《计算机视觉》（本科，2024）作业 1

1. 将附带的彩色图像（I0）转为灰度图像（记为 I1）。

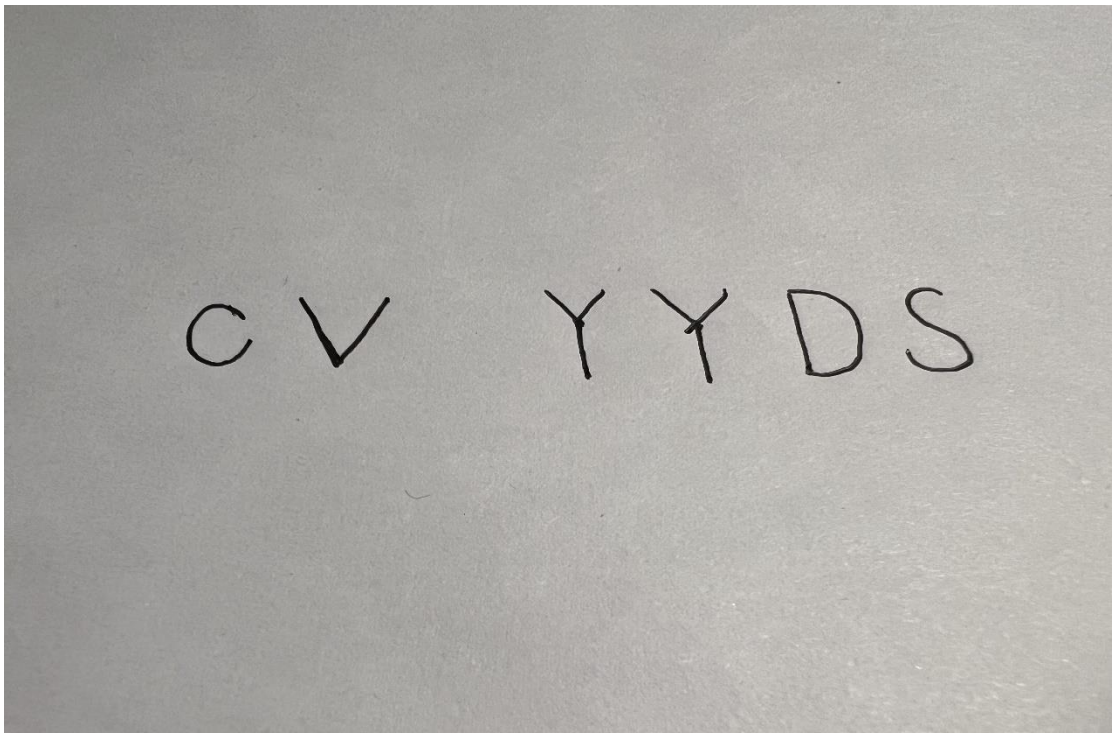
【代码贴这里】

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

# 将附带的彩色图像转化为灰度图像

image = cv2.imread(r'.\assert\I0.jpg')

# 1.计算公式 Gray = 0.3R + 0.59G + 0.11B
image_gray = image[:, :, 0] * 0.11 + image[:, :, 1] * 0.59
    + image[:, :, 2] * 0.3
image_gray = image_gray.astype(np.uint8)

# 2.cv 库自动转化为灰度图
image_gray_cv2 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow("pic", image_gray)
cv2.waitKey(0)
```

【结果（灰度图像 I1）贴这里】

2. 在白纸上手写“CV YYDS”几个字，手机拍照，转为与 I1 分辨率相同的二值图像（记为 I2）。

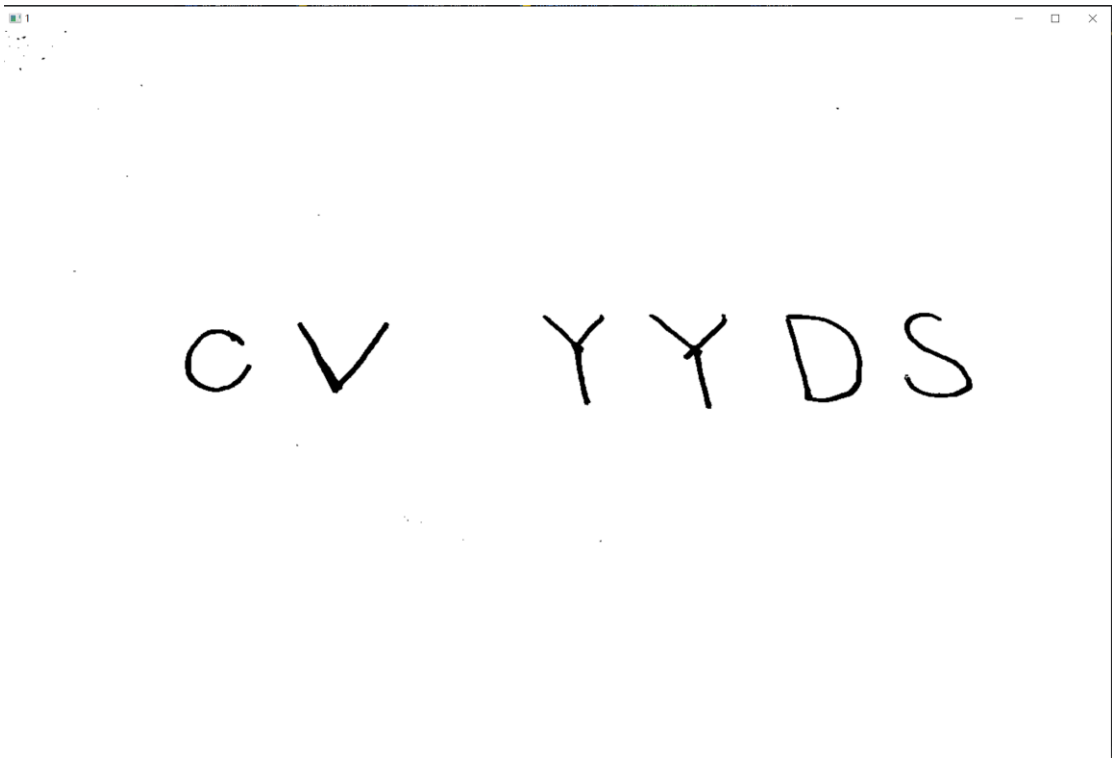
【手机拍的照片贴这里】



【代码贴这里】

```
import cv2
import numpy as np

# 将手写的图片转化为二值图像

img = cv2.imread("./assert/handwrite.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 方法1：将图片转化为灰度图像，并设置阈值，如果超过这个阈值就是黑色的，反之是白色的
theShot = 127
ret, img = cv2.threshold(img, theShot, 255, cv2.THRESH_BINARY)

cv2.imshow("1", img)
cv2.waitKey(0)
```

【结果（二值图像 I2）贴这里】



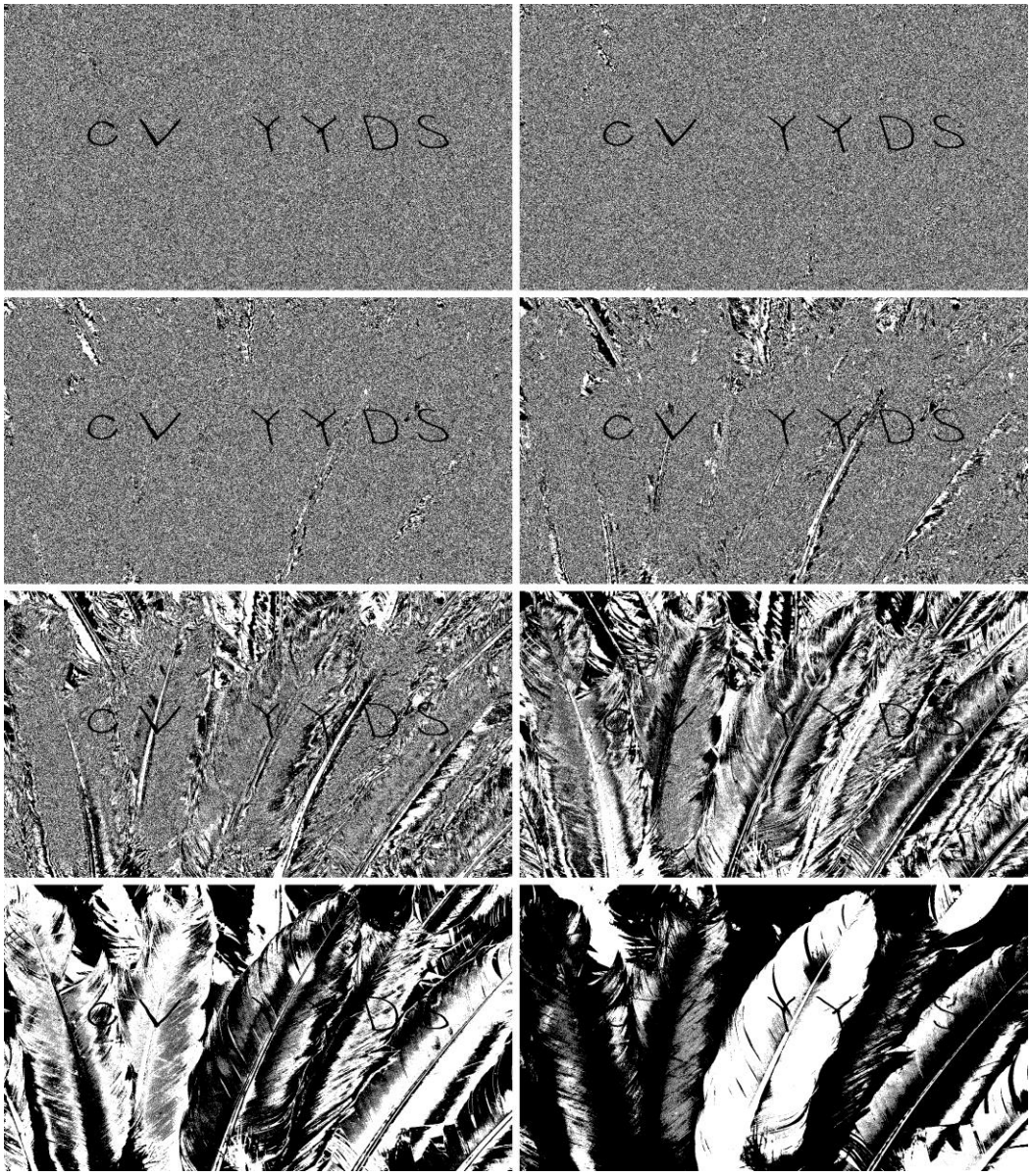
3. 灰度图像每个像素的灰度值为 1 个字节（8 位），按照从低到高记为 L1、L2、…、L8。将 I1 中每个像素的 L1、L2、…、L8 分别用 I2 替换。对结果进行分析。

【代码贴这里】

```
import cv2
import numpy as np
from PIL import Image


def merge_images(output_path, *image_paths):
    images = [Image.open(path) for path in image_paths]
    width, height = images[0].size
    padding = 20
    new_width = width * 2 + padding
    new_height = height * 4 + padding * 3
    new_image = Image.new('RGB', (new_width, new_height), color='white')
    for i, img in enumerate(images):
        x = (i % 2) * (width + padding)
        y = (i // 2) * (height + padding)
        new_image.paste(img, (x, y))
    new_image.save(output_path)


# 方法：通过将灰度照片进行与操作来提取每一位上的值，再进行替换
I1 = cv2.imread('./assert/I1.jpg')
```

```
I2 = cv2.imread('./assert/I2.jpg')
binary_images = [np.zeros_like(I1) for _ in range(8)]
for i in range(8):
    binary_images[i] = I1 & (1 << i)
    binary_images[i] = np.uint8(binary_images[i])
for i in range(8):
    binary_images[i] = np.where(binary_images[i] > 0, I2, binary_images[i])
    cv2.imwrite(f"binary_image_{i}.png", binary_images[i])

# 合并 8 张图片
image_paths = ['binary_image_{}.png'.format(i) for i in range(8)]
merge_images('./assert/merge_img.png', *image_paths)
```

【结果（L1、L2、…、L8 分别替换后生成的 8 张图像）贴这里】



【分析贴在这里】

本张图像的排列是由从左到右，从上到下的顺序替换 L1, L2, …, L8 所得来的。可以看到，随着替换的像素点值变大，图像也会从形似 I1 逐渐转化为形似 I2

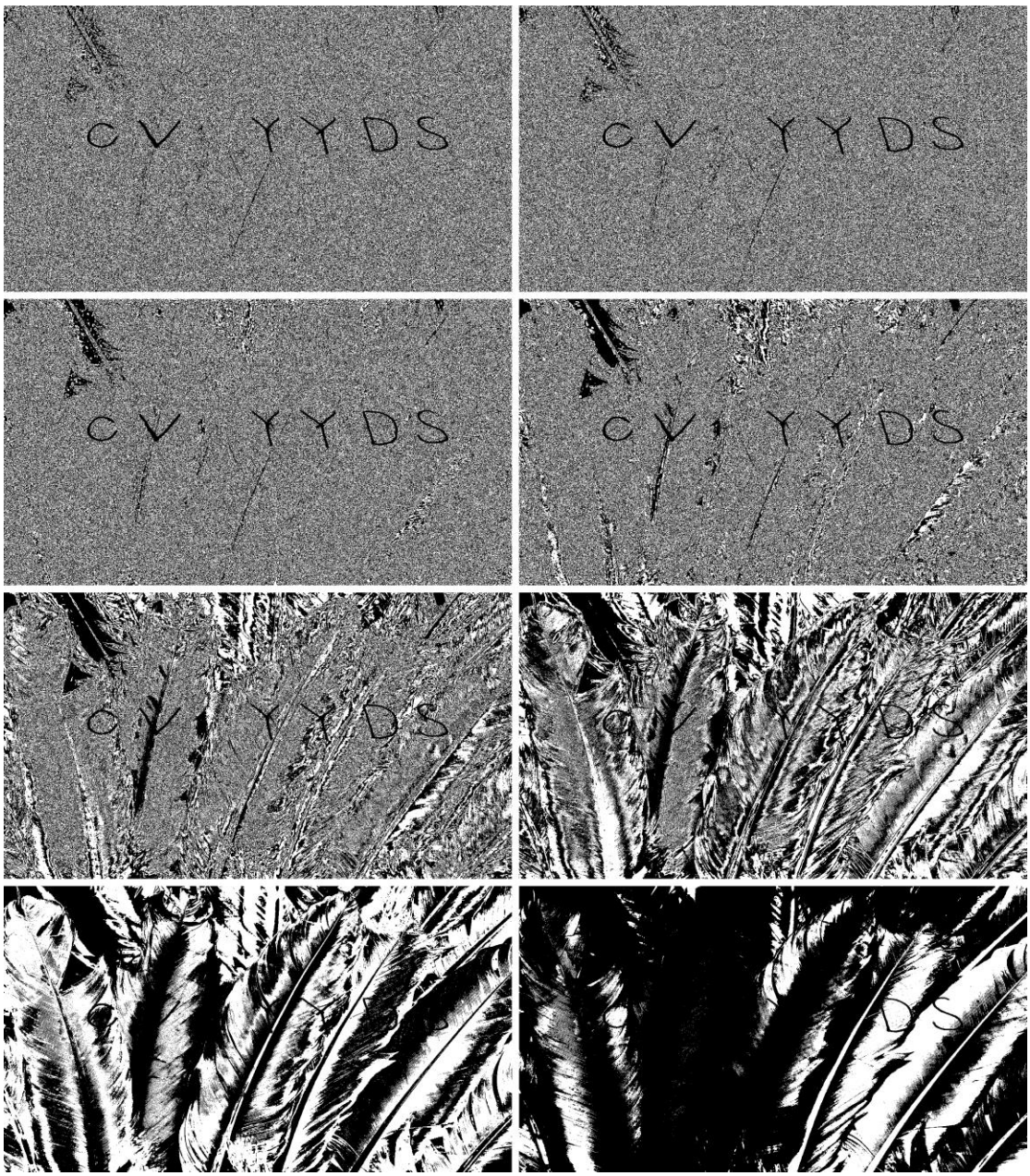
4. 将附带彩色图像（I0）的 R、G、B 通道中某个或某几个通道做与问题 3 类似的处理。对结果进行分析。

【代码贴这里】

```
import cv2
import numpy as np
from PIL import Image


def merge_images(output_path, *image_paths):
    images = [Image.open(path) for path in image_paths]
    width, height = images[0].size
    padding = 20
    new_width = width * 2 + padding
    new_height = height * 4 + padding * 3
    new_image = Image.new('RGB', (new_width, new_height), color='white')
    for i, img in enumerate(images):
        x = (i % 2) * (width + padding)
        y = (i // 2) * (height + padding)
        new_image.paste(img, (x, y))
    new_image.save(output_path)


# 方法：通过将灰度照片进行与操作来提取每一位上的值，再进行替换
I0 = cv2.imread('./assert/I0.jpg')
I2 = cv2.imread('./assert/I2.jpg')
blue_images = [np.zeros_like(I0) for _ in range(8)]
for i in range(8):
    blue_images[i] = I0[:, :, 0] & (1 << i)
    blue_images[i] = np.uint8(blue_images[i])
for i in range(8):
    blue_images[i] = np.where(blue_images[i] > 0, I2[:, :, 0], blue_images[i])
    blue_images[i] = cv2.merge([blue_images[i], blue_images[i], blue_images[i]])
    cv2.imwrite(f"blue_image_{i}.png", blue_images[i])

# 合并 8 张图片
image_paths = ['blue_image_{}.png'.format(i) for i in range(8)]
```

```
34.merge_images('./assert/blue_img.png', *image_paths)
```

【结果（替换后生成的若干张图像）贴这里】



【分析贴在这里】

这里分析的是蓝色单通道的图像变换，呈现的结果和问题 3 相似，细微之处在于原本图像中蓝色较多的地方在变换中多为白色，而蓝色少的地方多为黑色。

5. 对附带彩色图像（I0）进行亮度变换、对比度变换、饱和度变换（变换程度自行确定）。对结果进行分析。

【代码贴这里】

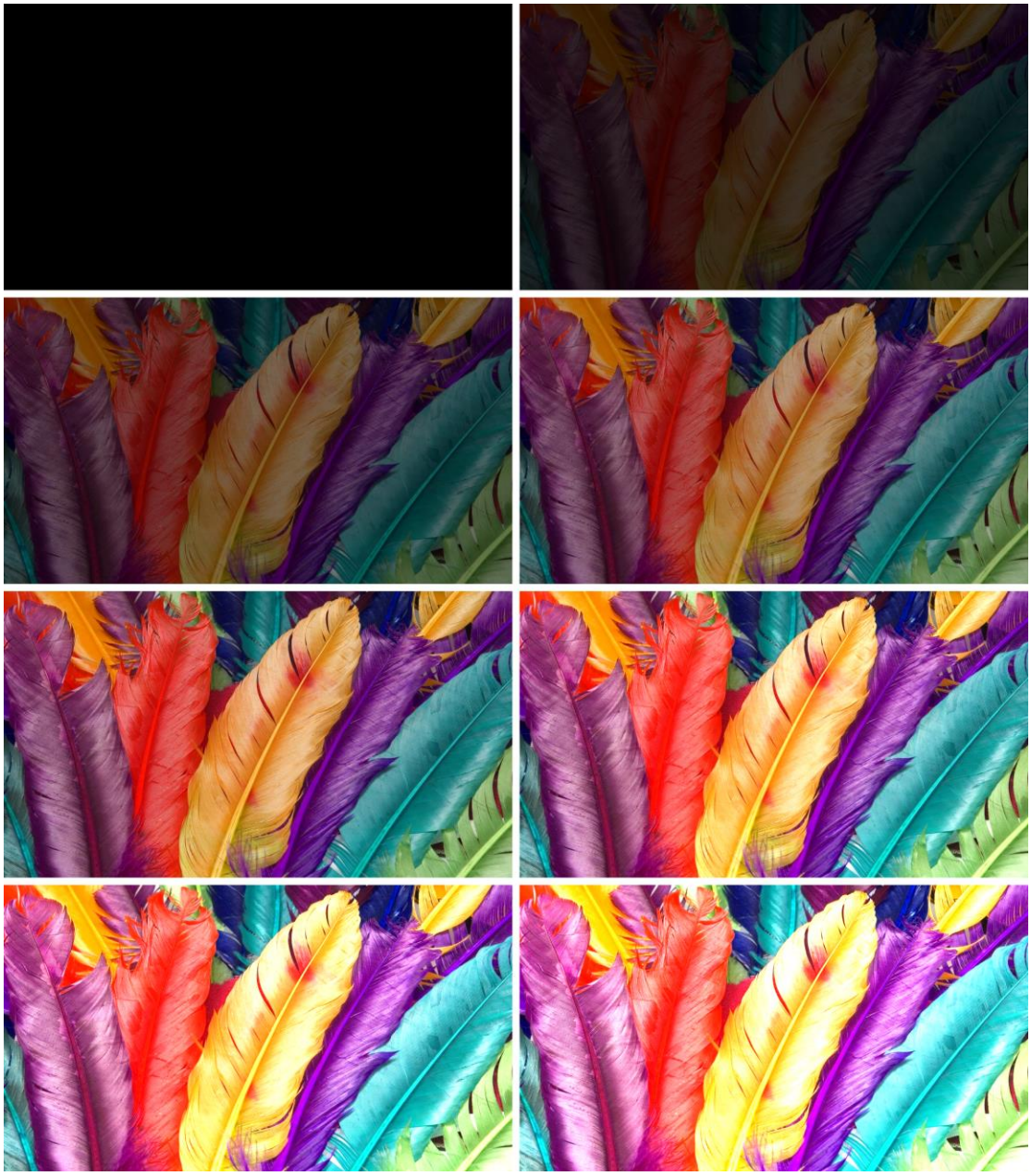
```
1. import cv2
2. import numpy as np
```

```
from PIL import Image


def merge_images(output_path, *image_paths):
    images = [Image.open(path) for path in image_paths]
    width, height = images[0].size
    padding = 20
    new_width = width * 2 + padding
    new_height = height * 4 + padding * 3
    new_image = Image.new('RGB', (new_width, new_height), color='white')
    for i, img in enumerate(images):
        x = (i % 2) * (width + padding)
        y = (i // 2) * (height + padding)
        new_image.paste(img, (x, y))
    new_image.save(output_path)


image = cv2.imread('./assert/I0.jpg')

# 亮度变换
for i in range(8):
    tmp = cv2.convertScaleAbs(image, beta=(-120 + i * 30))
    cv2.imwrite(f"light{i}_img.png", tmp)
image_paths = ['light{}_img.png'.format(i) for i in range(8)]
merge_images('./assert/light_img.png', *image_paths)

# 对比度变换
for i in range(8):
    tmp = cv2.convertScaleAbs(image, alpha=(0 + i * 0.25))
    cv2.imwrite(f"contract{i}_img.png", tmp)
image_paths = ['contract{}_img.png'.format(i) for i in range(8)]
merge_images('./assert/contract_img.png', *image_paths)

# 饱和度变化
for i in range(8):
    hls = cv2.cvtColor(image, cv2.COLOR_BGR2HLS)
    hls[:, :, 2] = np.clip(30 * i, 0, 255).astype(np.uint8)
    tmp = cv2.cvtColor(hls, cv2.COLOR_HLS2BGR)
    cv2.imwrite(f"saturation{i}_img.png", tmp)
image_paths = ['saturation{}_img.png'.format(i) for i in range(8)]
```

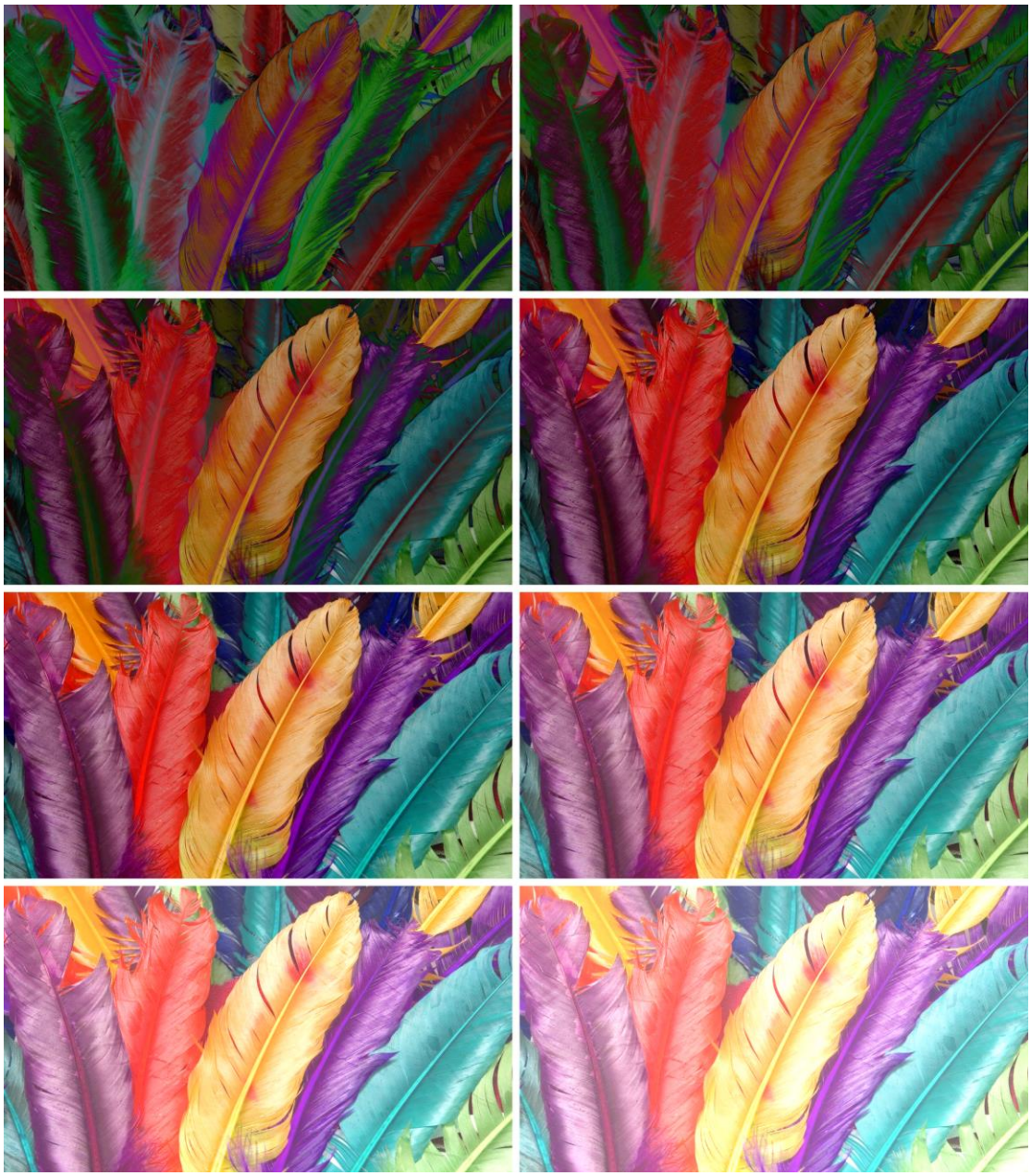
```
43.merge_images('./assert/saturation_img.png', *image_paths)
```

【结果（变换后生成的若干张图像）贴这里】

对比度图片



亮度图片



饱和度图片

【分析贴在这里】

饱和度：增加图像的饱和度会使颜色更加鲜艳，减少饱和度会使颜色变得更加灰暗。饱和度的范围通常是［0，1］或［0，100］，其中 0 表示灰度图像，1 或 100 表示最大饱和度。

亮度：增加图像的亮度可以改善图像的视觉效果，使图像更加清晰和易于观察；降低亮度可以增强图像的对比度，使图像的细节更加突出。

对比度：增加对比度会增强图像的细节和清晰度，使图像更加生动；降低对比度会减少图像的细节，使图像看起来更加平坦。调整对比度可以改变图像的视觉效果和外观。