**实验四：立体视觉**

**本实验单人完成**

# 实验环境

python：3.7.4

numpy：1.16.4

opencv：3.4.2

pillow：6.1.0

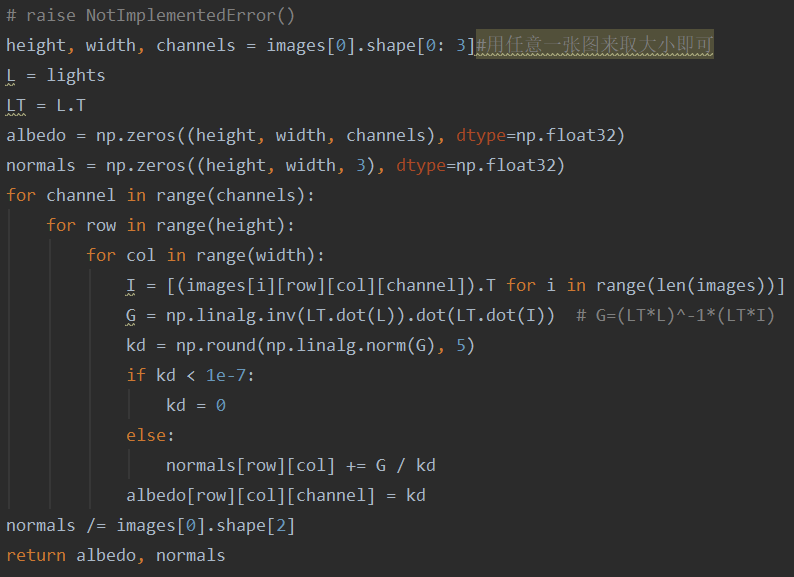
pytest：5.0.1

nose：1.3.7

imageio：2.6.0

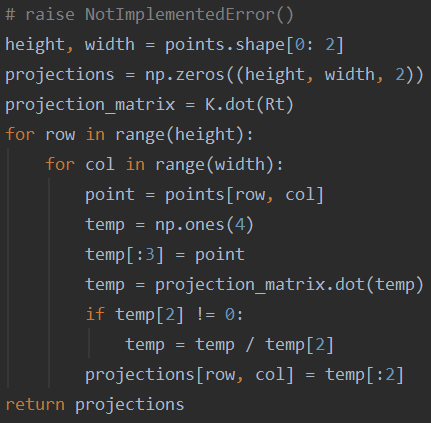
# 实现过程

## 实现compute\_photometric\_stereo\_impl函数



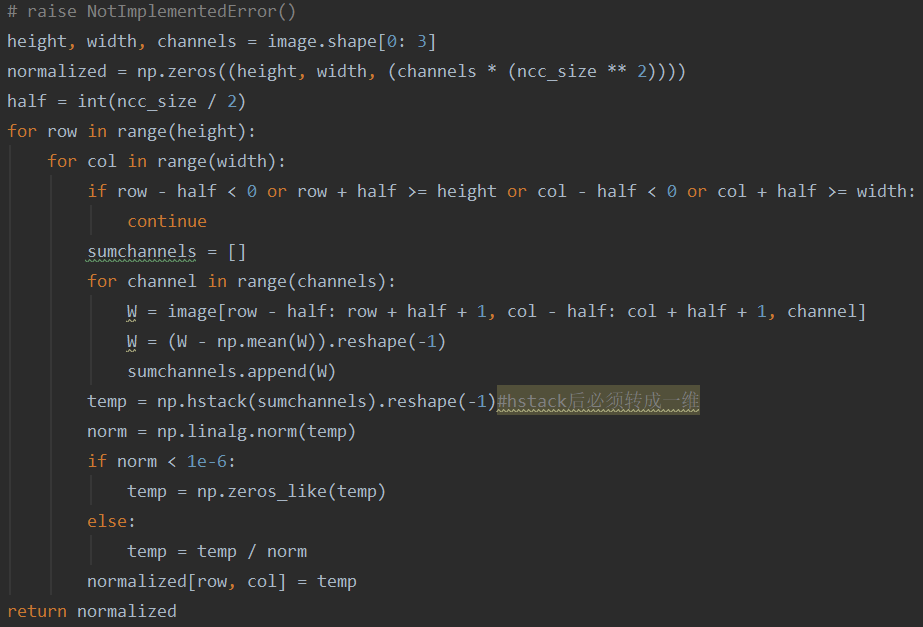
* 对每一个像素点利用公式G=(LT\*L)^-1\*(LT\*I)，进而计算出G的反照率kd。
* 按照函数给出的提示，在除以kd时进行判断，如果kd过小，则将其kd设为零。

## 实现project\_impl函数



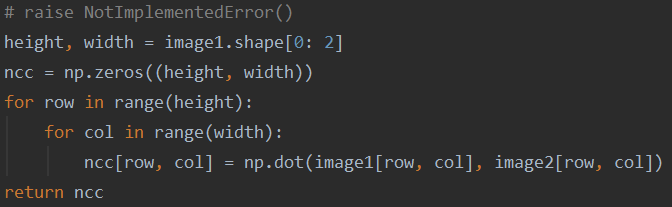
* 首先计算相机的投影矩阵。
* 再将世界坐标系中的每一个点转换成齐次坐标后进行投影。
* 最后对投影后的坐标进行归一化。

## 实现preprocess\_ncc\_impl函数



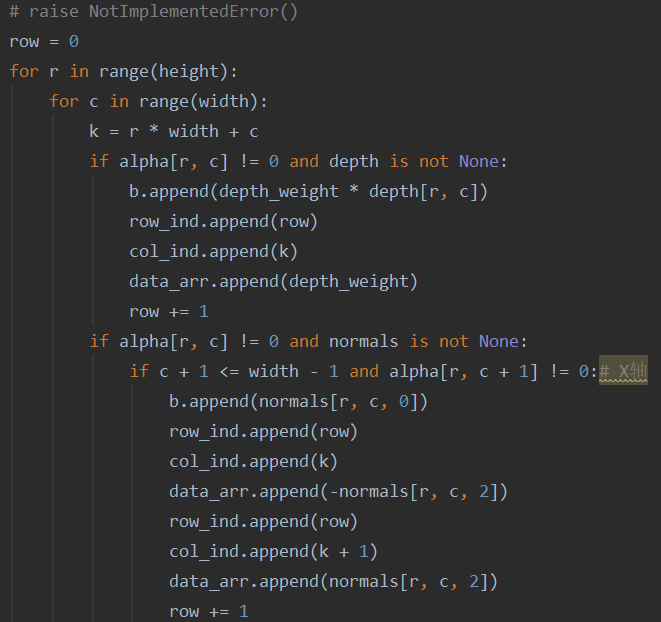
* 根据实验要求描述，跳过该点对应区域内有部分再边界外的点（因为创建的是全零矩阵，所以自动将对应向量设为全零）。
* 对每个通道的像素点单独求平均值。
* 再将各通道合起来计算欧几里得范数（即）。

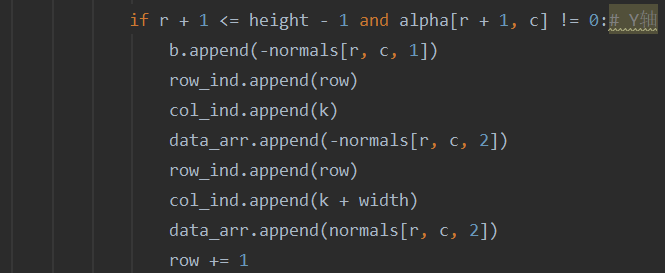
## 实现compute\_ncc\_impl函数



* 对两幅图片中每一对对应像素点计算ZNCC的值（用上一个函数预处理好的向量做内积），从而衡量这一对像素点的相关性。

## 实现form\_poisson\_equation\_impl函数

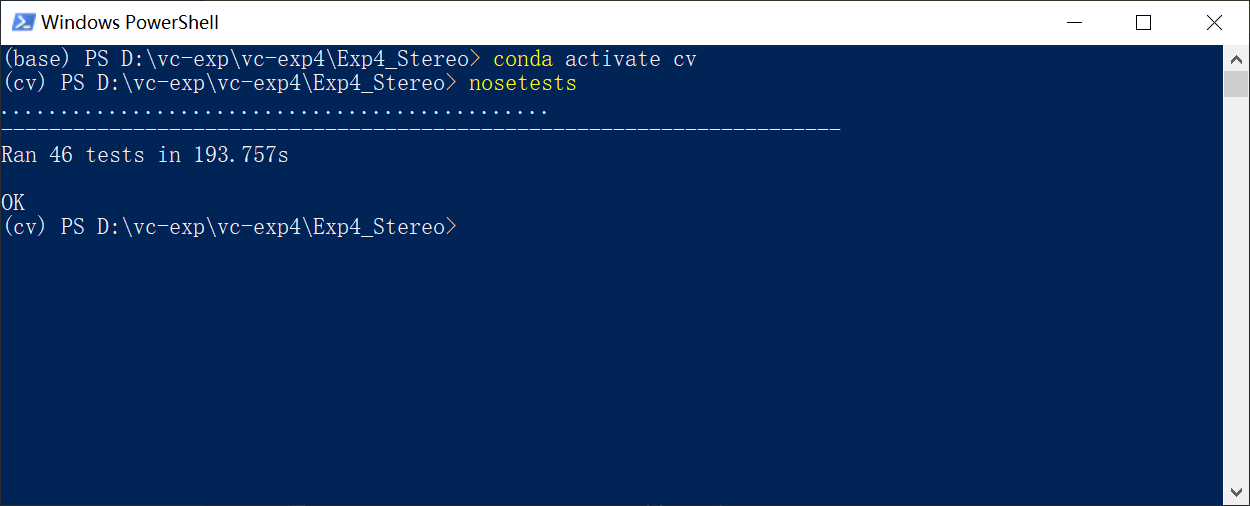




* 仔细阅读函数的提示，该函数本质是在为泊松方程Ax = b中的A，b进行赋值。
* 分析三种模式depth、normals和both发现both模式是前两种的结合，所以仅需按照函数提示对A（row\_ind、col\_ind、data\_arr构成A的稀疏矩阵）和b进行赋值即可。

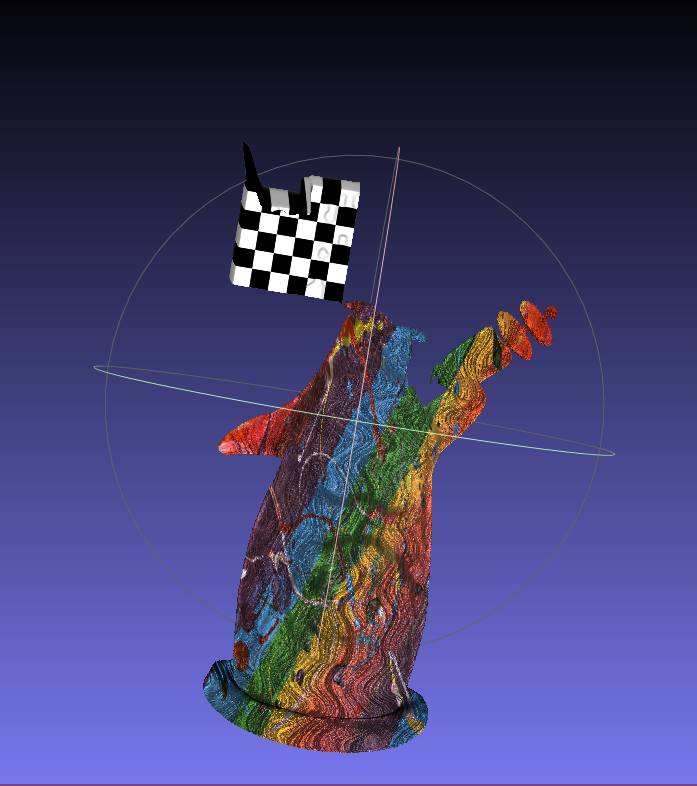
# 实验结果

## nosetests测试结果

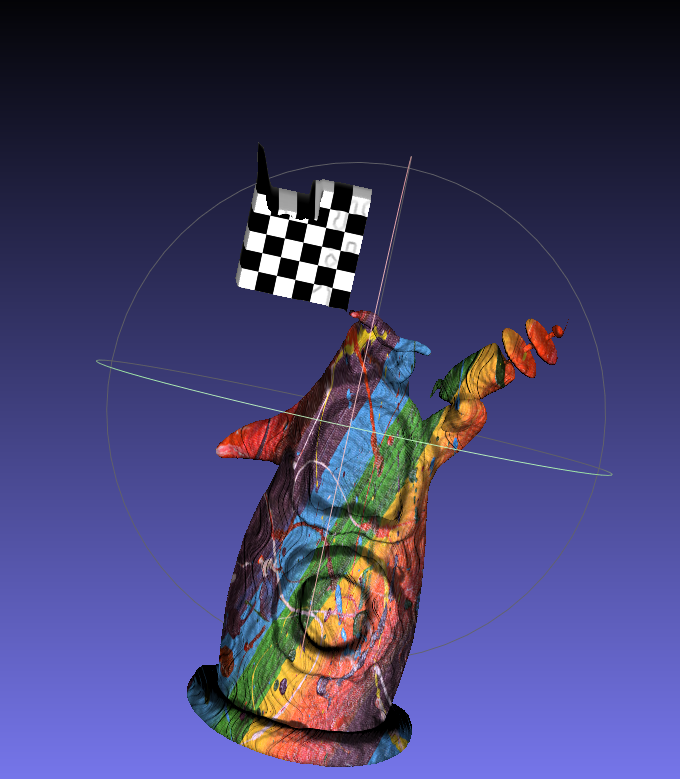


## Meshlab结果

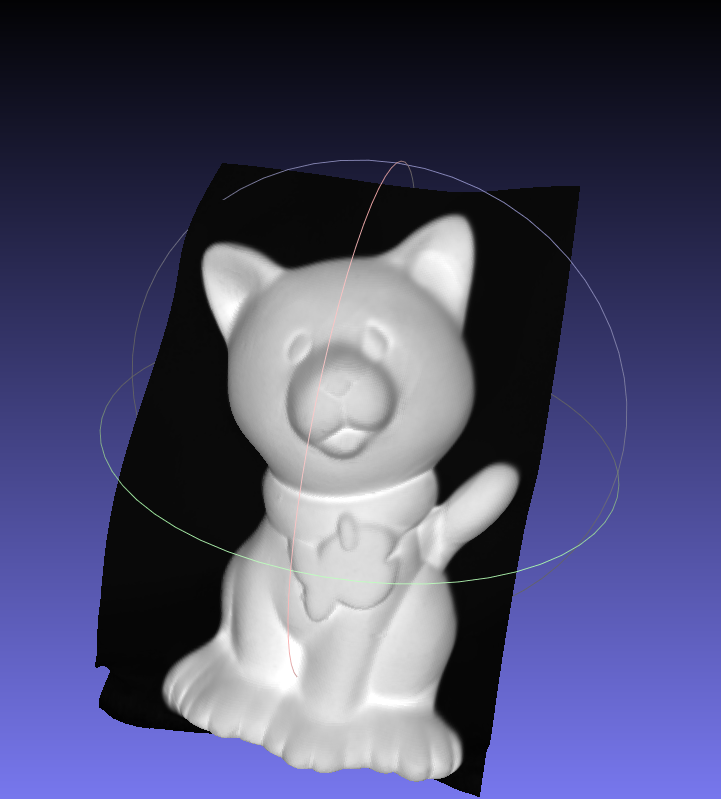
### 数据集tentacle，模式both



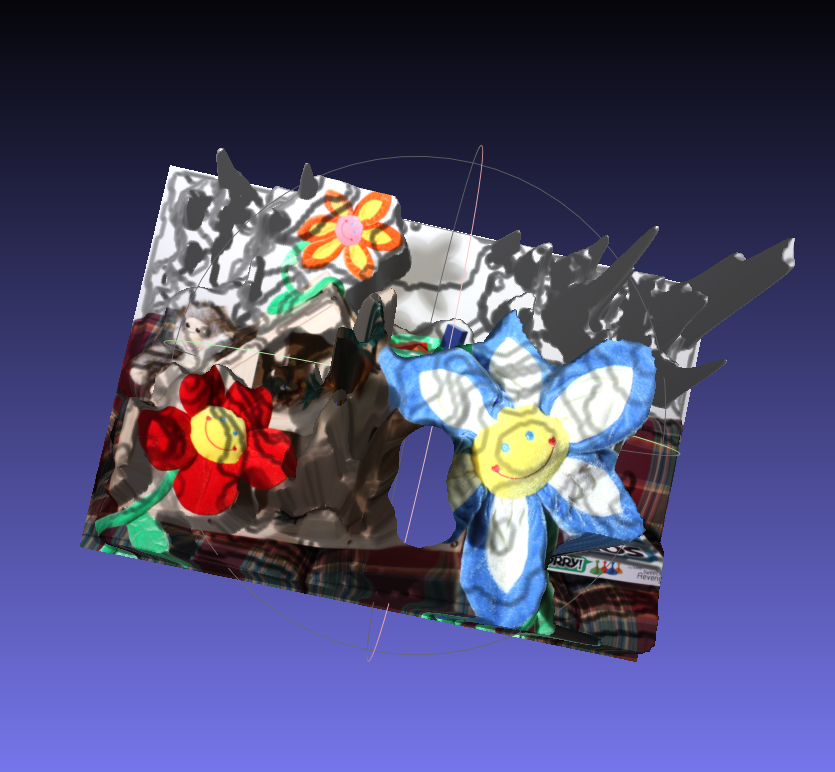
### 数据集tentacle，模式depth



### 数据集cat，模式normals



### 数据集flowers，模式depth



### 分析

不难看出，使用depth模式生成的图像非常不好，tentacle数据集使用both模式的图像边缘明显比使用depth模式更加细腻，而且flowers数据集在depth模式下甚至出现了紫色的大块斑点。可以说normals模式进一步修正了图片预先给出的depth中的误差。

## 光度测量立体视觉算法的问题

光度的量立体视觉的算法是基于朗伯反射的算法，但是朗伯反射假设了光的入射能量在所有方向均匀反射，但是现实中这个假设是不成立的。

## 平面扫描立体视觉算法的问题

对于平面扫描立体视觉算法，它是根据两幅图中相应位置点的相关性来选取该点的深度平面的，所以如何衡量相关性是这个算法的核心。如果能找到比ZNCC更好的匹配代价函数来衡量相关性，则可以还原更好的深度图像。

# 实验遇到的问题及心得

一开始在实现最后一个函数时发现该函数跟泊松方程关系不大，而且看不懂函数提示中为什么那样赋值。经过逆向工程，分析todo外的代码和上网查阅csr\_matrix函数，发现其实本质上就是一个稀疏矩阵的赋值问题。再次分析函数提示，理解了其赋值操作的意义。

还没有开始实验的时候看到别人的powershell无法运行nosetests，准备去装linux前抱着侥幸心理试了一下，成功通过，所以没有安装linux系统。