# 立体匹配生成图像视差图

## 实验目的：

通过立体匹配算法生成图像视差图，获取场景深度信息。

## 实验步骤：

1.数据准备：

准备一对经过校准的立体图像，包括左右两个视角拍摄的对应场景。

2.预处理：

对左右图像进行灰度化处理，去除噪声等预处理操作，以便后续算法运行。

3.特征提取：

使用SIFT算法在左右图像中提取特征点，并计算描述子。

4.特征匹配：

使用暴力匹配器对左右图像的特征点进行匹配。

5.立体匹配：

使用半全局匹配（SGM）算法计算视差。

6.视差图生成：

根据立体匹配算法计算得到的视差值，生成视差图。

7.视差图后处理：

对生成的视差图进行后处理，去除无效视差、填充视差空洞等。

8.可视化：

展示生成的视差图，观察场景的深度信息。

## 实验代码：

import stereo  
import scipy.misc  
from PIL import Image  
from pylab import \*  
from scipy.ndimage import \*  
  
  
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(roll(norm\_l, -displ - start) \* norm\_r, wid, s) # 和归一化  
 filters.uniform\_filter(roll(norm\_l, -displ - start) \* 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 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(roll(norm\_l, -displ - start) \* norm\_r, wid, 0, s) # 和归一化  
  
 filters.gaussian\_filter(roll(norm\_l, -displ - start) \* 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 / sqrt(s\_l \* s\_r)  
 # 为每个像素选取最佳深度  
 return argmax(dmaps, axis=2)  
  
  
im\_l = array(Image.open('1.png').convert('L'), 'f')  
im\_r = array(Image.open('2.png').convert('L'), 'f')  
# 开始偏移，并设置步长  
steps = 50  
start = 4  
  
# ncc 的宽度  
wid = 13  
  
res = plane\_sweep\_ncc(im\_l, im\_r, start, steps, wid)  
  
imsave('result'.jpg', res)

## 实验结果：

通过以上步骤，我们成功生成了一幅图像视差图，显示了场景中不同像素点的视差值。视差图可以用来推断场景中物体的深度信息，为三维重建和场景理解提供重要参考。



