一、 实验目的

本实验将运用平面扫描立体视觉与光度测量立体视觉的方法，来恢复图像深度，并建立立体图。实验包含三个部分：

1. 光度测量立体视觉：给定在不同的已知光照方向下从相同视角拍摄的一组图像，从中恢复物体表面的反照率(albedo)和法线方向(normals)。

2. 平面扫描立体视觉：给定同一场景从不同的视角拍摄的两幅校准图像，从中恢复出粗略的深度图。

3. 基于泊松方程重建深度图：根据法线图及粗略深度图，恢复出物体每个点的深度，并重建 3D 网格。

二、 实验环境

1．Python 3.7.4

2．Windows 10 17134.950 安装了 Git

3．全部独立完成

三、 实验过程

1. 在光度测量立体视觉中，出现最大的问题是没注意导 iamges 是个向量。实现时先将 images 转成矩阵，然后同时对所有图片上同一个位置上的点计算反照率。利用公式 $G = (L^TL)^{-1}(L^TI)$，其中 L 是直接给出的，I 是 images 图像中所有图片同一个点处的像素值。由于法线 N 是一个单位向量，所以用 G/||G||得到 N。||G|| 就是反照率 kd，按题目要求，对于小于 1e-7 的 kd 直接看作 0。

2. 平面扫描立体视觉中，实现主要包括三个部分，分别是反向卷绕到参考平面，实现 ZNCC 预处理，计算两种图片间的归一化互相关，得到每个像素处的归一化向量。分别对应于函数 project_impl、 preprocess_ncc_impl、compute_ncc_impl。

   (1) 首先说 project_impl。这个函数的实现与上一次实现类似，都是将每个点先转成齐次坐标，再与内外参数相乘卷绕到参考平面。这里要注意的问题是 K*Rt 得到 3*4 的向量，所以要将像素升一维再乘，这里升维我使用的是 np.r_[]

   (2) 对于 preprocess_ncc_impl 函数。实现了 ZNCC 预处理，求平均值时对每个通道单独做，归一化时对所有通道一起做。我在每个通道的循环开始前，开了一个(height, width, ncc_size ** 2)的矩阵，来存每个通道中所有小区域内的信息。最终通过 dstack 把这些矩阵拼在一起形成(height,width,channels*ncc_size**2)矩阵。遍历每个通道的每个像素点，取像素点周围 ncc_size*ncc_size 区域(如果有超出图像范围的直接看作 0)，求区域内的平均值，并用区域内像素点减去这个平均值并记录到矩阵中。出现的问题是当我归一化时，一开始只保证不出现除 0，但是这样一直 F。后来通过输出结果矩阵，发现出现了 nan，猜测是分母太小导致溢出。最后通过 clip 来限制下限解决了溢出问题，但是对于 clip 的上限，我写 np.max()时就会 F，直接写一个很大的数时就通过，不太理解。

   (3) compute_ncc_impl 相对简单，只需要计算两张图片每个点除的内积即可。

3. 利用泊松方程重建深度图感觉应该是最难的，但是文档说的很清楚。在这个函数中，对于稀疏举证采用了三元组来存储。

   (1) 在 depth 模式中，对于反照率不为 0 的像素点 k，如果像素 k 的值为零，不要添加任何新的方程。否则，用 depth_weight*depth[k]填充 b 中的行，并用 depth_weight 填充 A 中相应行的 k 列。

(2) 在 normals 模式中， 对于每个像素 k 和它在 x 轴上的相邻像素 l，如果像素 k 或像素 l 的值为 0，则不添加任何新方程。否则，用 nx[k]填充 b 行(nx 是法向量的 x 分量)用-nz[k]填充 A 中相应行的 k 列，用值 nz[k]填充 k+1 列，沿着 y 轴重复上面的步骤，除了 nx[k]应该是-ny[k]。

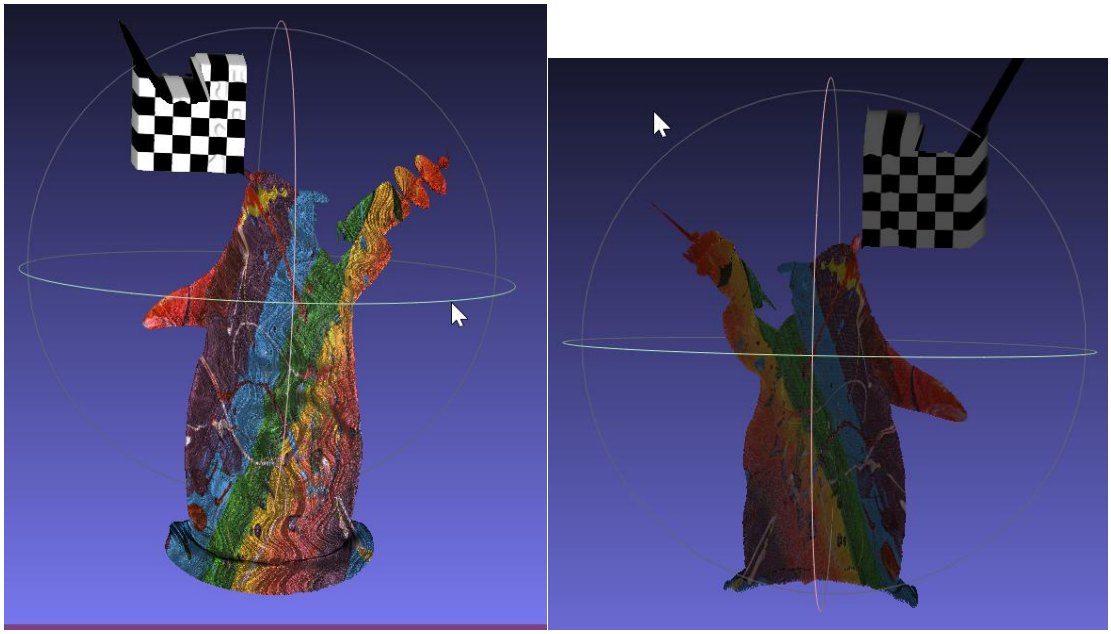
(3) 对于 both 模式，就是上述两种模式都要执行。

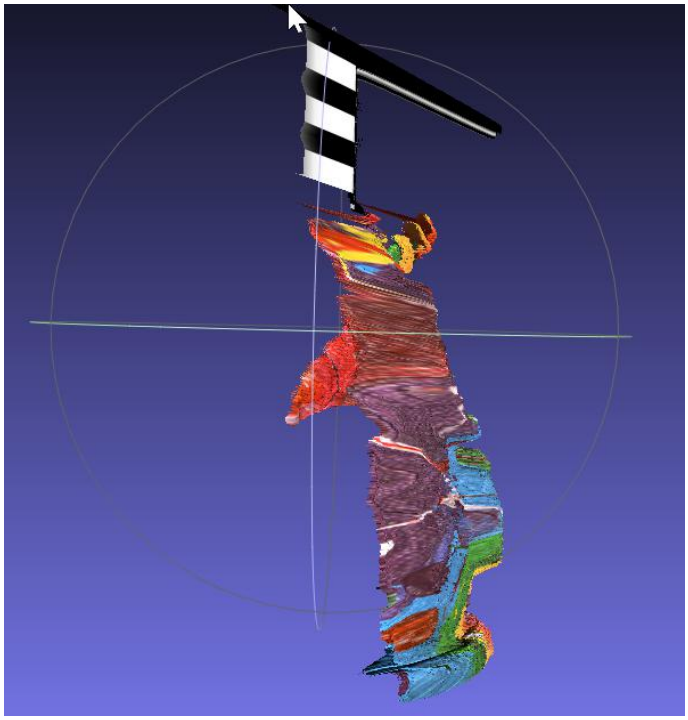
四、 实验结果

1. 所有测试通过

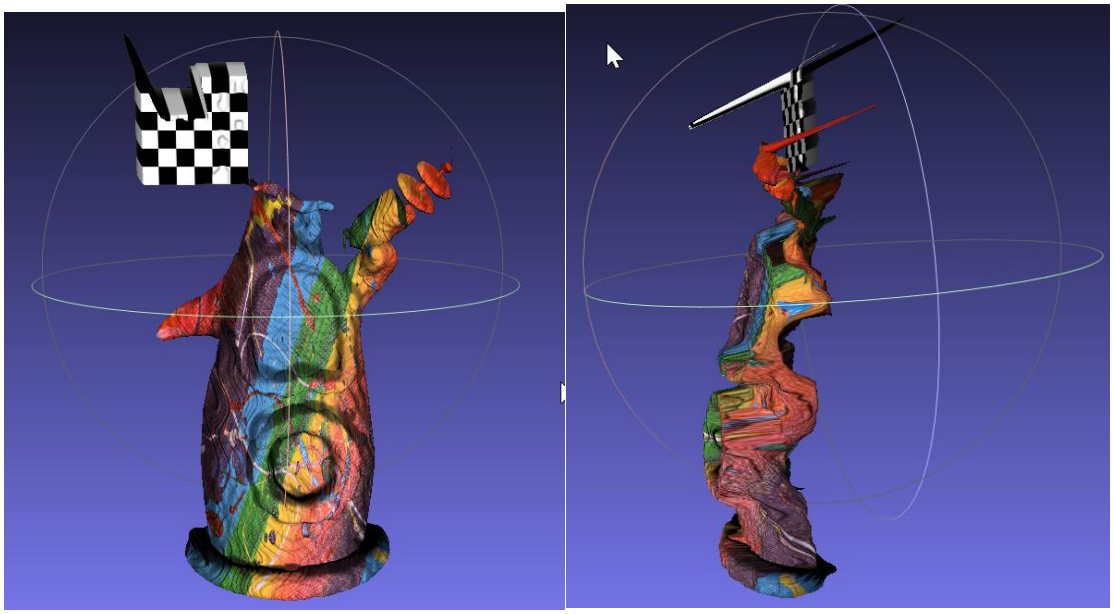
```
Ran 14 tests in 4.329s

OK (SKIP=13)
PS D:\CV_resorce\Exp4_Stereo> nosetests
.............................................
----------------------------------------------------------------------
Ran 46 tests in 358.547s

OK
```

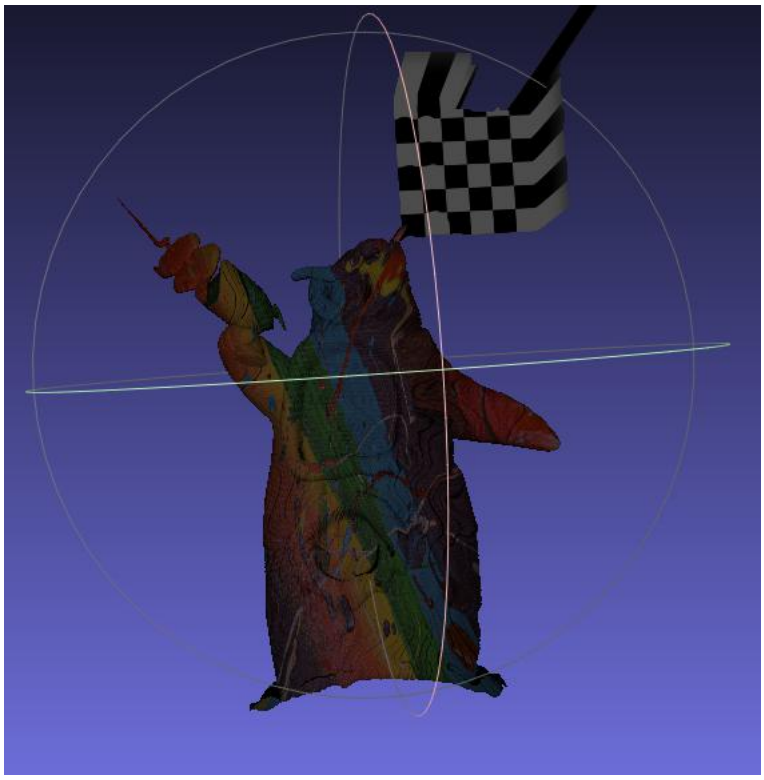
2. Both 模式下的 tentacle



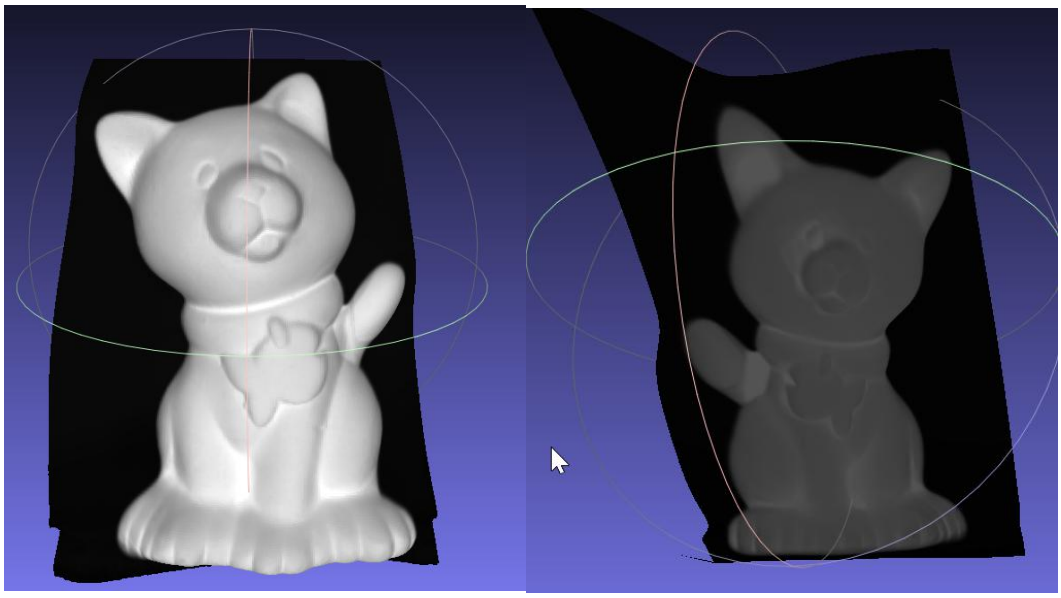


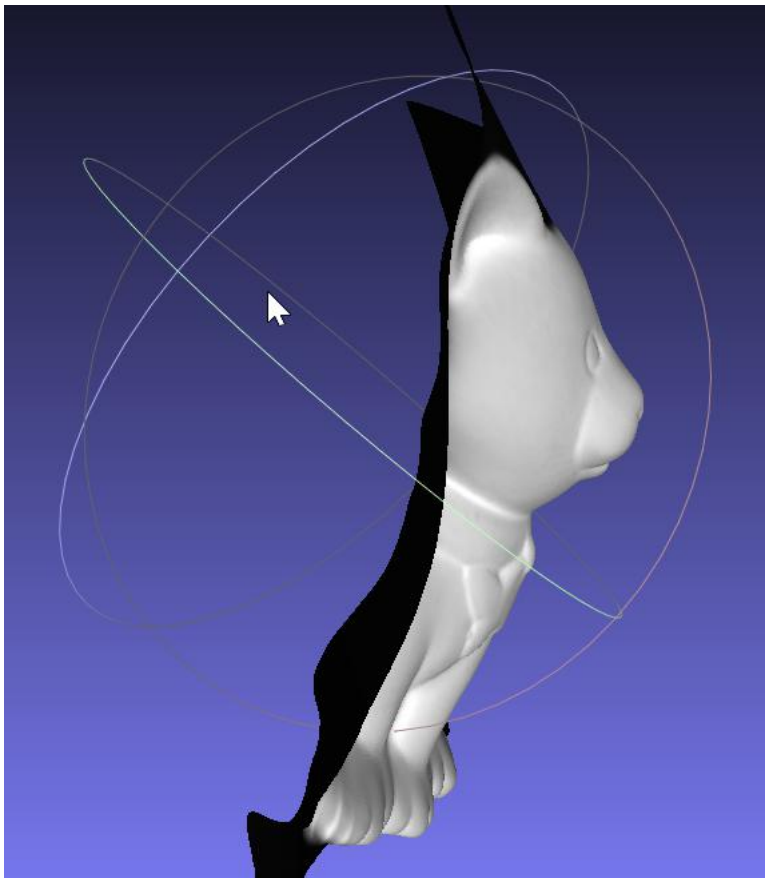
3. depth 模式 tentacle



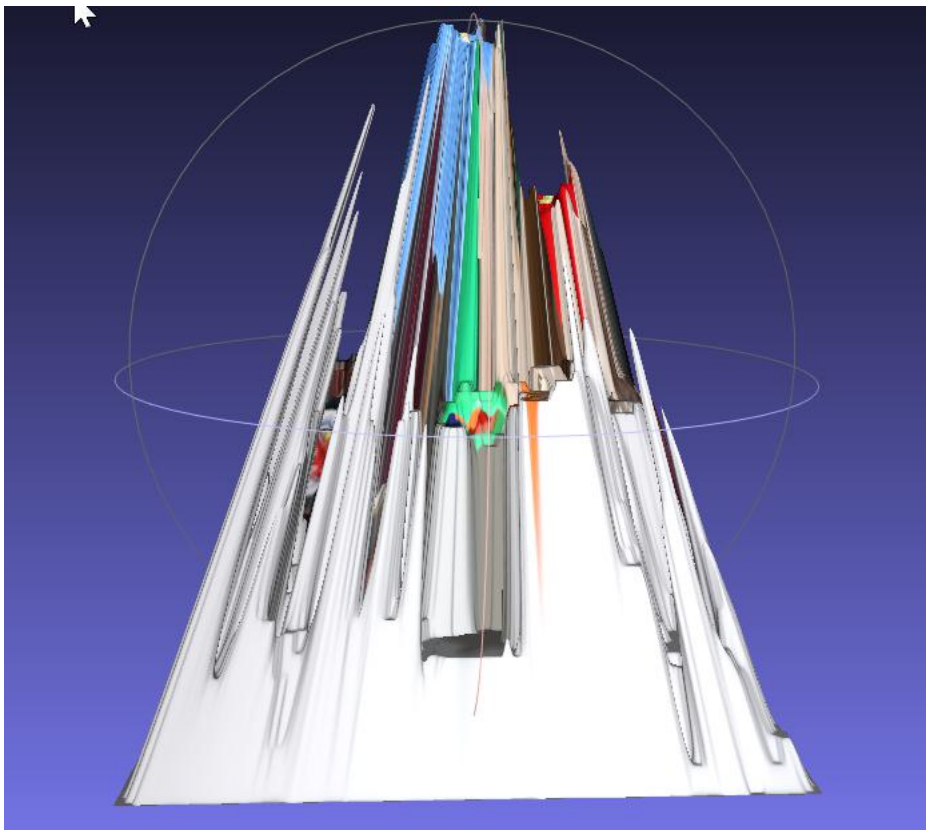


4. Normals 模式下 Cat





5. Depth 模式下 Flowers



五、 结果分析

1. 比较 both 模式和 depth 模式，发现 both 模式下，阴影不明显，深度差异区分较小。猜测可能是由于在平面扫描立体视觉中是通过试验各种离散的深度平面来寻找最佳深度，这样会导致深度差异不明显。
2. 对于 tentacle，两种模式下都发现黑白格处出现了凹凸，以及到输入数据边缘处，会出现很大的错误(一个大长条)。Depth 肚子处的圆圈还原的比较好，可能是因为源图像中这个位置突出，光线变化较为明显。
3. 对于 cat 数据集，整体感觉还不错，运行时候一直担心耳朵和脚趾的位置可能深度差不明显，不过结果看来还可以。
4. 前两个数据集都是做的图片，效果应该是不错的，但是 flowers 这个实际拍摄的图片结果就不太理想。尤其是墙面处人能看出来是墙面但是结果来看并不像。猜测可以通过阈值过滤使之看起来更平一些。
5. 总的来说，我认为平面扫描立体视觉得到的深是“相对的”，并不能很好的反应实际的深度。深度测量立体视觉基于朗伯漫反射，对于含有镜面反射，折射等图像不适用。

六、 实验心得

通过本次实验实现了平面扫描立体视觉和光度测量立体视觉，以及泊松方程计算深度。平面扫描立体视觉上课时候还不太懂，通过本次实验学习了一些细节，加深了理解。对于实验结果中的问题，理解还不深刻，不能给出完整的解释。在以后的学习中要注意积累经验，将这些问题解决。

编程能力还需要加强，我的实现运行起来挺慢的，还有努力学习。

另外，实验过程中发现提供的文件中 dataset.py 的_parse_K 函数的倒数第二行，要改成 lines = list(map(lambda x: list(map(float, x.strip().split())), lines))才能在 python3 中运行 flowers 数据集。