Algorithms of Image Matching ——Implementation & Comparison

Group 4

Emma (Ziyi Guo)

Luna (Hailu Qiu)

CPS*3320 W01

Instructor: Nasser Mustafa

Content

00 ⊕ Intuitive

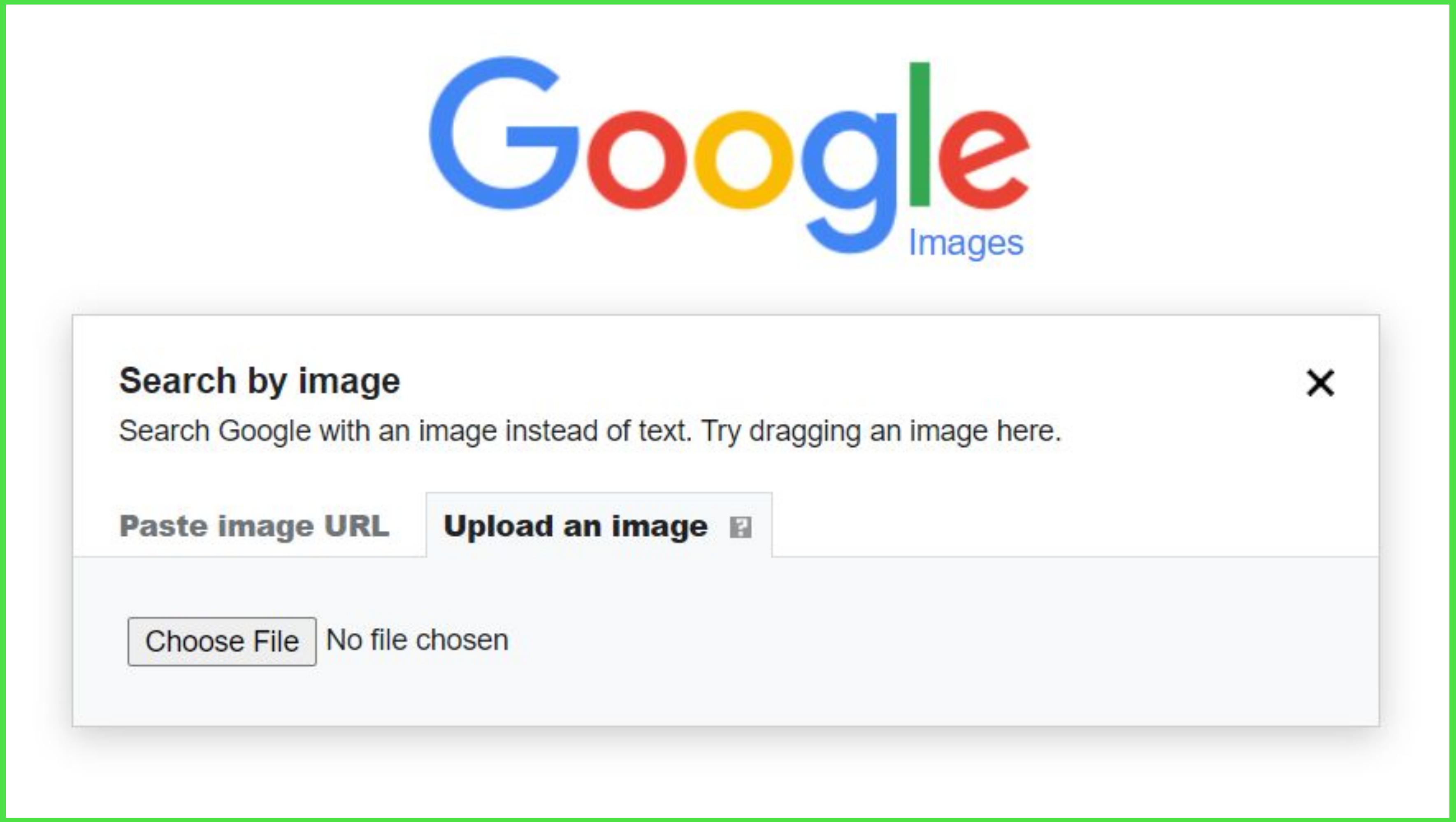
01 ⊕ General Description

02 ⊕ Algorithm & Preform Results

- 00 -

Intuitive

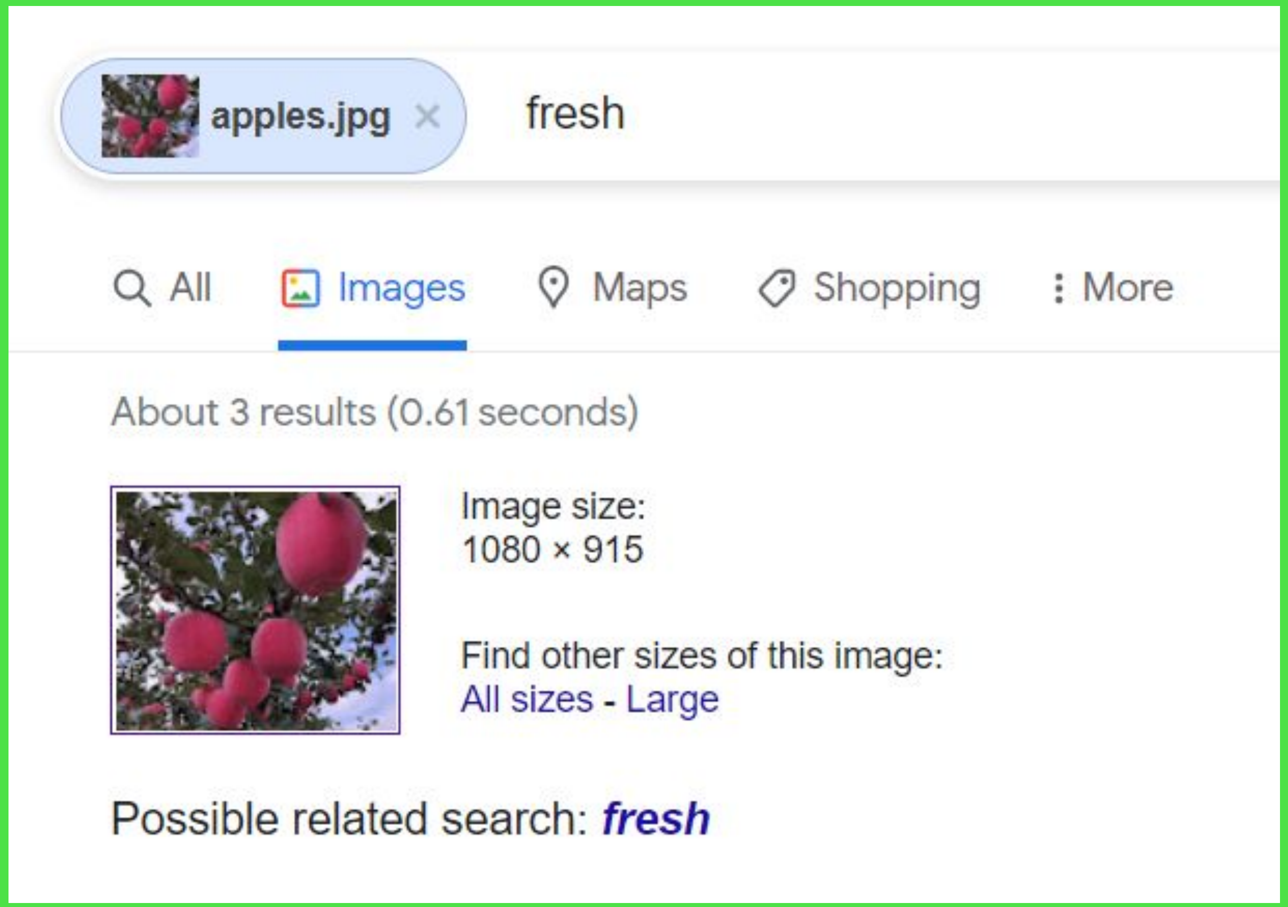
Intuitive



Search by image

Search Google with an image instead of text. Try dragging an image here.

No file chosen



apples.jpg fresh

About 3 results (0.61 seconds)

Image size:
1080 × 915

Find other sizes of this image:
All sizes - Large

Possible related search: ***fresh***

- 01 -

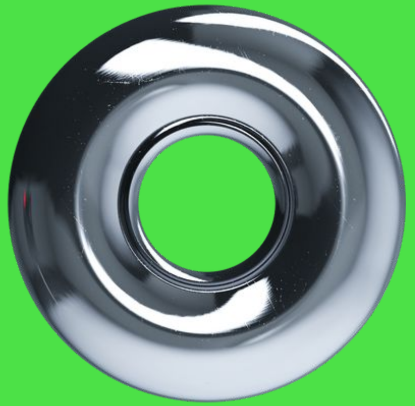
General Description

Level 1: Pixel-Level Similarity

Abstract

Each corresponding pixel value of the two images is exactly equal, which is directly manifested by the fact that the two image files are identical in terms of binary content

Key words 2D Grayscale Array



Level 2: Global Visual Similarity

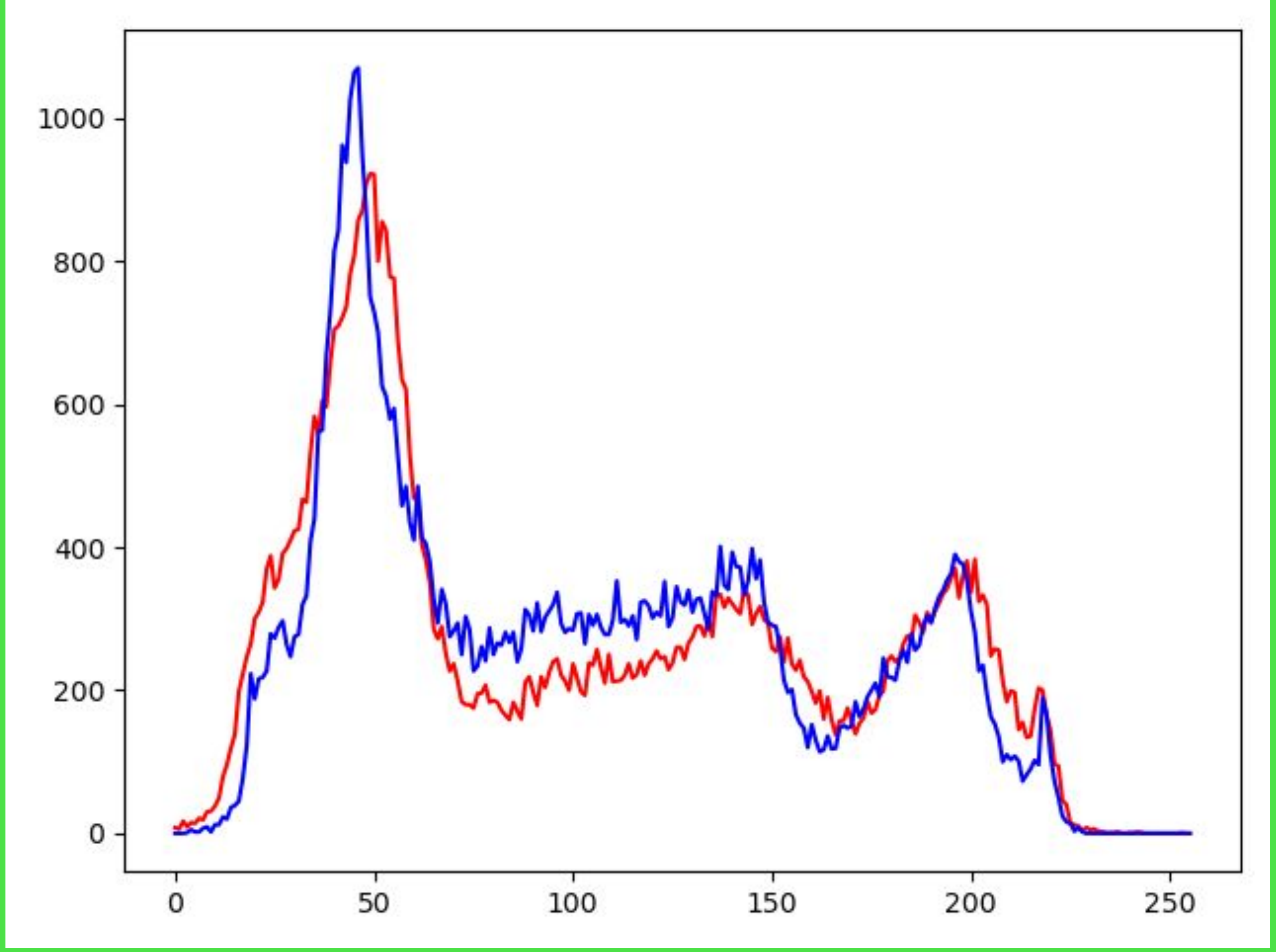
Abstract

After two pixel-level similar images are scaled and compressed respectively, their corresponding pixel values also change to some extent due to scaling or compression, but remain visually identical.

Key words

Simple standardization of difference, grayscale histogram, hash-perceptual image hashing

Histogram Calculation



- **gi** and **si** are **ith** point on two curves, respectively
- The result is the similarity

$$\frac{1}{N}\sum_{i=1}^{N}\left(1-\frac{\left|g_i-s_i\right|}{Max(g_i,s_i)}\right)$$

Histogram Calculation

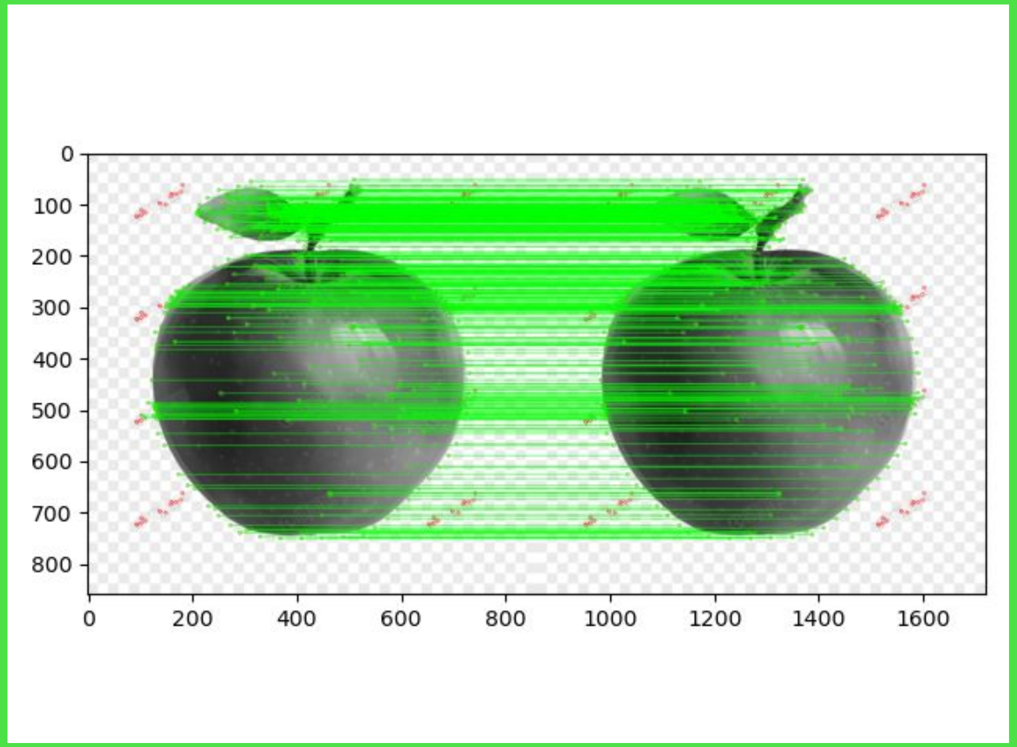
Weakness: it looks at the global distribution of colors, unable to describe the local distribution of colors and where they are located.

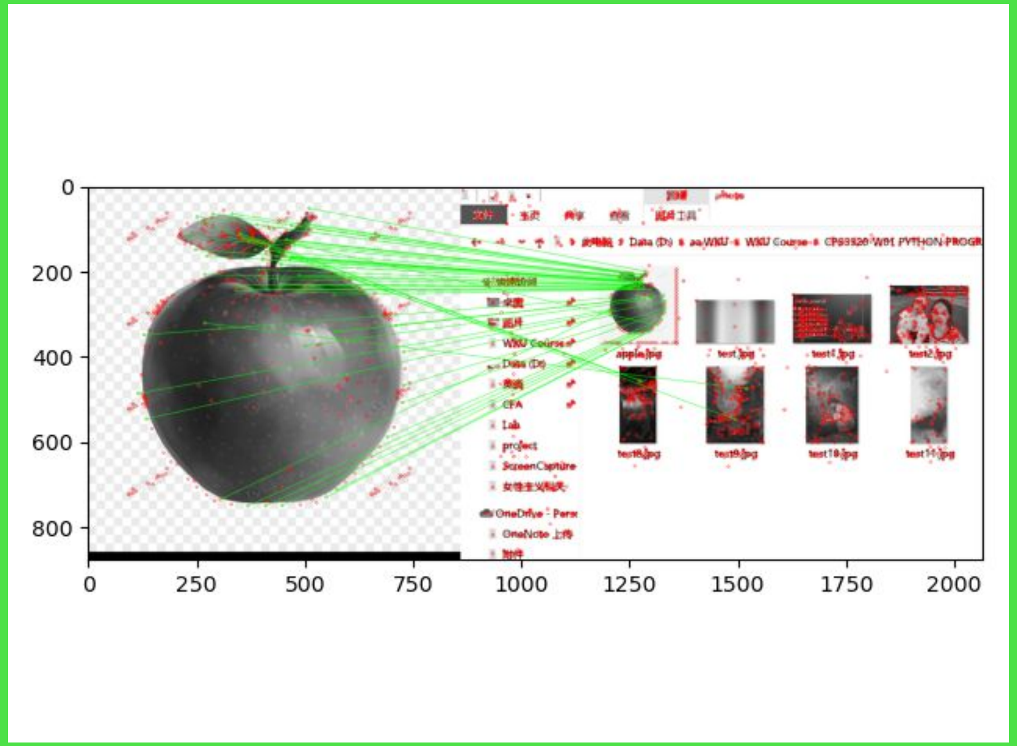
Level 3: Feature Similarity

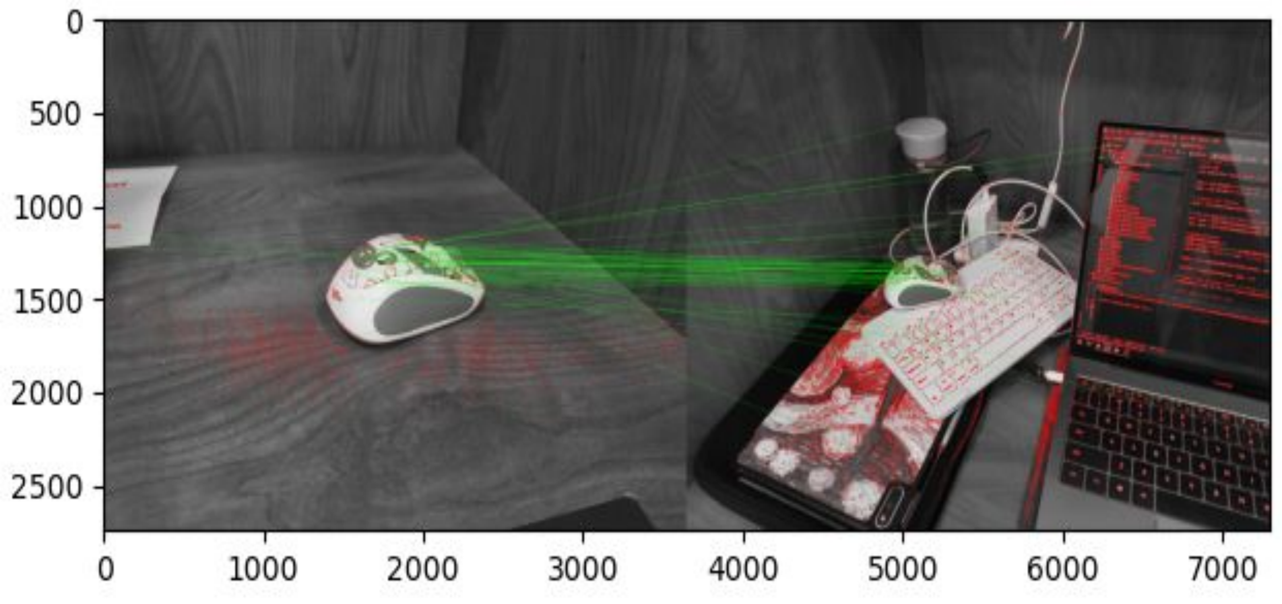
Abstract Determines whether the features are matching.

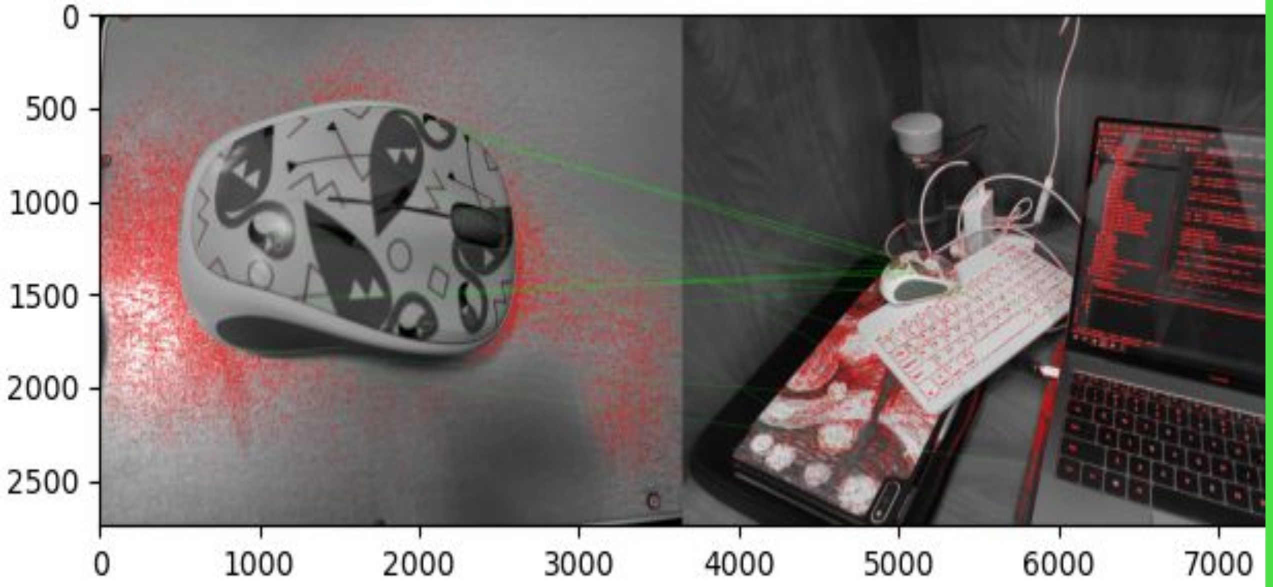
Key words ORB algorithm, Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF)

Feature Similarity: One example









Output

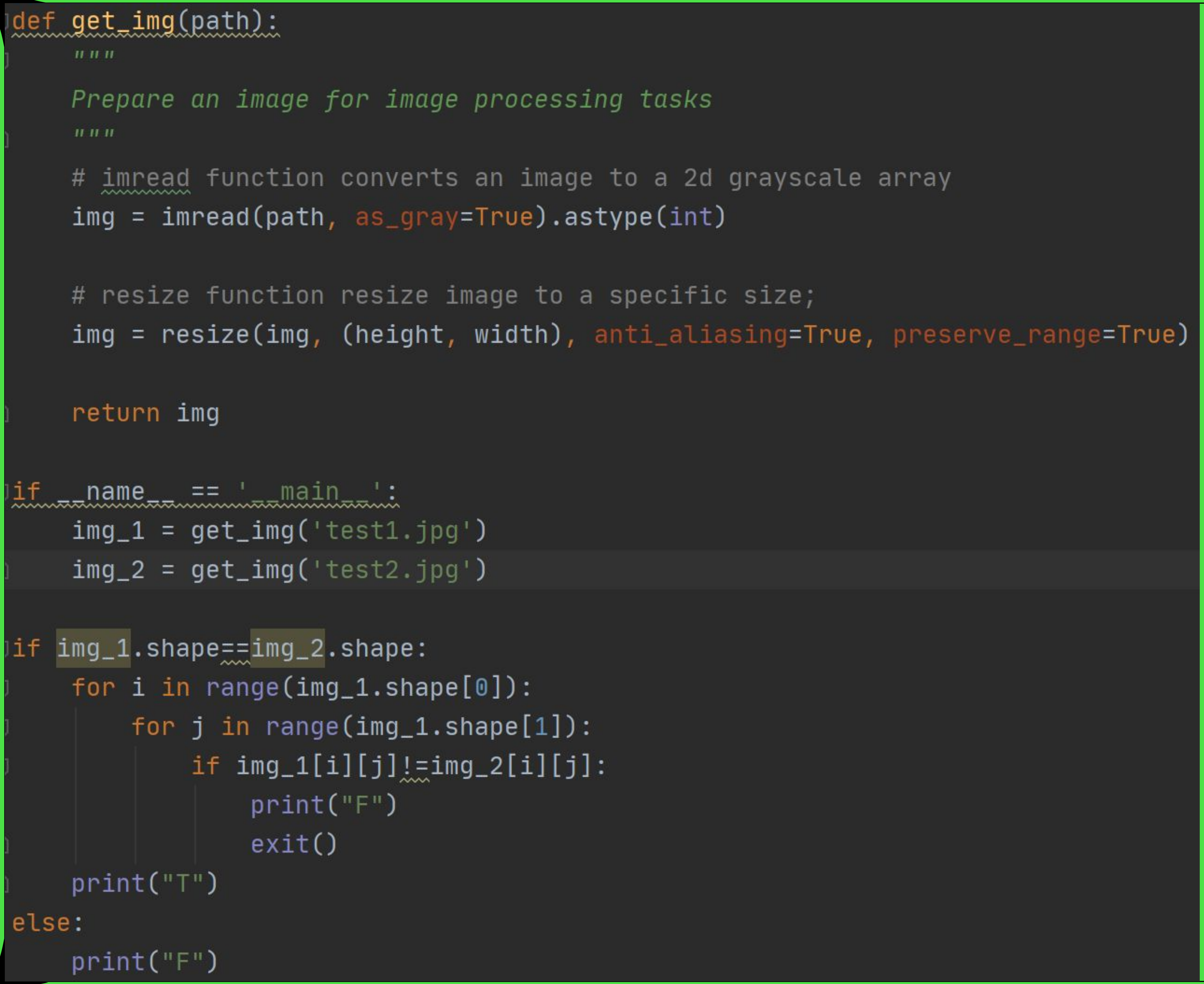
Print “T” or “F” for above similarity in each level. The interpretation of content we will print show in below rows:

1. TTT: Similar in pixel-level; Similar in global visual; Similar in feature matching
2. FTT: Dissimilar in pixel-level; Similar in global visual; Similar in feature matching
3. FFT: Dissimilar in pixel-level; Dissimilar in global visual; Similar in feature matching
4. FFF: Dissimilar in pixel-level; Dissimilar in global visual; Dissimilar in feature matching

- 02 -

Algorithms & perform results

Level 1: Pixel-Level Similarity

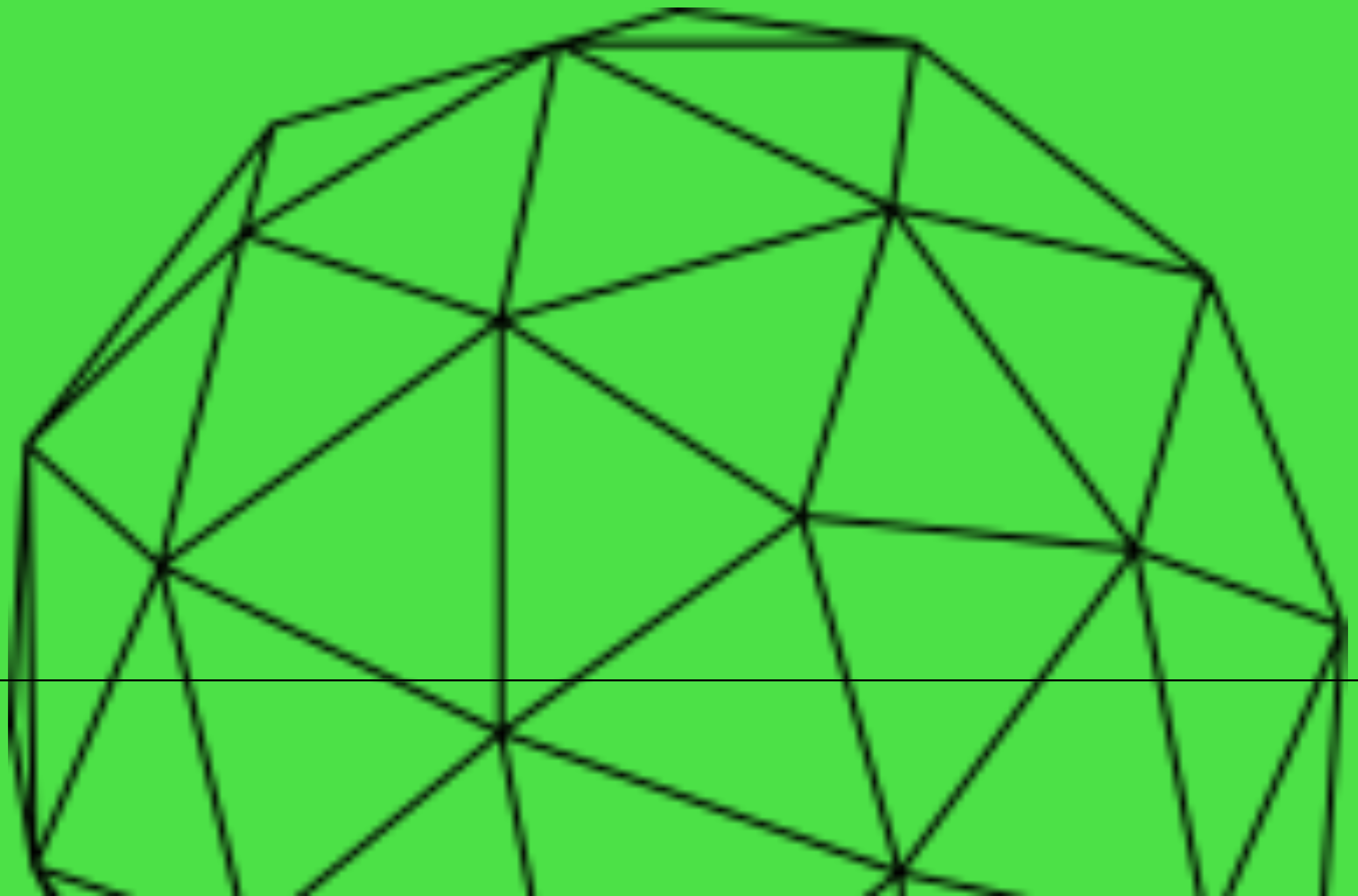


```python
def get_img(path):
    """
    Prepare an image for image processing tasks
    """
    # imread function converts an image to a 2d grayscale array
    img = imread(path, as_gray=True).astype(int)

    # resize function resize image to a specific size;
    img = resize(img, (height, width), anti_aliasing=True, preserve_range=True)

    return img

if __name__ == '__main__':
    img_1 = get_img('test1.jpg')
    img_2 = get_img('test2.jpg')

if img_1.shape==img_2.shape:
    for i in range(img_1.shape[0]):
        for j in range(img_1.shape[1]):
            if img_1[i][j]!=img_2[i][j]:
                print("F")
                exit()
    print("T")
else:
    print("F")
```

- Only return “T” when 2 images are exactly same.

- Simply compare 2 image data, if it finds 1 different value, then it will return “F”.

Level 2: Global Visual Similarity



Algorithms involving :

- Simple standardization of difference
- Grayscale histogram
- Hash - Perceptual image hashing
  - ○ Average hash (aHash)
  - ○ Perceptual hash (pHash)
  - ○ Dynamic hash (dHash)

L2: ① Simple Standardization of Difference

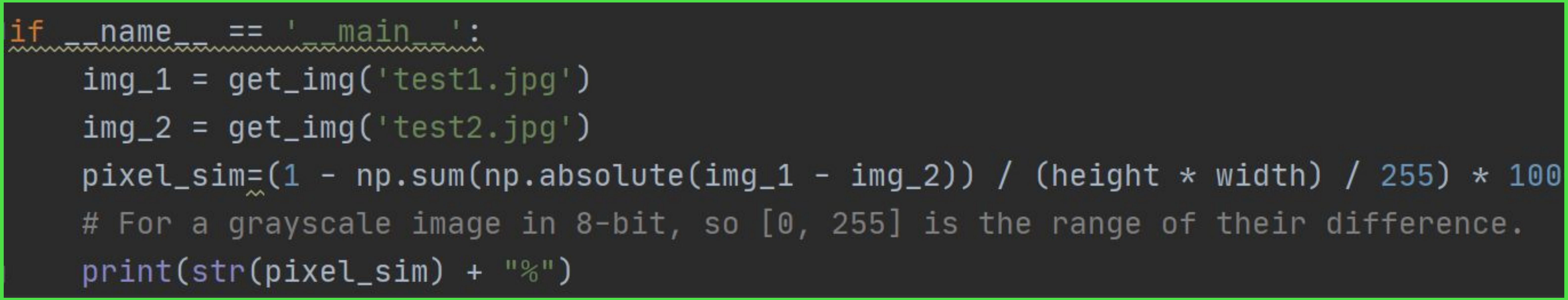
⊕ Steps:

1. convert images to 2d grayscale arrays
2. resize images to a specific size
3. define the similarity by a function as below:

$$\frac{1}{N}\sum_{i-1}^{N}\left(1-\frac{|g_i - s_i|}{Max(g_i, s_i)}\right)$$

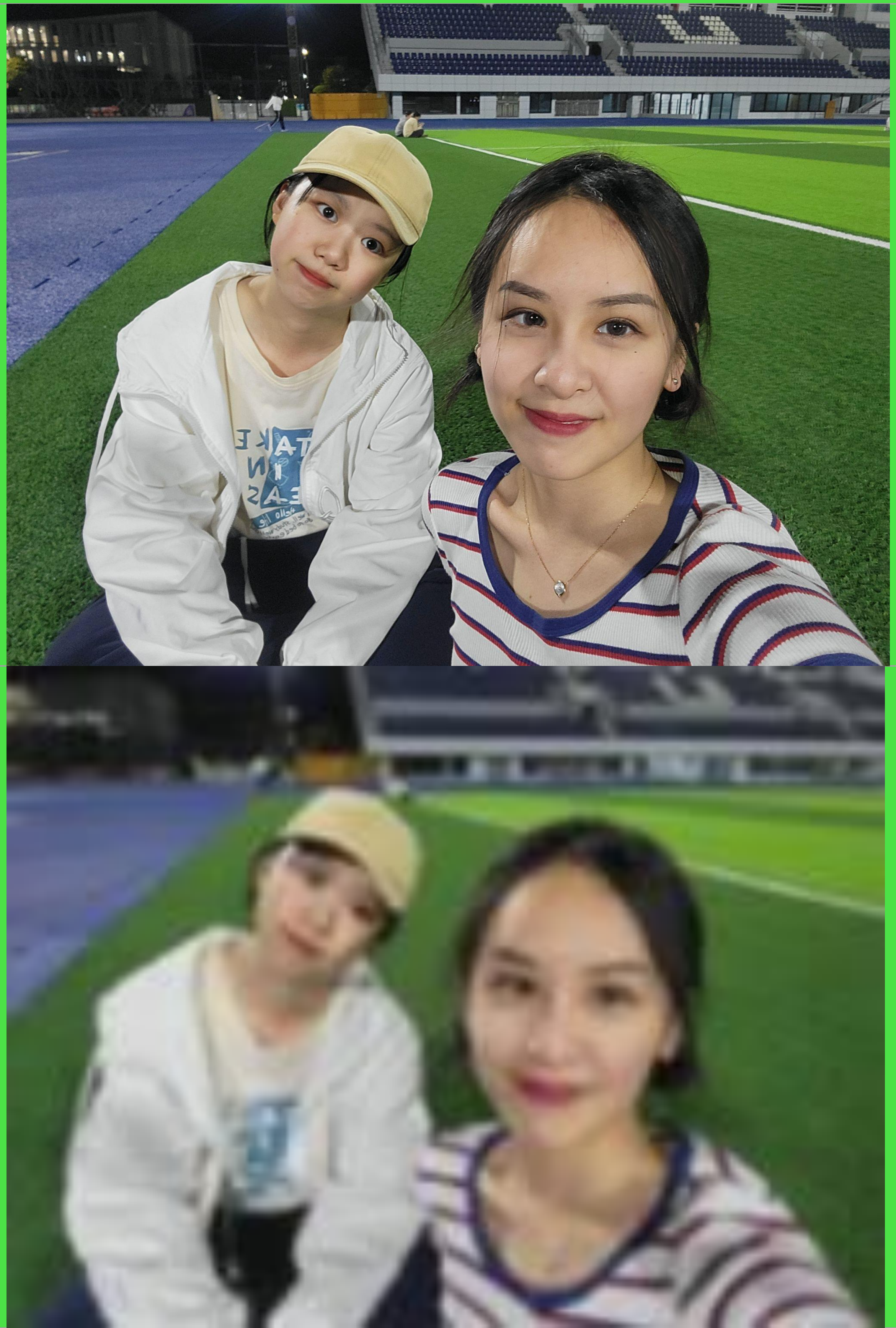
⊕ After many trials, we defined that:

similarity >=90%	same
similarity [80, 90]%	similar
similarity <= 80%	unacceptable

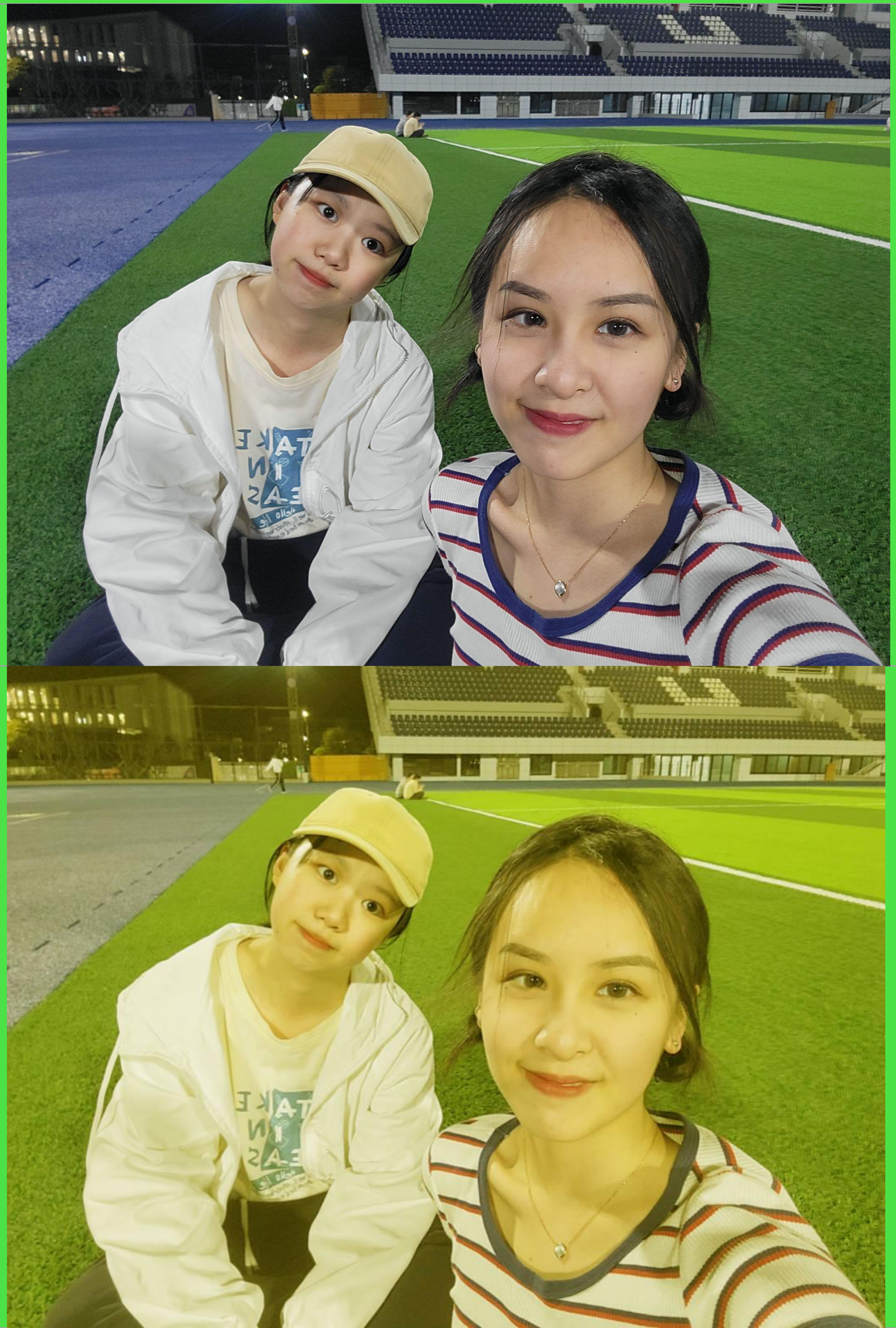


```
if __name__ == '__main__':
    img_1 = get_img('test1.jpg')
    img_2 = get_img('test2.jpg')
    pixel_sim=(1 - np.sum(np.absolute(img_1 - img_2)) / (height * width) / 255) * 100
    # For a grayscale image in 8-bit, so [0, 255] is the range of their difference.
    print(str(pixel_sim) + "%")
```

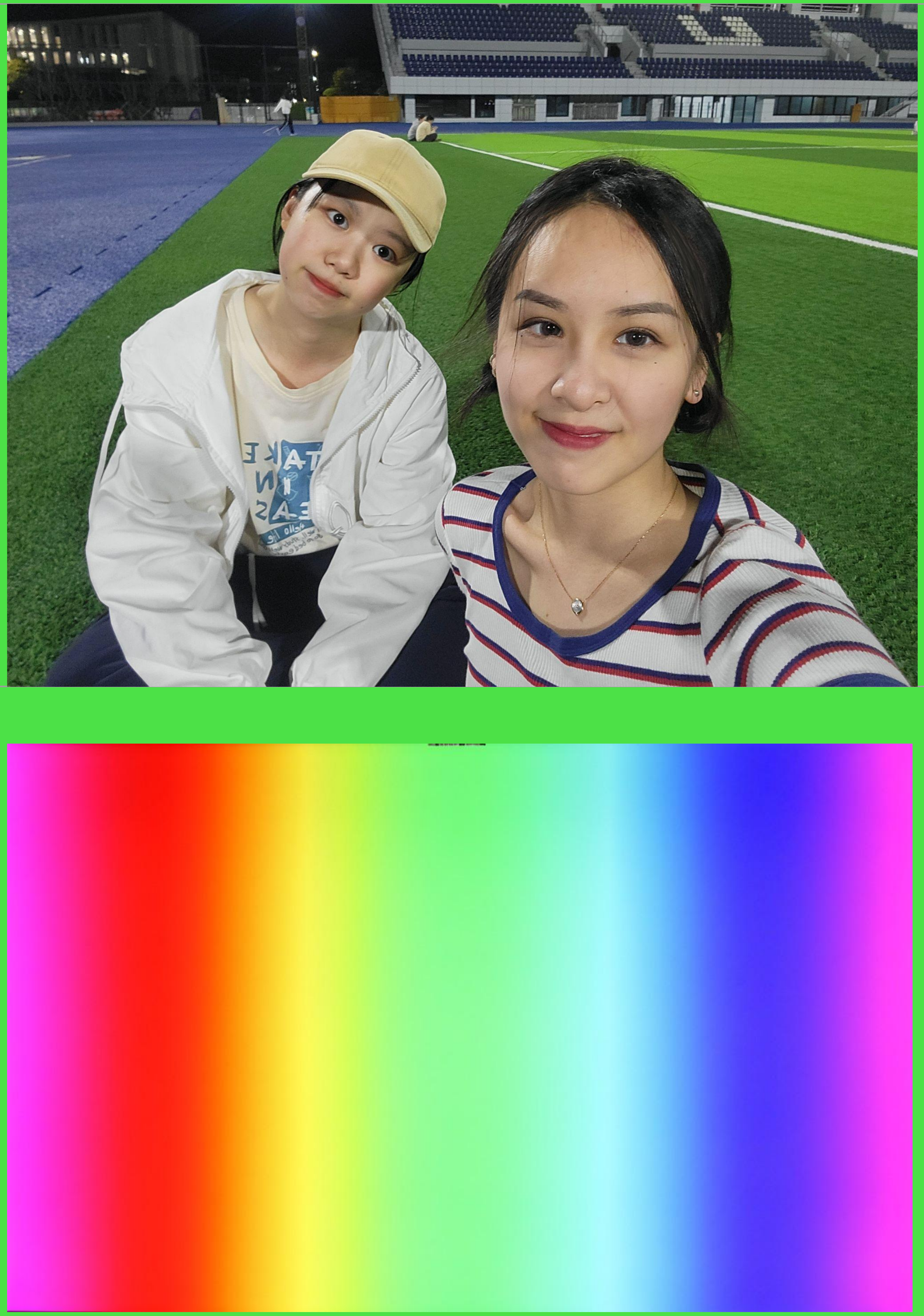
similarity = 91.7%
blurred



similarity = 84.7%
color intensity



similarity = 72.4%
control group



L2: ① Simple Standardization of Difference

Advantages:

- Easy & fast
- Less affected by common filters
- Efficiently find out stretched or blurred images

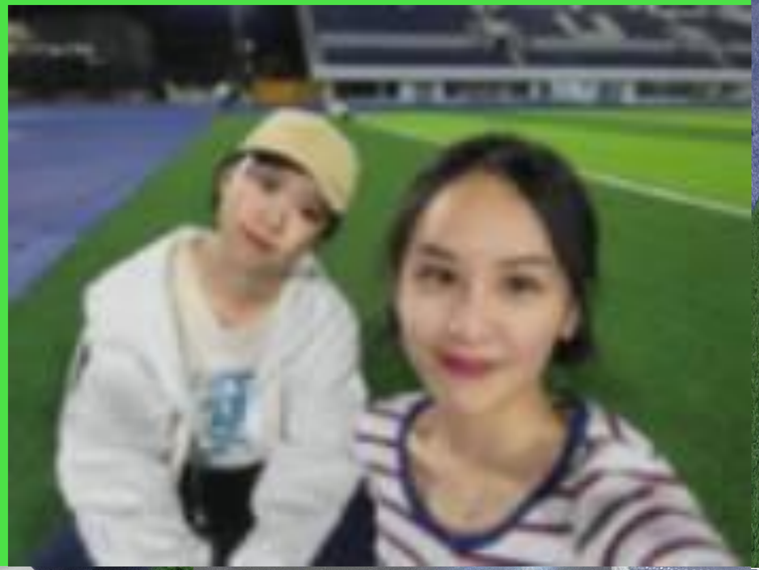
Drawbacks:

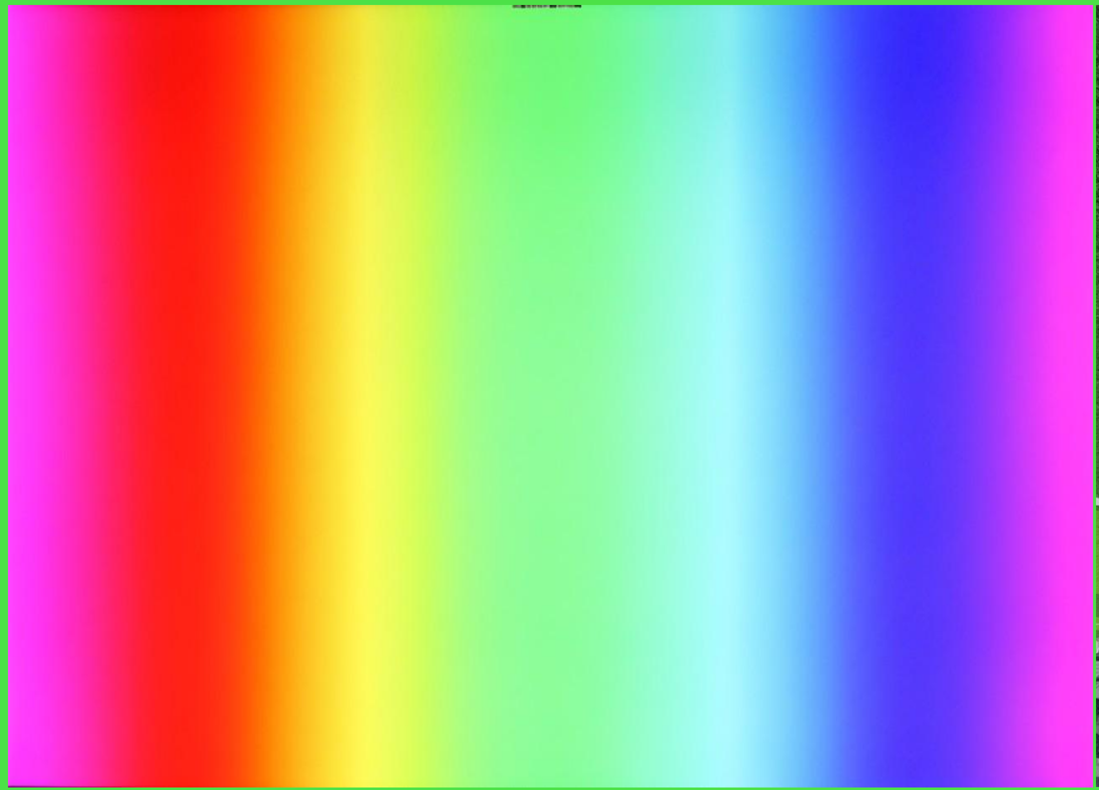
- **Rotation** and **cropping** are not supported.
- **Exposure** has a great impact of results.

similarity = **94.6%** (stretched)

→

similarity =





similarity = **72.4%**

control group

similarity =71.2%

rotate

similarity = 62.9%

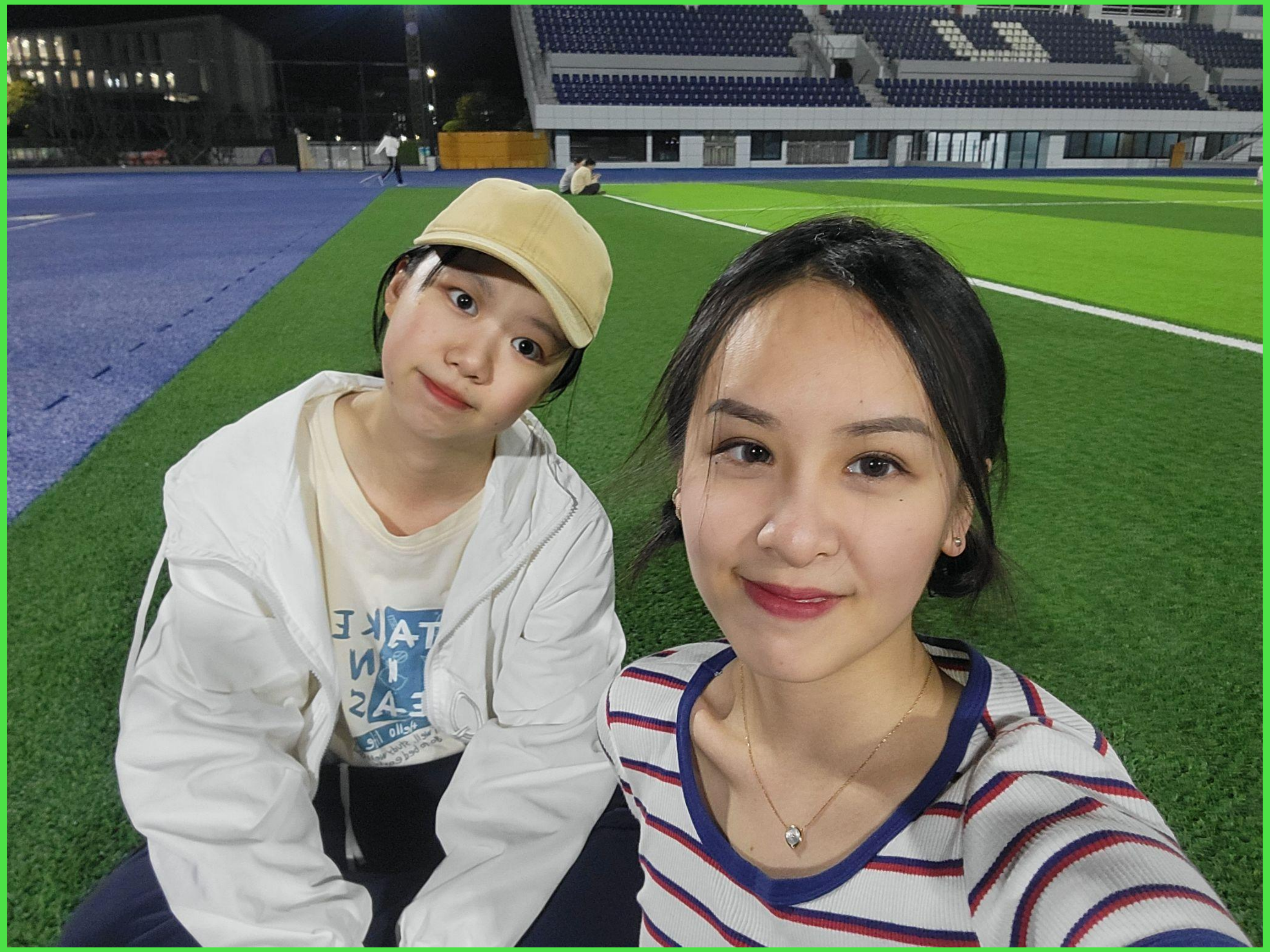
brightness (exposure)

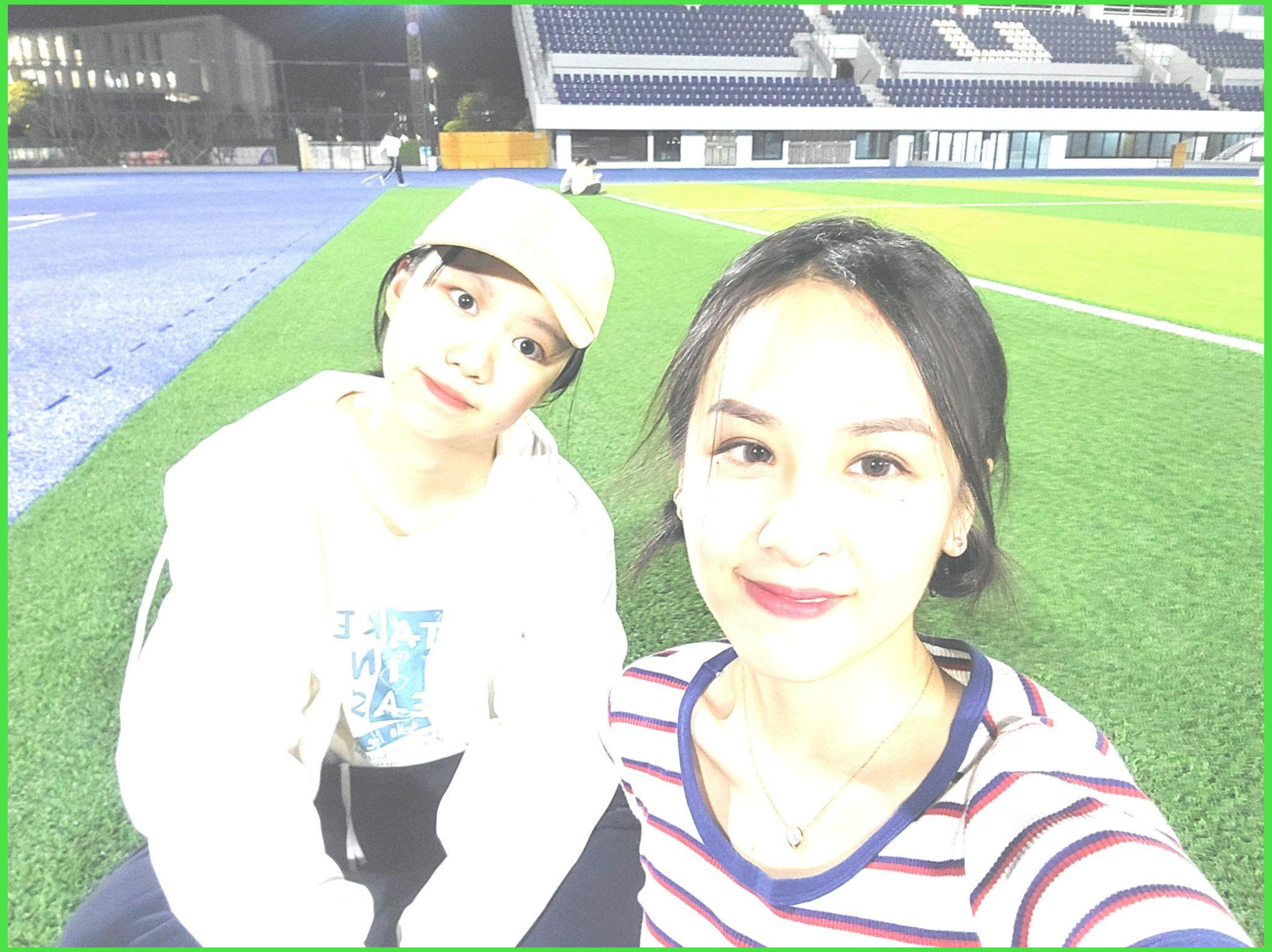
L2: ② Grayscale Histogram

Improve the drawback about exposure:

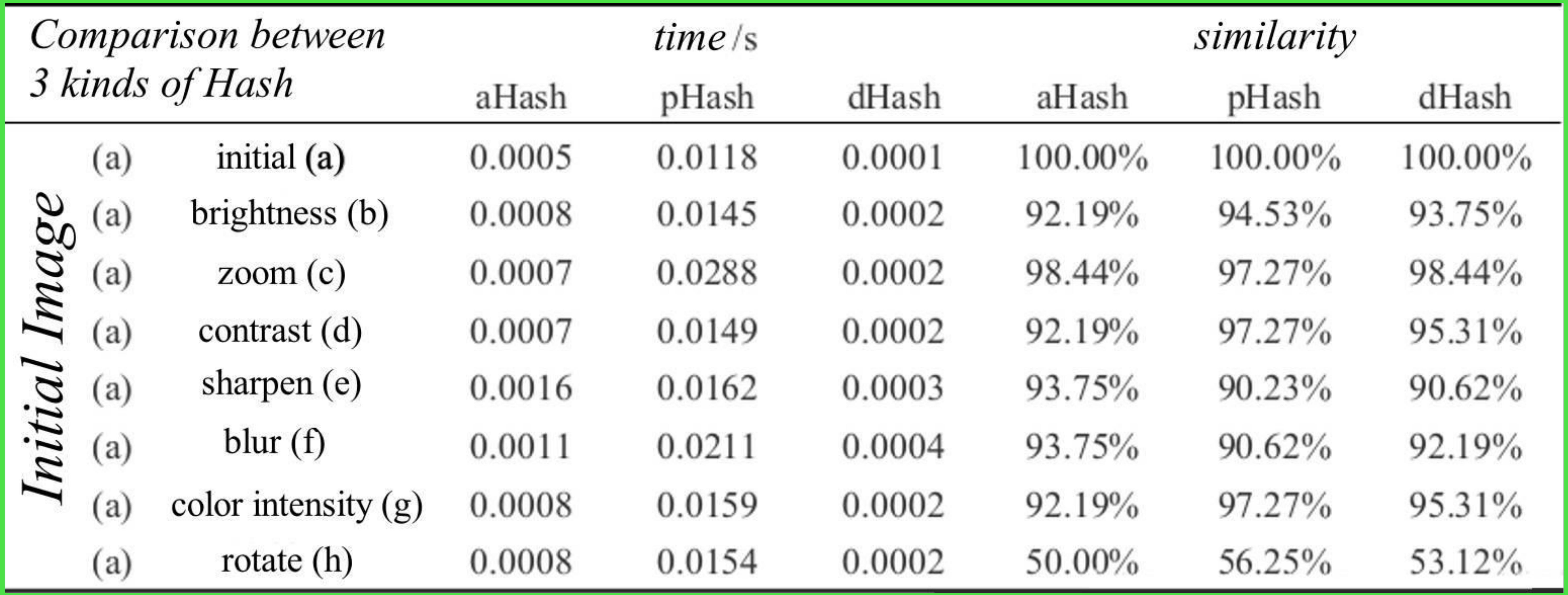
Simple standardization of difference: similarity = 62.9%

Grayscale histogram: similarity = 89.2%





L2: ③ Hash



Comparison between 3 kinds of Hash			*time* /s			*similarity*		
			aHash	pHash	dHash	aHash	pHash	dHash
Initial Image	(a)	initial **(a)**	0.0005	0.0118	0.0001	100.00%	100.00%	100.00%
	(a)	brightness (b)	0.0008	0.0145	0.0002	92.19%	94.53%	93.75%
	(a)	zoom (c)	0.0007	0.0288	0.0002	98.44%	97.27%	98.44%
	(a)	contrast (d)	0.0007	0.0149	0.0002	92.19%	97.27%	95.31%
	(a)	sharpen (e)	0.0016	0.0162	0.0003	93.75%	90.23%	90.62%
	(a)	blur (f)	0.0011	0.0211	0.0004	93.75%	90.62%	92.19%
	(a)	color intensity (g)	0.0008	0.0159	0.0002	92.19%	97.27%	95.31%
	(a)	rotate (h)	0.0008	0.0154	0.0002	50.00%	56.25%	53.12%

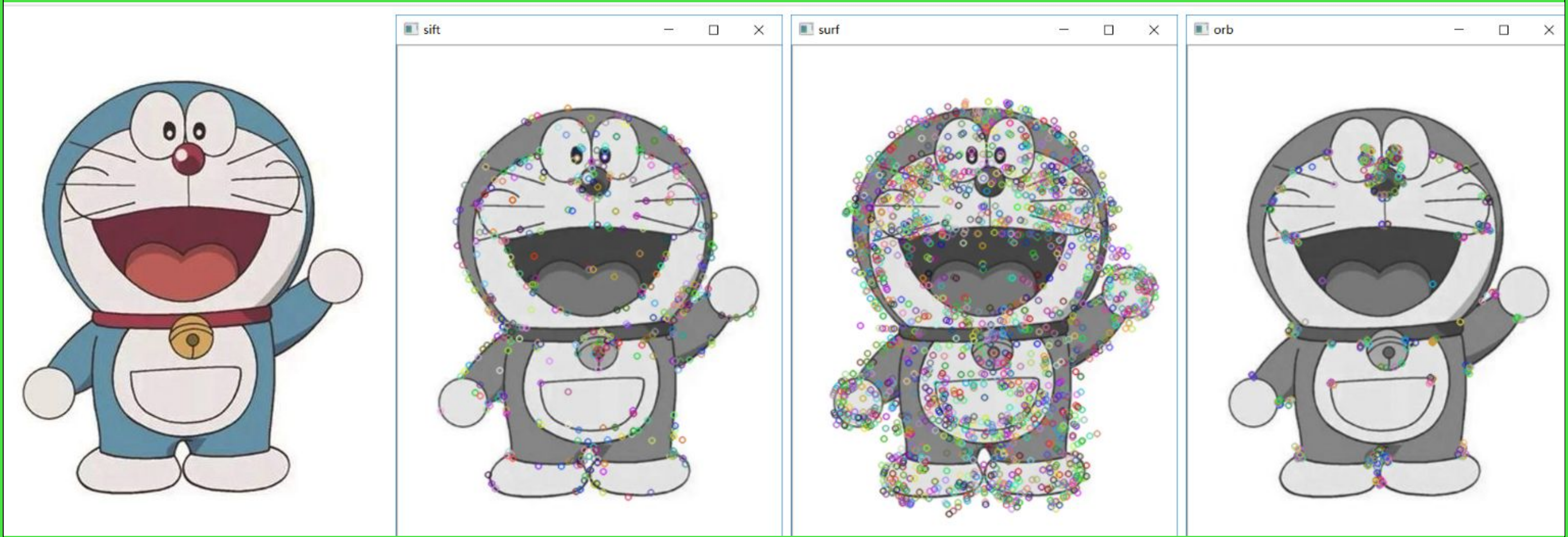
In conclusion, hash algorithms have better performance in global visual and cost less time. However, rotation still is a difficult problem to solve.

Level 3: Feature Matching

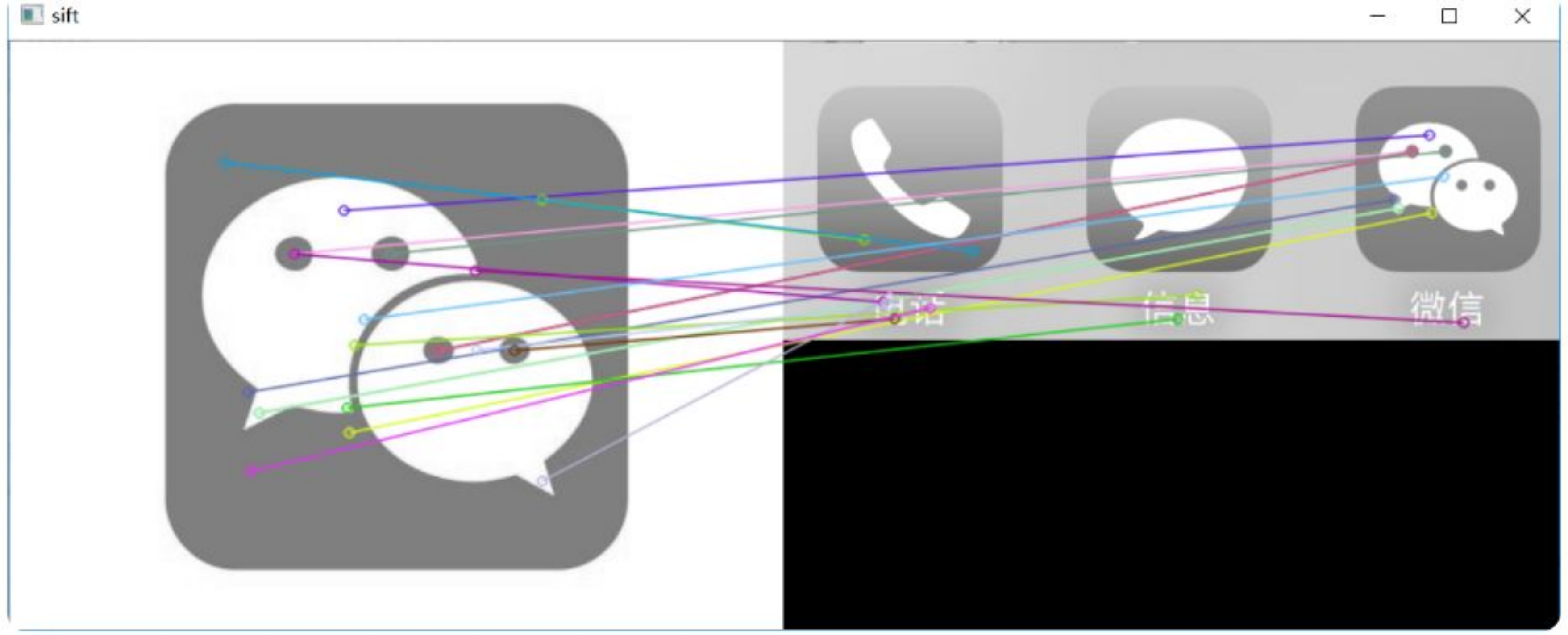
- ORB algorithm
- Scale Invariant Feature Transform (SIFT)
- Speeded-Up Robust Features (SURF)

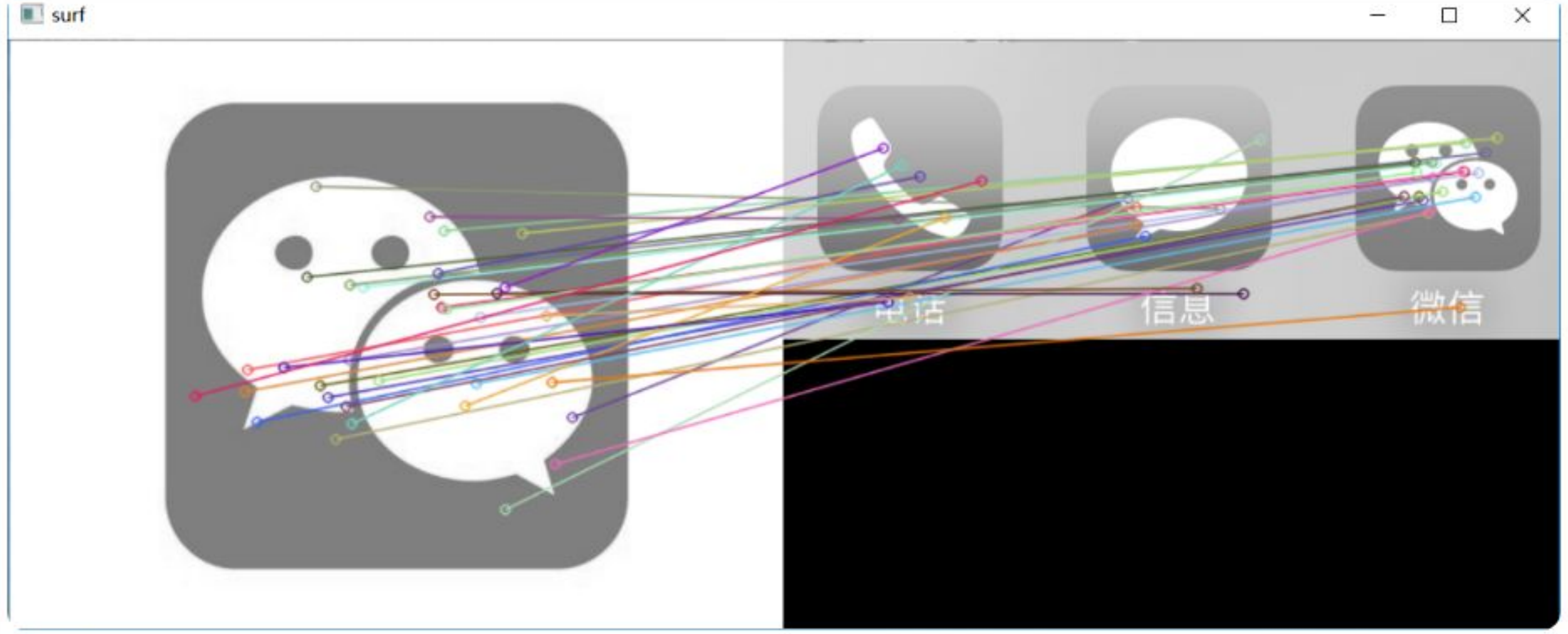
Level 3: Feature Extraction

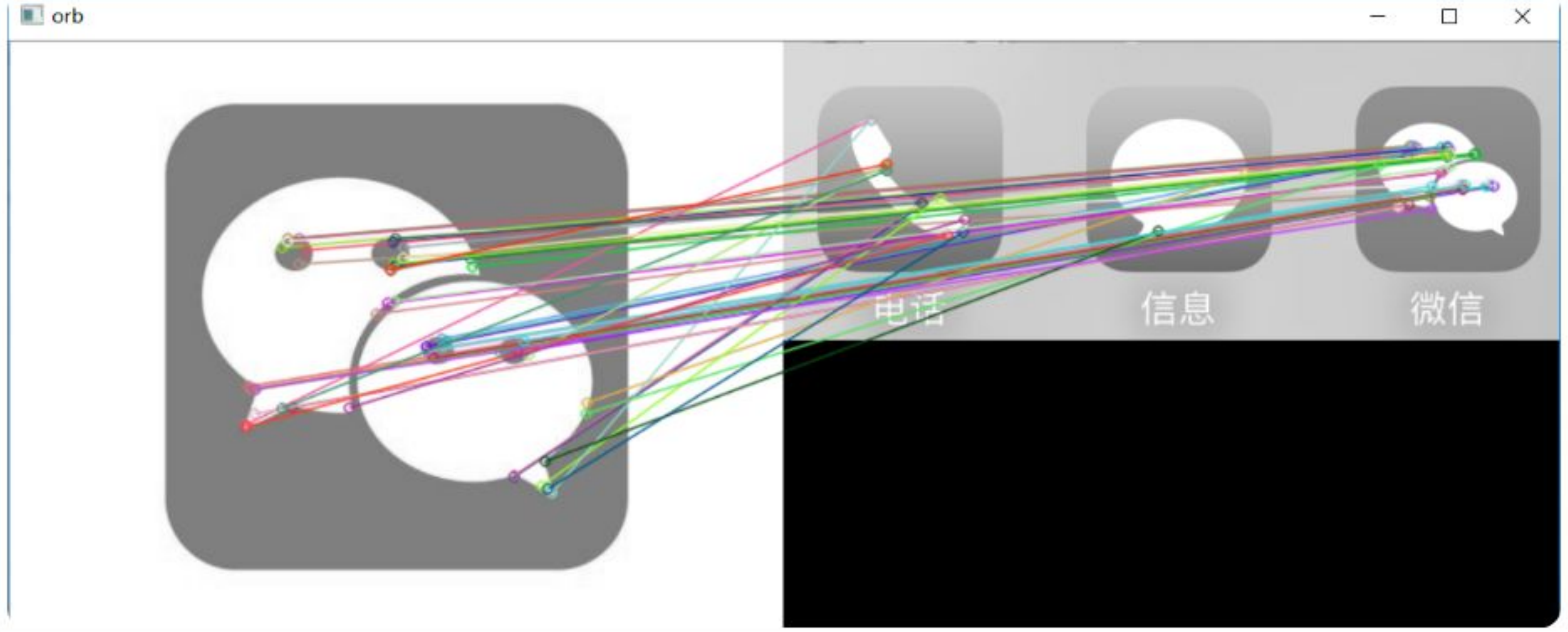
Original↓ SIFT↓ SURF↓ ORB↓



We can find that: SIFT algorithm, although it extracts the fewest feature points, has the best results.



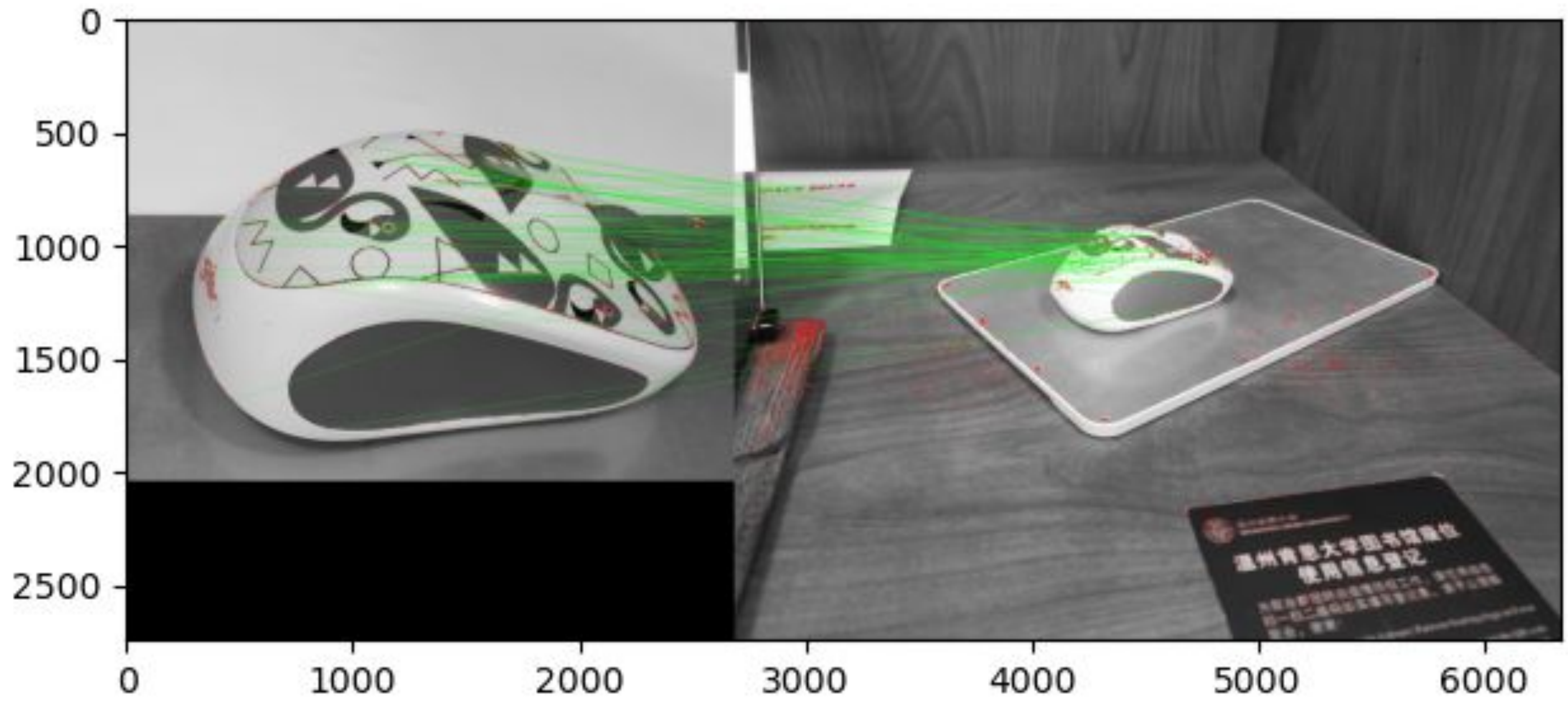




Level 3: Feature Matching

- Judging from the results of the output, ORB works best.

- ORB result of matching my mouse on the desk↓



Level 3: Conclusion

After numerous trials and reading some researches, we got conclusion that:

- Computational speed: ORB>>SURF>>SIFT
- Robustness of rotation：SURF>ORB~SIFT (similar)
- Fuzzy robustness：SURF>ORB~SIFT
- Robustness of Scale transformation: SURF>SIFT>ORB（ORB unsupported）

THANK YOU