《计算机视觉》实验报告

# 实验05：图像匹配

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 庞晓宇 | 学号 | 2020118100 |
| 实验地点 | 信息学院310 | 实验日期 | 2023年10月8日 |

**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)。

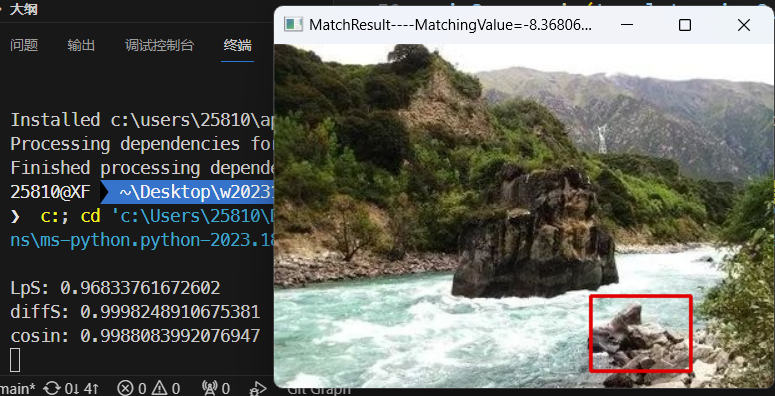
1. **实验过程以及结果分析**

一些图像匹配相似度算法实现：

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
# Lp相似度  
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)  
  
  
# 差相似度  
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  
  
  
# 余弦相似度  
def cosin(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.linalg.norm(vector\_a) \* np.linalg.norm(vector\_b)  
 sim = num / denom  
 return sim  
  
  
# 古本（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  
  
  
# 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]  
  
  
# 结构相似度（SSIM）  
def ssim(X, Y, data\_range=255.0, K=(0.01, 0.03)):  
 K1, K2 = K  
 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)  
 ssim\_ = (2 \* mu12 + C1) / (mu1\_sq + mu2\_sq + C1) \* cs\_  
 return ssim\_  
  
  
# 平均结构相似度（MSSIM）  
def Mssim(img1, img2):  
 C1 = (0.01 \* 255) \*\* 2  
 C2 = (0.03 \* 255) \*\* 2  
 img1 = img1.astype(np.float64)  
 img2 = img2.astype(np.float64)  
 hws = 5  
 sigma = 0.3 \* hws  
 kernel = cv2.getGaussianKernel(2 \* hws + 1, sigma)  
 window = np.outer(kernel, kernel.transpose())  
 mu1 = cv2.filter2D(img1, -1, window)[hws:-hws, hws:-hws]  
 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()  
  
  
# 峰值信噪比（PSNR）  
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 \* np.log10(1.0\*\*2 / mse)  
  
  
# 获取直方图、显示结果  
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()  
  
  
# Lp相似度 for 直方图  
def LpS\_hist(hist1, hist2, p=2.0):  
 x = np.float64(hist1.reshape(-1))  
 y = np.float64(hist2.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)  
  
  
# SIFT特征匹配  
# https://blog.csdn.net/wu\_zhiyuan/article/details/126028766  
# https://www.freesion.com/article/7751388844/

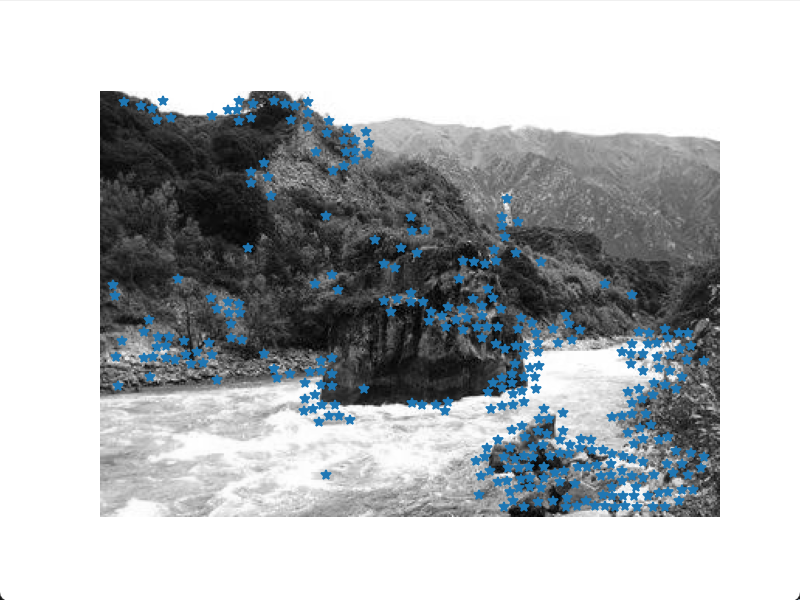
定位图片并使用多种算法计算相似度：

from pptcode import \*  
import cv2  
import numpy as np  
  
  
# 读取目标图片  
target = cv2.imread("./imgs/scenery4.jpg")  
# 读取模板图片  
template = cv2.imread("./imgs/scenery4-0.jpg")  
# 获得模板图片的高宽尺寸  
theight, twidth = template.shape[:2]  
# 执行模板匹配，采用的匹配方式cv2.TM\_SQDIFF\_NORMED  
result = cv2.matchTemplate(target, template, cv2.TM\_SQDIFF\_NORMED)  
# 归一化处理  
cv2.normalize(result, result, 0, 1, cv2.NORM\_MINMAX, -1)  
# 寻找矩阵（一维数组当做向量，用Mat定义）中的最大值和最小值的匹配结果及其位置  
min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(result)  
  
  
# 截取  
width, height = twidth, theight # 显示卡片的宽和高  
pts1 = np.float32(  
 [  
 [min\_loc[0], min\_loc[1]],  
 [min\_loc[0] + twidth, min\_loc[1]],  
 [min\_loc[0], min\_loc[1] + theight],  
 [min\_loc[0] + twidth, min\_loc[1] + theight],  
 ]  
) # 截取对片中的哪个区域  
pts2 = np.float32([[0, 0], [width, 0], [0, height], [width, height]]) # 定义显示的卡片的坐标  
matrix = cv2.getPerspectiveTransform(pts1, pts2) # 两个区域坐标绑定  
imgOutput = cv2.warpPerspective(target, matrix, (width, height)) # 转换为图片  
  
  
# 匹配值转换为字符串  
# 对于cv2.TM\_SQDIFF及cv2.TM\_SQDIFF\_NORMED方法min\_val越趋近与0匹配度越好，匹配位置取min\_loc  
# 对于其他方法max\_val越趋近于1匹配度越好，匹配位置取max\_loc  
strmin\_val = str(min\_val)  
# 绘制矩形边框，将匹配区域标注出来  
# min\_loc：矩形定点  
# (min\_loc[0]+twidth,min\_loc[1]+theight)：矩形的宽高  
# (0,0,225)：矩形的边框颜色；2：矩形边框宽度  
cv2.rectangle(  
 target, min\_loc, (min\_loc[0] + twidth, min\_loc[1] + theight), (0, 0, 225), 2  
)  
  
  
sim1 = LpS(template, imgOutput)  
print("LpS:", sim1)  
sim2 = diffS(template, imgOutput)  
print("diffS:", sim2)  
sim3 = cosin(template, imgOutput)  
print("cosin:", sim3)  
  
  
# 显示结果,并将匹配值显示在标题栏上  
cv2.imshow("MatchResult----MatchingValue=" + strmin\_val, target)  
cv2.waitKey()  
cv2.destroyAllWindows()

运行结果： 

角点检测算法：

# 特征提取  
# 局部图像描述子  
# Harris角点检测  
from pylab import \*  
from PIL import Image  
from numpy import \*  
from scipy.ndimage import filters  
  
  
def compute\_harris\_response(im, sigma=3):  
 """在一幅灰度图像中，对每个像素计算Harris角点检测器响应函数"""  
 # 计算导数  
 imx = zeros(im.shape)  
 filters.gaussian\_filter(im, (sigma, sigma), (0, 1), imx)  
 imy = zeros(im.shape)  
 filters.gaussian\_filter(im, (sigma, sigma), (1, 0), imy)  
 # 计算Harris矩阵的分量  
 Wxx = filters.gaussian\_filter(imx \* imx, sigma)  
 Wxy = filters.gaussian\_filter(imx \* imy, sigma)  
 Wyy = filters.gaussian\_filter(imy \* imy, sigma)  
 # 计算特征值和迹  
 Wdet = Wxx \* Wyy - Wxy\*\*2  
 Wtr = Wxx + Wyy  
 return Wdet / Wtr  
  
  
def get\_harris\_points(harrisim, min\_dist=10, threshold=0.1):  
 """从一幅Harris响应图像中返回角点。min\_dist为分割角点和图像边界的最小像素数目"""  
 # 寻找高于阈值的候选角点  
 corner\_threshold = harrisim.max() \* threshold  
 harrisim\_t = (harrisim > corner\_threshold) \* 1  
 # 得到候选点的坐标  
 coords = array(harrisim\_t.nonzero()).T  
 # 以及它们的Harris响应值  
 candidate\_values = [harrisim[c[0], c[1]] for c in coords]  
 # 对候选点按照Harris响应值进行排序  
 index = argsort(candidate\_values)  
 # 将可行点的位置保存到数组中  
 allowed\_locations = zeros(harrisim.shape)  
 allowed\_locations[min\_dist:-min\_dist, min\_dist:-min\_dist] = 1  
 # 按照min\_distance原则，选择最佳Harris点  
 filtered\_coords = []  
 for i in index:  
 if allowed\_locations[coords[i, 0], coords[i, 1]] == 1:  
 filtered\_coords.append(coords[i])  
 allowed\_locations[  
 (coords[i, 0] - min\_dist) : (coords[i, 0] + min\_dist),  
 (coords[i, 1] - min\_dist) : (coords[i, 1] + min\_dist),  
 ] = 0  
 return filtered\_coords  
  
  
def plot\_harris\_points(image, filtered\_coords):  
 """绘制图像中检测到的角点"""  
 figure()  
 gray()  
 imshow(image)  
 plot([p[1] for p in filtered\_coords], [p[0] for p in filtered\_coords], "\*")  
 axis("off")  
 show()  
  
  
# 调用展示  
im = array(Image.open("./imgs/scenery4.jpg").convert("L"))  
harrisim = compute\_harris\_response(im)  
filtered\_coords = get\_harris\_points(harrisim, 6, 0.05)  
plot\_harris\_points(im, filtered\_coords)  
  
  
# 图像匹配  
from PIL import Image  
from numpy import \*  
# import harris  
from PCV.localdescriptors import harris  
from pylab import \*  
  
wid = 5  
im1 = array(Image.open("./imgs/scenery4-0.jpg").convert("L"))  
im2 = array(Image.open("./imgs/scenery4.jpg").convert("L"))  
harrisim = harris.compute\_harris\_response(im1, 5)  
filtered\_coords1 = harris.get\_harris\_points(harrisim, wid + 1, 0.2)  
d1 = harris.get\_descriptors(im1, filtered\_coords1, wid)  
harrisim = harris.compute\_harris\_response(im2, 5)  
filtered\_coords2 = harris.get\_harris\_points(harrisim, wid + 1, 0.2)  
d2 = harris.get\_descriptors(im2, filtered\_coords2, wid)  
print("starting matching")

运行结果： 

多目标检测：

# opencv模板匹配----多目标匹配  
import cv2  
import numpy  
  
# 读取目标图片  
target = cv2.imread("target.jpg")  
# 读取模板图片  
template = cv2.imread("template.jpg")  
# 获得模板图片的高宽尺寸  
theight, twidth = template.shape[:2]  
# 执行模板匹配，采用的匹配方式cv2.TM\_SQDIFF\_NORMED  
result = cv2.matchTemplate(target, template, cv2.TM\_SQDIFF\_NORMED)  
# 归一化处理  
# cv2.normalize( result, result, 0, 1, cv2.NORM\_MINMAX, -1 )  
# 寻找矩阵（一维数组当做向量，用Mat定义）中的最大值和最小值的匹配结果及其位置  
min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(result)  
# 绘制矩形边框，将匹配区域标注出来  
# min\_loc：矩形定点  
# (min\_loc[0]+twidth,min\_loc[1]+theight)：矩形的宽高  
# (0,0,225)：矩形的边框颜色；2：矩形边框宽度  
cv2.rectangle(  
 target, min\_loc, (min\_loc[0] + twidth, min\_loc[1] + theight), (0, 0, 225), 2  
)  
# 匹配值转换为字符串  
# 对于cv2.TM\_SQDIFF及cv2.TM\_SQDIFF\_NORMED方法min\_val越趋近与0匹配度越好，匹配位置取min\_loc  
# 对于其他方法max\_val越趋近于1匹配度越好，匹配位置取max\_loc  
strmin\_val = str(min\_val)  
# 初始化位置参数  
temp\_loc = min\_loc  
other\_loc = min\_loc  
numOfloc = 1  
# 第一次筛选----规定匹配阈值，将满足阈值的从result中提取出来  
# 对于cv2.TM\_SQDIFF及cv2.TM\_SQDIFF\_NORMED方法设置匹配阈值为0.01  
threshold = 0.01  
loc = numpy.where(result < threshold)  
# 遍历提取出来的位置  
for other\_loc in zip(\*loc[::-1]):  
 # 第二次筛选----将位置偏移小于5个像素的结果舍去  
 if (temp\_loc[0] + 5 < other\_loc[0]) or (temp\_loc[1] + 5 < other\_loc[1]):  
 numOfloc = numOfloc + 1  
 temp\_loc = other\_loc  
 cv2.rectangle(  
 target,  
 other\_loc,  
 (other\_loc[0] + twidth, other\_loc[1] + theight),  
 (0, 0, 225),  
 2,  
 )  
str\_numOfloc = str(numOfloc)  
# 显示结果,并将匹配值显示在标题栏上  
strText = (  
 "MatchResult----MatchingValue="  
 + strmin\_val  
 + "----NumberOfPosition="  
 + str\_numOfloc  
)  
cv2.imshow(strText, target)  
cv2.waitKey()  
cv2.destroyAllWindows()

1. **实验总结**

**本次实验主要是进行图像匹配的相关练习，课件上面提供了一些相似度计算的函数。挑选了一组图片，使用cv2模板匹配，采用的匹配方式为cv2.TM\_SQDIFF\_NORMED。输出匹配的相关结果，并分离出匹配的部分，使用课件上提供的相似度计算函数进行相似度的计算并输出。通过对比发现不同的相似度函数对同样的输入计算的相似度是不同的，可能不同的函数由其相对应的偏好吧。其实有了相似度计算函数之后，我们可以自己实现一个图像匹配的函数，输入目标图像和模板图像，循环匹配找到在相应相似度计算函数下的最大相似度的图像切片。另外，利用相关直方图的获取函数，使用类似的方法可以基于直方图对图像进行匹配。相对于常规的图像匹配，基于直方图的匹配不受局部细节影响、对图像内容变化有适应性。最后，查阅相关资料使用SIFT算法进行了图像的特征点检测和识别。**

1. **材料提交**

1.实验要求内容完备(实验代码、实验结果及分析)、格式规范、排版美观。

2.实验过程中遇到问题需记录具体问题和解决方法；

3.把相关材料(包括实验报告、实验代码、实验使用到的图片等数据)压缩打包为“计算机视觉实验05\_学号\_姓名.zip”，提交到邮箱pengshenglin@nwu.edu.cn；

4.截止时间为实验课当周周日24点前(如实验课在周六周日，截止时间为下周周二24点前)。

5.不要迟交，不要抄袭(迟交当次作业最多70分，抄袭整个课程记0分！)。实验报告整体雷同且存在以下情况判为抄袭：程序仅有极少字符与变量的不同且；程序仅有空格和分行的不同；存在从网页复制导致的乱码，全角符号，非ASCII符号，&nbsp;等；代码高度相似并且程序存在完全相同的错误。(重复教材上的代码不计入抄袭)