## 计算机视觉工程实践报告5：图像视差匹配

**姓名：卓威 学号：123106222862**

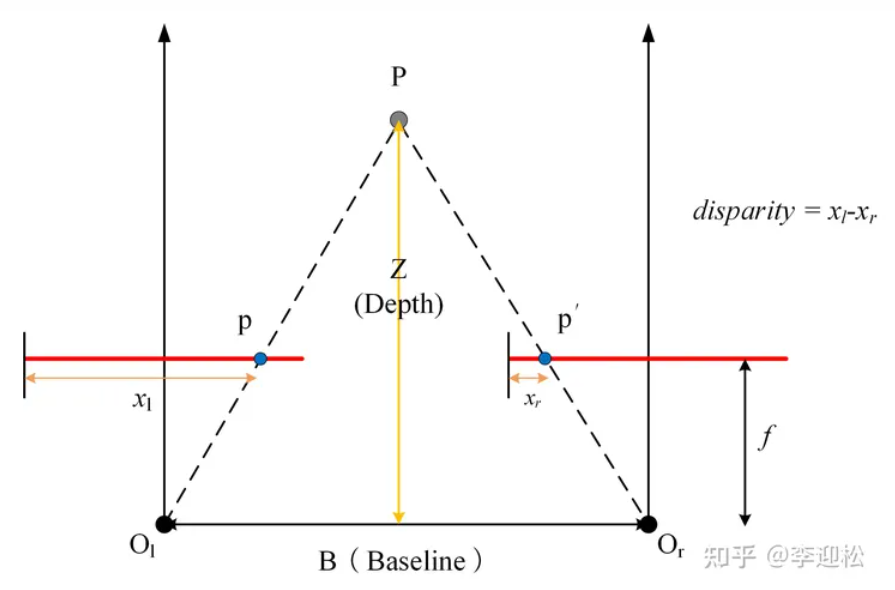
#### 实验目标

1. 理解视差的概念
2. 理解视差图匹配算法NCC
3. 使用python设计视差图计算算法
4. 对实验结果进行记录和分析

#### 实验原理

###### 2.1 视差（disparity）

视差等于同名点对在左视图的列坐标减去在右视图上的列坐标，是像素单位。在立体视觉里，视差概念在极线校正后的像对里使用。简单来说，视差图同一个场景在两个相机下成像的像素的位置偏差，因为通常下两个双目相机是水平放置的，所以该位置偏差一般体现在水平方向。比如场景中的点在左相机是坐标，那么在右相机成像则是坐标。d就是视差图中x坐标点的值。



2-1 视差原理图

###### 2.2 归一化相关性（NCC）

NCC是归一化相关性(normalization cross-correlation)的简称，NCC,就是用于归一化待匹配目标之间的相关程度，比较的是原始像素。通过在待匹配像素位置构建匹配窗口，与目标像素位置同样构建邻域匹配窗口的方式建立目标函数来对匹配窗口进行度量相关性，**注意这里构建相关窗口的前提是两帧图像之间已经校正到水平位置**，即光心处于同一水平线上，此时极线是水平的，否则匹配过程只能在倾斜的极线方向上完成。度量方式由如下式:



其中点表示图像待匹配像素坐标，表示在图像I2被查询像素位置在水平方向上与的距离。左边为图像，右边为图像。左视图和右视图两幅图像大小相同，只有水平方向上的视角变换。

#### 实验过程与结果分析

图4-1 实验原图img1、img2

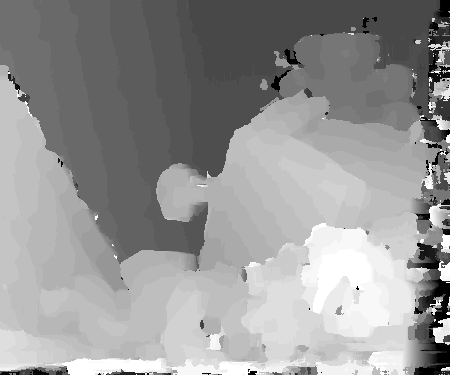
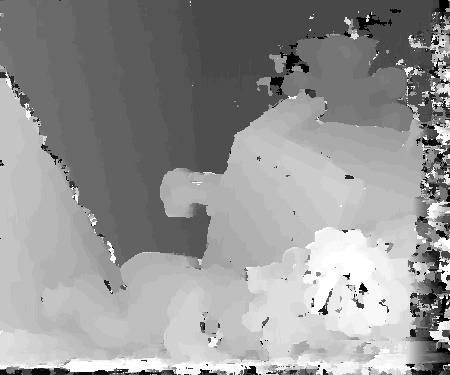


图4-2 NCC均匀滤波器视差图（width 3、9、15）

分析：当窗口较小时（width=3）匹配窗口中出现很多不连续的点，图像边缘特征也很模糊。

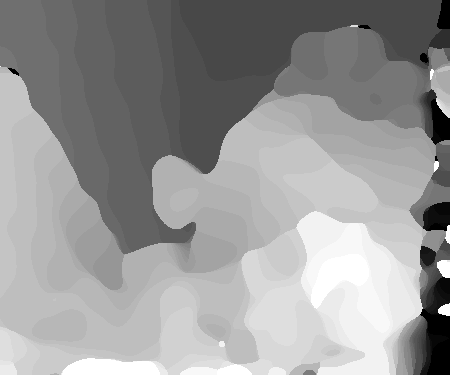


图 4-3 CC高斯滤波器视差图（width 3、9、15）

分析总结：从以上实验结果可以看出，当窗口值为3时视差匹配结果很模糊，基本看不清；把窗口值大小增大到9后，运行的匹配结果图片稍微清晰了一点，但仍无法看出图中是何物品，继续窗口值，可以很明显地看出，运行结果中的图片越来越靠近实物图像。所以，窗口值大小越大，视差图更清晰。但窗口值也不是越大运行结果就越清晰，当窗口值过大时，图片噪声也增多，从而使图片与实物图片相比更为杂乱，运行耗时越长。

#### 实验总结

从以上对im1和im2的计算结果可以看出窗口值过小时，在低纹理区域出现误匹配，匹配精度较低，视差图特别模糊；随着窗口值增大，匹配区分度逐渐清晰，误匹配区域得到矫正，匹配精度随着窗口值增大而变高。但当窗口值过大时，在深度区域容易出现误匹配。窗口值过小时，匹配代价区分度过低，在低纹理区域容易出现误匹配，匹配精度较低；窗口值的大小应适中，不宜过大也不宜过小。随着窗口值增大，匹配区分度逐渐清晰，误匹配区域得到矫正，匹配精度随着窗口值增大而变高。

#### 实验代码

from PIL import Image

from pylab import \*

import cv2

from numpy import \*

from numpy.ma import array

from scipy.ndimage import filters

import matplotlib.pyplot as plt

def plane\_sweep\_ncc(im\_l,im\_r,start,steps,wid):

""" 使用归一化的互相关计算视差图像 """

m,n = im\_l.shape

# 保存不同求和值的数组

mean\_l = zeros((m,n))

mean\_r = zeros((m,n))

s = zeros((m,n))

s\_l = zeros((m,n))

s\_r = zeros((m,n))

# 保存深度平面的数组

dmaps = zeros((m,n,steps))

# 计算图像块的平均值

filters.uniform\_filter(im\_l,wid,mean\_l)

filters.uniform\_filter(im\_r,wid,mean\_r)

# 归一化图像

norm\_l = im\_l - mean\_l

norm\_r = im\_r - mean\_r

# 尝试不同的视差

for displ in range(steps):

# 将左边图像移动到右边，计算加和

filters.uniform\_filter(np.roll(norm\_l, -displ - start) \* norm\_r, wid, s) # 和归一化

filters.uniform\_filter(np.roll(norm\_l, -displ - start) \* np.roll(norm\_l, -displ - start), wid, s\_l)

filters.uniform\_filter(norm\_r\*norm\_r,wid,s\_r) # 和反归一化

# 保存 ncc 的分数

dmaps[:,:,displ] = s / sqrt(s\_l \* s\_r)

# 为每个像素选取最佳深度

return np.argmax(dmaps, axis=2)

def plane\_sweep\_gauss(im\_l,im\_r,start,steps,wid):

""" 使用带有高斯加权周边的归一化互相关计算视差图像 """

m,n = im\_l.shape

# 保存不同加和的数组

mean\_l = zeros((m,n))

mean\_r = zeros((m,n))

s = zeros((m,n))

s\_l = zeros((m,n))

s\_r = zeros((m,n))

# 保存深度平面的数组

dmaps = zeros((m,n,steps))

# 计算平均值

filters.gaussian\_filter(im\_l,wid,0,mean\_l)

filters.gaussian\_filter(im\_r,wid,0,mean\_r)

# 归一化图像

norm\_l = im\_l - mean\_l

norm\_r = im\_r - mean\_r

# 尝试不同的视差

for displ in range(steps):

# 将左边图像移动到右边，计算加和

filters.gaussian\_filter(np.roll(norm\_l, -displ - start) \* norm\_r, wid, 0, s) # 和归一化

filters.gaussian\_filter(np.roll(norm\_l, -displ - start) \* np.roll(norm\_l, -displ - start), wid, 0, s\_l)

filters.gaussian\_filter(norm\_r\*norm\_r,wid,0,s\_r) # 和反归一化

# 保存 ncc 的分数

dmaps[:,:,displ] = s / np.sqrt(s\_l \* s\_r)

# 为每个像素选取最佳深度

return np.argmax(dmaps, axis=2)

im\_l = array(Image.open('data/teddy/im2.png').convert('L'), 'f')

im\_r = array(Image.open('data/teddy/im6.png').convert('L'), 'f')

# 开始偏移，并设置步长

steps = 40

start = 4

# ncc 的宽度

wid = 2

# res = plane\_sweep\_ncc(im\_l,im\_r,start,steps,wid)

res = plane\_sweep\_gauss(im\_l,im\_r,start,steps,wid)

import scipy.misc

scipy.misc.imsave('depth.png',res)