计算机视觉

彭盛霖

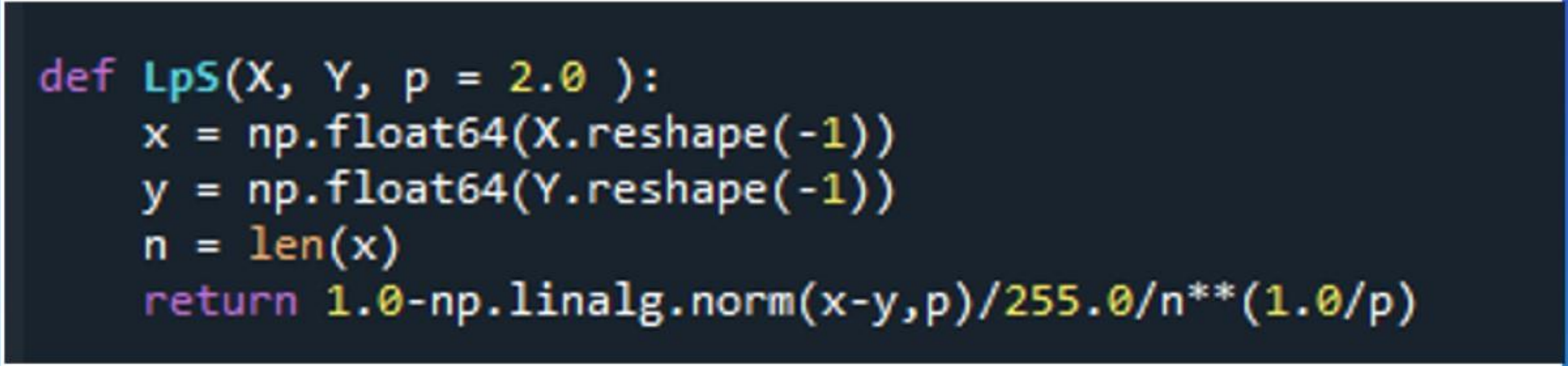
西北大学信息学院

pengshenglin@nwu.edu.cn

5.1 图像匹配

5.1 图像匹配

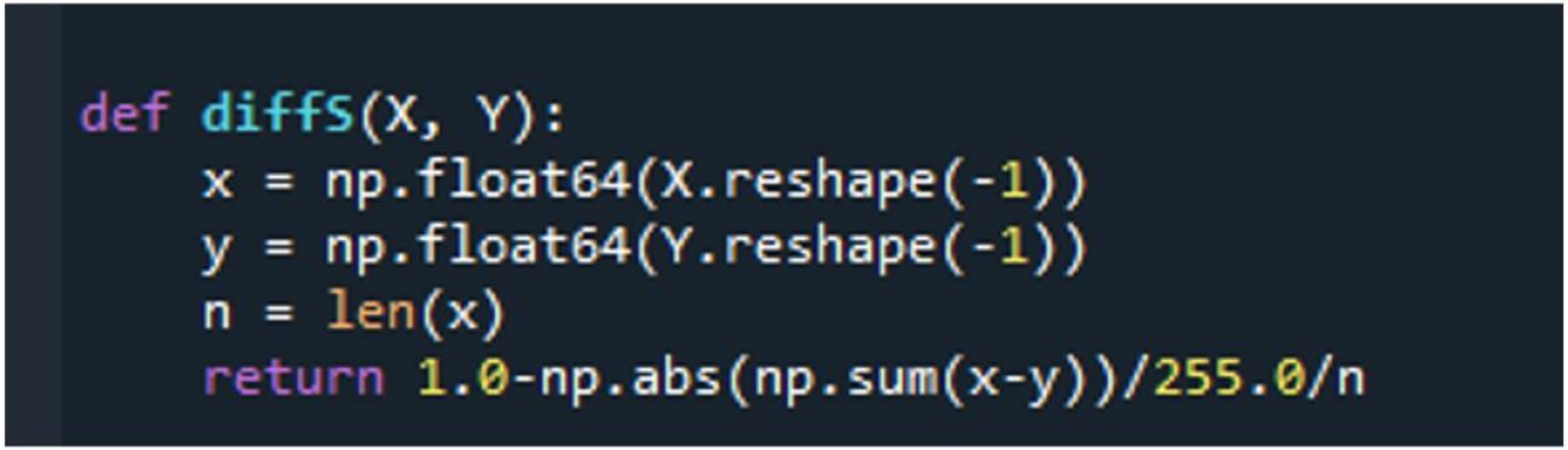
*Lp*相似度



```python
def LpS(X, Y, p = 2.0 ):
    x = np.float64(X.reshape(-1))
    y = np.float64(Y.reshape(-1))
    n = len(x)
    return 1.0-np.linalg.norm(x-y,p)/255.0/n**(1.0/p)
```

5.1 图像匹配

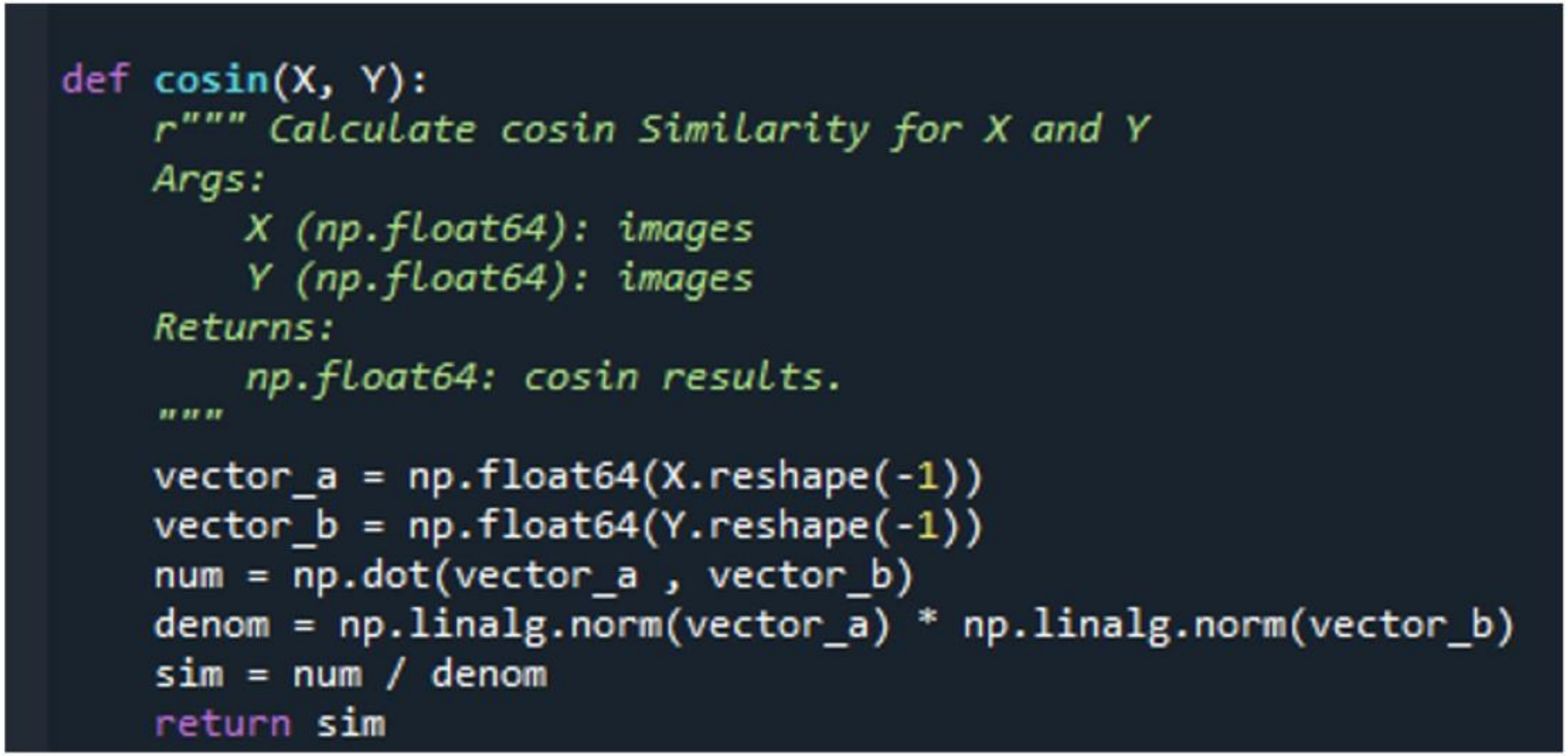
差相似度



```python
def diffS(X, Y):
    x = np.float64(X.reshape(-1))
    y = np.float64(Y.reshape(-1))
    n = len(x)
    return 1.0-np.abs(np.sum(x-y))/255.0/n
```

5.1 图像匹配

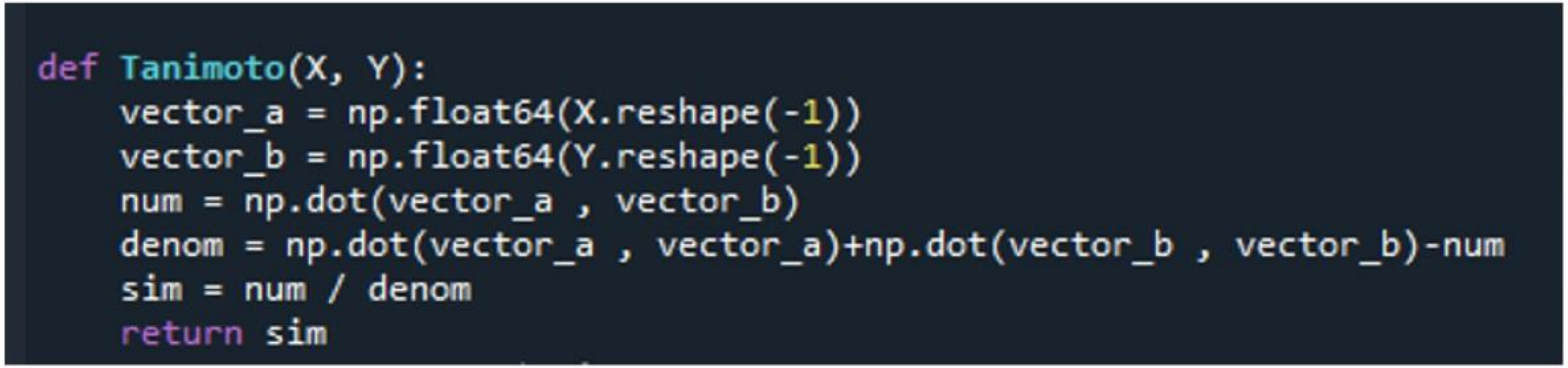
余弦相似度



```
def cosin(X, Y):
    r""" Calculate cosin Similarity for X and Y
    Args:
        X (np.float64): images
        Y (np.float64): images
    Returns:
        np.float64: cosin results.
    """
    vector_a = np.float64(X.reshape(-1))
    vector_b = np.float64(Y.reshape(-1))
    num = np.dot(vector_a , vector_b)
    denom = np.linalg.norm(vector_a) * np.linalg.norm(vector_b)
    sim = num / denom
    return sim
```

5.1 图像匹配

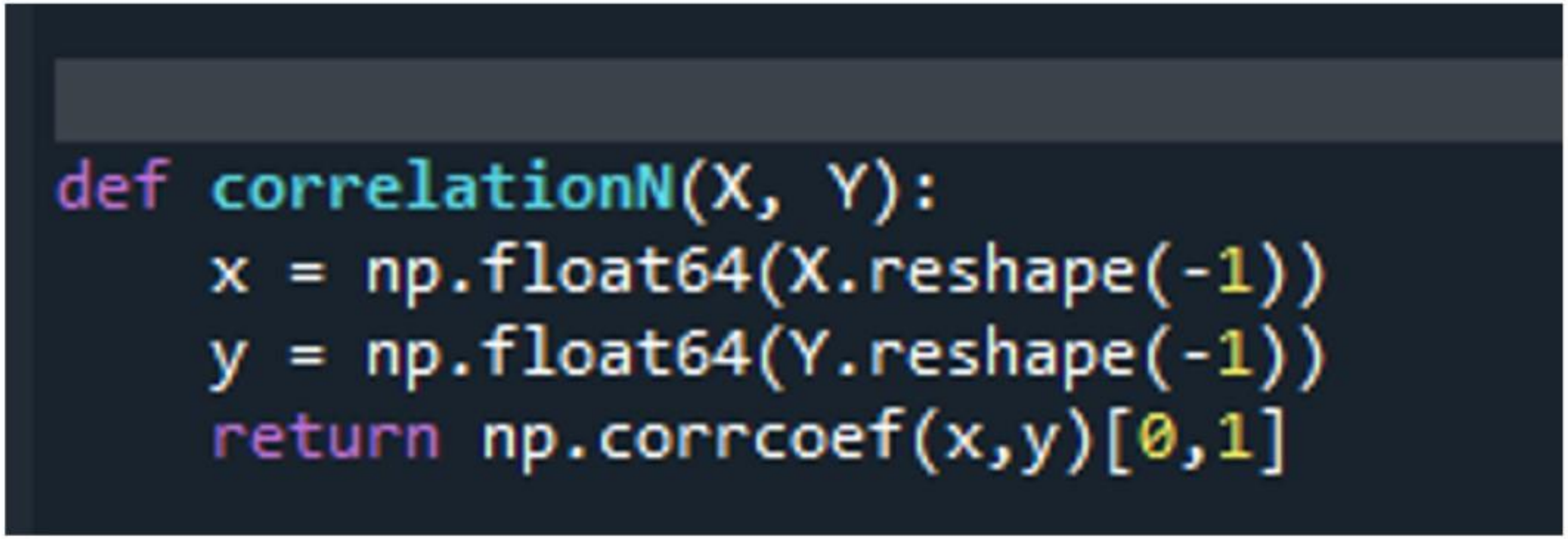
谷本(Tanimoto)相似度



```
def Tanimoto(X, Y):
    vector_a = np.float64(X.reshape(-1))
    vector_b = np.float64(Y.reshape(-1))
    num = np.dot(vector_a , vector_b)
    denom = np.dot(vector_a , vector_a)+np.dot(vector_b , vector_b)-num
    sim = num / denom
    return sim
```

5.1 图像匹配

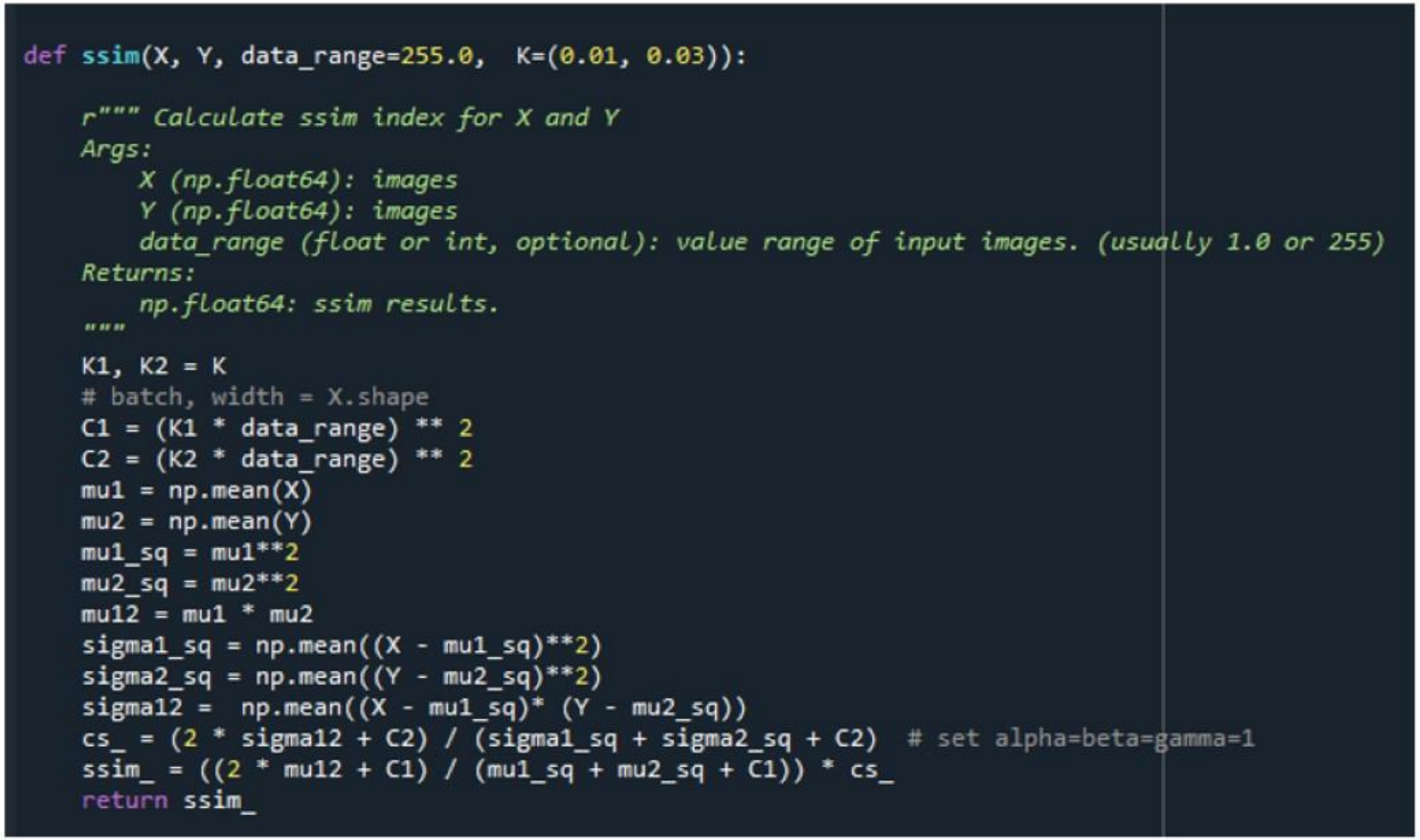
皮尔逊(Pearson)相似度



```
def correlationN(X, Y):
    x = np.float64(X.reshape(-1))
    y = np.float64(Y.reshape(-1))
    return np.corrcoef(x,y)[0,1]
```

5.1 图像匹配

结构相似度(SSIM)

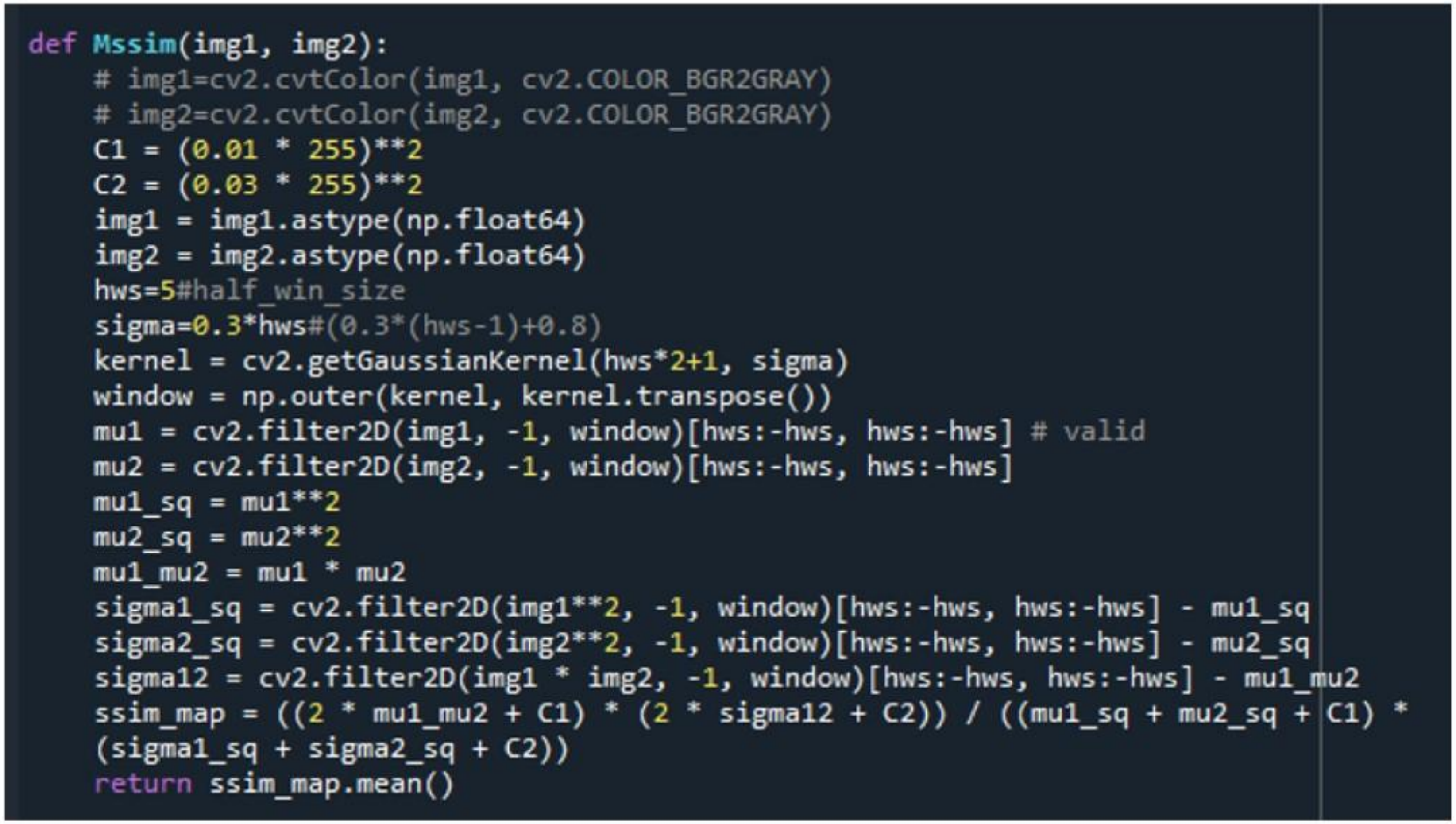


```
def ssim(X, Y, data_range=255.0,  K=(0.01, 0.03)):

    r""" Calculate ssim index for X and Y
    Args:
        X (np.float64): images
        Y (np.float64): images
        data_range (float or int, optional): value range of input images. (usually 1.0 or 255)
    Returns:
        np.float64: ssim results.
    """
    K1, K2 = K
    # batch, width = X.shape
    C1 = (K1 * data_range) ** 2
    C2 = (K2 * data_range) ** 2
    mu1 = np.mean(X)
    mu2 = np.mean(Y)
    mu1_sq = mu1**2
    mu2_sq = mu2**2
    mu12 = mu1 * mu2
    sigma1_sq = np.mean((X - mu1_sq)**2)
    sigma2_sq = np.mean((Y - mu2_sq)**2)
    sigma12 =  np.mean((X - mu1_sq)* (Y - mu2_sq))
    cs_ = (2 * sigma12 + C2) / (sigma1_sq + sigma2_sq + C2)  # set alpha=beta=gamma=1
    ssim_ = ((2 * mu12 + C1) / (mu1_sq + mu2_sq + C1)) * cs_
    return ssim_
```

5.1 图像匹配

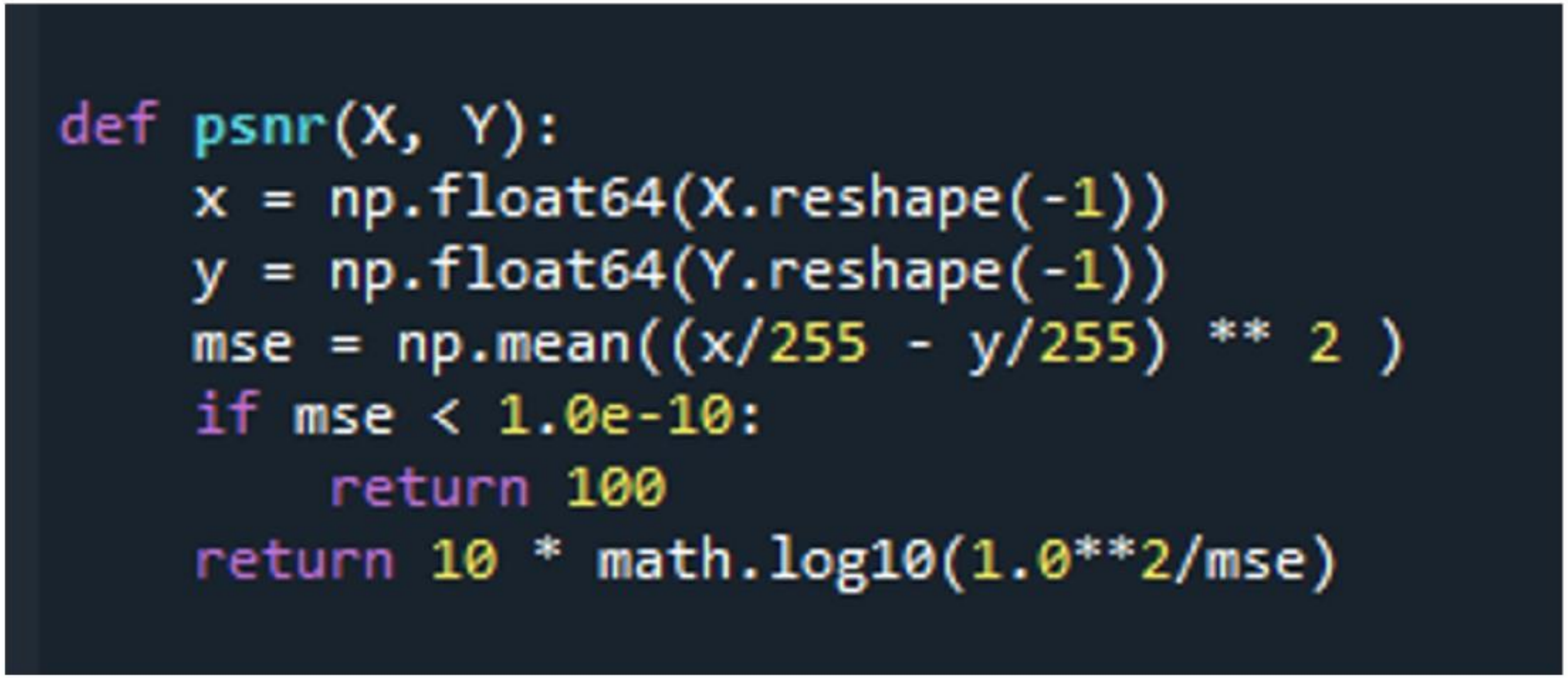
平均结构相似度(MSSIM)



```
def Mssim(img1, img2):
    # img1=cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    # img2=cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    C1 = (0.01 * 255)**2
    C2 = (0.03 * 255)**2
    img1 = img1.astype(np.float64)
    img2 = img2.astype(np.float64)
    hws=5#half_win_size
    sigma=0.3*hws#(0.3*(hws-1)+0.8)
    kernel = cv2.getGaussianKernel(hws*2+1, sigma)
    window = np.outer(kernel, kernel.transpose())
    mu1 = cv2.filter2D(img1, -1, window)[hws:-hws, hws:-hws] # valid
    mu2 = cv2.filter2D(img2, -1, window)[hws:-hws, hws:-hws]
    mu1_sq = mu1**2
    mu2_sq = mu2**2
    mu1_mu2 = mu1 * mu2
    sigma1_sq = cv2.filter2D(img1**2, -1, window)[hws:-hws, hws:-hws] - mu1_sq
    sigma2_sq = cv2.filter2D(img2**2, -1, window)[hws:-hws, hws:-hws] - mu2_sq
    sigma12 = cv2.filter2D(img1 * img2, -1, window)[hws:-hws, hws:-hws] - mu1_mu2
    ssim_map = ((2 * mu1_mu2 + C1) * (2 * sigma12 + C2)) / ((mu1_sq + mu2_sq + C1) *
    (sigma1_sq + sigma2_sq + C2))
    return ssim_map.mean()
```

5.1 图像匹配

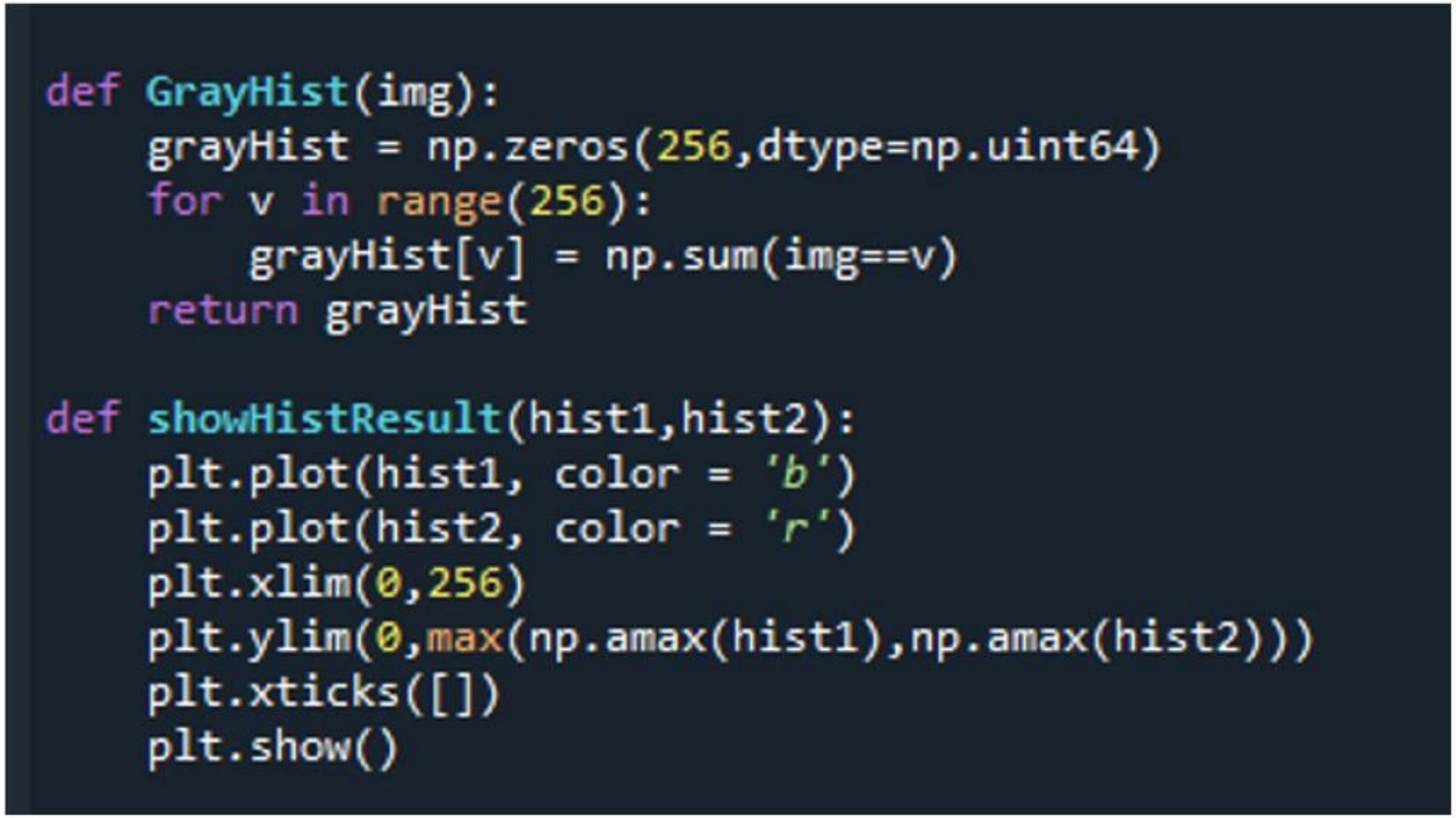
峰值信噪比(PSNR)



```python
def psnr(X, Y):
    x = np.float64(X.reshape(-1))
    y = np.float64(Y.reshape(-1))
    mse = np.mean((x/255 - y/255) ** 2 )
    if mse < 1.0e-10:
        return 100
    return 10 * math.log10(1.0**2/mse)
```

5.1 图像匹配

获取直方图、显示结果

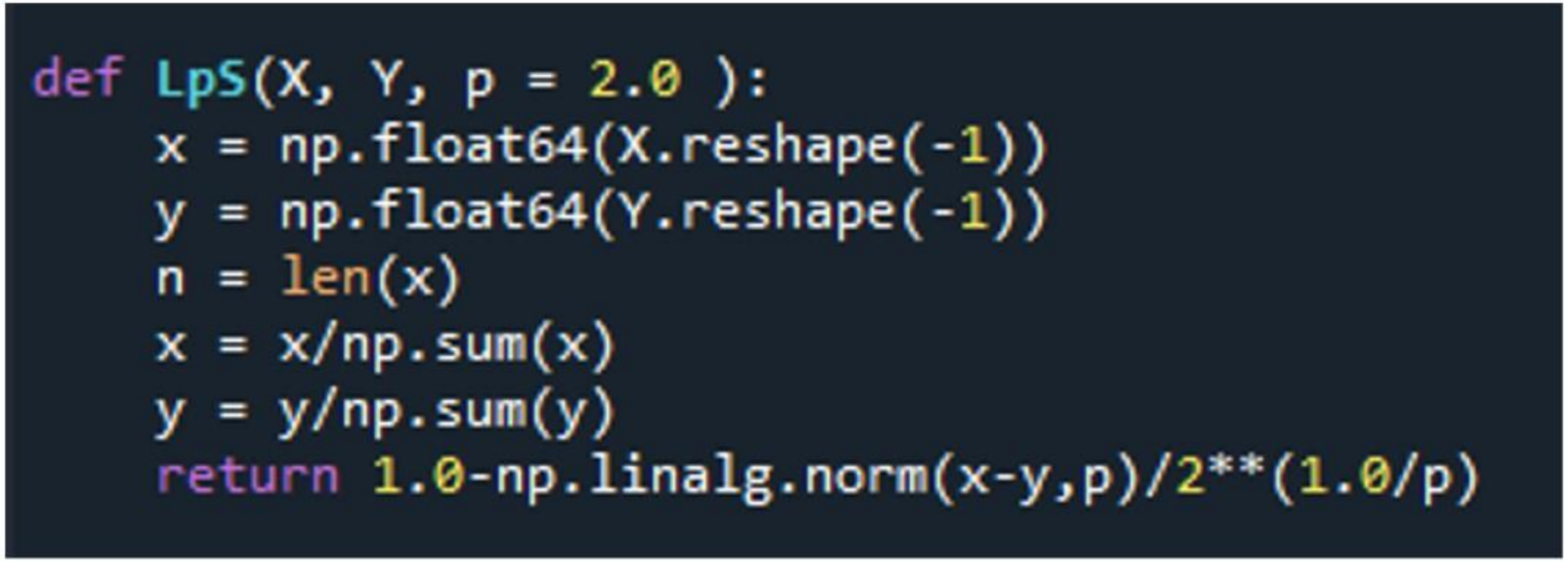


```python
def GrayHist(img):
    grayHist = np.zeros(256,dtype=np.uint64)
    for v in range(256):
        grayHist[v] = np.sum(img==v)
    return grayHist

def showHistResult(hist1,hist2):
    plt.plot(hist1, color = 'b')
    plt.plot(hist2, color = 'r')
    plt.xlim(0,256)
    plt.ylim(0,max(np.amax(hist1),np.amax(hist2)))
    plt.xticks([])
    plt.show()
```

5.1 图像匹配

*Lp*相似度 for 直方图



```
def LpS(X, Y, p = 2.0 ):
    x = np.float64(X.reshape(-1))
    y = np.float64(Y.reshape(-1))
    n = len(x)
    x = x/np.sum(x)
    y = y/np.sum(y)
    return 1.0-np.linalg.norm(x-y,p)/2**(1.0/p)
```

5.1 图像匹配

SIFT特征匹配

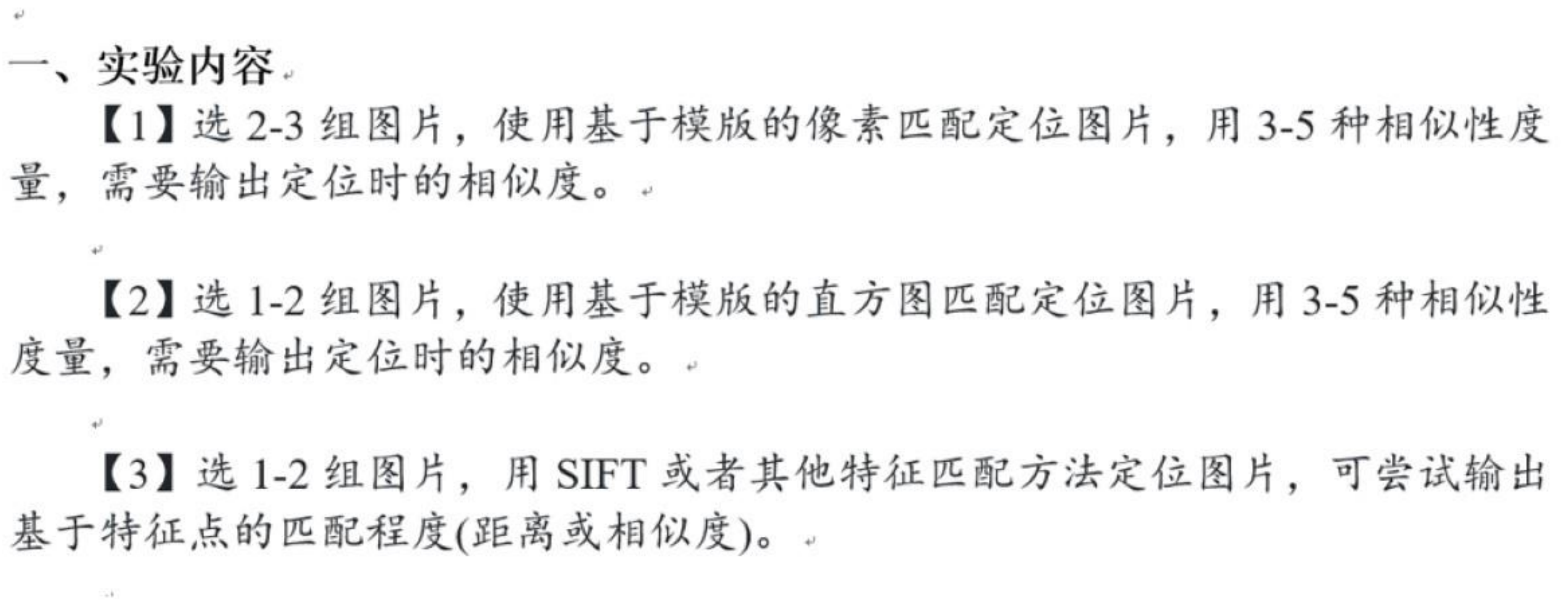
- 参考

https://blog.csdn.net/wu_zhiyuan/article/details/126028766

https://www.freesion.com/article/7751388844/

5.1 图像匹配

◆ 接下来的时间：上机实验并完成实验报告



一、实验内容

【1】选 2-3 组图片，使用基于模版的像素匹配定位图片，用 3-5 种相似性度量，需要输出定位时的相似度。

【2】选 1-2 组图片，使用基于模版的直方图匹配定位图片，用 3-5 种相似性度量，需要输出定位时的相似度。

【3】选 1-2 组图片，用 SIFT 或者其他特征匹配方法定位图片，可尝试输出基于特征点的匹配程度(距离或相似度)。