



# 城市空间建模与仿真

第十讲 城市空间三维数据表达和重建-边沿特征检测与前后景分割

任课教师：汤圣君  
建筑与城市规划学院 城市空间信息工程系



# 目录

## CONTENTS

- 01 边沿特征提取与检测**
- 02 前后景分离**
- 03 课堂练习**



# 目录

## CONTENTS

01

边沿特征提取与检测

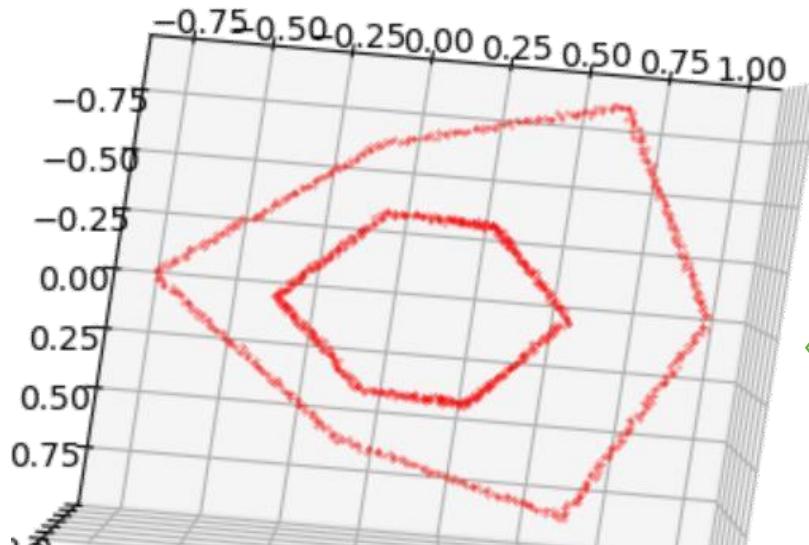
# 特征提取与检测-边沿检测

- 应用举例——边沿检测是测量和识别的关键
  - 螺丝丢失识别、焊缝位置检测、安装错位检测
  - 机器人移动过程中道路边线识别，阶梯检测
- 两个解决思路
  - 基于深度图的边沿检测
    - 直接接口3D传感器的数据，实时处理，效率高
    - 多种算法依赖近邻计算，深度图搜索快
    - 检测结果受视角影响（因为深度图看成是2.5D视图，有遮挡效应）
  - 基于点云的边沿检测
    - 应用于稀疏或者多传感器/多视角合成的点云
    - 需要解决点云无序性和密度不均匀
    - 需要解决近邻查询效率问题



# 特征提取与检测-边沿检测

- 把深度图当成强度图直接处理
- 边沿检测的方法
  - Canny/Sobel算子
  - 局部方差计算（在之前滤除飞散点噪声时用过）

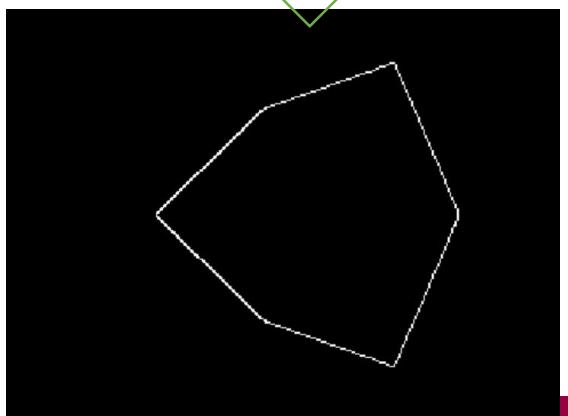
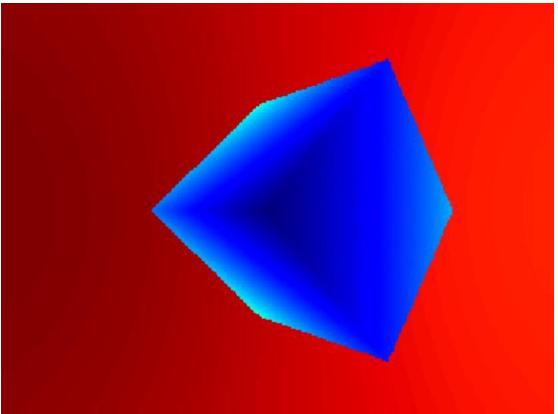


注意：图中外圈多边形是立方体遮挡在成的点云“空洞”的边界

将深度图转成点云，并对检出边沿的深度图像素用红色点云表示

`cv2.Canny(*)`  
`cv2.Sobel(*)`

用边沿检测算子检测边沿



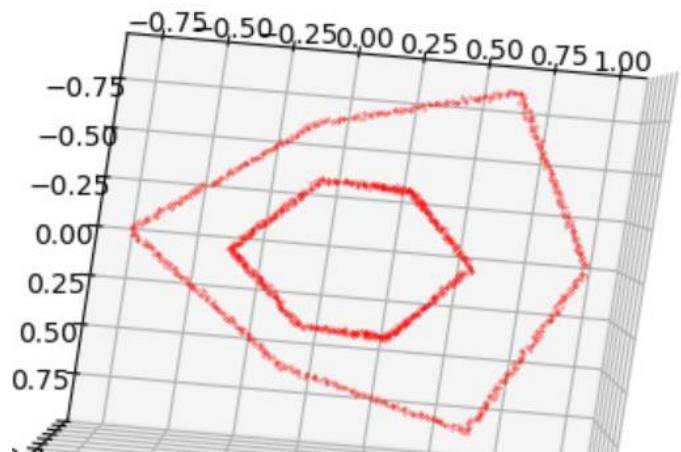
# 特征提取与检测-边沿检测

## 代码实现

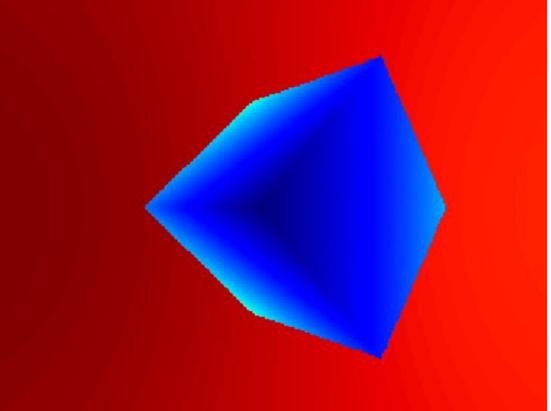
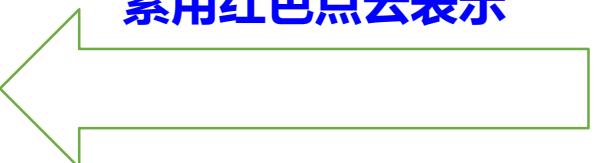
```

17  ## 基于深度图的边沿检测
18 def img_z_edge_det(img_z,win=5,th1=100,th2=200,mode='sobel'):
19     if mode=='sobel':
20         sobelx = cv2.Sobel(img_z.astype(np.float32),cv2.CV_32F,1,0,ksize=win)
21         sobely = cv2.Sobel(img_z.astype(np.float32),cv2.CV_32F,0,1,ksize=win)
22         return np.sqrt(sobely**2+sobelx**2)
23     if mode=='canny':
24         vmax=np.max(img_z)
25         vmin=np.min(img_z)
26         img_u8=((img_z-vmin)/(vmax-vmin)*255).astype(np.uint8)
27         return cv2.Canny(img_u8,th1,th2)
28     if mode=='var':
29         return cv2.blur(img_z**2,(win,win))-cv2.blur(img_z,(win,win))**2
  
```

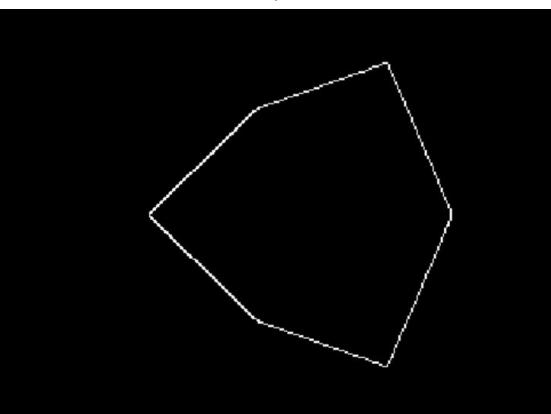
注意：上面代码返回边沿“强度”，需要将他和门限比较，来确定属于边界的点



将深度图转成点云，并对检出边沿的深度图像素用红色点云表示

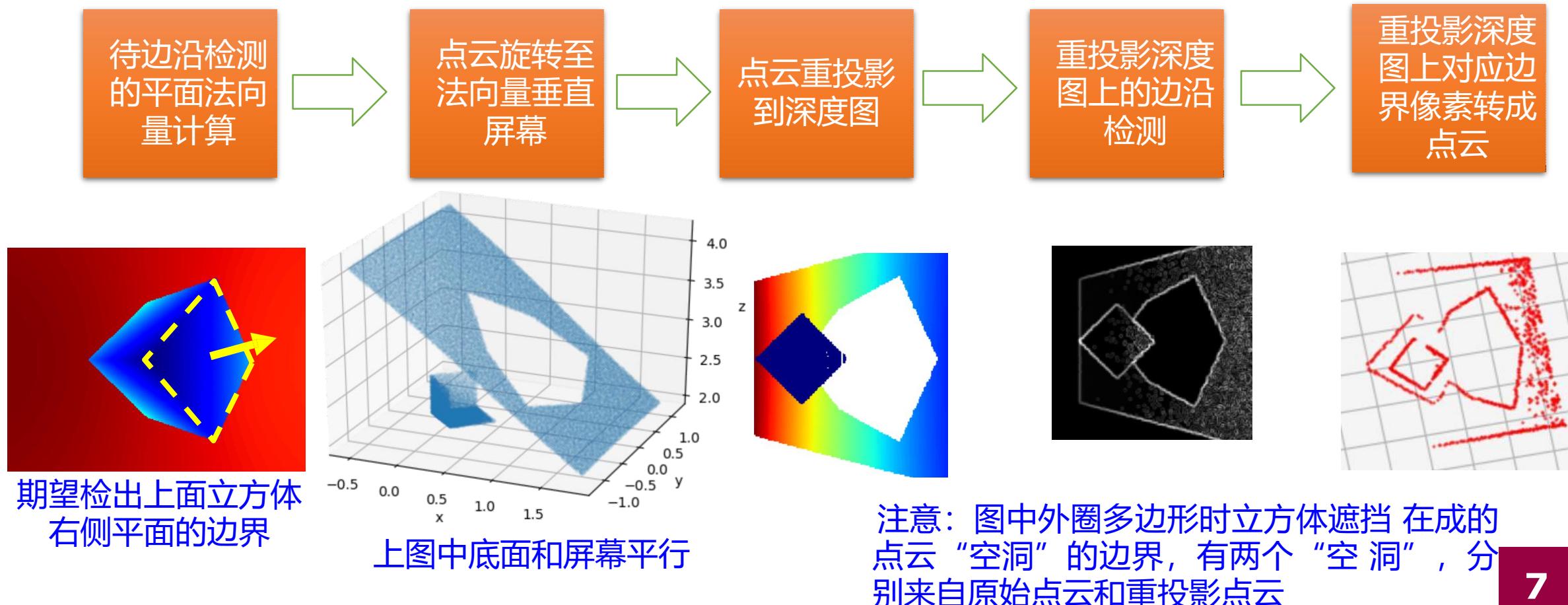


用边沿检测算子检测边沿



# 特征提取与检测-基于深度图的边沿检测

- :( 前后景边界被检出，但立方体棱边如何检测
- : 提高边沿的“深度”对比度，使得棱边容易通过边沿运算检出

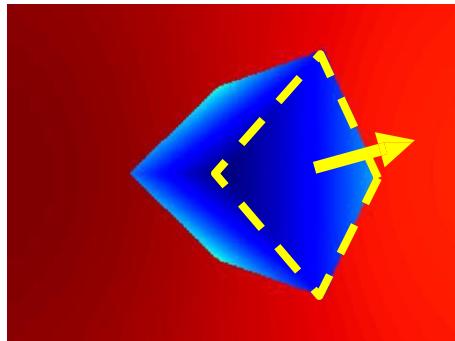


# 特征提取与检测-基于深度图的边沿检测



- :( • 平面法向量旋转——如何找到合适的旋转矩阵使得它指向屏幕?

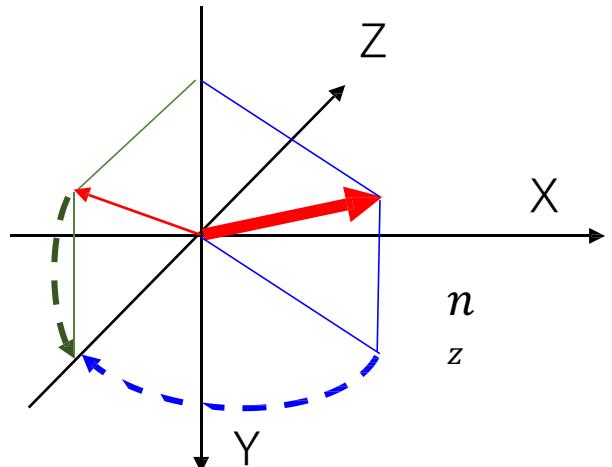
- :) • 根据坐标轴夹角旋转即可



- 先绕Y轴旋转到XOZ平面，旋转角
- 再绕X轴旋转到和Z轴平行，方向相反
- 旋转使用点云坐标和旋转矩阵相乘得到

$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

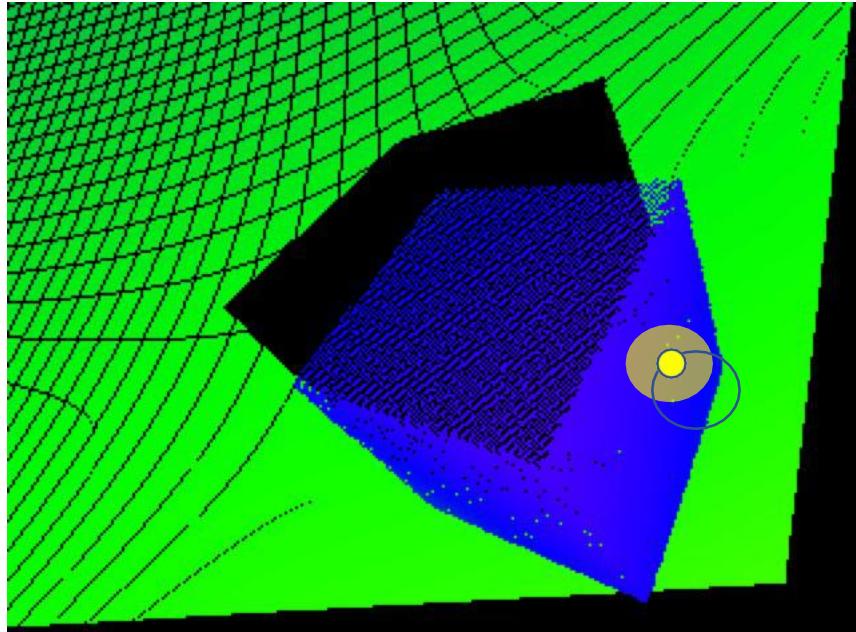
旋转后的点云坐标      旋转矩阵      原点云坐标



生成旋转矩阵的代码（注意，这里针对的点云运算使用行向量表示坐标）

```
13 def get_rot_mat(ax=0,ay=0,az=0):  
14     Rx=np.array([[1, 0, 0],  
15                  [0, np.cos(ax), np.sin(ax)],  
16                  [0,-np.sin(ax), np.cos(ax)]])  
17     Ry=np.array([[np.cos(ay), 0, -np.sin(ay)],  
18                  [0, 1, 0],  
19                  [np.sin(ay), 0, np.cos(ay)]])  
20     Rz=np.array([[np.cos(az), np.sin(az), 0],  
21                  [-np.sin(az), np.cos(az), 0],  
22                  [0, 0, 1]])  
23     return np.dot(np.dot(Rx,Ry),Rz)
```

# 特征提取与检测-基于点云的边沿检测



- 注意：邻域选取需要考虑点云密度来确定半径（确保领域内有足够的点）
- 当噪声增加时，需要加大r，以保证噪声被“平均”滤除

- 之前几何参数拟合用到了PCA分解结果的特征向量
- 特征值可以用用于进行形状分析
- 具体步骤
  - 对每个点，以半径 $r$ 找出其邻域内的点云
  - 对邻域内点云做PCA分析，得到特征值
- 计算特征值的例程（下面程序中的E）

```

pc_w=pc_nn-np.mean(pc_nn, axis=0)
M=np.dot(pc_w.T, pc_w)
E,F=np.linalg.eig(M)      # E: 特征值, F: 特征向量
idx=np.argsort(E)
uz=F[:, idx[0]].ravel()   # 法向量方向 (对应最小特征值)
ux=F[:, idx[1]].ravel()   # 平面方向x (对应次小特征值)
uy=F[:, idx[2]].ravel()   # 平面方向y (对应最大特征值)
    
```

# 特征提取与检测-基于点云的边沿检测

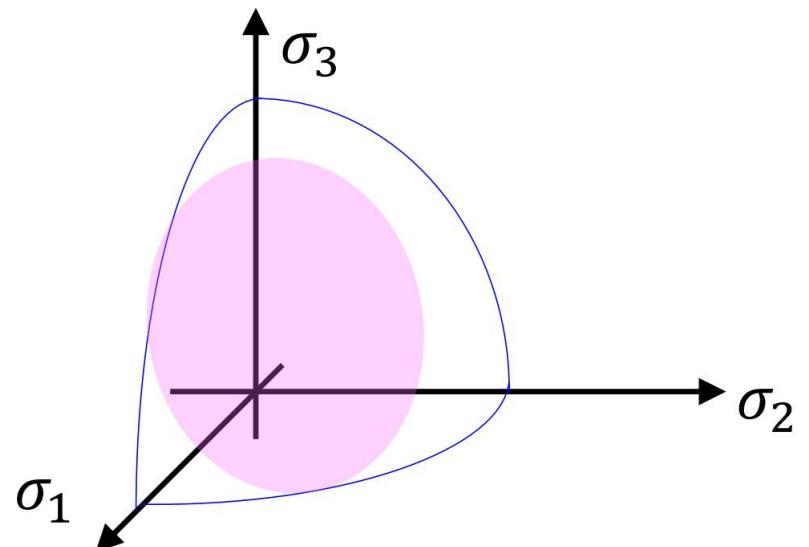
- $\sigma_1, \sigma_2, \sigma_3$ 是计算得到的3个特征值的归一化结果（不是原始特征值），满足：

1. 排序，即： $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$
2. 归一化，即： $\sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2} = 1$

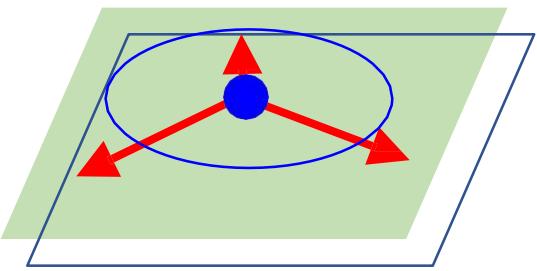


对于不同物体表面的点云，归一化了的局部特征值 $\sigma_1, \sigma_2, \sigma_3$ 能告诉我们什么？

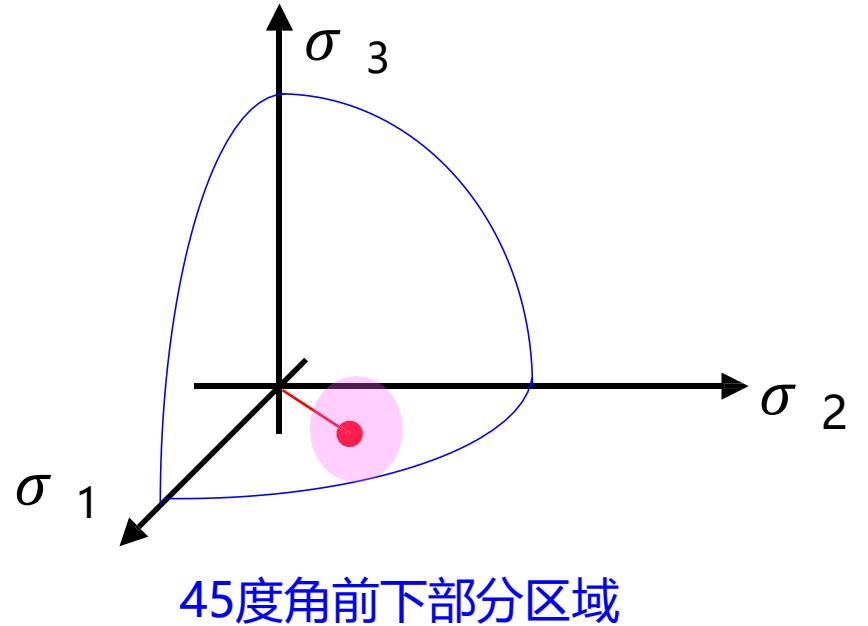
- 以 $\sigma_1, \sigma_2, \sigma_3$ 为坐标轴，
- 将每组 $\sigma_1, \sigma_2, \sigma_3$ 显示在这个坐标系下，
- 他们落在1/8的单位球面上
- 主要落在图中1/8球面的前下方



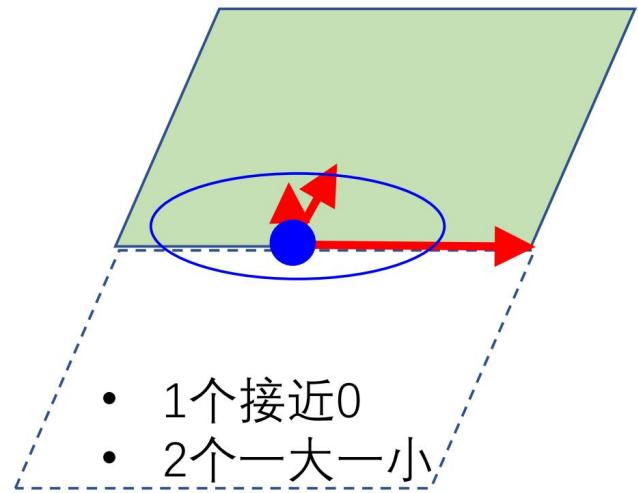
# 特征提取与检测-基于点云的边沿检测



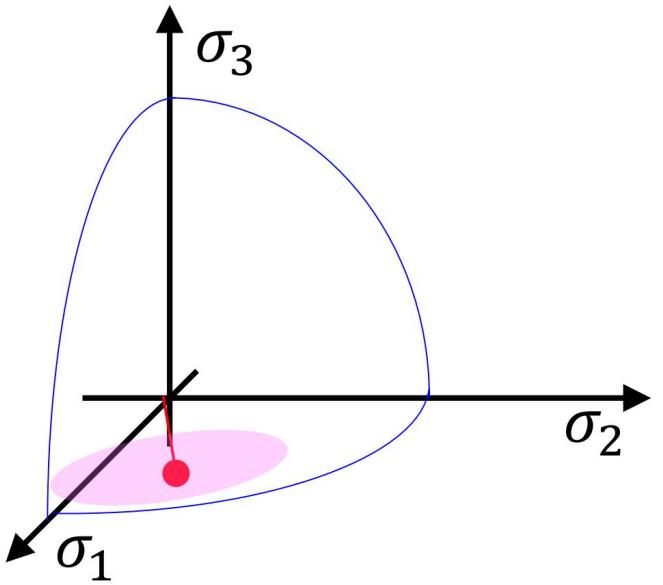
- 1个接近0
- 2个相同值



# 特征提取与检测-基于点云的边沿检测

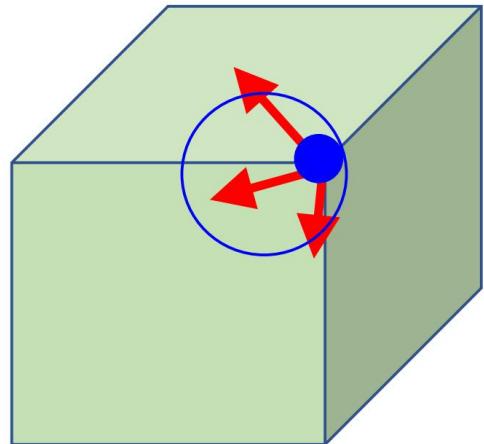


- 1个接近0
- 2个一大一小

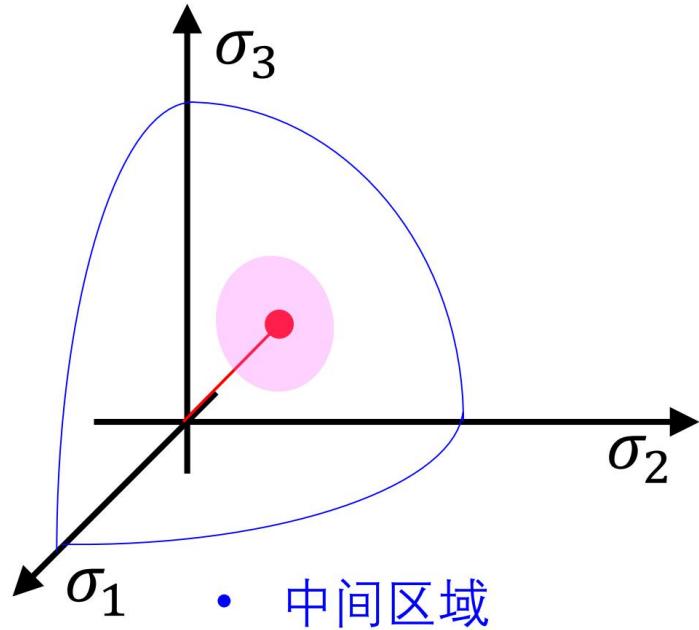


- 下方扁长区域, 更靠近 $\sigma_1$ 轴

# 特征提取与检测-基于点云的边沿检测

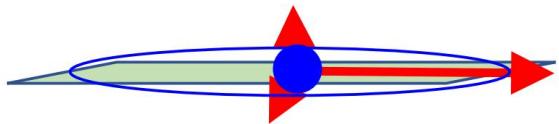


- 3个非零

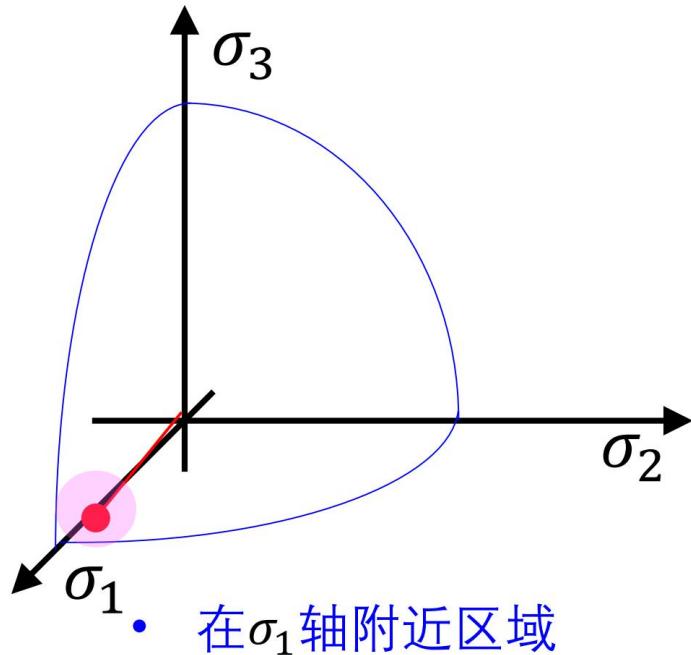


- 中间区域

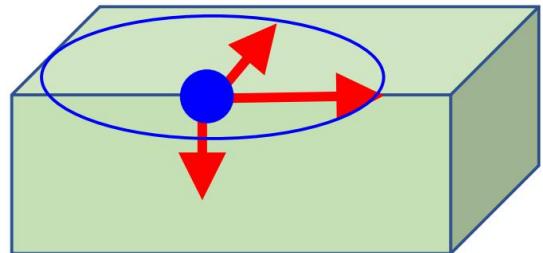
# 特征提取与检测-基于点云的边沿检测



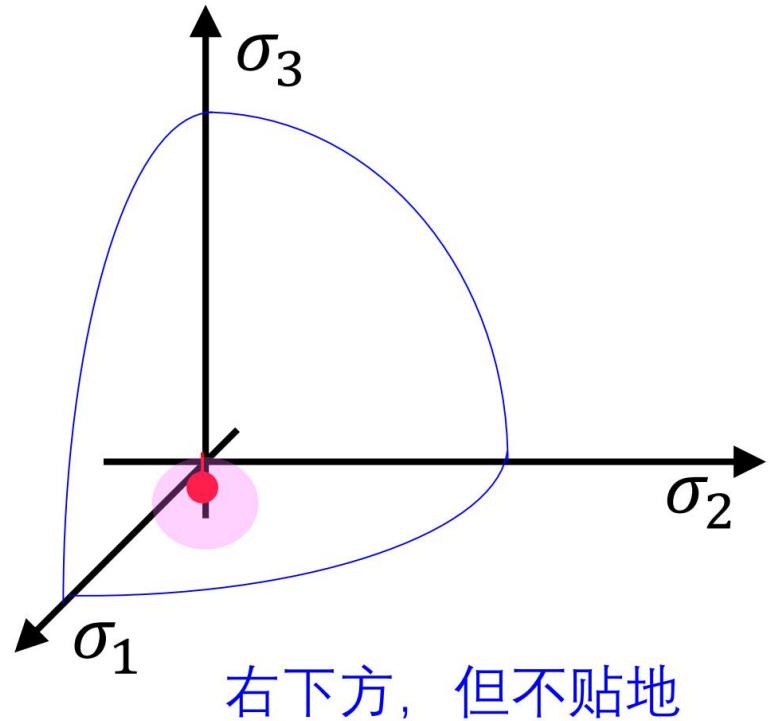
- 2个接近0
- 1个大特征值



# 特征提取与检测-基于点云的边沿检测



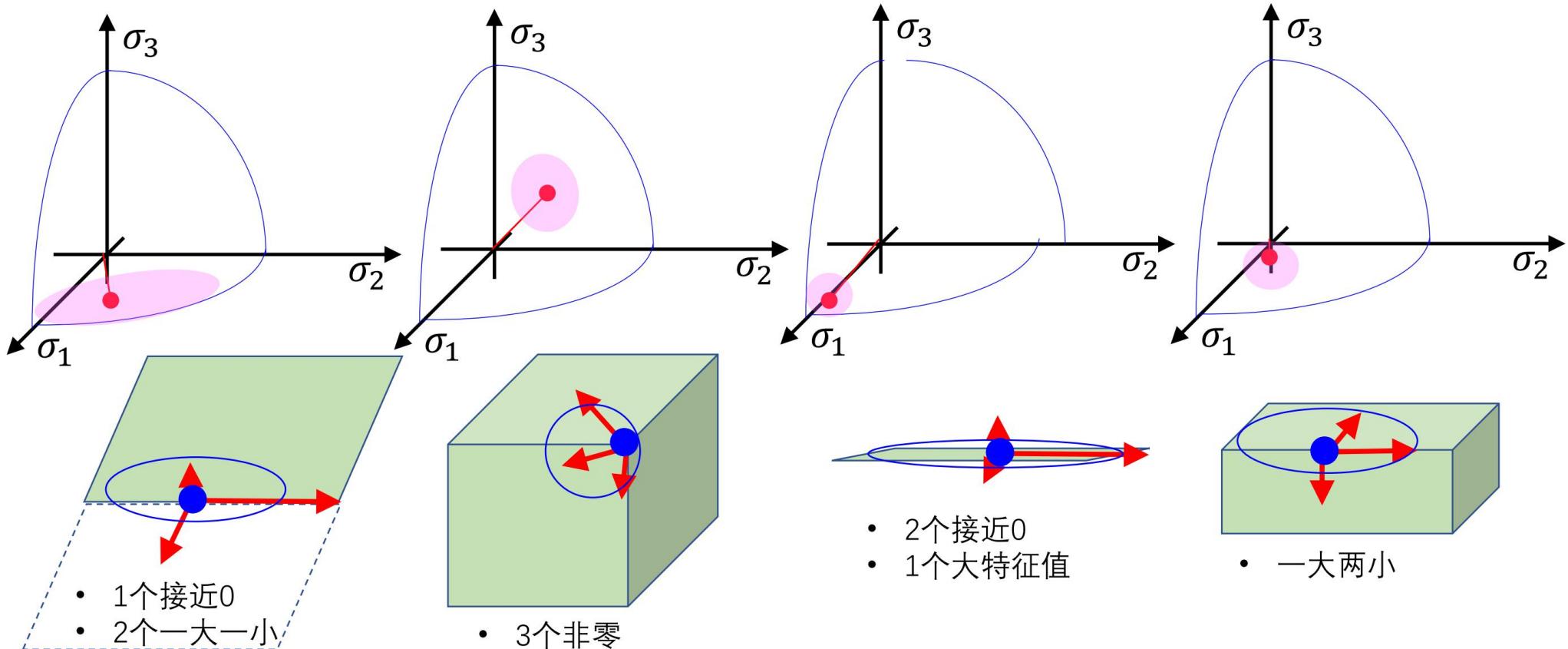
- 一大两小



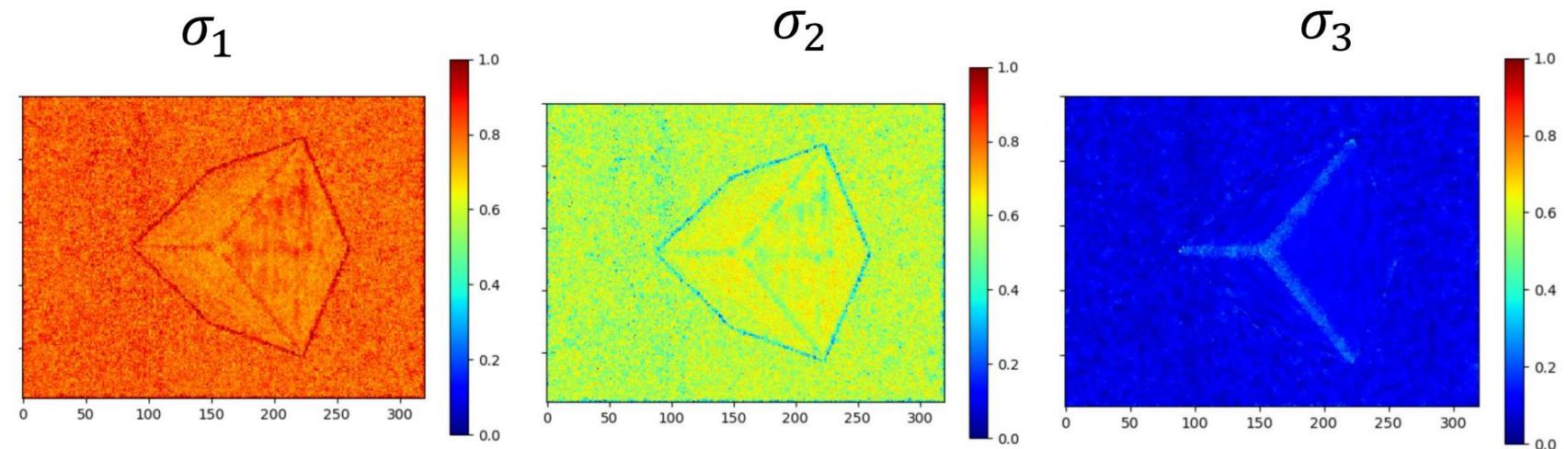
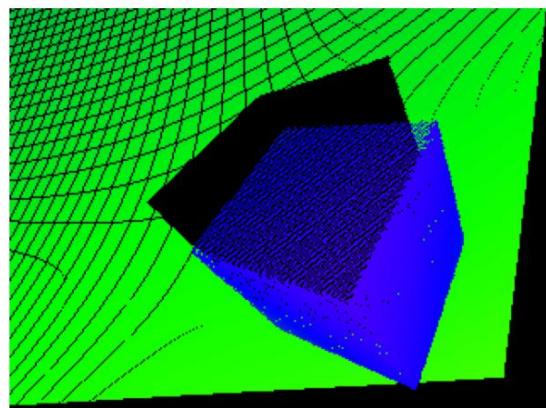
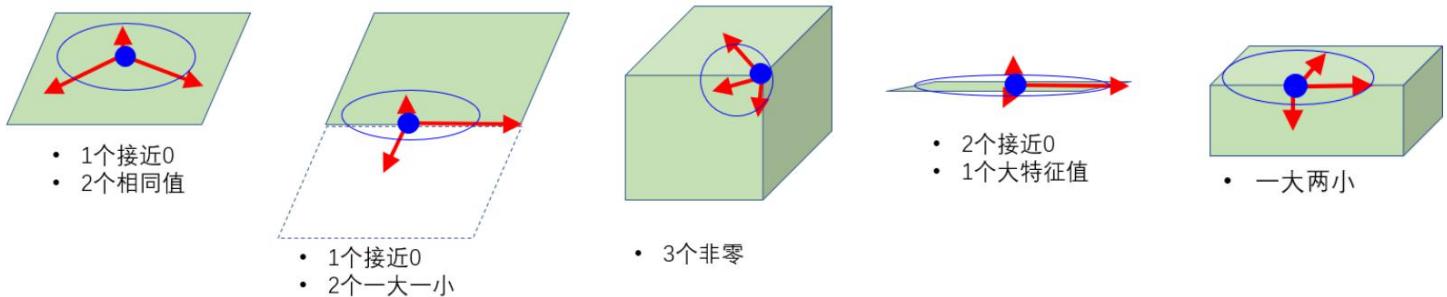
# 特征提取与检测-基于点云的边沿检测



- 设计的算法通过将球面分为若干区域，可以区分这几种情况，

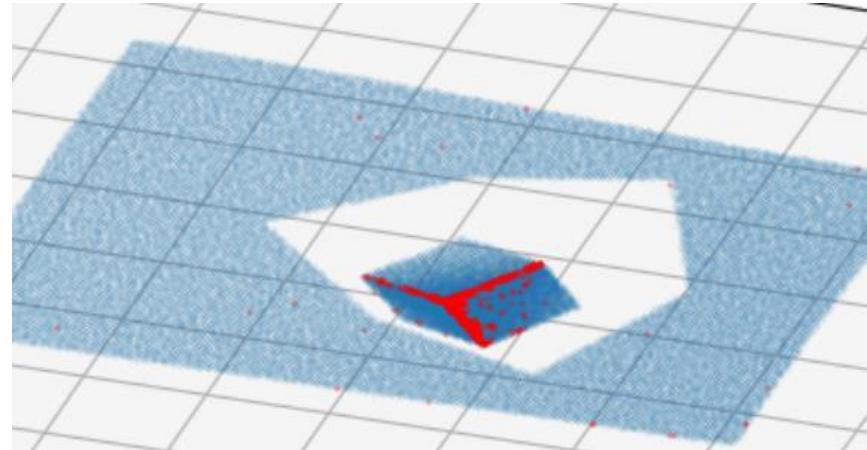
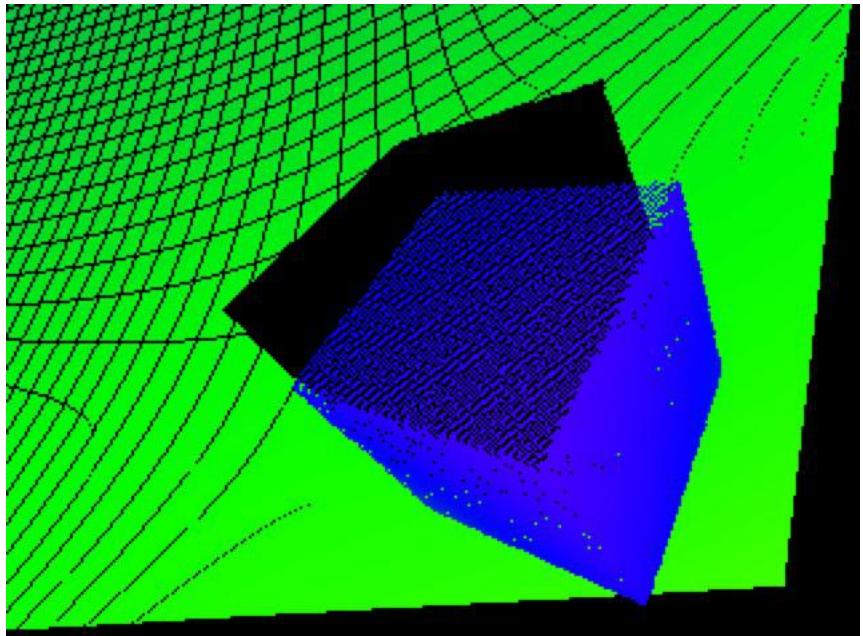


# 特征提取与检测-基于点云的边沿检测



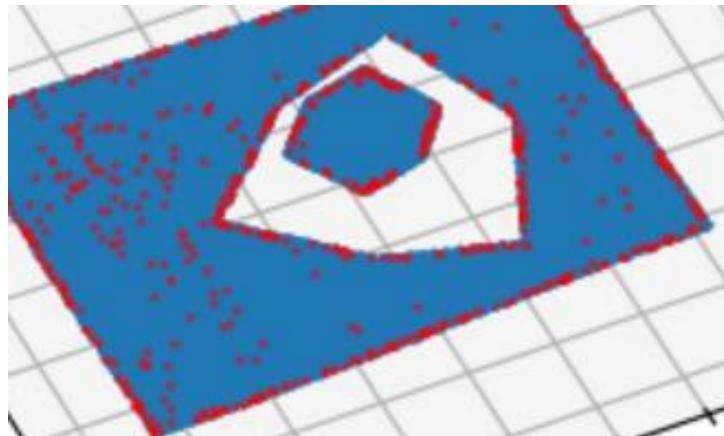
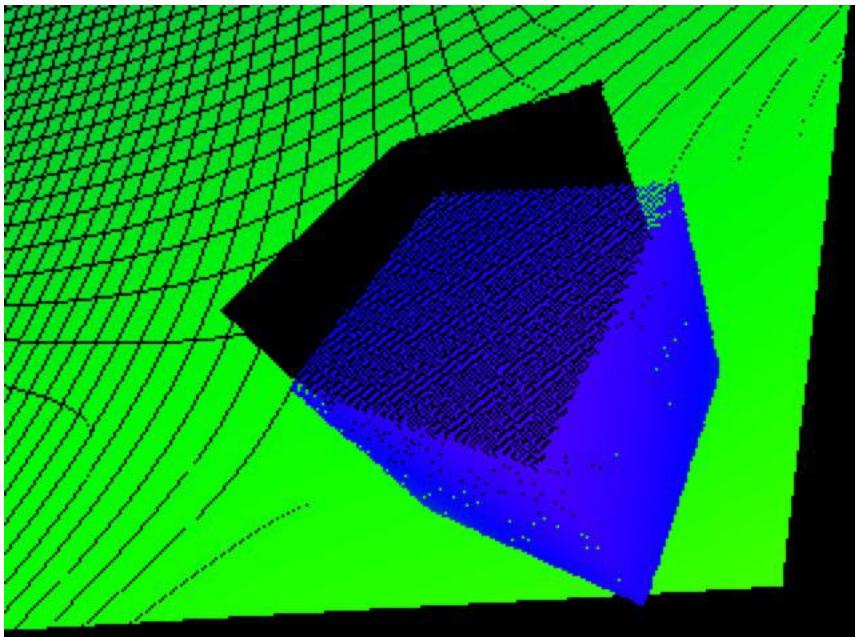
由于噪声，实际情况并不是那么理想

# 特征提取与检测-基于点云的边沿检测



- 领域半径0.03
- $\sigma_3 > 0.22$

# 特征提取与检测-基于点云的边沿检测



- 领域半径0.03
- $\sigma_1 > 0.9$
- $\sigma_2 < 0.3$
- 表面噪声使得识别过程中有孤立的噪声点
- 如果需要进一步识别直线，可以使用Hough算法或者RANSAC直线拟合算法来避免噪声点的影响





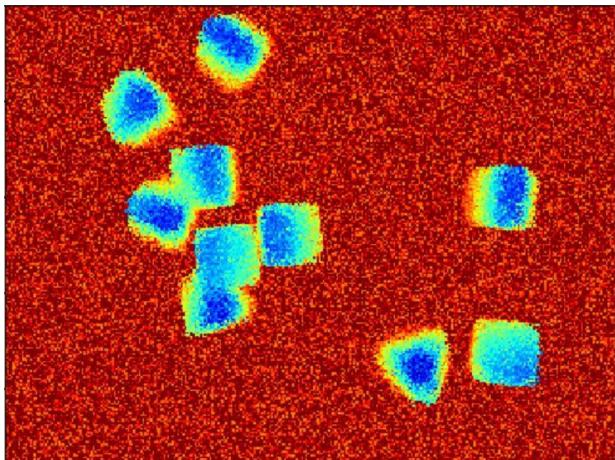
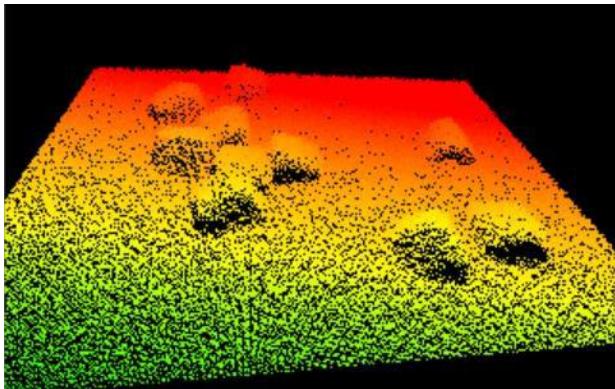
# 目录

## CONTENTS

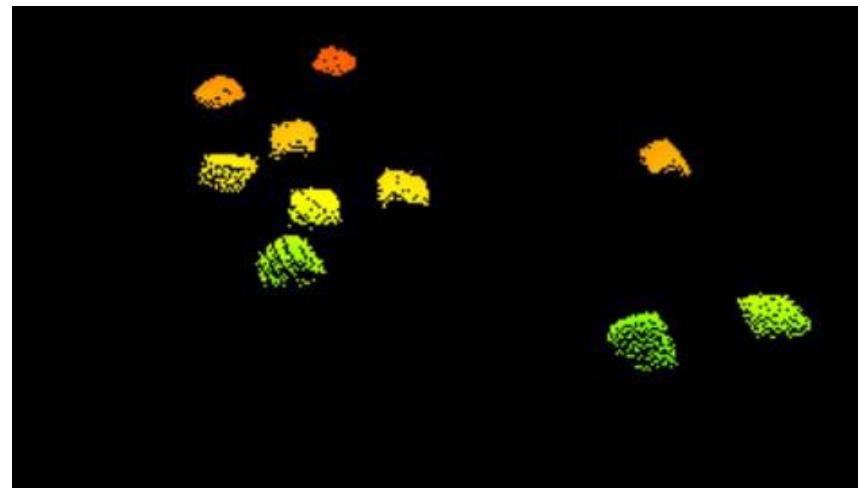
02

前后景分离

# 前后景分离-地面检测



- 目标——提取平面上散布的物体
- 计算地平面模型，扣除地表点云（删除离平面近的点云）
- 缺点：会误检出非平整地面，物体较多时RANSAC可能失效



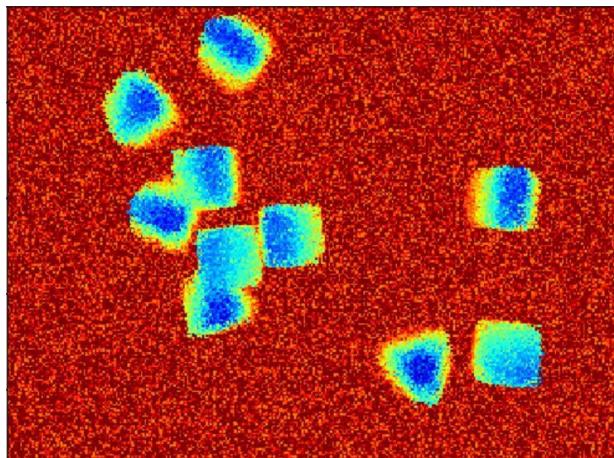
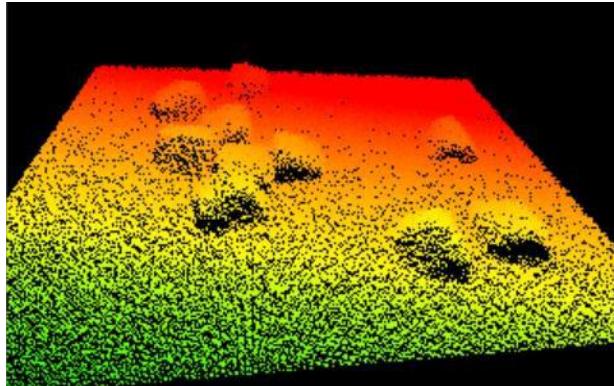
```

n,D=plane_det_pc_ransac(pc,r=0.2,k=0.4,th=0.06,it=200, it1=3, verbose=True)

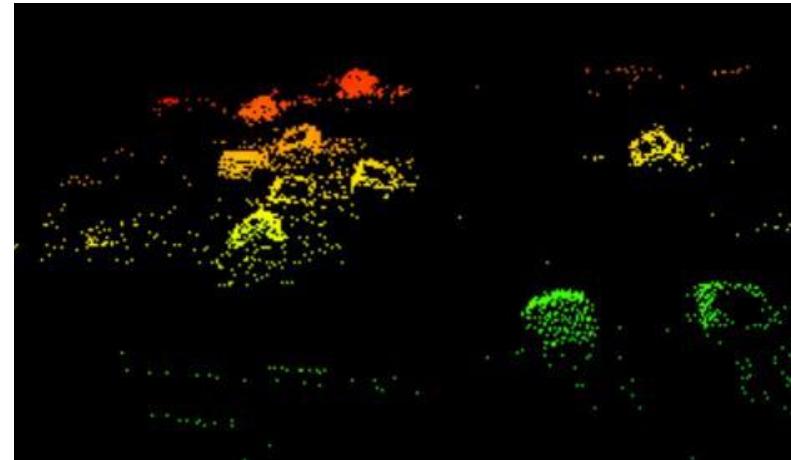
# 选出平面上的点
dist=pc_to_plane_dist(n,D,pc)

# 扣除了平面上的点
TH=0.02
pc_view(pc[dist>TH],CAM_FX,CAM_FY,CAM_CX,CAM_CY,CAM_WID,CAM_HGT,dmin=0.0,dmax=3.0)
    
```

# 前后景分离-PCA地面检测



- 计算较大邻域的点云PCA（邻域尺寸和物体相当），保留特征值大于门限的点云
- 用于检测人、非规则立柱等物体

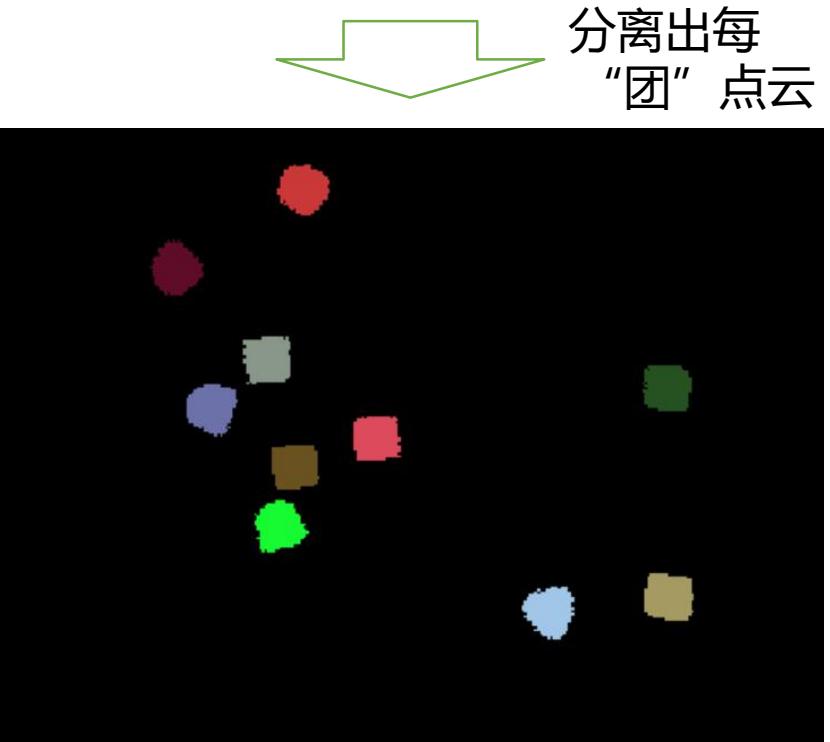
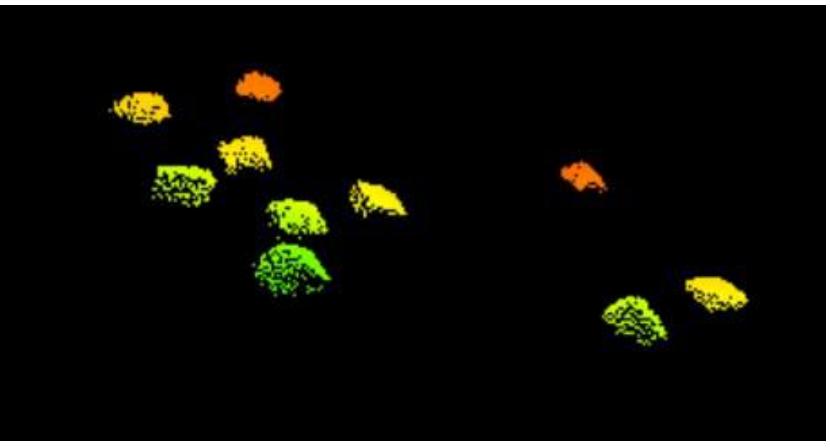


- 缺点：
  - 对于有较大平面的物体表面出现空洞
  - 噪声带来“飞散点”
- 用点云的过滤算法可以清除

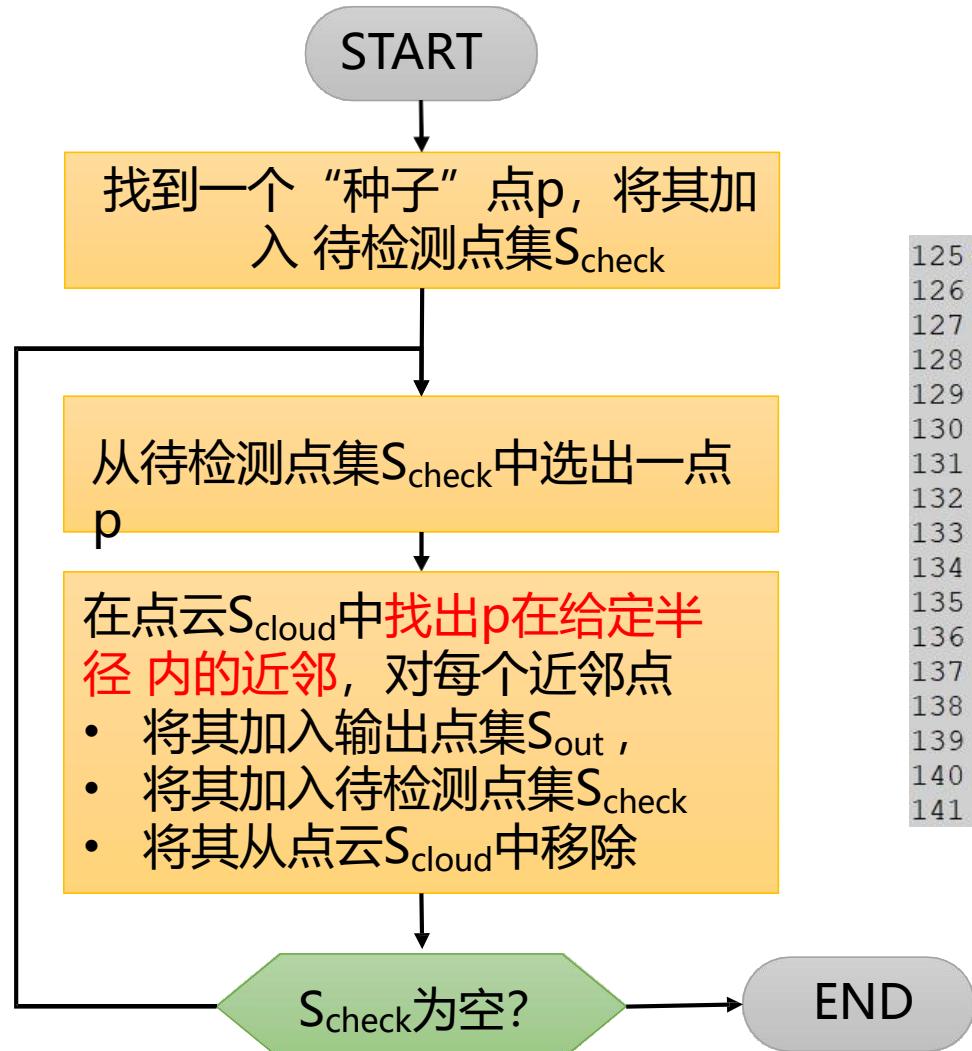


# 前后景分离-区域增长法点云提取

- 目的——将提取的点云按物体划分
- 物体分割是物体测量的前提



# 前景分离-区域增长法点云提取



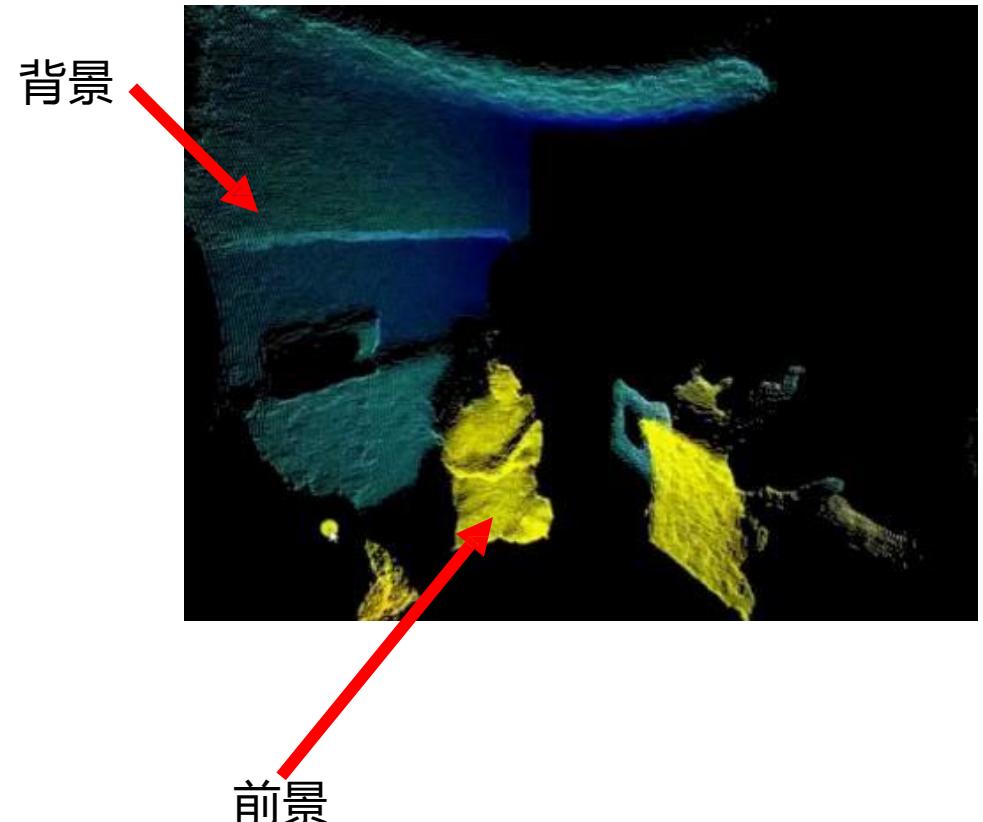
```

125 def pc_obj_merge(pc, p0, k, r, mask=None, ret_idx=False):
126     flann = cv2.FlannBasedMatcher(dict(algorithm=1, trees=5), dict(checks=50))
127
128     # 指示尚未分割的点云点(这里用copy()是为了防止修改输入的mask对象)
129     mask=np.ones(len(pc), dtype=bool) if mask is None else mask.copy()
130
131     pc_chk=[p0]即: Scheck
132     idx_out=[] 对应Sout
133     while len(pc_chk)>0:
134         p=pc_chk.pop()
135         # 找到半径r领域内的未检查过的点的序号
136         idx_nn=[m.trainIdx for m in flann.knnMatch(p,pc,k=k)[0] \
137                 if m.distance<r and mask[m.trainIdx]]
138         idx_out+=idx_nn
139         pc_chk+=[pc[i] for i in idx_nn]
140         mask[idx_nn]=False
141
142     return (pc[idx_out],idx_out) if ret_idx else pc[idx_out]
  
```

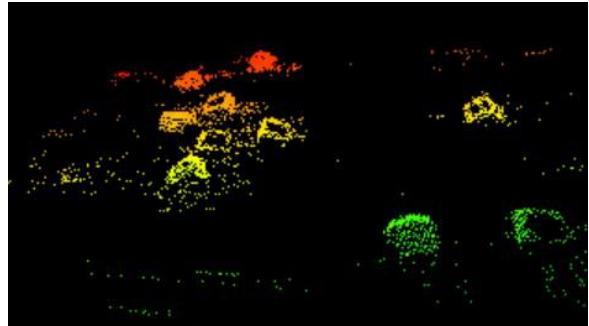
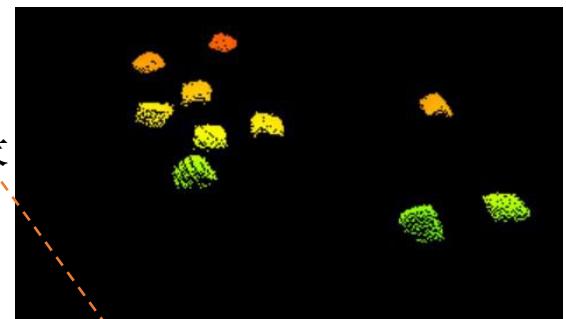
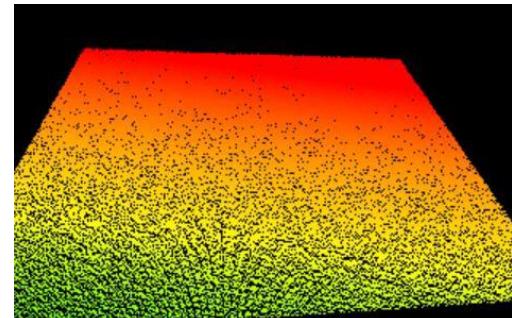
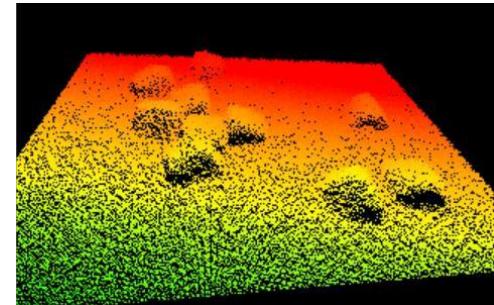
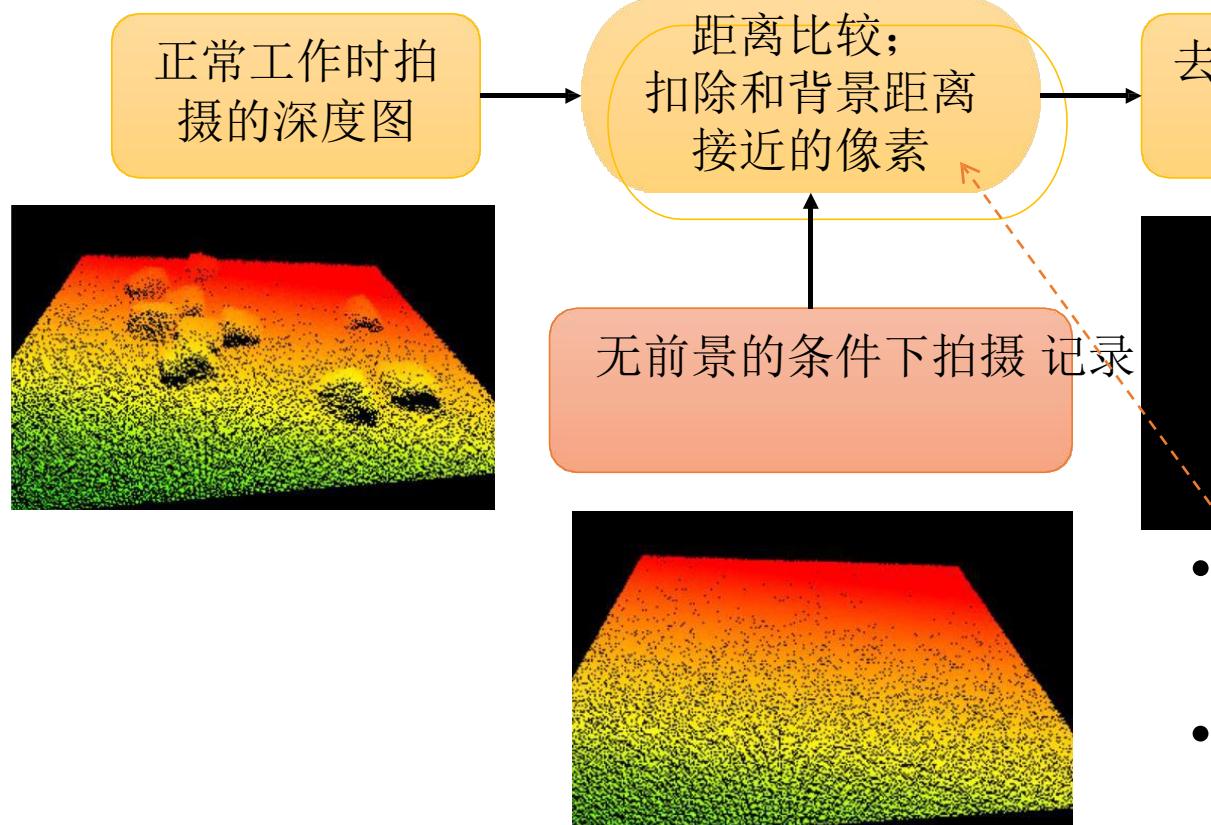
通过mask指示哪些点从S<sub>cloud</sub>中移除了  
 新增的点云序号加入输出集合  
 新增点云加入待检测集合  
 标注已被处理的点 从S<sub>cloud</sub>中移除

# 深度图的前后景分离

- 前面讨论的时前后景分离的一个例子，下面进一步介绍方法
- 目的——将关注对象从背景干扰中提取
- 有几种方法
  - 固定背景扣除的分离方法
  - 统计建模前背景分离方法
  - 自适应背景建模和获取



# 深度图的前景分离-背景扣除前背景分离方法

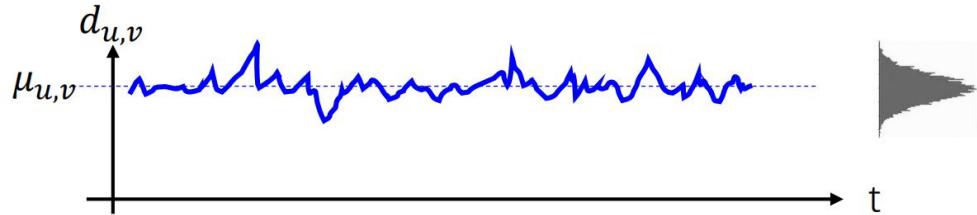


- 和之前的方法不同之处在于，这里**不需要平面参数拟合**，只需要采集并记录背景图
- 问题
  - 这个距离门限到底为多少？
  - 噪声使得背景去除不彻底
  - 相机或者背景改变，被错认为前景



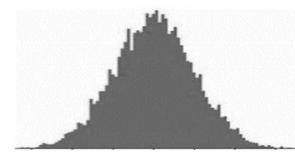
# 深度图的前背景分离-统计建模前背景分离方法

- 观察不同时间拍摄的特定位置 $(u, v)$ 的背景像素点深度值 $d_{u,v}$
- 他们带有随机波动
- 均值 $\mu_{u,v} = \frac{1}{T} \sum_{t=1}^T d_{u,v}(t)$ 是背景物体到相机的距离
- 画出测量结果的直方图（概率估计），得到类似高斯的形状



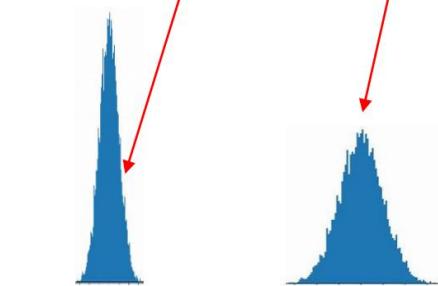
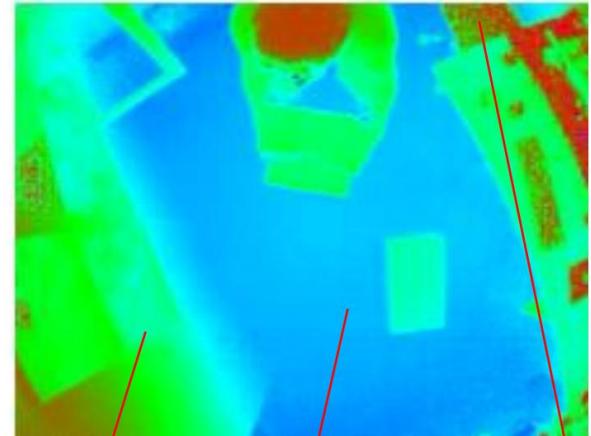
高斯型状的描述：

- 均值： $\mu_{u,v} = \frac{1}{T} \sum_{t=1}^T d_{u,v}(t)$
- 方差： $\sigma_{u,v}^2 = \frac{1}{T} \sum_{t=1}^T (d_{u,v}(t) - \mu_{u,v})^2$



- 背景不同地方有不同的均值的方差

顶视相机，拍摄在屋子里摆放物件

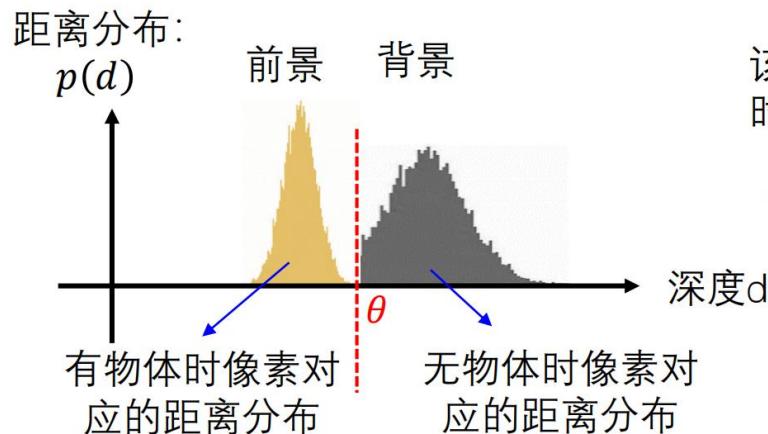


近距离，测量  
噪声方差小

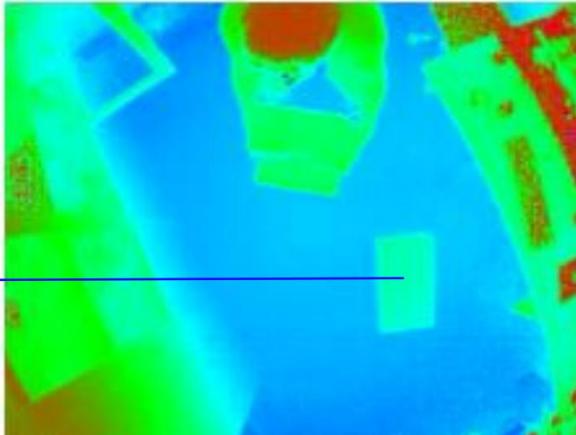
远距离，测量  
噪声方差大

# 深度图的前背景分离-统计建模前背景分离方法

观察有无前景时，距离分布的改变



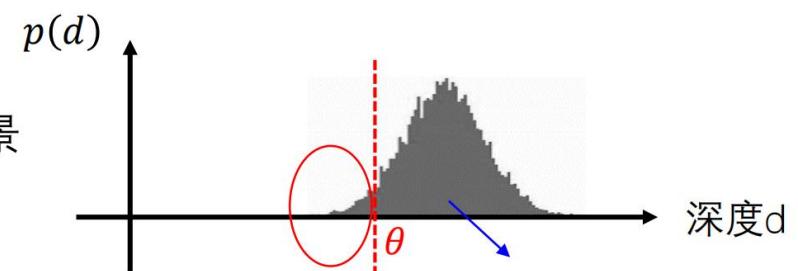
该位置像素有无物体时距离测量值的概率分布的改变



- 通过设定门限  $\theta$  来确定当前像素是否是前景
- 测量值小于门限  $\theta$  表示前景物体出现，大于  $\theta$  表示目前只有背景
- 背景被误认为前景的错误率——右图红色圈中的面积占比

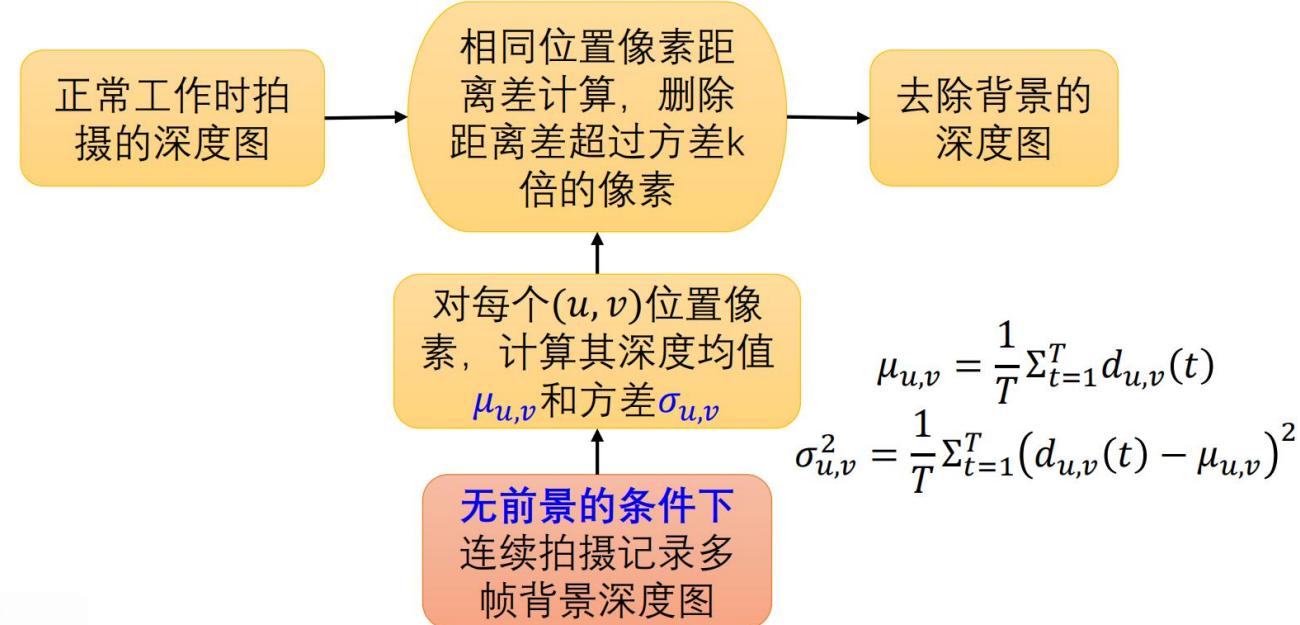
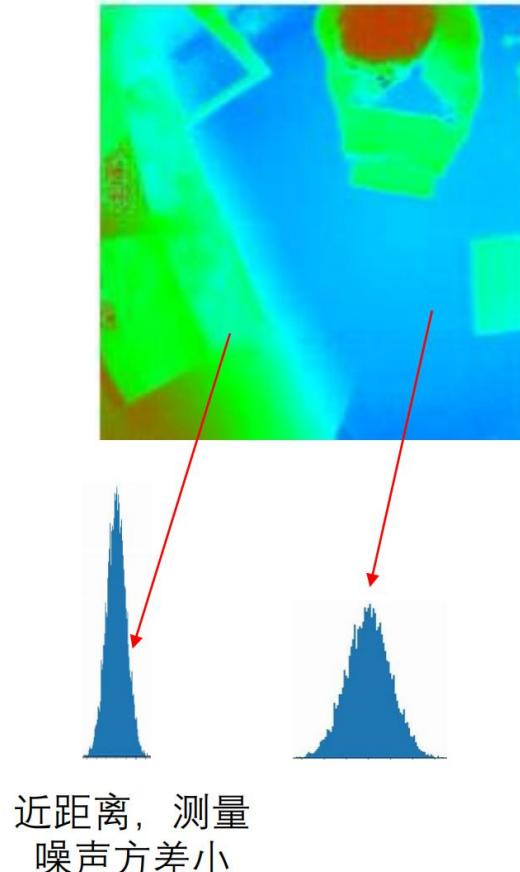
如何选择门限  $\theta$

- 数值分布在  $(\mu - \sigma, \mu + \sigma)$  中的概率为 0.653
- 数值分布在  $(\mu - 2\sigma, \mu + 2\sigma)$  中的概率为 0.954
- 数值分布在  $(\mu - 3\sigma, \mu + 3\sigma)$  中的概率为 0.997



高斯分布的特性

# 深度图的前背景分离-统计建模前背景分离方法

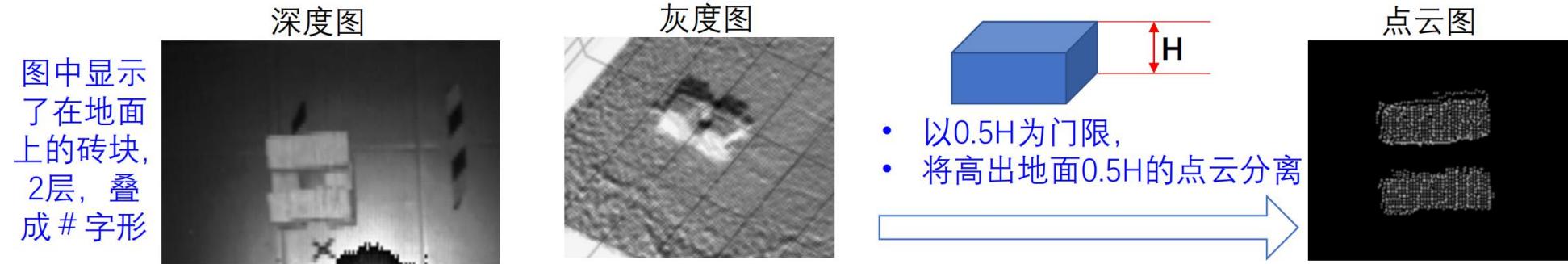


## 问题

- :( ) 基于背景的门限设定给出了误识别率, 但还是会错将背景噪声当成前进的概率
- : ) 结合额外的孤立点检测消除误识别和漏识别

# 深度图的前背景分离-统计建模前背景分离方法

## 应用例子——检测叠放在地面的砖块位置

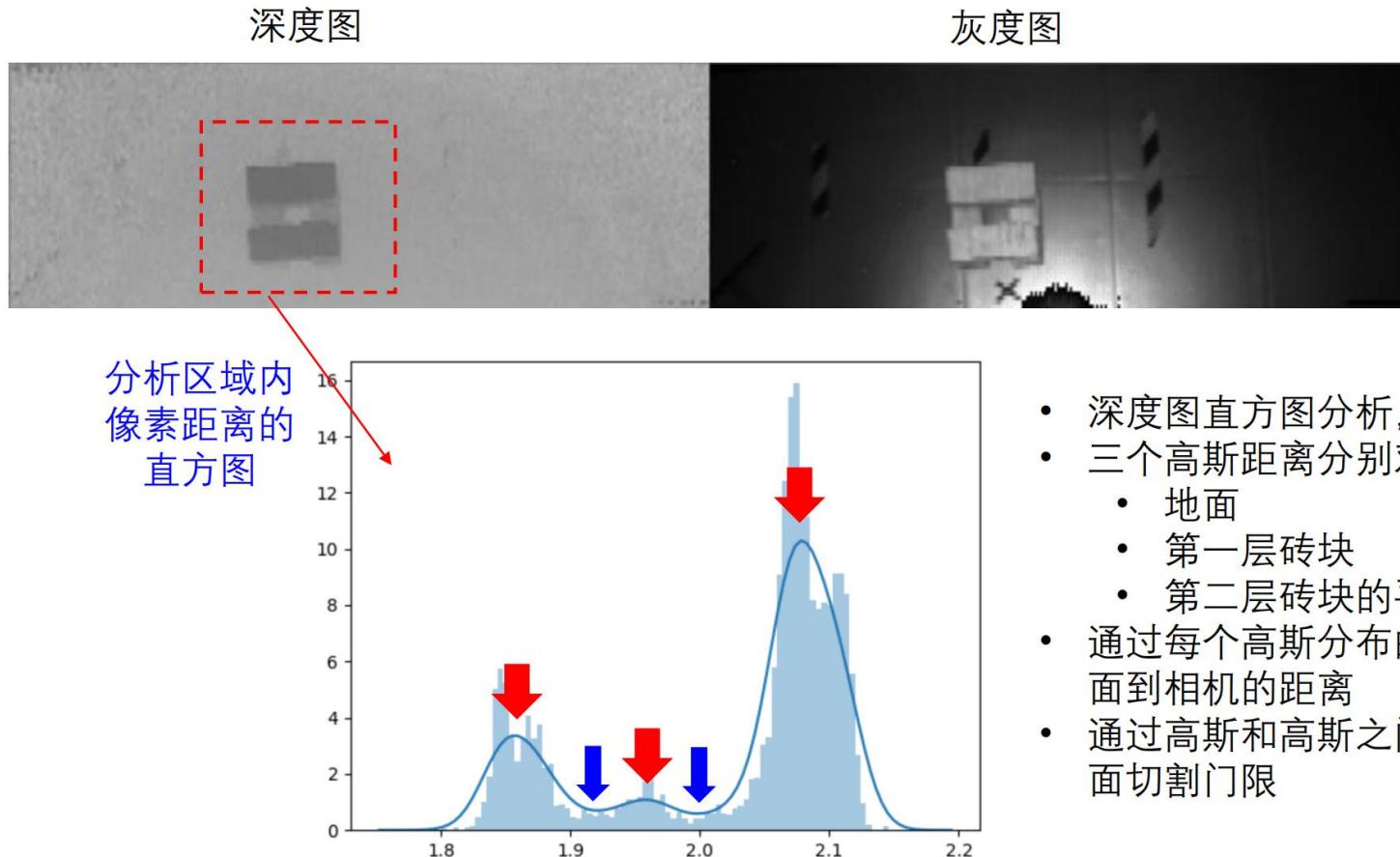


### 问题

- :(上述门限是否合理?
- :(不同材质物体测量值波动是否一致
  - 比如地面测量结果波动很大
- :上面摆放的前景物体测量波动很小
- :此时，为降低地面误判为前景物体，期望提高分割门限，比如设为 $0.75H$
- :(到底设多少合适?)

# 深度图的前背景分离-统计建模前背景分离方法

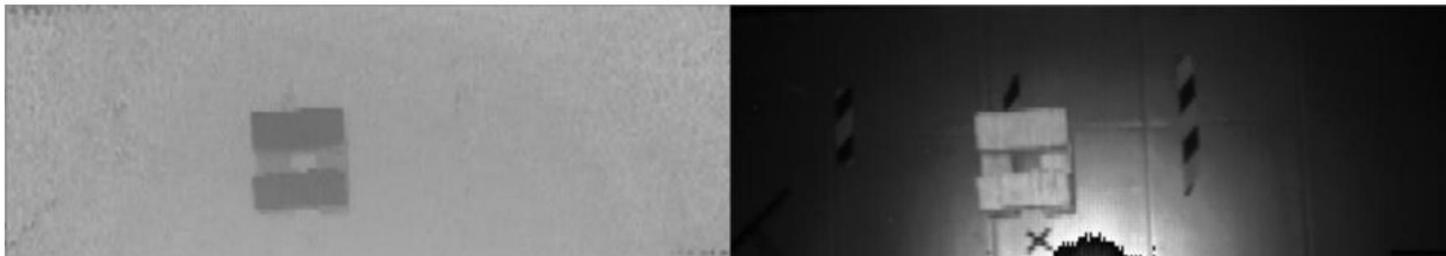
使用空间区域内的直方图分析来找到分割门限



# 深度图的前背景分离-统计建模前背景分离方法

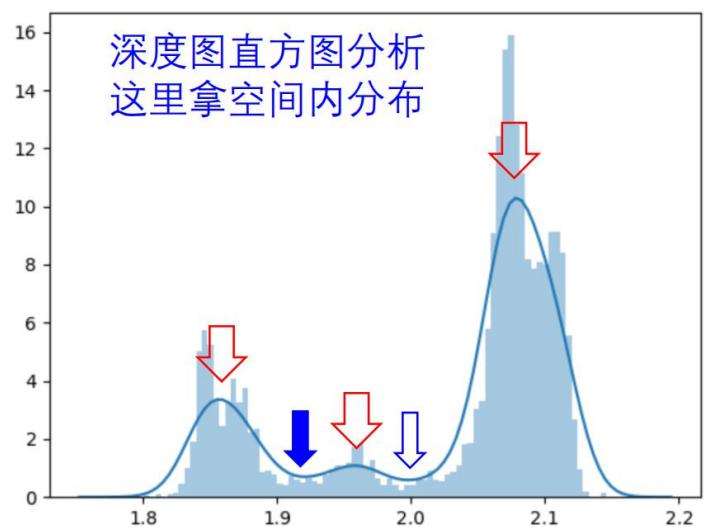
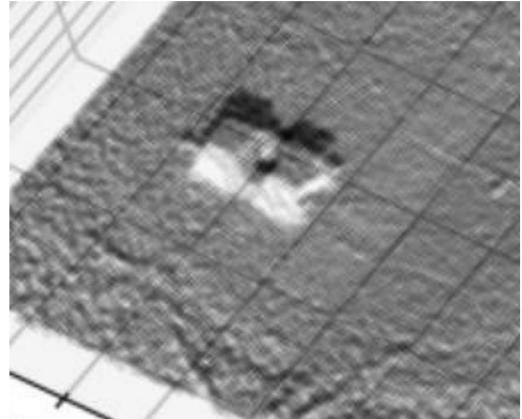
检测地面叠放的砖块距离相机距离和位置

深度图



灰度图

点云图



通过第一个“谷底”为门限  
分离出来的砖块顶端

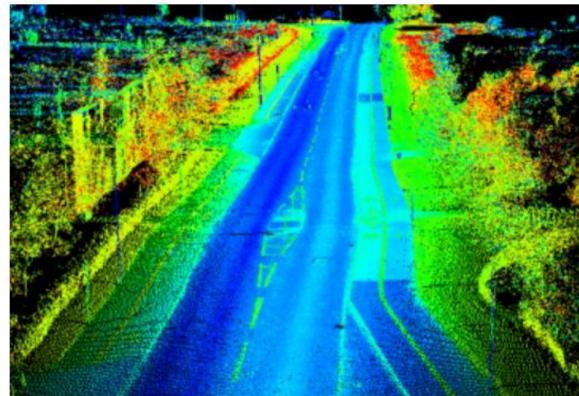


# 背景的自适应更新

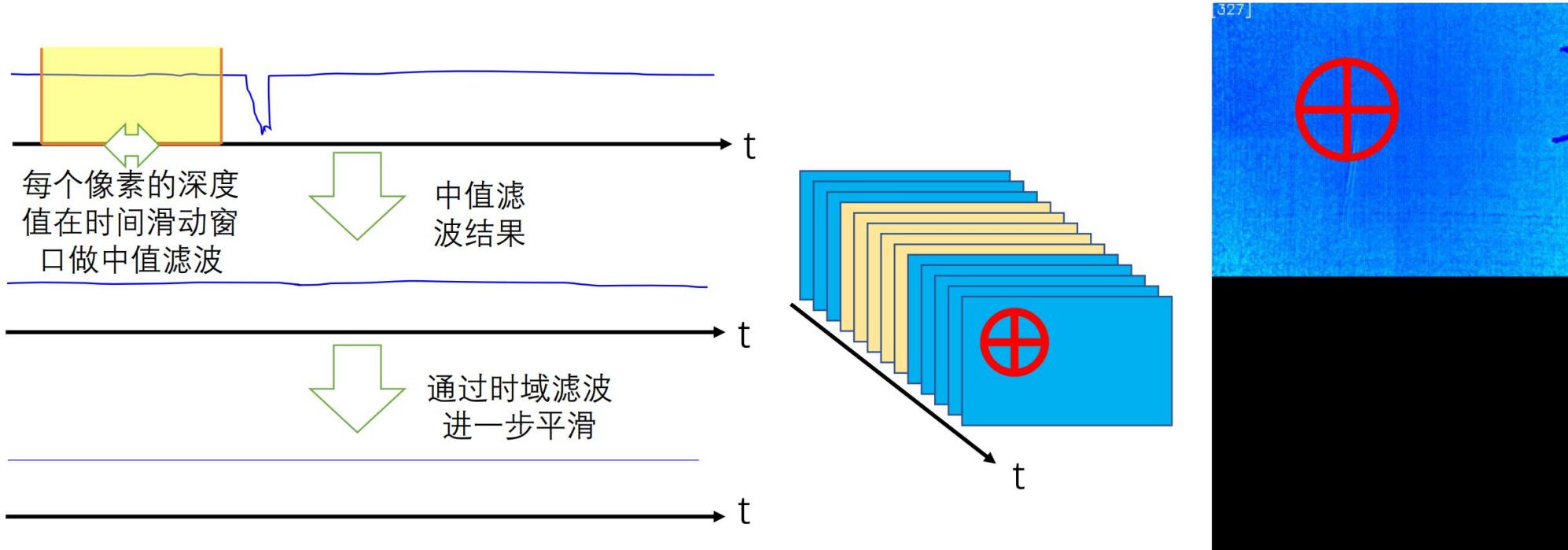
问题描述——无法直接采集“干净”的背景



- 背景不完全固定
  - 缓慢变化——比如由于设备温度等因素，导致深度图整体的偏移
  - 局部突变——比如场景内大型物体的搬移
  - 无法找到前景不存在的时刻点——比如公路场景



# 背景的自适应更新-时域中值滤波



- :(  
• 假设条件——每个像素连续出现前景的时间不超过时间滑动窗口的1/2
- :(  
• 需要很多图像帧吗？
- :)  
• 可以使用降采样后的数据帧，比如每20帧图像抽取1帧用于时域中值滤波，滤波滑动窗口是60帧，对于20fps的图像，滤波窗口对应了1分钟的数据



# 目录

## CONTENTS

03

课堂练习

# 课堂练习



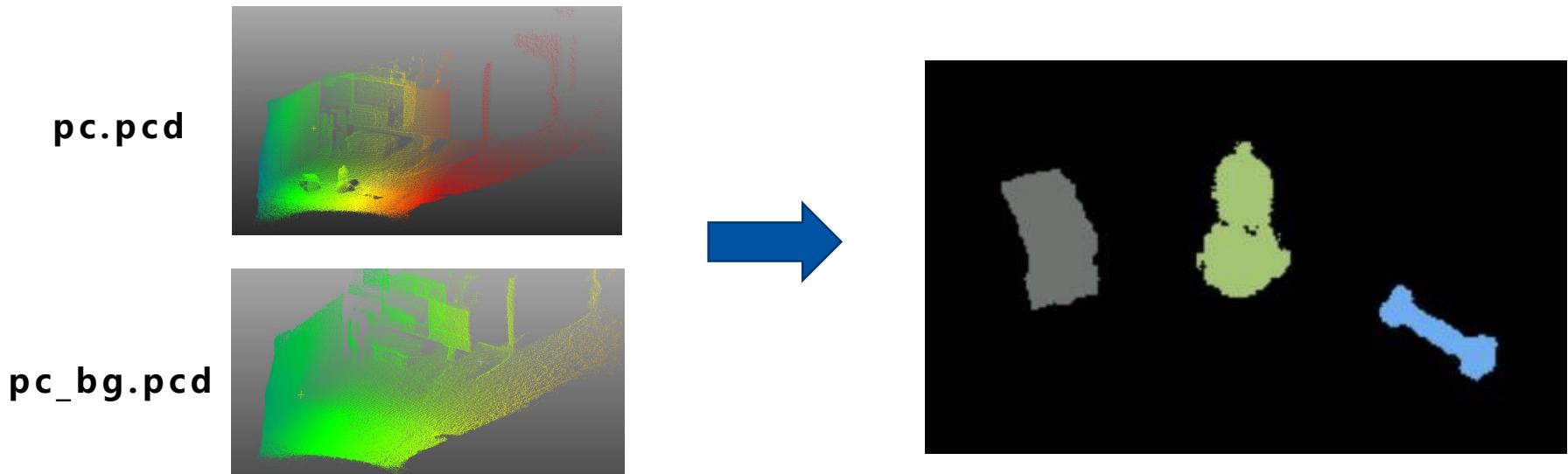
## 练习：从点云中分离特定物体

<https://pcl.readthedocs.io/projects/tutorials/en/latest/#octree>

[https://pcl.readthedocs.io/projects/tutorials/en/latest/region\\_growing\\_segmentation.html#region-growing-segmentation](https://pcl.readthedocs.io/projects/tutorials/en/latest/region_growing_segmentation.html#region-growing-segmentation)

要求：

1. 从给定点云中检测并分离出3个前景物体；
2. 需要你对比pc.pcd和pc\_bg.pcd这两个点云，并从pc.pcd中提取3个地面上的物体；
3. 将提取的3个物体的点云进行分割并分别保存；





谢谢