#### 计算机视觉实践-练习5

学号：122106222783 姓名：王心怡

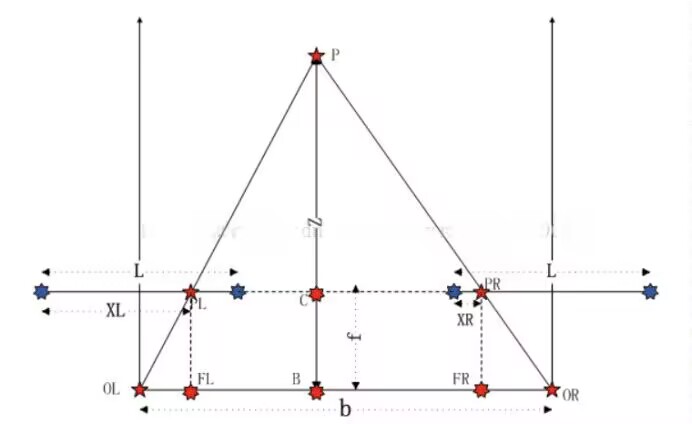
1 实验目标

图像视差匹配，通过立体匹配（stereo matching）得到两张图像的视差图。

2 实验说明

**·视差图和景深的关系推导**

经过双目相机标定和校准后，双目相机的主光轴到达平行，如图所示是双目相机模型，世界坐标系中的任意一点都满足，该点与它在左右相机的成像点在同一个极平面上。OL和OR是左右相机的的光心，长为L的两条线段（端点为蓝色星星的线段）表示的是左右相机的像面。则光心到像面的最短距离就是焦距长度f。若P是世界坐标系中的一点，它在左右像面上的成像点是PL和PR。PL和PR距各自像面的左边缘的距离是XL和XR。视差就是XR-XL或者是XL-XR。标定和匹配后f，b，XR，XL都能够得到。



在三角形OL-OR-P中。三角行PL-PR-P相似于三角形OL-OR-P，则有比例关系：

推导过程：

得证：

只要求得XR-XL就可以求得Z了。

**·SAD算法**： SAD（Sum of absolute differences）是一种图像局部匹配算法。

公式表示为：SAD(u,v) = Sum{|Left(u,v) - Right(u,v)|} 选择最小值。

此种方法就是以左目图像的源匹配点为中心，定义一个窗口D，其大小为（2m+1）（2n+1），统计其窗口的灰度值的和，然后在右目图像对应点，沿极线方向（水平方向）按照不同视差确定窗口，逐步计算其左右窗口的灰度和的差值，最后搜索到的差值最小的区域的中心像素即为匹配点。

**·读取图片**

os.chdir(r'C:\Users\wangxinyi\Documents\jupyter-notebook\计算机视觉实践-练习5\pic')

lraw=np.asanyarray(Image.open(r"view0.png"))

rraw=np.asanyarray(Image.open(r"view1.png"))

·**转换彩色图像为灰度图像，并且转为double格式**

limg= cv.cvtColor(lraw,cv.COLOR\_BGR2GRAY)

rimg= cv.cvtColor(rraw,cv.COLOR\_BGR2GRAY)

limg=np.asanyarray(limg,dtype=np.double)

rimg=np.asanyarray(rimg,dtype=np.double)

img\_size=np.shape(limg)[0:2]

plt.imshow(limg)

plt.show()

plt.imshow(rimg)

plt.show()

**·加速后的SAD算法，具体做法是先计算右图按照视差由0到maxDisparity减去左图所得的矩阵**

tic1=time.time()

imgDiff=np.zeros((img\_size[0],img\_size[1],maxDisparity))

e = np.zeros(img\_size)

for i in range(0,maxDisparity):

e=np.abs(rimg[:,0:(img\_size[1]-i)]- limg[:,i:img\_size[1]]) #视差为多少，那么生成的图像就会少多少像素列,e负责计算视差为i时，两张图整体的差距

e2=np.zeros(img\_size) #计算窗口内的和

for x in range((window\_size),(img\_size[0]-window\_size)):

for y in range((window\_size),(img\_size[1]-window\_size)):

e2[x,y]=np.sum(e[(x-window\_size):(x+window\_size),(y-window\_size):(y+window\_size)]) #其实相当于用111 111 111的卷积核去卷积，其实就是先做差再加和以及先加和再做差的关系

imgDiff[:,:,i]=e2

dispMap=np.zeros(img\_size)

**·找到使灰度差最小的视差，并绘图**

for x in range(0,img\_size[0]):

for y in range(0,img\_size[1]):

val=np.sort(imgDiff[x,y,:])

if np.abs(val[0]-val[1])>10:

val\_id=np.argsort(imgDiff[x,y,:])

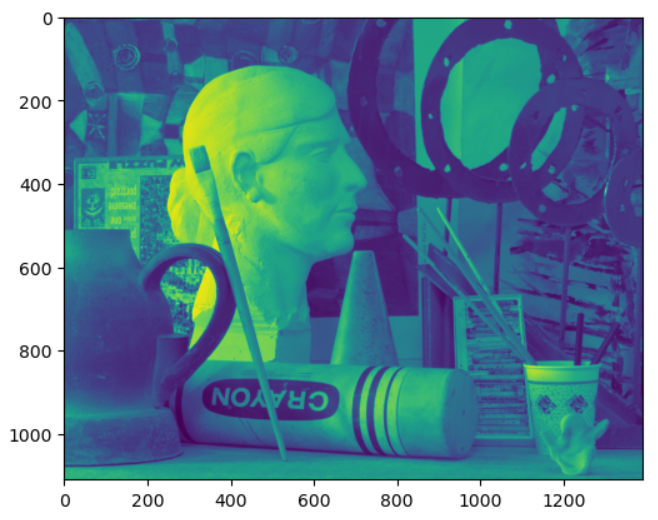
dispMap[x,y]=val\_id[0]/maxDisparity\*255#其实dispmap计算的是视差的大小，如果视差大，那么就相当于两张图片中同样物品的位移大，就是距离近

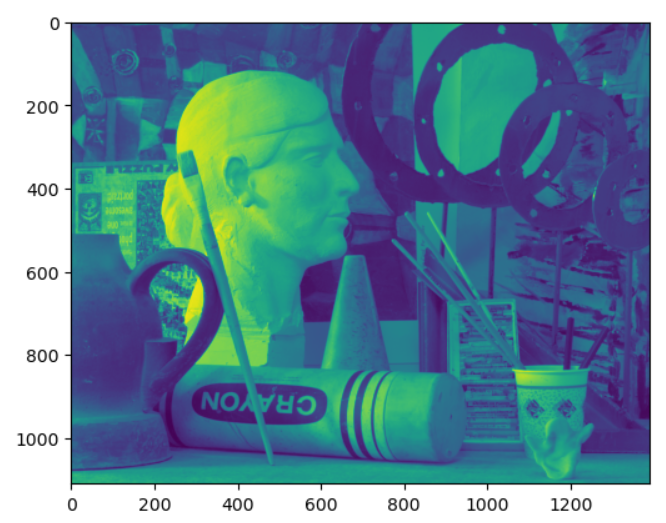
print('用时:',time.time()-tic1)

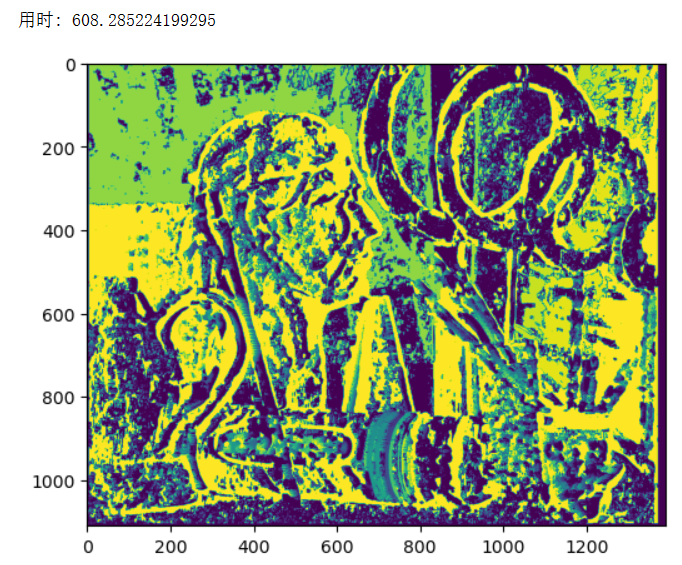
plt.imshow(dispMap)

plt.show()

3 结果截图







4 结果分析

由结果图可以看出，SAD算法虽然得到了视差图，但效果不够理想。SAD算法虽然简单高效，但是容易受光照、曝光等噪声影响，视差图的效果往往不是特别鲁棒。