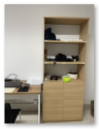
三维重建指导手册

概述

三维重建分为很多方法，比如激光雷达，结构光，基于相机的重建，我们这里做的基于单目相机的三维重建，也称之为多视图立体匹配（Multiple View Stereo，MVS）。也就是说通过相机拍摄物体不同视角来用点云恢复出物体结构，如下图所示：


00000000.jpg


00000001.jpg


00000002.jpg

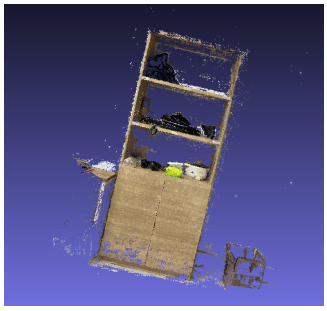

00000003.jpg


00000004.jpg


00000005.jpg


00000006.jpg


00000007.jpg



实现方法

不管使用哪种方法，首先要了解相机模型，传统方法中分享的网课有详细的介绍

传统方法

step1：对图像取特征点

step2：特征点匹配

step3：去除错误匹配点

step4：sfm恢复相机位姿

step5：稠密重建（也称mvs），方法推荐patchmatch（PMVS）

可以调的库：colmap（深度学习方法中详述）/openMVG（实现的是sfm）+openMVS（实现的是mvs）

可以参考的代码（用python调用的openMVG+openMVS）：

链接：https://pan.baidu.com/s/1yyCdkb6zgrXO4Awbnh8Beg
提取码：z1wi

可以参考的资料：

三维重建传统方法网课：

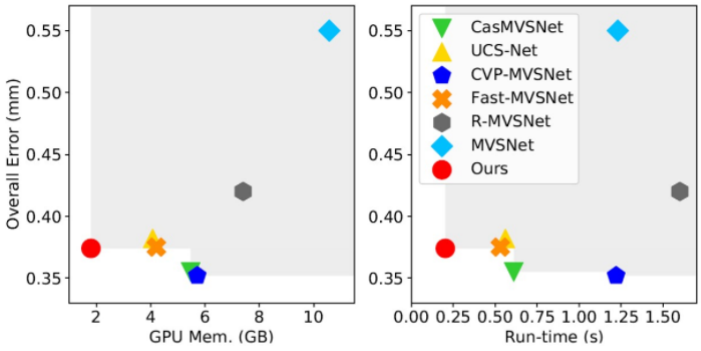
链接：https://pan.baidu.com/s/1PWRl3Uo08jzUMXeNr087PA
提取码：m5n0

《slam十四讲》（对各种重建的方法也有具体介绍，推理易懂，很好的工具书，强烈推荐）：

链接：https://pan.baidu.com/s/18RiSyCACpBz1fRpJy-Lsew
提取码：fkjx

深度学习方法

深度学习方法使用的是patchmatchnet，是精度，运算速度，存储都非常好的方法，下面是不同深度学习方法的比较（红色圆圈为patchmatchnet）：



(b) Comparison with state-of-the-art learning-based multi-view stereo methods on DTU. Relationship between error, GPU memory and run-time with image size 1152 x 864.

论文地址：链接：https://pan.baidu.com/s/1t_rbMP5WomeZNZhfh9IFpw
提取码：3cxx

代码地址：https://github.com/FangjinhuaWang/PatchmatchNet

深度学习方法数据集准备

深度学习方法的工作是做mvs的，所以还是需要借助传统方法sfm获得相机的内参和外参，这里使用colmap封装好的windows版图形界面实现。

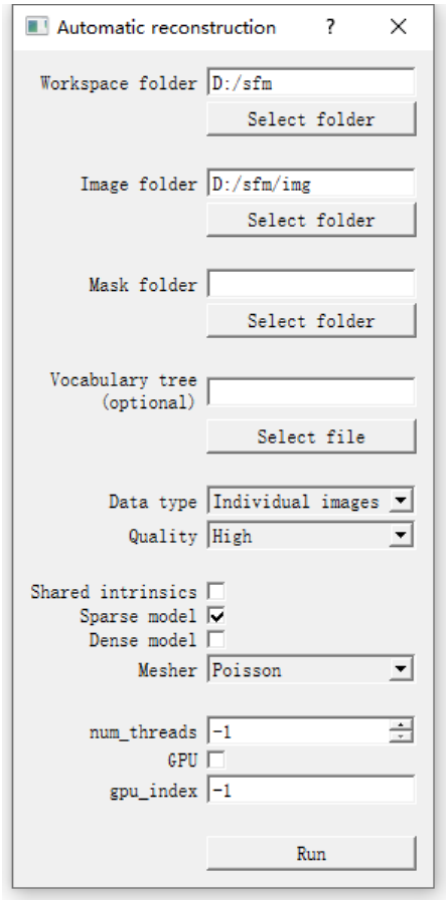
下载链接:https://github.com/colmap/colmap/releases
根据电脑是否支持cuda下载对应版本

重建示例：

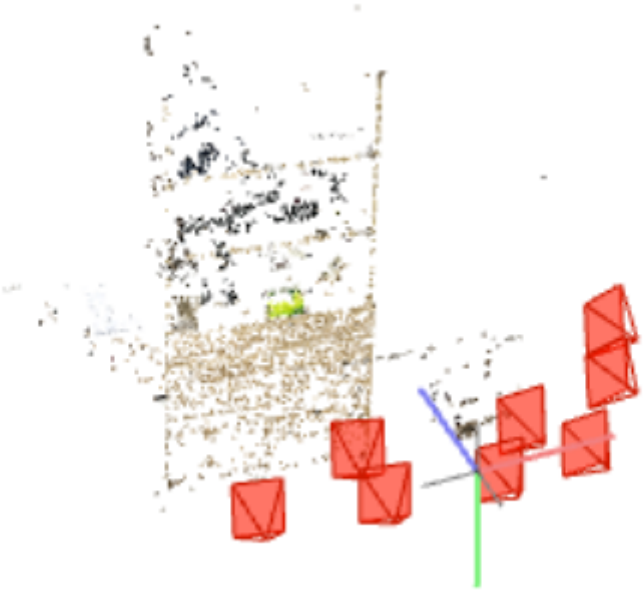
要对D:\sfm\img下的图片获得每张照片对应的位姿，图片如下所示：

step1：相机标定获得相机内参+去畸变+获得相机外参

选择reconstruction中的auto reconstruction出现下面的页面，选择Image folder，和Workspace folder，电脑没有cuda的话把GPU的勾去掉，点击run（如果run超出电脑屏幕可以调一下分辨率到1280*720）

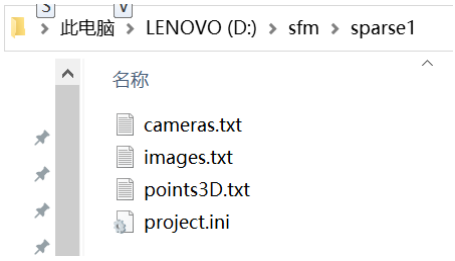


得到了相机的位姿（以其中一个特征点最多的相机坐标系作为世界坐标系），如下图所示：



step2：导出txt文本

点击file中的export moodel as txt保存到指定文件夹得到txt文件：



txt文件解读：

cameras.txt：

```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
# Number of cameras: 8
1 SIMPLE_RADIAL 3024 4032 3033.87 1512 2016 0.0217839
```

数据的每一行分别为相机序号，相机模型，图像的宽和高，相机的参数（焦距，相机成像平面到像素坐标系的位移（x和y方向），畸变系数）。每种相机对应的参数不同，这里是简单的径向相机，手机相机一般是这个模型。如果是别的相机模型，在colmap中应首先选择对应的相机模型。

images.txt：

```
# Image list with two lines of data per image:
#   IMAGE_ID, QW, QX, QY, QZ, TX, TY, TZ, CAMERA_ID, NAME
#   POINTS2D[] as (X, Y, POINT3D_ID)
# Number of images: 8, mean observations per image: 2335.5
1 0.994599 -0.00245424 -0.103759 0.000806696 5.16103 -0.434791 0.997857 4 702FA455E115F930B5CE239AC394649D.jpg
1086.1 1310.89 -1 2582.81 701.823 -1 1570.09 2390.62 -1 1508.24 1043.11 -1 650.102 2837.16 -1 1961.27 1136.42 -1 1911.21 1804.19 -1 2475.93 2242.73 -1 670.094
```

IMAGE_ID：图像的序号，Q利用四元数表示旋转，T表示平移，CAMERA_ID表示对应的相机，NAME表示图片名，紧接着给出每张图片上得到的特征点（X，Y）和对应的3D点PIOINT3D_ID，-1表示没有此2D点对应的3D点

points3D.txt：

```
# 3D point list with one line of data per point:
#   POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)
# Number of points: 5549, mean track length: 3.3670931699405298
1 -0.88000163333869874 -0.56842209674271682 12.687085008194769 2 4 3 1.7981421322273892 5 13 6 5 2 13
```

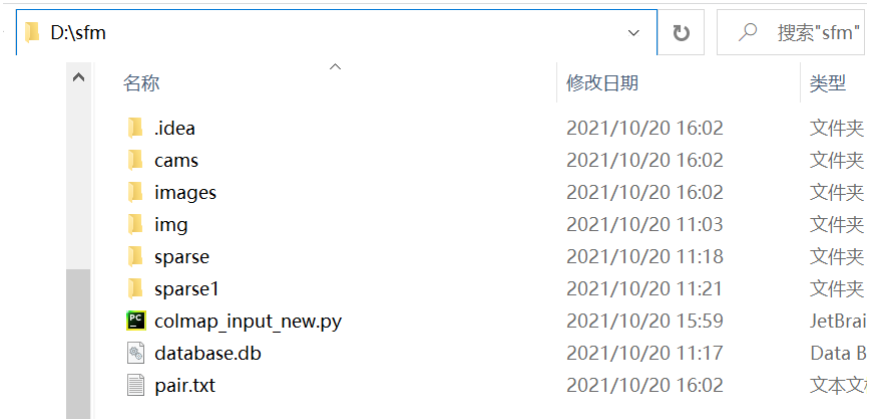
每一行表示的是3D点的ID，空间坐标，颜色信息，错误（在不同视角看一个点的视线不可能完全重合为一个点），该3D点对应的图像和上面的2D点的ID

step3：将数据格式整理为patchmatchnet的输入格式

在patchmatchnet中有一个input_custom.py，把这三个txt文件输入进入就会得到patchmatchnet的输入格式，但是这个文件有一块是并行操作的可能会运行不出来，我改写了一下，地址为：

链接：https://pan.baidu.com/s/1gWYs4OvuunO2z6rKPk0tYg
提取码：xp5r

将这个代码放在工作目录下可直接运行生成三个文件：images，cams，pair.txt，现在的文件夹内容如下所示：



images里存放的信息是重命名后图片的信息，patchmatchnet读取图片会按照这种命名方式读取

cams存储的是每个照片的内参，外参和该照片对应的三维点的最大和最小深度

00000000_cam.txt	2021/10/20 16:02	文本文档	1 KB
00000001_cam.txt	2021/10/20 16:02	文本文档	1 KB
00000002_cam.txt	2021/10/20 16:02	文本文档	1 KB
00000003_cam.txt	2021/10/20 16:02	文本文档	1 KB
00000004_cam.txt	2021/10/20 16:02	文本文档	1 KB
00000005_cam.txt	2021/10/20 16:02	文本文档	1 KB
00000006_cam.txt	2021/10/20 16:02	文本文档	1 KB
00000007_cam.txt	2021/10/20 16:02	文本文档	1 KB

pair.txt是视角的配对文件，里面包含了每个视角和其他视角的匹配程度和排名，用来选择视角进行重建

```
8
0
8 3 688.994902 5 677.288037 4 55.712159 7 24.857942 1 2.724577 6 0.453897 2 0.007346 0 0.000000
1
8 7 1197.270157 6 1140.549115 4 1132.056299 2 522.517228 5 153.701223 3 58.229365 0 2.724577 1 0.000000
2
8 6 1075.863405 1 522.517228 7 244.573680 4 85.037574 5 1.960610 3 0.743746 0 0.007346 2 0.000000
3
8 5 1376.852151 0 688.994902 4 470.292380 7 279.135252 1 58.229365 6 10.883371 2 0.743746 3 0.000000
4
8 7 1414.982317 1 1132.056299 5 1041.342204 3 470.292380 6 425.353226 2 85.037574 0 55.712159 4 0.000000
5
8 3 1376.852151 4 1041.342204 0 677.288037 7 493.179033 1 153.701223 6 24.343724 2 1.960610 5 0.000000
6
8 1 1140.549115 2 1075.863405 7 794.707805 4 425.353226 5 24.343724 3 10.883371 0 0.453897 6 0.000000
7
8 4 1414.982317 1 1197.270157 6 794.707805 5 493.179033 3 279.135252 2 244.573680 0 24.857942 7 0.000000
```

第一行说明图片的数量，下面是说和每个视角匹配的视角（按分数从高到低排列），例如和0视角匹配的视角为3视角，分数为688.994902，接下来是5视角，分数为677.288037以此类推。

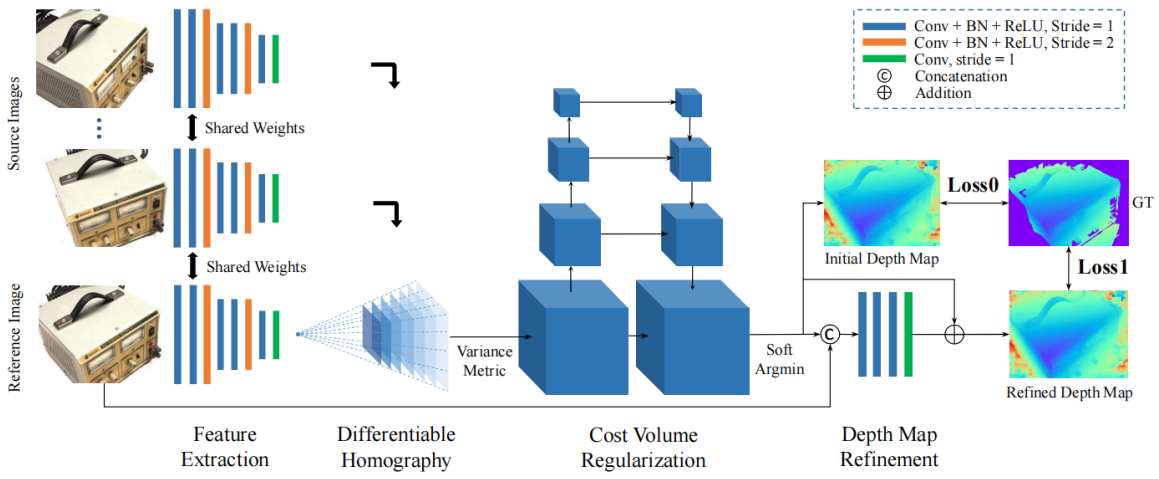
step4：将准备好的三个数据放在data文件夹中放到patchmatchnet项目下，放入神经网络即可得到结果（也可修该代码自定义文件路径）

深度学习算法介绍

三维重建的深度学习思路基本相同，拿最简单的神经网络MVSnet来说：

论文地址：

链接：https://pan.baidu.com/s/1ByIsZJzogvG7R0vOsTCiZQ
提取码：relh



step1：根据pair.txt选择视角，将选择的视角经过一个CNN得到特征图，目的是通过卷积考虑邻域像素。设原图大小为[B,C,H,W]，经过卷积层之后变为[B,32,H/4,W/4]

step2：假设平面有D个深度，将所有的src（Source Image）warp到ref（Reference Images）上并进行代价聚合得到cost volume，维度为[B,32,D,H/4,W/4]，注意：论文中的Differentiable Homography公式是错的，大家自己推导一下，实际上也用不到，根据相机模型可以将src对应的像素warp到ref上。代码里做这一步也有错误，希望大家能自己推导一下。

step3：3D卷积回归+softmax得到概率体，维度为[B,D,H/4,W/4]

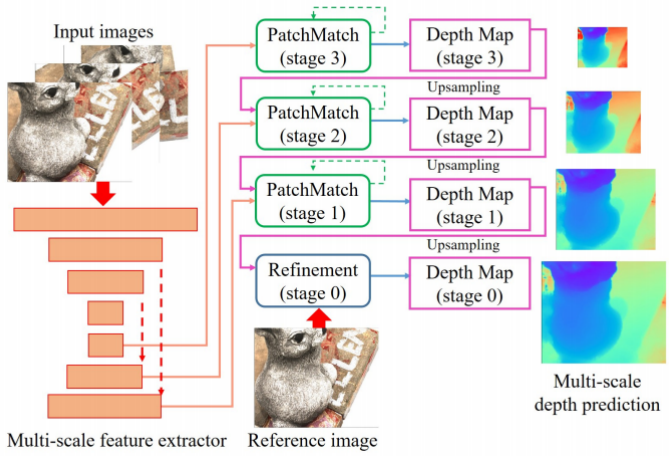
setp4：回归将每个像素的深度和该深度概率乘积求和得到每个像素的深度，再上采样，得到初始深度图[B,1,H,W]

step5：将归一化的初始深度图和归一化的原图拼接送入残差网络得到refine之后的深度图，计算loss

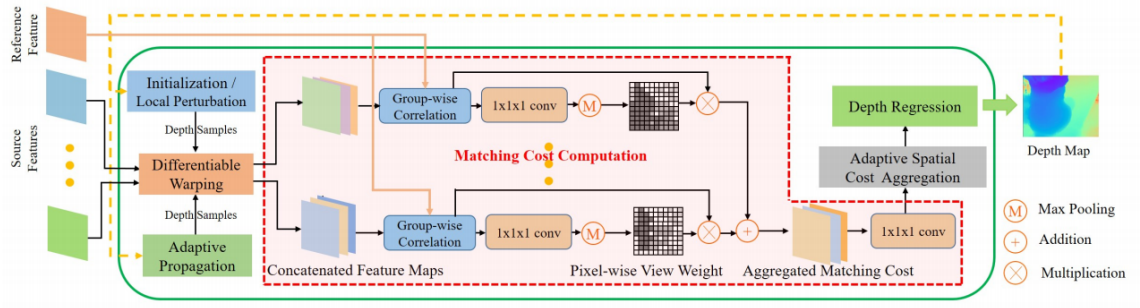
step6：通过几何一致性和光度一致性滤波，将深度图杂点滤掉，再利用原图的颜色，生成RGB点云

patchmatchnet是对MVSnet做了几处修改：

首先采取了金字塔结构逐层提取特征迭代，实现coarse to dense

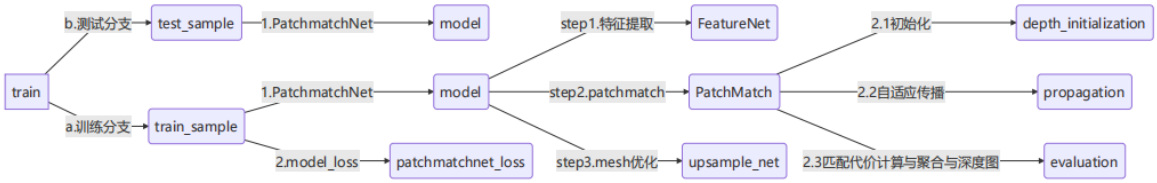


每一层的结构其实是MVSnet的基本流程，但对其进行了很大程度的优化：



最大的优化就是采用了自适应的邻域信息，体现在两个方面。一个是通过邻域的信息和上次生成的深度图来确定每个像素的深度范围而不像MVSnet对一张图像做平面假设，另一个是用邻域信息回归代价体，大大加快了3D卷积的回归速度。

patchmatchnet代码框架：



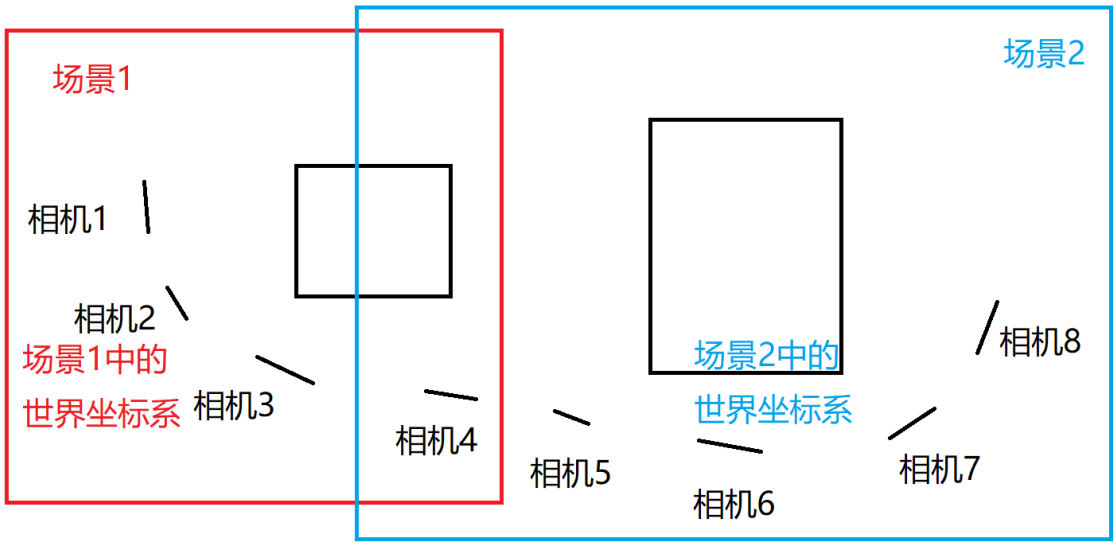
训练数据集的网址：https://polybox.ethz.ch/index.php/s/ugDdJQIuZTk4S35，数据集比较大，这里我提供训练了16轮的参数（代码里也提供了训练了8轮的参数）：

链接：https://pan.baidu.com/s/1YzInYdfGwm03QV4CPuEGRQ
提取码：vcs2

局部重建实现全局重建（初步构想）

多视角立体匹配是对一个物体或场景多角度拍摄图片得到三维重建的结果，本质上是一种局部重建。校园中包含很多场景和建筑，如何实现校园重建呢？很自然的想法是通过匹配局部重建得到全局重建的结果，给出两种理论上可行的构想：

考虑以下情景，通过相机1、2、3、4建立出来了场景1的点云（以相机2作为世界坐标系），通过4、5、6、7、8得到了场景2的点云（以相机6作为世界坐标系），如何将场景1的点云和场景2的点云进行融合呢？很简单的想法是点云的配准，但是这两个点云处在两个不同的坐标系中，是不同尺度的，配准无法完成这样的工作。



那么解决的方法就是将场景1和场景2的点云转换为同一个坐标系下

第一种构想是通过公共相机4来实现

step1：将场景2的点云由相机6转换到相机4下

设 $\begin{bmatrix} x \\ y \\ z \end{bmatrix}_6$ 为相机6坐标下的点，设相机6到相机4的变换矩阵为$T_{64} = \begin{bmatrix} R_{64} & t_{64} \\ 0 & 1 \end{bmatrix}$，相机4到相机6的变换矩阵为

$T_{46} = \begin{bmatrix} R_{46} & t_{46} \\ 0 & 1 \end{bmatrix}$，则$T_{46} = T_{64}^{-1} = \begin{bmatrix} R_{64}^{\top} & -R_{64}^{\top} t_{64} \\ 0 & 1 \end{bmatrix}$，那么在相机4下的点云坐标为

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_4 = T_{46} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_6$$

step2：将相机4下的坐标变换到相机2下，同step1

如此即可得到场景2的点云在场景1中的世界坐标系下的坐标，通过直接融合或者通过配准方式融合都可以。

第二种构想是通过ICP算法来实现。可以参考《slam十四讲》184页3D-3D：ICP

step1：通过特征图寻找匹配点，方法是先利用几何一致性和光度一致性寻找两个点云的匹配点（即两个点云中的公共点）

step2：假设点云二到点云一的坐标变换为R、t，优化坐标变换后的误差

这只是提供了两个理论可行的方案，具体实践上可能会有较大误差，希望同学们在实践中摸索出一套可行的方法并把结果反馈给我

后续对点云处理

目标检测和实例分割

这里主要是实现对得到的点云进行零部件的尺寸测量，那么可以想到的方法为对点云进行目标检测和实例分割用box把零部件框出来后再进行测量。

参考资料：三维点云课程，第五章开始是深度学习处理点云的基础

链接：https://pan.baidu.com/s/1oCcsoZ3ZrGHpawttzCZY0g
提取码：073x

曲面重建

把点云用曲面拟合起来，重建的表现性会更强。具体实现在传统方法做三维重建的网课中有讲