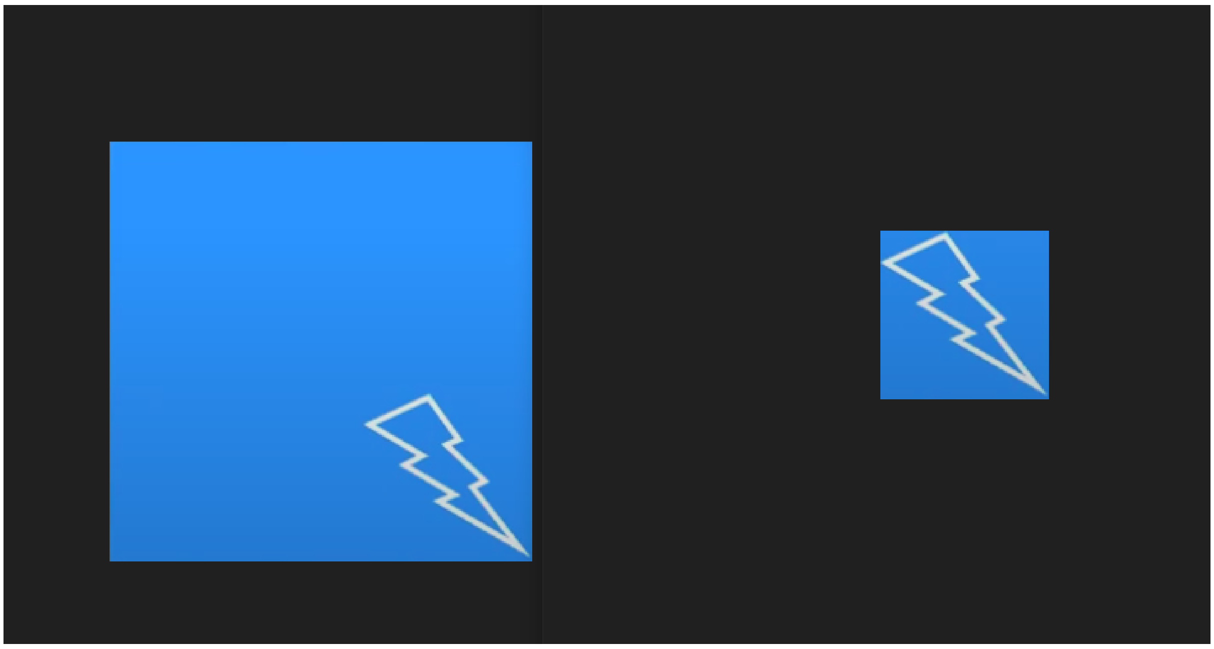
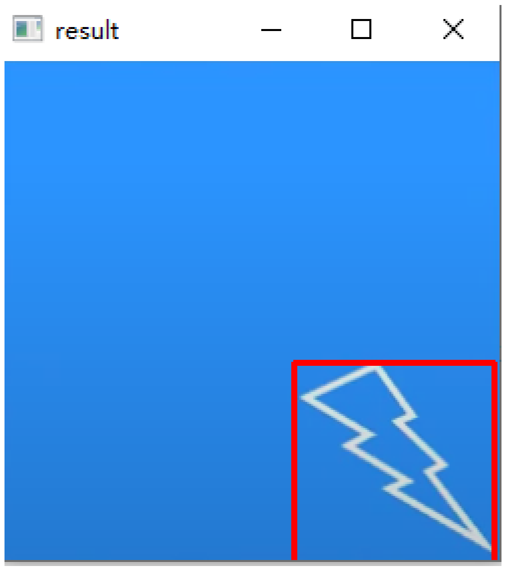
9テンプレートマッチングupload版

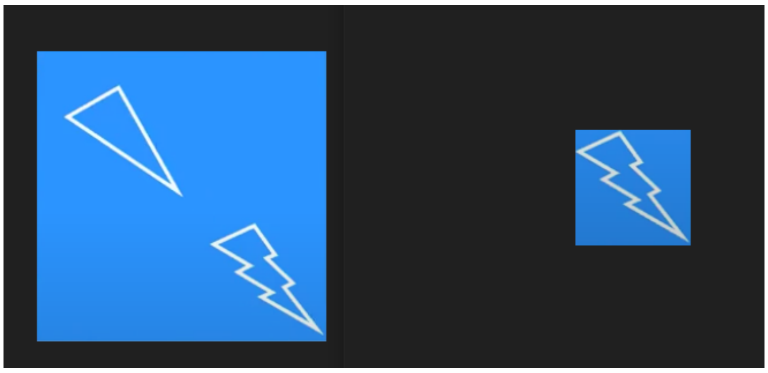
Today, I used Python for image similarity calculation and matching. While SSD and SAD methods worked well, the Pyramid Image Matching method was faster and more accurate, which is very impressive.

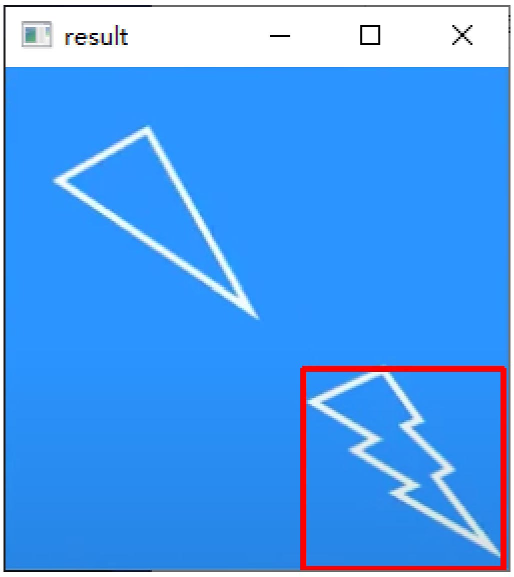
SSD

```
import cv2 as cv
import numpy as np

img = cv.imread('images/11.png').astype(np.float32)
H, W, C = img.shape
mi = np.mean(img)
temp = cv.imread('images/12.png').astype(np.float32)
Ht, Wt, Ct = temp.shape
mt = np.mean(temp)
i, j = -1, -1
v = -1
for y in range(H - Ht):
    for x in range(W - Wt):
        _v = np.sum((img[y:y+Ht, x:x+Wt] - temp) ** 2)
        if _v < v or v == -1:
            v = _v
            i, j = x, y
out = img.copy()
# rectangle draw function in opencv
cv.rectangle(out, (j, i), (j+Wt, i+Ht), (0, 0, 255), 2)
out = out.astype(np.uint8)
cv.imshow('result', out)
cv.waitKey(0)
cv.destroyAllWindows()
```



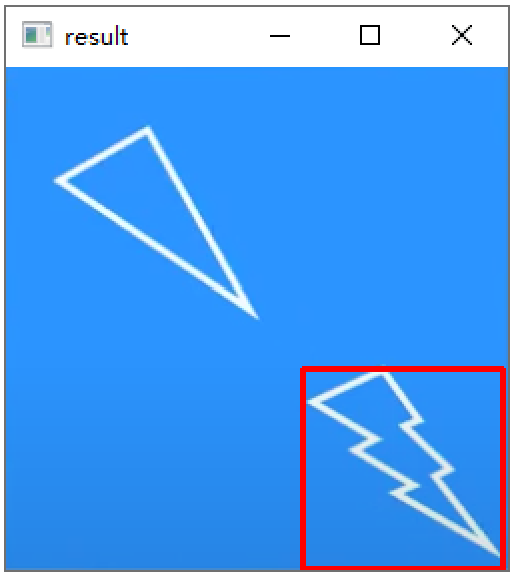

result





SAD

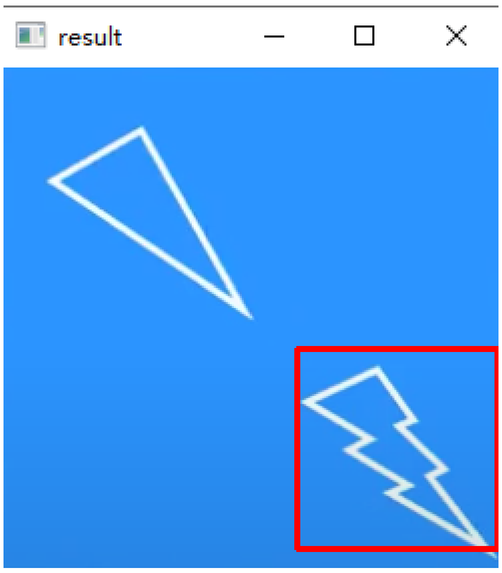
```
_v = np.sum(np.abs(img[y:y+Ht, x:x+Wt] - temp))
```



NCC（Normalization Cross Correlation）

The running time is slightly longer

```
_v = np.sum(img[y:y+Ht, x:x+Wt] * temp)
_v /= (np.sqrt(np.sum(img[y:y+Ht, x:x+Wt]**2)) * np.sqrt(n
p.sum(temp**2)))
```



Calculating similarity

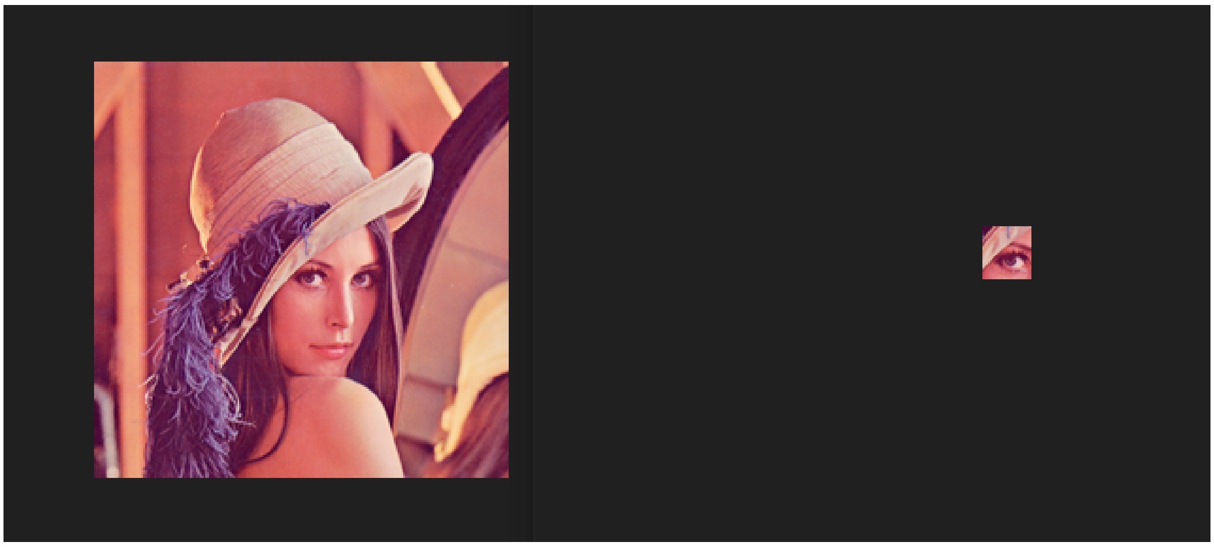
```
from PIL import Image
from matplotlib.pyplot import show
from numpy import average, dot, linalg
import numpy as np

def get_thum(image, size=(64, 64), greyscale=False):
    image = image.resize(size, Image.LANCZOS)
    if greyscale:
        image = image.convert('L')
    return image

def MSE(image1,image2):
    image1 = get_thum(image1)
    image2 = get_thum(image2)
    image1 = np.array(image1)
    image2 = np.array(image2)
    mse = np.mean( (image1 - image2) ** 2 )
    return mse

def image_similarity_voctors_via_numpy(image1, image2):
    image1 = get_thum(image1)
    image2 = get_thum(image2)
    images = [image1, image2]
    vectors = []
    norms = []
    for image in images:
        vector = []
        for pixel_tuple in image.getdata():
            vector.append(average(pixel_tuple))
        vectors.append(vector)
        # linalg=linear+algebra,norm
        norms.append(linalg.norm(vector, 2))
    a, b = vectors
    a_norm, b_norm = norms
    res = dot(a / a_norm, b / b_norm)
    return res

image1 = Image.open('images/1.png')
image2 = Image.open('images/2.png')
image3 = Image.open('images/3.png')
image4 = Image.open('images/4.png')

print(" 1 and 2 similarity:",image_similarity_voctors_via_n
umpy(image1, image2))
print(" 1 and 3 similarity:",image_similarity_voctors_via_n
umpy(image1, image3))
print(" 1 and 4 similarity:",image_similarity_voctors_via_n
umpy(image1, image4))
print(" 1 and 2 similarity:",MSE(image1, image2))
print(" 1 and 3 similarity:",MSE(image1, image3))
print(" 1 and 4 similarity:",MSE(image1, image4))
```

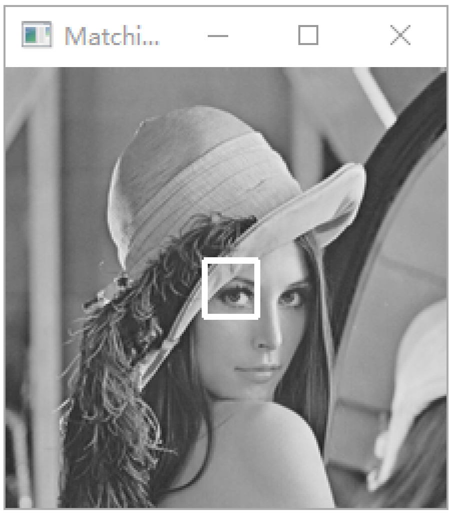
```
1 and 2 similarity: 0.9980938645843469
1 and 3 similarity: 0.9979211475013584
1 and 4 similarity: 0.9198669259513603
1 and 2 similarity: 53.384765625
1 and 3 similarity: 56.45947265625
1 and 4 similarity: 76.5054931640625
```

Pyramid Image Matching

```
import cv2
import numpy as np

def build_gaussian_pyramid(image, levels):
    pyramid = [image]
    for i in range(levels - 1):
        image = cv2.pyrDown(image)
        pyramid.append(image)
    return pyramid

def pyramid_match(template, target, pyramid_levels):
    template_pyramid = build_gaussian_pyramid(template, pyr
amid_levels)
    target_pyramid = build_gaussian_pyramid(target, pyramid
_levels)

    best_match = None
    best_location = None
    for level in range(pyramid_levels - 1, -1, -1):
        template_resized = template_pyramid[level]
        target_resized = target_pyramid[level]
        result = cv2.matchTemplate(target_resized, template
_resized, cv2.TM_SQDIFF_NORMED)
        min_val, _, min_loc, _ = cv2.minMaxLoc(result)

        if best_match is None or min_val < best_match:
            best_match = min_val
            best_location = (min_loc[0] * (2 ** level), min
_loc[1] * (2 ** level))
    return best_location

if __name__ == "__main__":
    target_image = cv2.imread("images/lenna.png", cv2.IMREA
D_GRAYSCALE)
    template_image = cv2.imread("images/lenna2.png", cv2.IM
READ_GRAYSCALE)
```

```
    if target_image is None or template_image is None:
        print("Error: Could not load images.")
        exit()
    levels = 3
    match_location = pyramid_match(template_image, target_i
mage, levels)
    h, w = template_image.shape[:2]
    top_left = match_location
    bottom_right = (top_left[0] + w, top_left[1] + h)
    result_image = cv2.rectangle(target_image.copy(), top_l
eft, bottom_right, 255, 2)
    cv2.imshow("Matching Result", result_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```




Matchi...