        smg = int(input('Edge magnitude (range=0-255, default=255): '))
        if smg < 0: smg = 0
        elif smg > 255: smg = 255

        self.new_gaussian = gaussian_kernel(g_size, g_sigma)
        self.low_threshold_ratio = ltr
        self.high_threshold_ratio = htr
        self.strong_mag = smg

    elif mode.lower() in ['y', 'yes']:
        self.set_default()
    else:
        raise Exception('Incorrect Input. Please try again.')
```

3.5.2 ส่วนการทำงานหาตำแหน่งเส้นขอบ Soft edge และ Hard edge ในรูป

```
def get_edge(self):
    '''return (hard_edge, soft_edge)'''

    grey = cv2.cvtColor(self.image, cv2.COLOR_RGB2GRAY)

    gaussian_filter = self.new_gaussian if self.new_gaussian is not None else self.gaussian_default
    blur = convolve2d(grey, gaussian_filter, 'same', 'symm')

    grad, theta = sobel_filter(blur)
    grad = cv2.convertScaleAbs(grad)

    angle = gradient_direction(theta)
    nms = non_max_suppression(grad, angle)
    self.soft_edge = nms.copy()

    thresh, weak, strong = double_thresholding(nms, self.low_threshold_ratio, self.high_threshold_ratio, self.weak_mag, self.strong_mag)
    canny = hysteresis(thresh, weak, strong)
    self.hard_edge = canny.copy()

    return self.hard_edge, self.soft_edge
```

3.5.3 Input เลือกการใช้งานเส้นขอบ Soft edge หรือเส้นขอบ Hard edge และเลือกค่าสี (RGB) ของเส้นขอบที่ต้องการสุดท้าย

```
def edge_enhance(self):
    hard, soft = self.get_edge()

    print('Choose Enhance Mode')
    print('[1]: Soft edge')
    print('[2]: Hard edge')
    mode = int(input('Please Select Mode: '))
    if mode not in [1,2]:
        raise Exception('Incorrect Mode. Please Try Again')

    color = [max(0, min(int(x), 255)) for x in input("Input color of edges (RGB) ex. xxx xxx xxx : ").split(' ')]
    edge = soft if mode == 1 else hard

    # Alpha Blending Method
    edge_rgb = cv2.cvtColor(edge.copy(), cv2.COLOR_GRAY2RGB)
    # Normalize the alpha mask to keep intensity between 0 and 1
    alpha = edge_rgb.astype(float) / 255

    color_img = np.zeros(self.image.shape)
    color_img[:] = color
    foreground = color_img.astype(float)

    background = self.image.astype(float)

    foreground = cv2.multiply(alpha, foreground)
    background = cv2.multiply(1.0-alpha, background)
    out = cv2.add(foreground, background) / 255

    out = cv2.normalize(out, None, 255, 0, cv2.NORM_MINMAX, cv2.CV_8U)

    return out
```

ส่วนที่ 3.6 ฟีเจอร์ Blur จะรับ input
เลือกเพื่อกำหนดปริมาณความเข้มของการทำให้ภาพมัวขึ้นกว่าเดิม

```python
def gaussian_kernel(size, sigma=1):
    if size % 2 != 1:
        return None
    size = size // 2
    x, y = np.mgrid[-size:size+1, -size:size+1]
    normal = 1 / (2.0 * np.pi * sigma**2)
    g =  np.exp(-((x**2 + y**2) / (2*sigma**2))) * normal
    return g
```

```python
def get_blurRGB(image:np.array):
    blur_mode = {'1':[3,0.55],'2':[3,1],'3':[5,1.5],'4':[7,2],'5':[9,3]}

    mode = input('Choose blur intensity (1-5): ')
    if not blur_mode.get(mode):
        mode = str(int(min(5, max(1, int(mode)))))
        print(f'Warning: Intensity outside of range 1-5. Using intensity of {mode}')

    ksize, ksig = blur_mode.get(mode)
    kernel = gaussian_kernel(ksize, ksig)

    img = image.copy()
    img = img.astype(float)

    r = img[:,:,0]
    g = img[:,:,1]
    b = img[:,:,2]

    r_blur = convolve2d(r, kernel, 'same', 'symm')
    g_blur = convolve2d(g, kernel, 'same', 'symm')
    b_blur = convolve2d(b, kernel, 'same', 'symm')

    blur = np.dstack((r_blur,g_blur,b_blur))
    blur = cv2.normalize(blur, None, 255, 0, cv2.NORM_MINMAX, cv2.CV_8U)

    return blur
```

ส่วนที่ 4 เป็นส่วนการแปลงเมทริกซ์ไปเป็นเวกเตอร์เพื่อนำไปคำนวณค่า Cosine Similarity โดยเริ่มจากหาค่าความต่างของขนาดรูปต้นฉบับ และรูปที่ตกแต่งแล้ว ถ้าขนาดของรูปต้นฉบับและรูปที่ตกแต่งแล้วแตกต่างกัน จะส่งผลให้ไม่สามารถหาค่า Cosine Similarity ได้ จึงต้องทำการ Zero Padding รูปที่มีขนาดเล็กกว่า
ให้มีขนาดเท่ากับรูปที่มีขนาดใหญ่เพื่อให้สามารถนำรูปทั้งสองไปคำนวณค่า Cosine Similarity ได้ต่อไป

```
# Get the dimensions of the original image
original_height, original_width, _ = img.shape

# Get the dimensions of the resized image
resized_height, resized_width, _ = edited_image.shape

# Calculate the padding needed to match sizes
vertical_padding = abs(original_height - resized_height)
horizontal_padding = abs(original_width - resized_width)

# Padding a smaller image to match image size
if mode in [4,5,6]:       ReallyWarm, 2 hours ago • add blur function
    A = img.flatten()
    B = edited_image.flatten()
else:
    if (original_height > resized_height) and (original_width > resized_width):
        padding_edited = padding0(edited_image, vertical_padding, horizontal_padding)
        A = img.flatten()
        B = padding_edited.flatten()
    elif (original_height > resized_height) and (original_width < resized_width):
        padding_original = padding0(img, 0, horizontal_padding)
        padding_edited = padding0(edited_image, vertical_padding, 0)
        A = padding_original.flatten()
        B = padding_edited.flatten()
    elif (original_height < resized_height) and (original_width > resized_width):
        padding_original = padding0(img, vertical_padding, 0)
        padding_edited = padding0(edited_image, 0, horizontal_padding)
        A = padding_original.flatten()
        B = padding_edited.flatten()
    else:
        padding_original = padding0(img, vertical_padding, horizontal_padding)
        A = padding_original.flatten()
        B = edited_image.flatten()
```

```python
def padding0(edited_image, vert_padding, hori_padding):
        # Pad the resized image with zeros
        if vert_padding % 2 == 1 and hori_padding % 2 == 1:              # vertical ODD, Horizon is ODD
              resized_padded = cv2.copyMakeBorder(edited_image, vert_padding // 2 + 1, vert_padding // 2,
                                                  hori_padding // 2 + 1, hori_padding // 2, cv2.BORDER_CONSTANT, None, value=0)
        elif vert_padding % 2 == 1 and hori_padding % 2 == 0:              # vertical ODD , Horizon Even
              resized_padded = cv2.copyMakeBorder(edited_image, vert_padding // 2 + 1, vert_padding // 2,
                                                  hori_padding // 2, hori_padding // 2, cv2.BORDER_CONSTANT, None, value=0)
        elif vert_padding % 2 == 0 and hori_padding % 2 == 1:              # vertical EVEN , Horizon is ODD
              resized_padded = cv2.copyMakeBorder(edited_image, vert_padding // 2, vert_padding // 2,
                                                  hori_padding // 2 + 1, hori_padding // 2, cv2.BORDER_CONSTANT, None, value=0)
        else:                                                              # vertiacl EVEN , Horizon EVEN
              resized_padded = cv2.copyMakeBorder(edited_image, vert_padding // 2, vert_padding // 2,
                                                  hori_padding // 2, hori_padding // 2, cv2.BORDER_CONSTANT, None, value=0)

        return resized_padded
```

ส่วนที่ 5 การแสดงผล โดยจะแสดงรูปต้นฉบับ, รูปที่ตกแต่งแล้ว และค่า Cosine Similarity ระหว่างทั้งสองรูป และ User สามารถบันทึกรูปภาพที่ตกแต่งแล้วหรือไม่ก็ได้

```python
def cosine_similarity(vector1, vector2):
    cos_sim = 1 - spatial.distance.cosine(vector1, vector2)
    return cos_sim
```

```python
# Display images
plt.figure(figsize=(12, 6))
plt.subplot(2, 2, 1); plt.imshow(img)
plt.title("OLD IMAGE")
plt.subplot(2, 2, 2); plt.imshow(edited_image)
plt.title("NEW IMAGE")
plt.figtext(0.5, 0.3, f'Cosine Similarity : {cosine_similarity(A, B)}', fontsize=12, ha='center', va='center', color='blue')
plt.show(block=False)
plt.pause(1)

save = input("Save image (y/n):")
if save.lower() in ['y', 'yes']:
    edited_image = cv2.cvtColor(edited_image, cv2.COLOR_RGB2BGR)
    cv2.imwrite(f'{FLIE_PATH}/imgout/{img_name}_edited.jpg', edited_image)

print("Exiting program . . .")
plt.close()
```

2.5 การประยุกต์ใช้ทฤษฎี

2.5.1 การประยุกต์ใช้ทฤษฎีเวกเตอร์

2.5.1.1 Cosine similarity

การวัดความเหมือนของเวกเตอร์ 2 เวกเตอร์ด้วยองศา ว่ามีทิศทางไปทางเดียวกันหรือไม่ โดยที่เป็นจะการตัดการเทียบขนาดของเวกเตอร์ออกไป
และจะแสดงค่ามุมที่ผ่านการคำนวณแล้วออกมาเป็นค่า 0 ถึง 1 ซึ่งถ้าค่าเข้าใกล้ 1 มาก จะหมายความว่ามีความเหมือนกันมาก

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

โดยจะสามารถนำมาเทียบความเหมือนของรูปภาพที่เป็น Matrix ได้ด้วยการทำ Flatten กับ Matrix ของรูปภาพต้นฉบับและ Matrix ของ รูปภาพสุดท้าย ได้ผลลัพธ์เป็นเวกเตอร์ออกมา

ตัวอย่างการประยุกต์ใช้ Cosine similarity ในการคำนวนความเหมือนของภาพ

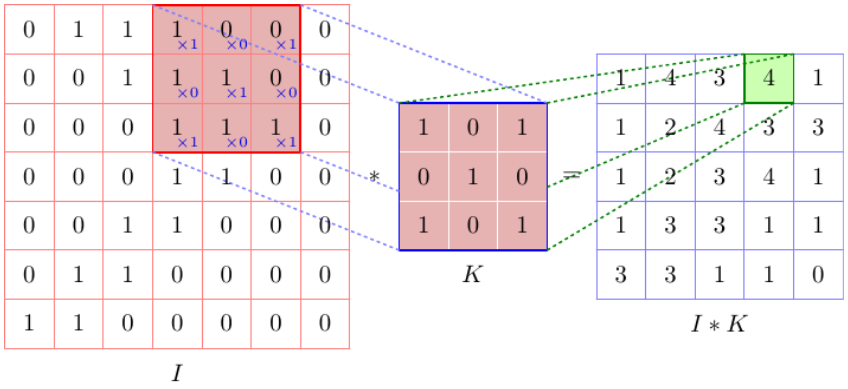
```
A = img.flatten()
B = edited_image.flatten()
```

```
def cosine_similarity(vector1, vector2):
    cos_sim = 1 - spatial.distance.cosine(vector1, vector2)
    return cos_sim
```

2.5.2 การประยุกต์ใช้ทฤษฎีเมทริกซ์

2.5.2.1 Matrix Convolution

เป็นการนำ Matrix ขนาดเล็ก เรียกว่า Kernel มาเลื่อนผ่านไปบน Matrix ข้อมูล Pixel ของภาพ โดยขณะที่มีการทาบ Kernel บนส่วนหนึ่งของภาพ ก็จะคูณค่าแต่ละ Pixel ของ Input Image กับ Kernel แล้วนำทั้งหมดมาบวกกันเป็น 1 จุด Pixel ของ Matrix ภาพผลลัพธ์

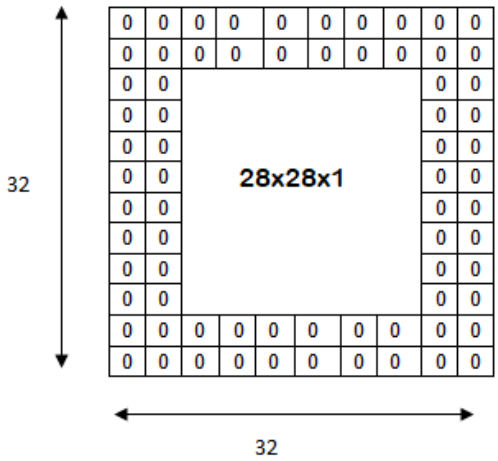


ตัวอย่างการประยุกต์ใช้ Metrix Convolution ในการทำฟีเจอร์ Blur

```
blur = convolve2d(grey, gaussian_filter, 'same', 'symm')
```

- Zero Padding

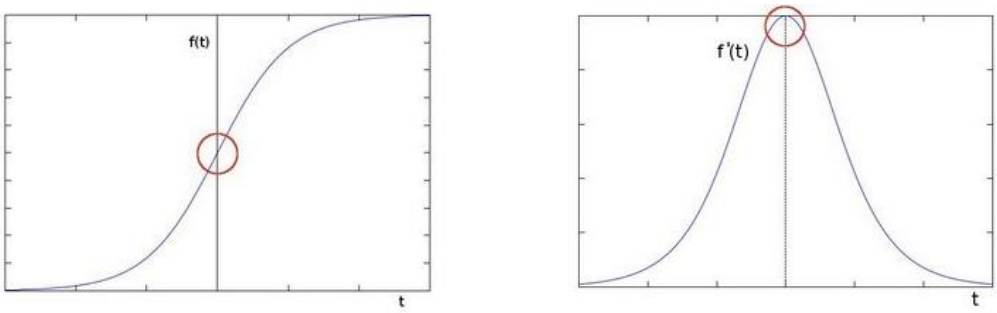
เป็นการทำให้ Matrix ที่มีขนาดเล็กนั้น มีขนาดใหญ่ขึ้นเป็นตามขนาดที่ต้องการ โดยการเสริมกรอบรอบ ๆ Matrix เดิมด้วยค่า 0



ข้อดีของการทำ Zero Padding ในการทำ Convolution จะทำให้ Matrix ภาพผลลัพธ์มีขนาดเล็กกว่าภาพต้นฉบับ ดังนั้นจะต้องมีการทำ Zero Padding ให้ภาพสุดท้ายมีขนาดเท่าต้นฉบับ

- Edge detection

การหาตำแหน่งของเส้นขอบในภาพสามารถดูได้จากตำแหน่งที่มีการเปลี่ยนแปลงความเข้มของ Pixel รอบ ๆ สูงกว่าตำแหน่งอื่น จะทำให้รู้ได้ว่าตำแหน่งนั้นเป็นส่วนขอบของเส้นในภาพนั้น โดยสามารถทำได้ด้วยการหา Derivatives ที่แต่ละตำแหน่ง Pixel ของภาพ จะได้ตำแหน่งที่มีความเข้มของ Pixel ต่างกันสูงออกมาเป็นส่วนเส้นขอบในภาพ



โดยเราสามารถใช้ Sobel kernels มาทำ Convolution กับภาพเพื่อหาค่าประมาณของ Derivatives ในภาพได้

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

เมื่อ I แทน Matrix รูปภาพ เราจะได้

Gx = Derivative ในแนวนอน

Gy = Derivative ในแนวตั้ง

จากนั้นนำ Derivatives ทั้งสองมารวมกันจะได้ Matrix ที่มีค่าประมาณตำแหน่งของเส้นขอบในภาพเป็นความเข้มของการเปลี่ยนแปลงของ Pixel อยู่

$$G = \sqrt{G_x^2 + G_y^2}$$

ตัวอย่างการประยุกต์ใช้ Metrix Convolution หาขอบในรูปโดยใช้ Sobel kernels

```
def sobel_filter(grey_img):
    sobelx = np.matrix([[-1, 0, 1],
                        [-2, 0, 2],
                        [-1, 0, 1]], np.float32)

    sobely = np.matrix([[ 1, 2, 1],
                        [ 0, 0, 0],
                        [-1,-2,-1]], np.float32)
```

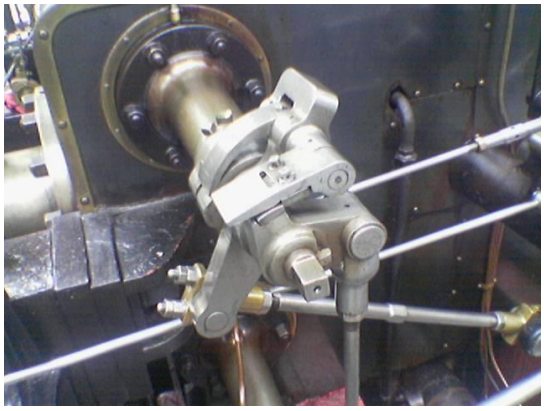
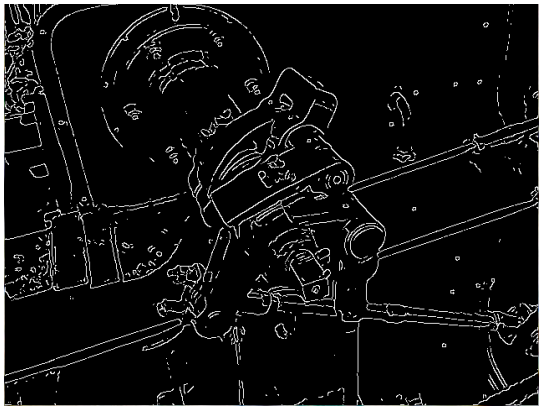
```
    Gx = convolve2d(grey_img, sobelx, "same", "symm")
    Gy = convolve2d(grey_img, sobely, "same", "symm")

    Gout = np.hypot(Gx, Gy) # sqrt(Gx**2 + Gy**2)
    theta = np.arctan2(Gy, Gx) # Edge direction

    return Gout, theta
```

- Canny Edge Detector

จะมีการทำงานของ Edge detection รวมเข้ากับ Algorithm ต่าง ๆ หลายขั้นตอนเพื่อให้การตรวจจับขอบในภาพมีความแม่นยำสูงขึ้น ลดความผิดพลาดจากคลื่นรบกวน และตรวจจับจุดกึ่งกลางของขอบได้แม่นยำมากขึ้น

จุดเด่นนี้จึงเป็นสาเหตุในการนำมาใช้หาตำแหน่งของเส้นขอบที่เห็นไม่ชัดเจน (Soft edge) และเส้นขอบเห็นได้ชัดเจน (Hard edge) ในส่วนของฟีเจอร์ Edge enhance
โดยมีขั้นตอนการทำงานดังนี้

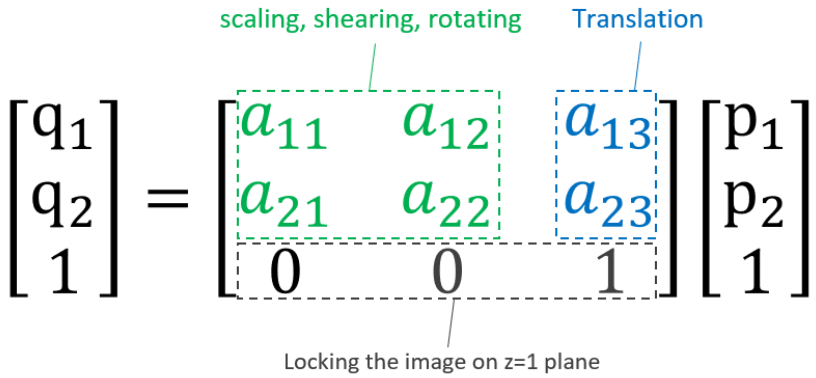
1. นำภาพผ่าน Gaussian filter เพื่อทำการ Blur ให้ภาพเรียบขึ้นและกำจัดคลื่นรบกวนออก
2. หาตำแหน่งของเส้นขอบในภาพโดยใช้ Sobel kernels ทำ Edge detection
3. ลดส่วนด้านข้างของเส้นขอบที่มีขนาดความเข้มของ Pixel ต่ำกว่าความเข้มของ Pixel ของเส้นขอบในทิศทางข้าง ๆ เพื่อให้เส้นขอบบางลงและแม่นยำขึ้น
โดยจะนำผลลัพธ์นี้ไปใช้ในส่วนของเส้นขอบ Soft edge
4. กำหนด Double thresholding เพื่อทำการหาส่วนที่มีความเข้มสูงมากและน้อยแต่ไม่ต่ำเกินไป
5. หาขอบ Hard edge ด้วยการใช้ Hysteresis โดยเชื่อมขอบส่วนที่มีความเข้มสูงกับเข้มน้อยที่อยู่ใกล้ ๆ เข้าด้วยกัน และตัดส่วนที่มีความเข้มน้อยอยู่เดี่ยว ๆ ออก

ตัวอย่าง รูปต้นฉบับ และรูปจาก Canny edge detector ส่วน Hard edge

2.5.2.2 Transformation matrix

เป็น Matrix ที่สามารถทำให้เกิดกระบวนการเปลี่ยนแปลง Vector หนึ่งไปเป็นอีกVector หนึ่งได้เช่น การให้ Vector ขยายขนาด, เลื่อนตำแหน่ง, หมุน เป็นต้น โดยอาศัยการนำ Vector ต้นฉบับ ไปทำการคูณกับ Trasformation Matrix



$$\begin{bmatrix} q_1 \\ q_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ 1 \end{bmatrix}$$

-Rotation

การหมุนภาพไปตามมุมที่ต้องการ ใช้หลักการ Vector Transformation เมื่อต้องการหมุนภาพไป theta องศาทวนเข็มนาฬิกา จะสามารถสร้าง transformation matrix ได้ดังรูป

$$R_{2D} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3\,x\,3}$$

และนำ transformation matrix ไปคูณกับ vector ตำแหน่งของแต่ละ pixel ในภาพ จะได้ vector ตำแหน่งของแต่ละ pixel ในภาพใหม่ที่หมุนแล้ว

แต่ใน opencv จะใช้ transformation matrix ที่ดัดแปลงจาก transformation matrix ภาพด้านบน โดยสามารถเปลี่ยนจุดศูนย์กลางการหมุนได้ ดังรูป

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

ตัวอย่างการคำนวณ

Vector ตำแหน่ง pixel 1 จุดบนภาพ = $\begin{bmatrix} 4 \\ 2 \end{bmatrix}$

หมุนทวนเข็มนาฬิกา 30 องศา

จะได้ Transformation Matrix ดังนี้

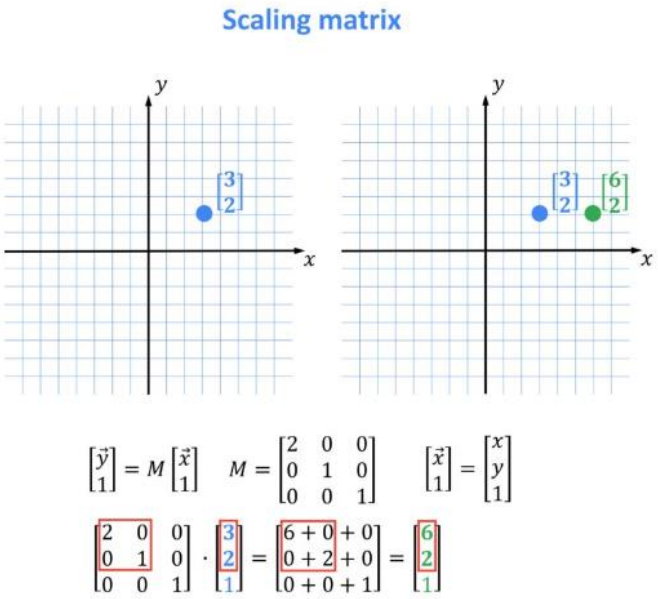
$$T_{2D} = \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$V' = \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4\cos 30^\circ - 2\sin 30^\circ \\ 4\sin 30^\circ + 2\cos 90^\circ \\ 1 \end{bmatrix}$$

จะได้ vector ตำแหน่งของแต่ละ pixel ในภาพใหม่ที่หมุนแล้ว = $\begin{bmatrix} 2.5 \\ 0.3 \end{bmatrix}$

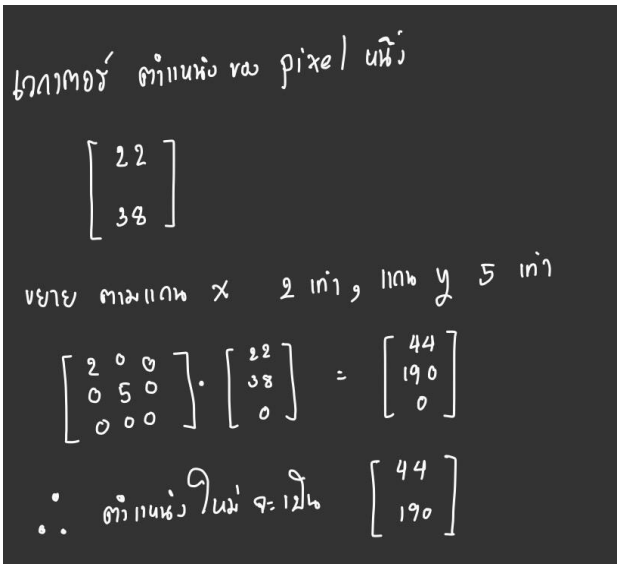
-Scaling

การย่อขยายรูปภาพไปตามขนาดที่ต้องการ ใช้หลักการ Vector Transformation เมื่อต้องการย่อหรือขยายภาพไปกี่เท่าจากรูปเดิม จะสามารถสร้าง transformation matrix ได้ ดังรูป



และนำ transformation matrix ไปคูณกับ vector ตำแหน่งของแต่ละ pixel ในภาพ จะได้ vector ตำแหน่งของแต่ละ pixel ในภาพใหม่ที่ถูกขยายแล้ว

ตัวอย่างการคำนวณ

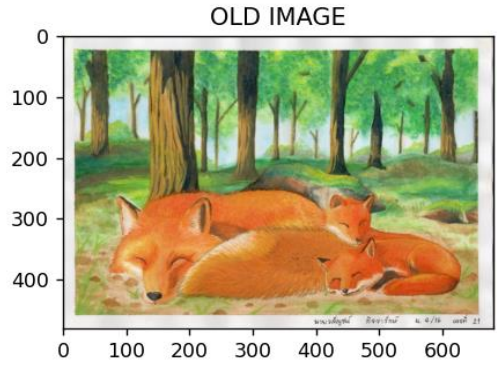


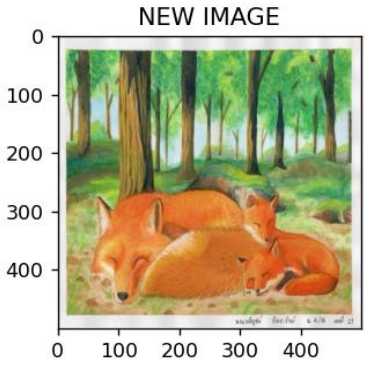
บทที่ 3

ผลลัพธ์ที่ได้รับ

3.1 ผลลัพธ์จากฟีเจอร์ Resize

3.1.1 ผลลัพธ์ Resize ภาพขนาด 600 x 480 เป็น 500 x 500

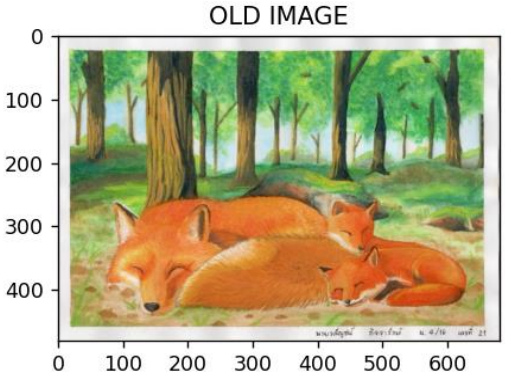


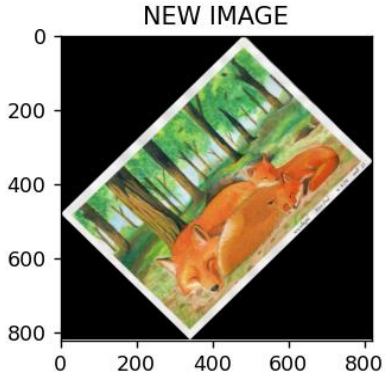


Cosine Similarity : 0.9593992401785643

3.2 ผลลัพธ์จากฟีเจอร์ Rotate

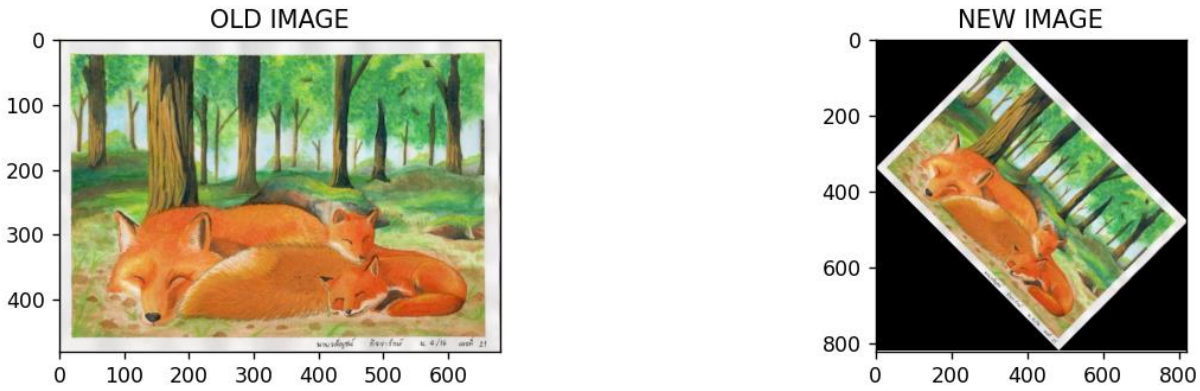
3.2.1 ผลลัพธ์ Rotate ภาพทวนเข็มนาฬิกา 45 องศา





Cosine Similarity : 0.9057293488398155

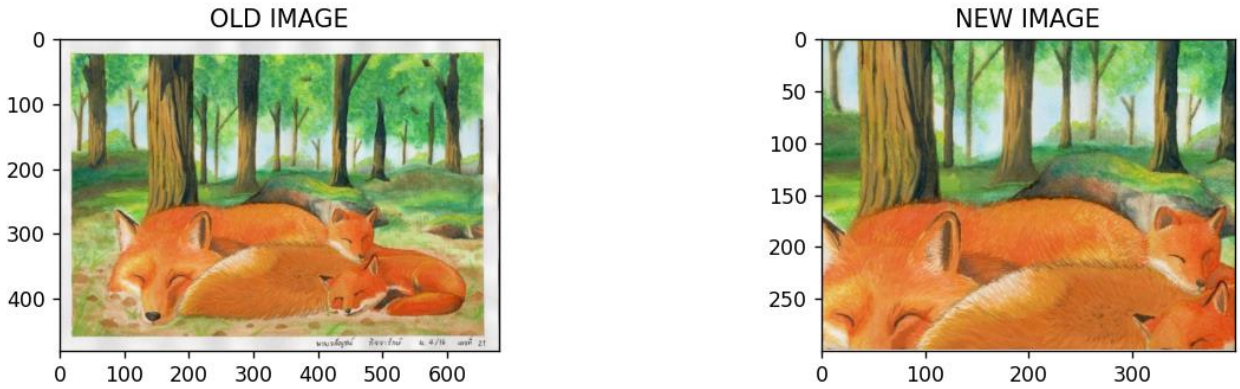
3.2.2 ผลลัพธ์ Rotate ภาพตามเข็มนาฬิกา 45 องศา



Cosine Similarity : 0.9059397317521873

3.3 ผลลัพธ์จากฟีเจอร์ Crop

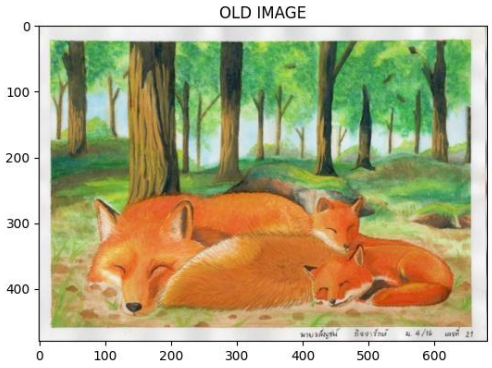
3.3.1 ผลลัพธ์ Crop ภาพโดยเริ่มที่พิกัด (x, y) = (100, 100) ไปจนถึงพิกัด (x + width, y + height) = (500, 400)

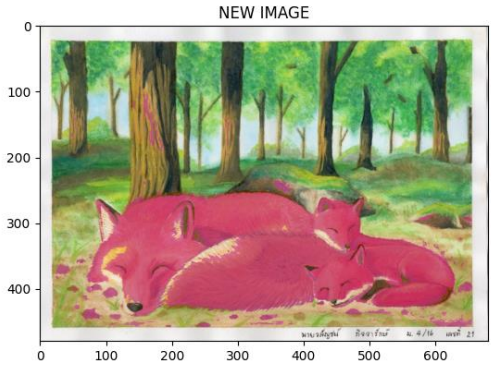


Cosine Similarity : 0.6975633888552832

3.4 ผลลัพธ์จากฟีเจอร์ Color Threshold

ช่วง Hue Threshold 14-50
Saturation Threshold 60 -100
Value Threshold 45-100
ต้องการเปลี่ยนเป็นสี (RGB) : 31 146 246
Weight : 45%

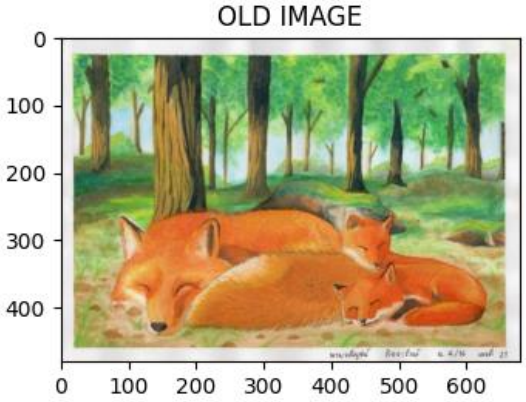


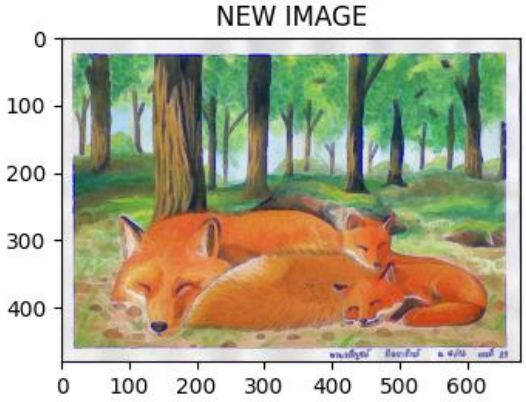


Cosine Similarity : 0.96504325777787

3.5 ผลลัพธ์จากฟีเจอร์ Edge Enhance

3.3.1 ผลลัพธ์ Soft edge โดยใช้ค่าเริ่มต้น และสีของเส้นขอบ 0 0 255





Cosine Similarity : 0.9595014742651291

3.3.1 ผลลัพธ์ Hard edge โดยใช้การกำหนดค่า

Guassian kernel size 7

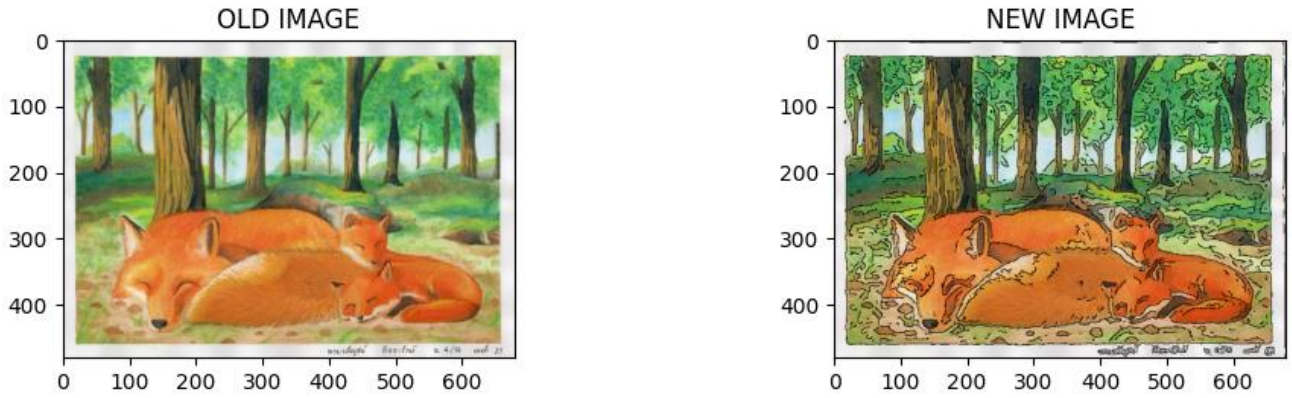
Guassian kernel intensity 3

Low threshold ratio 6

High threshold ratio 10

Edge magnitude 255

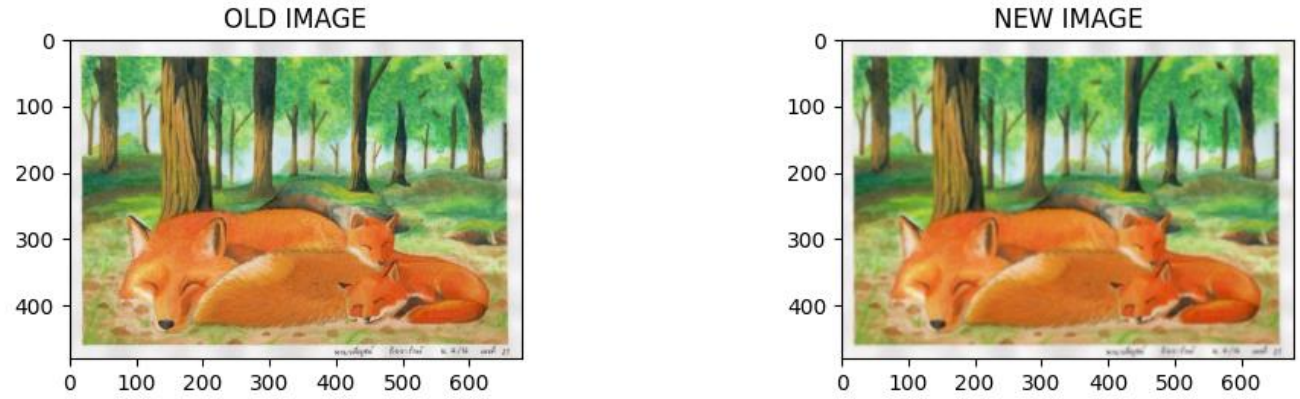
สีของเส้นขอบ 0 0 0



Cosine Similarity : 0.9428369411047185

3.4 ผลลัพธ์จากฟีเจอร์ Blur

ผลลัพธ์การกำหนดความเข้ม 3



Cosine Similarity : 0.8477863195040456

บทที่ 4

ปัญหาที่พบระหว่างการดำเนินโครงงาน

ปัญหาที่พบ

1.รูปภาพ จากแหล่งข้อมูล บางภาพนั้นติดลายน้ำ หรือไม่ชัด ทำให้ต้องกำจัดออก

2.รูปภาพผลลัพธ์จากฟีเจอร์ Rotate มีการตัดรูปภาพส่วนที่เกินขอบออก ทำให้ได้รูปภาพที่ถูก Rotate ไม่เต็ม

3.ภาพผลลัพธ์บางภาพมีขนาดไม่เท่ากับต้นฉบับ ทำให้ไม่สามารถคำนวณ Cosine Similarity ได้

แนวทางพัฒนา

1.พัฒนาเป็นโปรแกรมที่เป็น GUI เพื่อให้ User ใช้งานได้สะดวกมากขึ้น เพิ่มฟีเจอร์ที่ User สามารถเลือกพื้นที่ในรูปภาพที่ต้องการได้โดยกระบวนการ Edge Detection

2.พัฒนาฟังก์ชันให้สามารถเปลี่ยนแปลงรูปภาพได้แบบ Real-time และมีความลื่นไหลในการใช้งานมากขึ้น เช่นการ Crop ด้วยการลากจากเมาส์ หรือการเปลี่ยน Color Threshold แบบมีจานสีให้ User เลื่อนเป็นต้น

3.หลังจากที่มีการตัดแต่งรูปภาพ จะเพิ่มการวัดค่า Similarity ของภาพโดย ใช้ปริมาณสีของภาพก่อนที่ภาพจะถูกเปลี่ยนสีตามที่ผู้ใช้เลือกนั้นปริมาณสีที่ผู้ใช้ต้องการมีจำนวนกี่%ของภาพทั้งหมด และหลังจากเปลี่ยนสีภาพแล้ว % ปริมาณของสีเพิ่มขึ้นเป็นเท่าใด

บทที่ 5

แหล่งอ้างอิงข้อมูล

5.1 แหล่งข้อมูลที่จะนำมาใช้วิเคราะห์

https://www.kaggle.com
https://www.classicfootballshirts.co.uk/
https://cocodataset.org/#download

5.2 แหล่งอ้างอิงการสืบค้นข้อมูล

OpenCV. “Geometric Transformations of Images”, [ระบบออนไลน์].
https://docs.opencv.org/4.x/dd/d52/tutorial_js_geometric_transformations.html

OpenCV. “Adding borders to your images”, [ระบบออนไลน์].
https://docs.opencv.org/3.4/dc/da3/tutorial_copyMakeBorder.html

OpenCV. “Sobel Derivatives”, [ระบบออนไลน์].
https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html

Wikipedia. Sep 29, 2023. “Canny edge detector”, [ระบบออนไลน์].
https://en.wikipedia.org/wiki/Canny_edge_detector

Sofiane Sahir. Jan 25, 2019. “Canny Edge Detection Step by Step in Python - Computer Vision”, [ระบบออนไลน์].
https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Sunita Nayak. Apr 10, 2017. “Alpha Blending Using OpenCV (C++ / Python)”, [ระบบออนไลน์].
https://learnopencv.com/alpha-blending-using-opencv-cpp-python

ภาคผนวก

ภาคผนวก ก

ข้อมูลโครงงาน

1. ลิงค์ Source Code
https://github.com/ReallyWarm/photo-editor

2. ลิงค์ Dataset รูปภาพต่างๆ https://drive.google.com/drive/u/0/folders/17abhc-hOxxFktUEJZl8G6vGN3rvB7_Eh