# 立体匹配

立体匹配(Stereo Matching)是指利用两张图像中的像素点之间的关系来计算出两张图像中的像素点的视差，并通过视差图来恢复场景的深度信息。视差图广泛应用于三维重建、自动驾驶、机器人视觉、虚拟现实等领域。在三维重建中，通过对多张视差图的融合，可以得到更加精确的三维模型。在自动驾驶中，视差图可以用于障碍物检测和道路分割等任务。在机器人视觉和虚拟现实中，视差图可以用于场景重构和增强现实等应用。

## 实验过程：

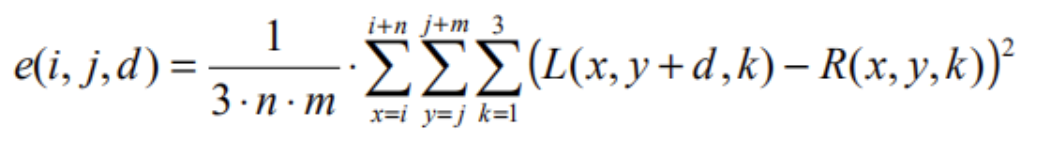
该算法属于基于灰度的匹配算法，这是一种区域相关方法， 在一幅图像中以一点为中心选定区域（窗口），在另一幅图像中寻找与该区域相关系数最大的区域，把该找到的区域的中心认为是原来那区域中心的对应点。它对噪声很敏感，所以需要滤波器来进行去噪。

数据准备

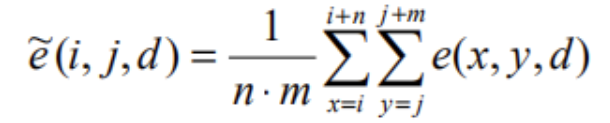
本次实验使用的是两张图片：一张为左相机图片（left.png），另一张为右相机图片（right.png）。

计算误差能量，生成具有最小平均误差能量的视差图。选取匹配计算区域窗口大小是（m\*n）

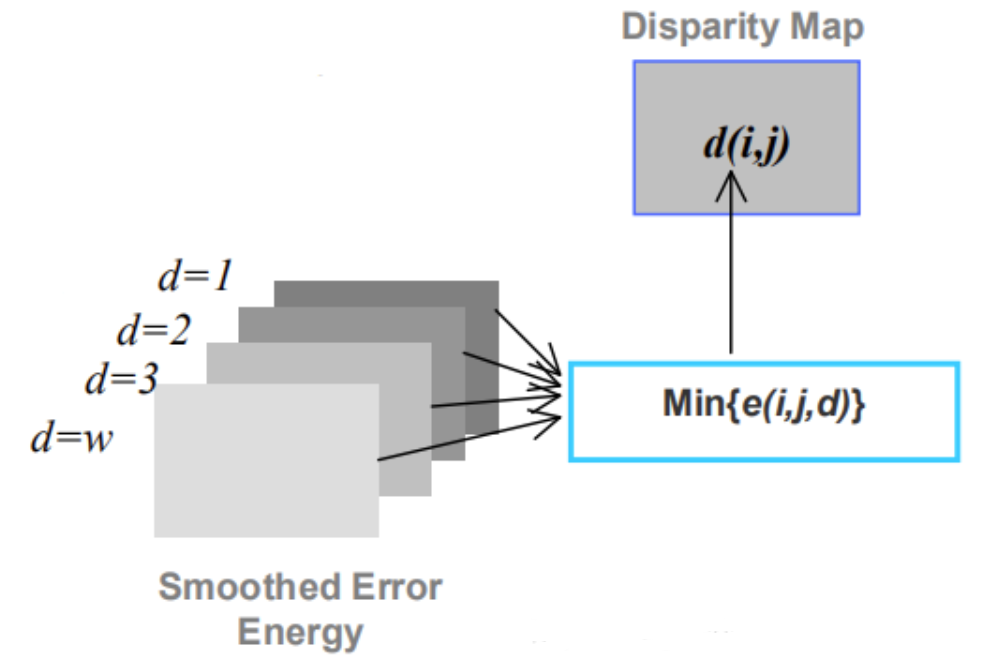
d是视差，我们需要先定一个视差搜寻范围如dmax=40。



由于该算法对噪声敏感，进一步计算平均error energy。



在计算后，选取error energy最小的d作为视差图中（i,j）点的d，得到视差图。



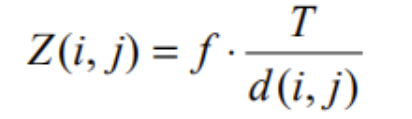
|  |
| --- |
| import numpy as np  import cv2  from matplotlib import pyplot as plt  from mpl\_toolkits.mplot3d import Axes3D  dmax=30   #搜寻匹配的范围  window\_size=2  #匹配窗大小  f=30 #焦距  T=20 #两个相机间距  alpha=1 #阈值ve的系数  left=cv2.imread('left.png')  right=cv2.imread('right.png')  #print(left.shape,right.shape)  size1,size2=left.shape[0],left.shape[1]  disparity=np.zeros((size1,size2), dtype=np.uint8)  #error\_energy  e=np.zeros\_like(disparity)  e\_avg=np.ones\_like(disparity)  e\_avg=e\_avg\*100000  for d in range(dmax):      ##先计算  e （i,j，d）      for i in range(size1):          for j in range(size2):              sum =0              for m in range(window\_size):                  for n in range(window\_size):                      for k in range(3):                          x=min(size1-1,i+m)                          y=min(size2-1,j+n)                          square\_diff=(int(left[x][min(y+d,size2-1)][k])-int(right[x][y][k]))\*\*2                          sum=sum+square\_diff              e[i][j]=sum/(3\*window\_size\*window\_size)      ##再计算e\_avg（i,j，d）      for i in range(size1):          for j in range(size2):              e\_temp = 0              for m in range(window\_size):                  for n in range(window\_size):                      x = min(size1-1, i + m)                      y = min(size2-1, j + n)                      e\_temp=e\_temp+e[x][y]              e\_temp=e\_temp/(window\_size\*window\_size)              if e\_temp<e\_avg[i,j]:                  e\_avg[i,j]=e\_temp                  disparity[i,j]=d |

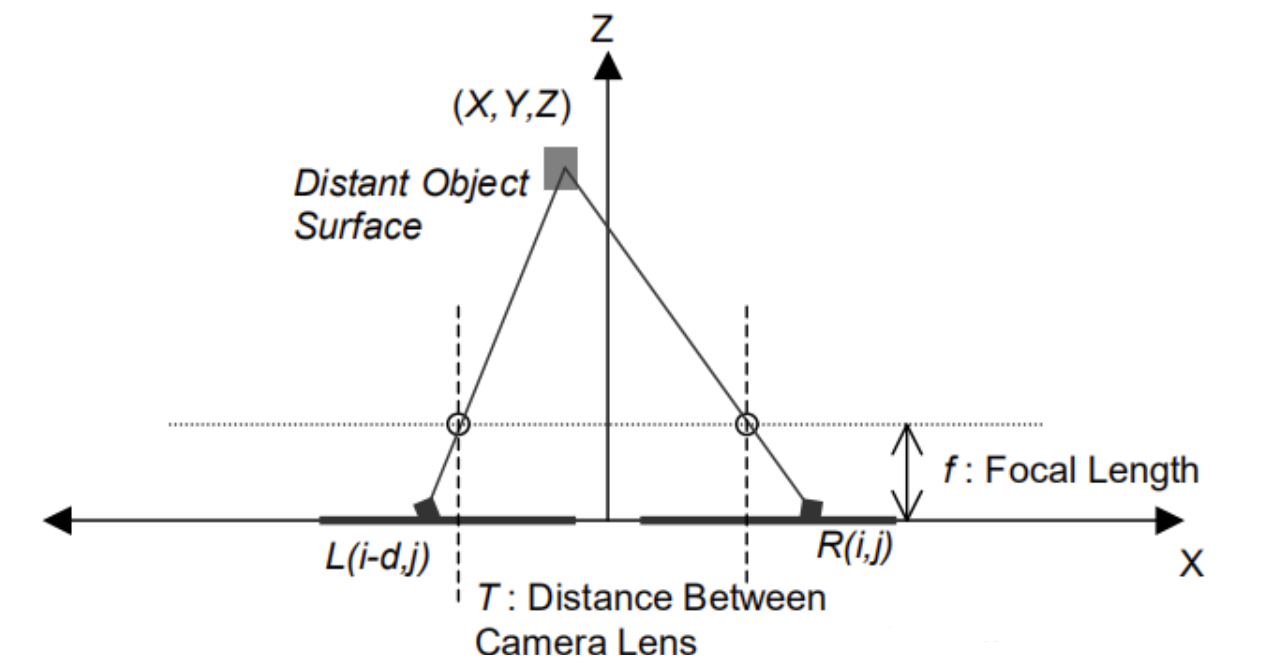
计算可靠度，生成具有可靠视差的视差图。

由于当先的视差图存在有噪音，因此，使用中值滤波的方法来对视差图进行处理。定一个阈值ve，将error energy低于它的筛掉。

|  |
| --- |
| ve=alpha\*e\_avg.mean()#设定可靠度阈值ve  count\_not\_ne=0  sum\_e=0  for i in range(size1):      for j in range(size2):          if e\_avg[i][j]>ve:              disparity[i][j]=0          else:              sum\_e=sum\_e+e\_avg[i][j]              count\_not\_ne=count\_not\_ne+1 |

由视差图生成深度图

根据这个三角关系，由视差图计算深度图，



其中将视差小于5的视为噪声处理，并对生成的图像进行对比度增强。

|  |
| --- |
| depth=np.ones\_like(disparity,dtype=np.uint8)  for i in range(size1):      for j in range(size2):          if disparity[i][j]<5: ##噪音              depth[i][j]=0          else:              depth[i][j]=f\*T//disparity[i][j]  cv2.medianBlur(depth,3)  cv2.imwrite('depth\_reliable\_medianblur.png', depth)  #######增强对比度显示  temp=cv2.imread('depth\_reliable\_medianblur.png')  gray = cv2.cvtColor(temp, cv2.COLOR\_RGB2GRAY) #opencv的直方图均衡化要基于单通道灰度图像  dst = cv2.equalizeHist(gray)  cv2.imwrite('depth\_reliable\_medianblur\_enhanced.png', dst)  temp2=cv2.imread('depth\_reliable\_medianblur\_enhanced.png')  print('具有可靠视差，中值滤波处理的深度图（原图->对比度增强显示）')  plt.imshow(np.hstack([temp,temp2])),plt.show() |

## 结果分析



1. （b）

如图所示，图a为左相机图像，图b为右相机图像。在执行程序后，得到视差图如下：



（c） （d）

其中，图(c)是未经过滤波和深度图噪声处理的视差图。图(d)则是经过中值滤波处理和深度图后生成的特征图，可以看出具有更加清晰的效果。

视差图中每个像素的取值范围是整数，通常为0到255之间。视差的最大值和最小值决定了深度的范围，即视差越大表示深度越浅，反之亦然。