

ข้อที่ 1) สิ่งที่น่าสนใจคือ reduce_gray_levels ซึ่งเป็น key หลักของข้อนี้

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

def reduce_gray_levels(img, levels):
    step = 256 / levels
    quantized = (img / step).astype(np.uint8)
    quantized = (quantized * step).astype(np.uint8)
    return quantized

def process_display_and_save_image(image_path, output_folder):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        print(f"Error: Couldn't load the image {image_path}.")
        return

    gray_levels = [8, 64, 128, 256]

    # Display in Jupyter Notebook
    plt.figure(figsize=(15, 5))

    for index, levels in enumerate(gray_levels, 1):
        processed_img = reduce_gray_levels(img, levels)

        # Display in Jupyter Notebook
        plt.subplot(1, 4, index)
        plt.imshow(processed_img, cmap='gray')
        plt.title(f"{levels} gray levels")
        plt.axis('off')

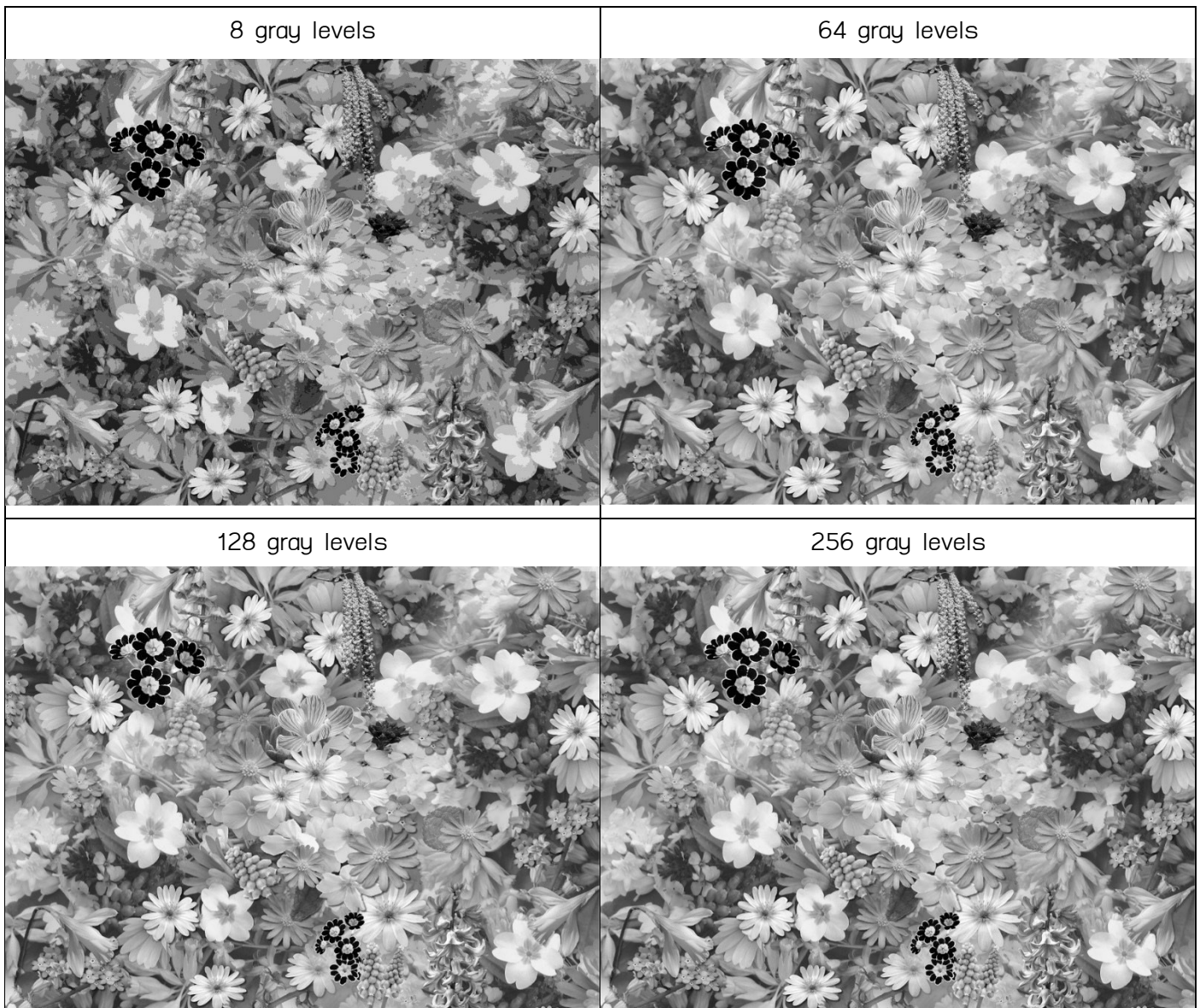
        # Save to the output folder
        base_name = os.path.basename(image_path)
        file_name_without_extension = os.path.splitext(base_name)[0]
        new_file_name = f"{file_name_without_extension}_{levels}_levels.jpg"

        output_path = os.path.join(output_folder, new_file_name)
        cv2.imwrite(output_path, processed_img)

    plt.tight_layout()
    plt.show()

# Ensure the output folder exists, if not create it
output_folder = "processed_images"
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

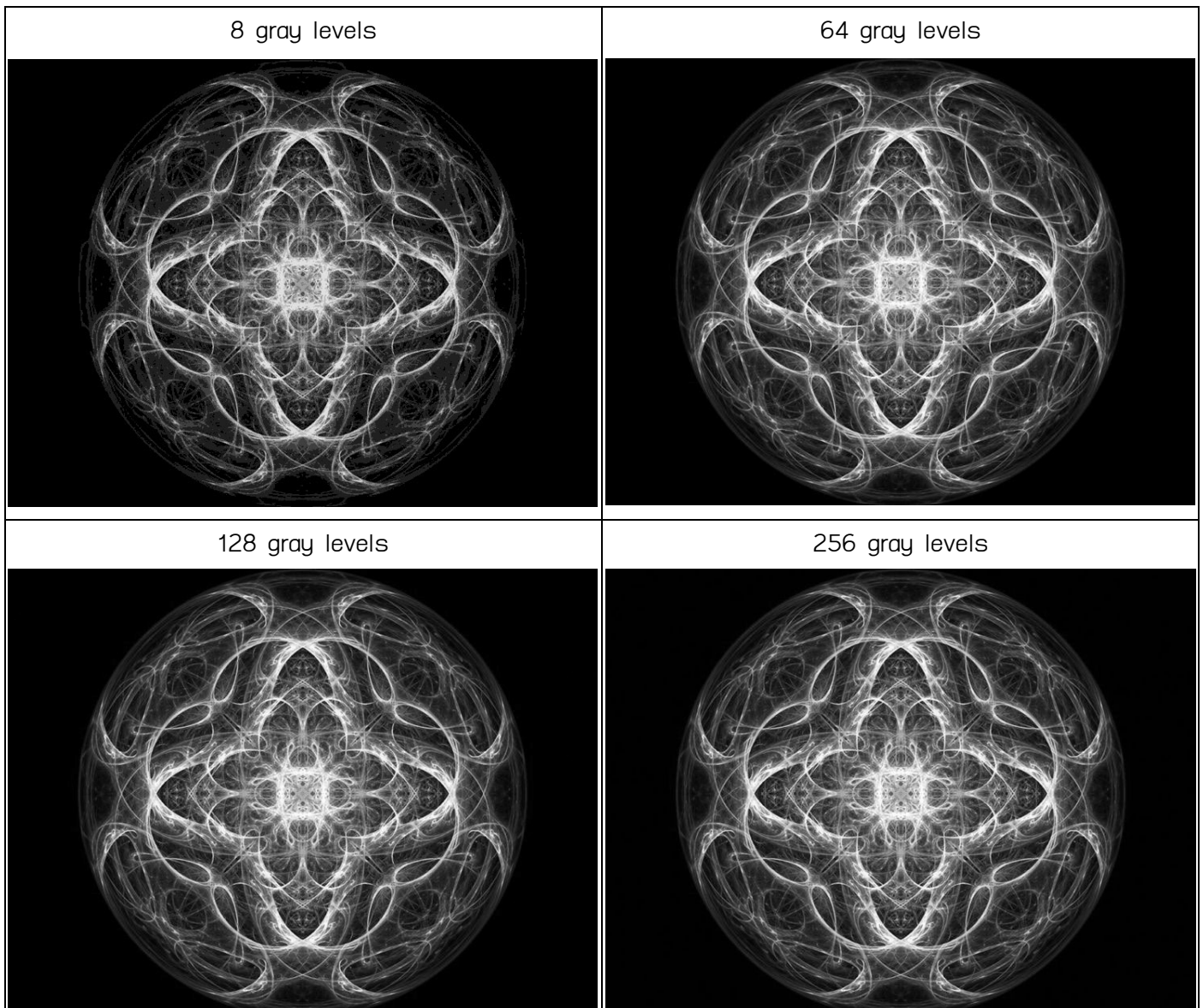
image_list = ["../original_images/flower.jpg",
              "../original_images/fractal.jpeg", "../original_images/fruit.jpg"]
```



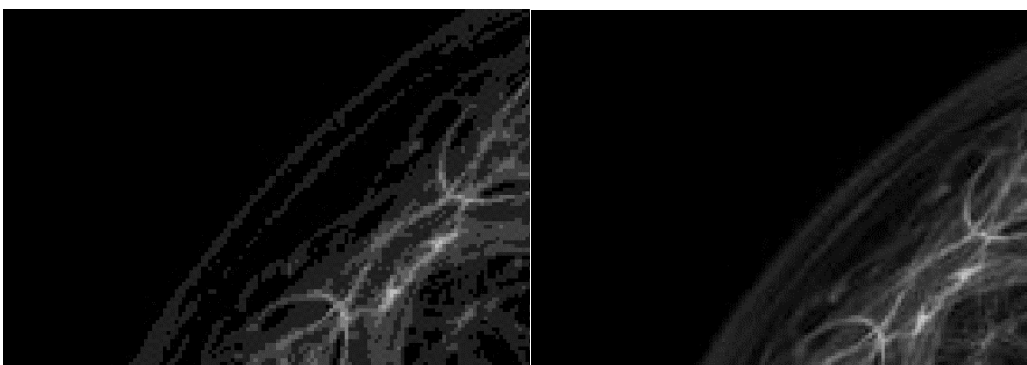
ภาพแรก flower เมื่อปรับเป็น 64,128 และ 256 gray levels จะพบว่าเราไม่สามารถแยกความแตกต่างออกกับภาพต้นฉบับได้ แต่เมื่อปรับเป็น 8 gray levels จะพบว่าเราเริ่มสามารถเห็นความแตกต่างได้แล้ว ดังแสดงในรูปที่ตัดออกมา



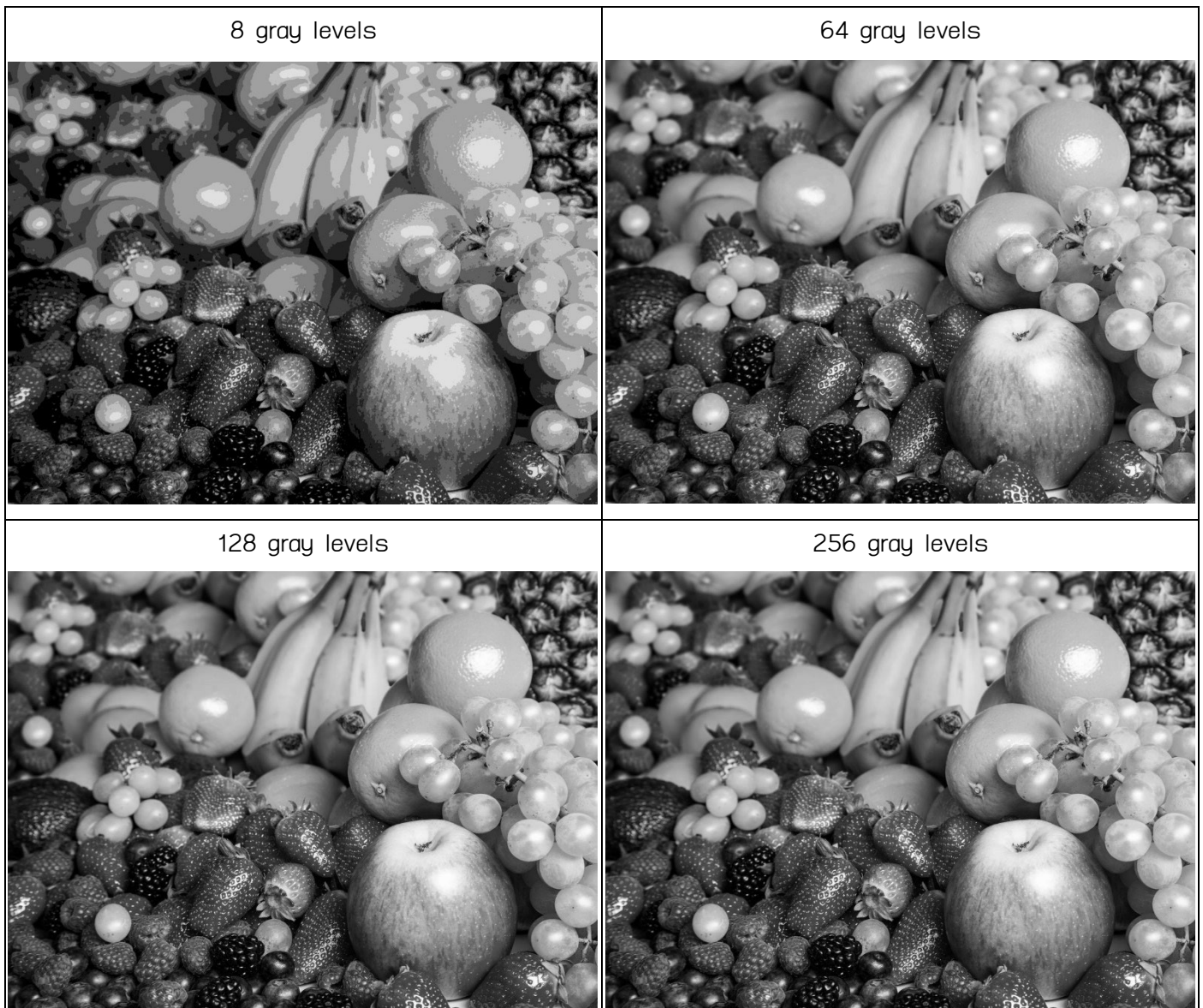
จะเห็นได้ว่าสีของกลีบดอกของ 8 gray level จะดูไม่ smooth เมื่อเทียบกับ 64 gray levels



ภาพที่สอง fractal เหมือนกับภาพแรกเมื่อปรับเป็น 64,128 และ 256 gray levels จะพบว่าเราไม่สามารถแยกความแตกต่างออกกับภาพต้นฉบับได้ แต่เมื่อปรับเป็น 8 gray levels จะพบว่าเราเริ่มสามารถเห็นความแตกต่างได้แล้ว ดังแสดงในรูปที่ตัดออกมา

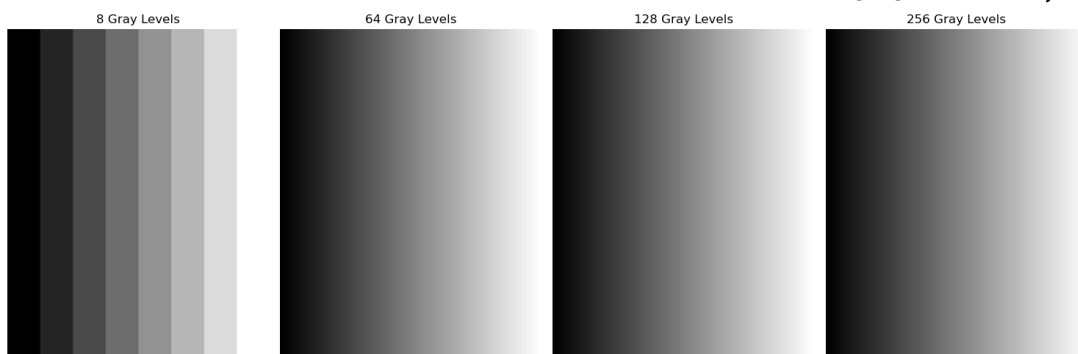


จะเห็นว่าสีบริเวณขอบ ที่เป็นสีดำ ทางด้านซ้ายที่เป็น 8 gray levels จะมีความดำสนิท มากกว่าภาพทางด้านขวาที่เป็น 64 gray levels



ภาพที่สุ่ดท่าย fruit เช่นเดียวกับภาพก่อนๆหน้า เมื่อปรับเป็น 64,128 และ 256 gray levels จะพบว่าเราไม่สามารถแยกความแตกต่างออกกับภาพต้นฉบับได้ แต่เมื่อปรับเป็น 8 gray levels จะพบว่าเราเริ่มสามารถเห็นความแตกต่างได้แล้ว ซึ่งสังเกตได้ว่าภาพ fruit นี้แสดงถึงความแตกต่างของ 8 gray levels กับภาพ gray level อื่นได้อย่างชัดเจนที่สุด เพราะเราจะเห็นว่าภาพที่เป็น 8 gray level จะเหมือนกับเป็นภาพวาดมากกว่าภาพที่ถ่ายออกมา

ถ้าเราลองเขียน code เพิ่มนิดหน่อยให้โปรแกรมแสดงความแตกต่างของ gray level ต่างๆซึ่งได้ผลดังภาพด้านล่าง



จะพบว่ามีเพียงแค่ระดับ 8 gray levels ที่เราแยกความแตกต่างอย่างเห็นได้ชัดเทียบกับ gray levels อื่นๆ

ข้อที่ 2) สิ่งที่น่าสนใจคือ enhance_image ซึ่งเป็น key หลักของข้อนี้

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

def enhance_image(img):
    L = 256 # 8-bit grayscale image
    # Create an empty image with the same dimensions for the output
    enhanced_img = np.zeros_like(img)
    # Define the conditions and apply the transformations
    cond1 = img < L/3
    cond2 = (img >= L/3) & (img < 2*L/3)
    cond3 = img >= 2*L/3
    enhanced_img[cond1] = 5*L/6
    enhanced_img[cond2] = 2*img[cond2] + L/6
    enhanced_img[cond3] = L/6
    return enhanced_img

def process_display_and_save_image(image_path, output_folder):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        print(f"Error: Couldn't load the image {image_path}.")
        return
    enhanced_img = enhance_image(img)
    # Display in Jupyter Notebook
    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    plt.imshow(img, cmap='gray')
    plt.title('Original Image')
    plt.axis('off')
    plt.subplot(1, 2, 2)
    plt.imshow(enhanced_img, cmap='gray')
    plt.title('Enhanced Image')
    plt.axis('off')
    # Save the enhanced image to the output folder
    base_name = os.path.basename(image_path)
    file_name_without_extension = os.path.splitext(base_name)[0]
    new_file_name = f"{file_name_without_extension}_enhanced.jpg"
    output_path = os.path.join(output_folder, new_file_name)
    cv2.imwrite(output_path, enhanced_img)
    plt.tight_layout()
    plt.show()

image_list =
["../original_images/flower.jpg", "../original_images/traffic.jpg", "../original_images/tram.jpg"]

for image in image_list:
    process_display_and_save_image(image, output_folder)
```


Original Image



Enhanced Image



Original Image



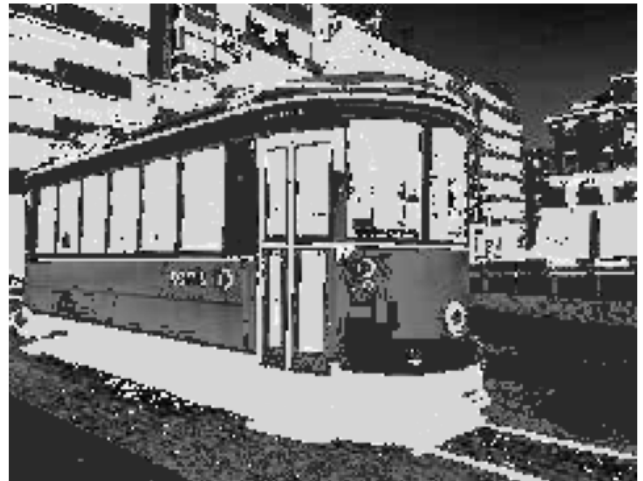
Enhanced Image



Original Image



Enhanced Image



ก่อนจะบอกว่าแต่ละภาพมีการเปลี่ยนแปลงอย่างไร อยากจะบอกก่อนว่าเนื่องจากเราใช้เป็น 8-bit gray scale ดังนั้น $L = 256$ และจากกราฟจะทำให้ได้ว่า ตั้งแต่ $r = 0$ ถึง $r = 85$ จะกลายเป็นระดับสีที่ 213 ซึ่งเป็นสีที่ดำขึ้น ดังนั้นเราจะน่าจะ expected ได้ว่า สีที่สว่างๆ(สีขาว)ควรจะเปลี่ยนกลายเป็นสีที่เข้มขึ้น(ดำ) ต่อมาในช่วงตั้งแต่ $r = 86$ ถึง $r = 170$ ซึ่งน่าจะเป็นสีออกเทาๆ จะกลายเป็นสีที่สว่างมากขึ้น และในที่สุดคือ ช่วง $r = 171$ จนถึง $r = 255$ ซึ่งจะได้ระดับสีที่ 42 ทำให้เราได้ว่าสีที่เป็นสีดำจะกลายเป็นสีขาว จากข้อสรุปนี้ ถ้าเราพิจารณาทีละภาพ จะได้ผลดังนี้

- ดอกไม้ จะเห็นว่าดอกไม้ที่เป็นสีดำ กลายเป็นสีขาว และ สีขาวกลายเป็นสีที่เข้มขึ้น
- รถยนต์ ถ้าสังเกตที่โครงรถเพียงอย่างเดียวจะเห็นได้ชัดว่า โครงรถจากขาวกลายเป็นดำ และดำกลายเป็นขาว
- รถไฟ สังเกตที่บริเวณโครงรถไฟ และหน้าต่าง จะเห็นว่าหน้าต่างจากที่มีสีเข้มๆ กลายเป็นสีขาวทั้งหมด

ข้อที่ 3) สิ่งที่น่าสนใจคือ function `power_law_transformation` ซึ่งเป็น key หลักของข้อนี้

```
import os
import cv2
import numpy as np
import pandas as pd
from IPython.display import Image, display, HTML

def power_law_transformation(img, c, gamma):
    img_normalized = img / 255.0
    transformed = c * np.power(img_normalized, gamma)
    transformed = np.clip(transformed * 255, 0, 255).astype(np.uint8)
    return transformed

def process_and_save_image(image_path, output_folder):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        print(f"Error: Couldn't load the image {image_path}.")
        return
    constants = [0.5, 1, 2]
    gammas = [0.4, 2.5]
    output_paths = []
    for c in constants:
        for gamma in gammas:
            enhanced_img = power_law_transformation(img, c, gamma)
            # Save the enhanced image
            base_name = os.path.basename(image_path)
            file_name_without_extension = os.path.splitext(base_name)[0]
            new_file_name =
f"{file_name_without_extension}_c{c}_gamma{gamma}.jpg"
            output_path = os.path.join(output_folder, new_file_name)
            cv2.imwrite(output_path, enhanced_img)
            output_paths.append(output_path)
    return output_paths

image_list =
["../original_images/cartoon.jpg", "../original_images/scenery1.jpg", "../origin
al_images/scenery2.jpg"]
all_output_paths = []







for image in image_list:
    all_output_paths.extend(process_and_save_image(image, output_folder))

# Now, display the images in a structured format using pandas
df = pd.DataFrame({
    'Image Filename': [os.path.basename(path) for path in all_output_paths],
    'Image': [f'' for path in all_output_paths]
})

# Display the DataFrame in Jupyter with images embedded
display(HTML(df.to_html(escape=False)))
```





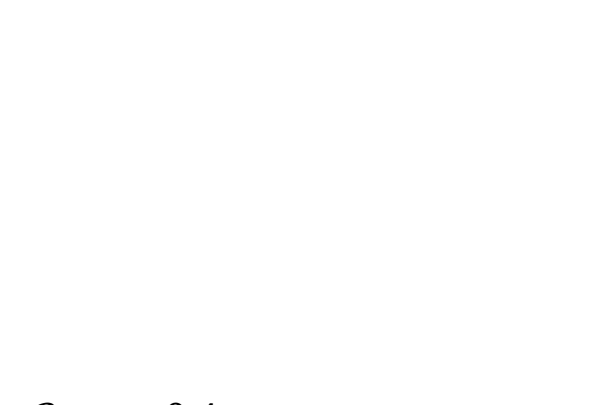

เนื่องจากเรามี c ทั้งหมด 3 ค่า และ γ ทั้งหมด 2 ค่า ดังนั้นในแต่ละภาพจะได้ combination ที่ต่างกัน 6 แบบ

ภาพแรก cartoon

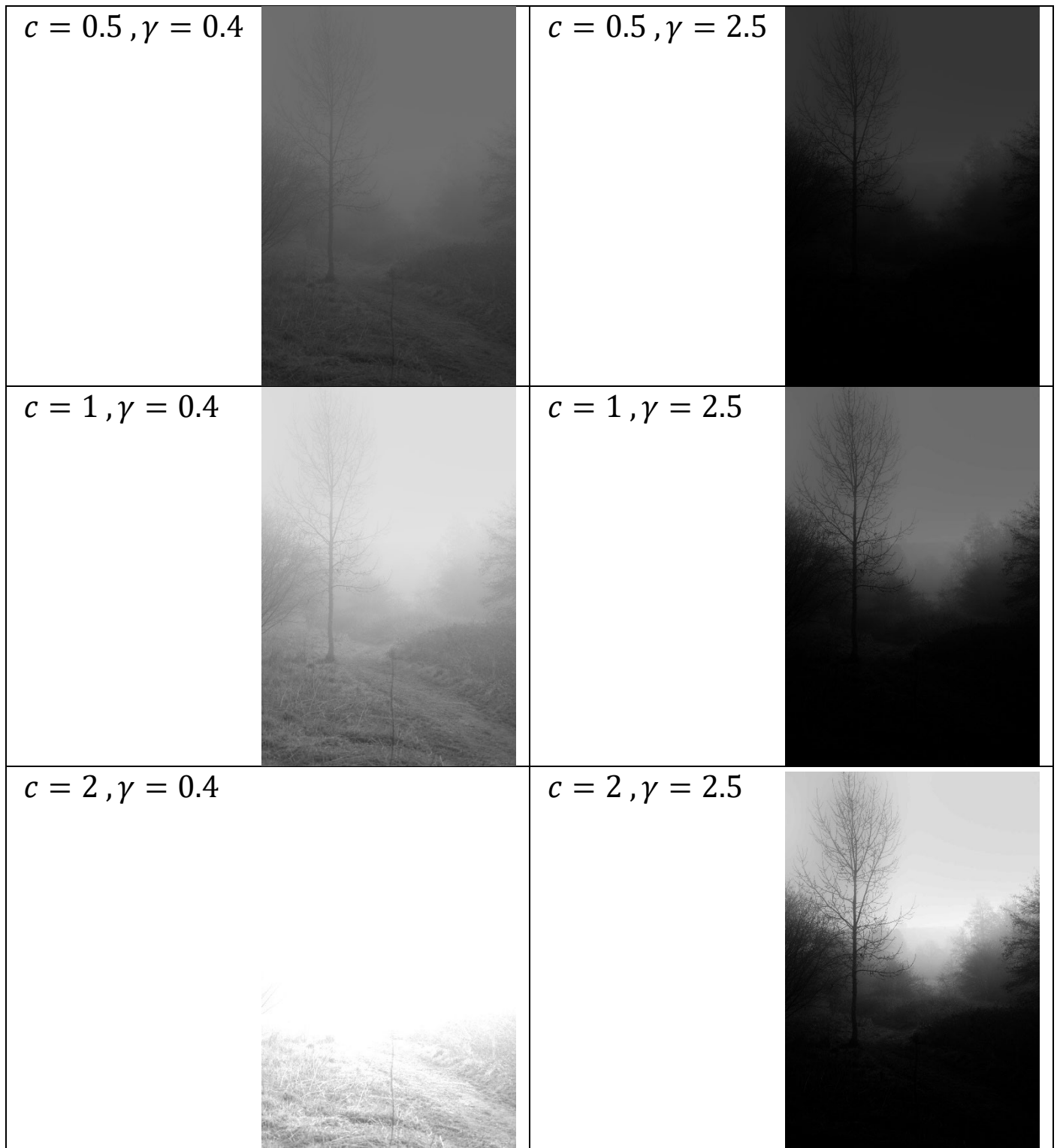
<p>Dark Aesthetic Anime PFPs</p> 	<p><i>Dark Aesthetic Anime PFPs</i></p> 
<p>$c = 0.5, \gamma = 0.4$</p>	<p>$c = 0.5, \gamma = 2.5$</p>
<p>Dark Aesthetic Anime PFPs</p> 	<p><i>Dark Aesthetic Anime PFPs</i></p> 
<p>$c = 1, \gamma = 0.4$</p>	<p>$c = 1, \gamma = 2.5$</p>
<p>Dark Aesthetic Anime PFPs</p> 	<p><i>Dark Aesthetic Anime PFPs</i></p> 
<p>$c = 2, \gamma = 0.4$</p>	<p>$c = 2, \gamma = 2.5$</p>

จาก power law transformation ถ้า γ ของเรามีค่าน้อยภาพที่ได้จะสว่างมากขึ้น และกลับกันถ้า γ มีค่ามากภาพที่ได้ก็จะมืดมากขึ้น ในขณะที่เรามีตัวแปร c โดยหาก c มีค่ามากขึ้นและเราให้ γ มีค่าเท่าเดิม ภาพที่ได้ก็จะสว่างมากขึ้น ดังนั้นจากภาพทั้ง 6 ภาพที่แสดงจะพบว่าถ้าเราดูในแต่ละแถว ที่มี c เท่ากันแต่ γ ต่างกัน ภาพที่มี γ สูงกว่าจะสว่างน้อยกว่า และถ้าพิจารณาในแต่ละคอลัมน์ จะพบว่าเมื่อค่า γ เท่ากันแต่ค่า c มากขึ้นจะทำให้ภาพสว่างมากขึ้น โดยจะเห็นว่าภาพที่มี $c = 2, \gamma = 0.4$ จะให้ภาพที่ออกมาชัดเจนมากกว่าภาพต้นฉบับ

ด้วยเหตุผลเดียวกันนี้ในภาพที่ 2 และ 3 ซึ่งก็คือ scenery1 และ scenery2 ก็ควรจะมึลักษณะของภาพตามที่กล่าวไว้ข้างต้น

	
$c = 0.5, \gamma = 0.4$	$c = 0.5, \gamma = 2.5$
	
$c = 1, \gamma = 0.4$	$c = 1, \gamma = 2.5$
	
$c = 2, \gamma = 0.4$	$c = 2, \gamma = 2.5$

ภาพนี้แสดงลายของไปไม้ เมื่อเทียบกับภาพต้นฉบับเราจะพบว่า เมื่อปรับ $c = 1, \gamma = 2.5$ เราจะได้ภาพที่มองเห็นที่ลายของไปไม้ได้อย่างชัดเจนมากขึ้น แต่ในขณะที่หากเราปรับค่า c จนสูงเกินไปอย่างในกรณีของ $c = 2, \gamma = 0.4$ จะเห็นว่าด้วยความที่ภาพถูกปรับให้มีความสว่างมากเกินไปจากค่า c เราจึงมองเห็นภาพเป็นสีขาว หรือก็คือรายละเอียดของภาพของเราได้หายไปทั้งหมด



ภาพสุดท้ายเป็นภาพป่าไม้ ในการปรับค่าแต่ละแบบก็มีข้อดีแตกต่างกัน อย่างในกรณีที่เรารับเป็น ค่า $c = 2, \gamma = 2.5$ เราจะได้รายละเอียดของกิ่งไม้ที่ชัดเจนมากขึ้น แต่หากปรับเป็นค่า $c = 1, \gamma = 0.4$ จะได้รายละเอียดในบริเวณพื้นป่าที่ชัดเจนกว่า

จากการเปรียบเทียบภาพทั้ง3ภาพ ภาพละ6แบบ พบว่าหากเรากำหนดค่า c และ γ ได้อย่างเหมาะสม จากภาพที่อาจจะไม่ค่อยชัดเจน ก็สามารถทำให้เห็นรายละเอียดได้ชัดเจนขึ้นได้