

3D感知技术与实践

特征提取、前后景分割、物体检测和识别

内容概要

- 特征提取与检测
- 前后景分离
- 几何形状识别

内容概要

- 特征提取与检测
 - 基于深度图的方法
 - 基于点云的方法
- 前后景分离
- 几何形状识别

边沿检测

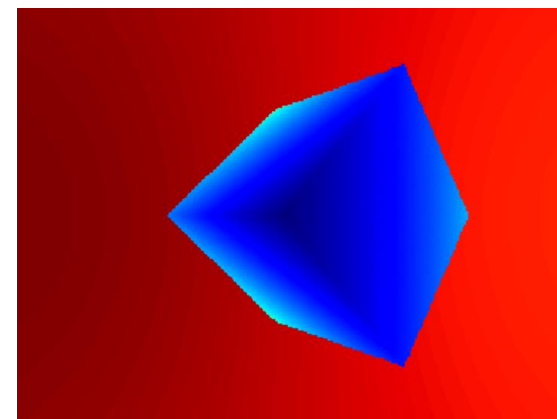
- 应用举例——边沿检测是测量和识别的关键
 - 螺丝丢失识别、焊缝位置检测、安装错位检测
 - 机器人移动过程中道路边线识别，阶梯检测
- 两个解决思路
 - 基于深度图的边沿检测
 - 直接接口3D传感器的数据，实时处理，效率高
 - 多种算法依赖近邻计算，深度图搜索快
 - 检测结果受视角影响（因为深度图看成是2.5D视图，有遮挡效应）
 - 基于点云的边沿检测
 - 应用于稀疏或者多传感器/多视角合成的点云
 - 需要解决点云无序性和密度不均匀
 - 需要解决近邻查询效率问题



边沿检测——基于深度图的边沿检测

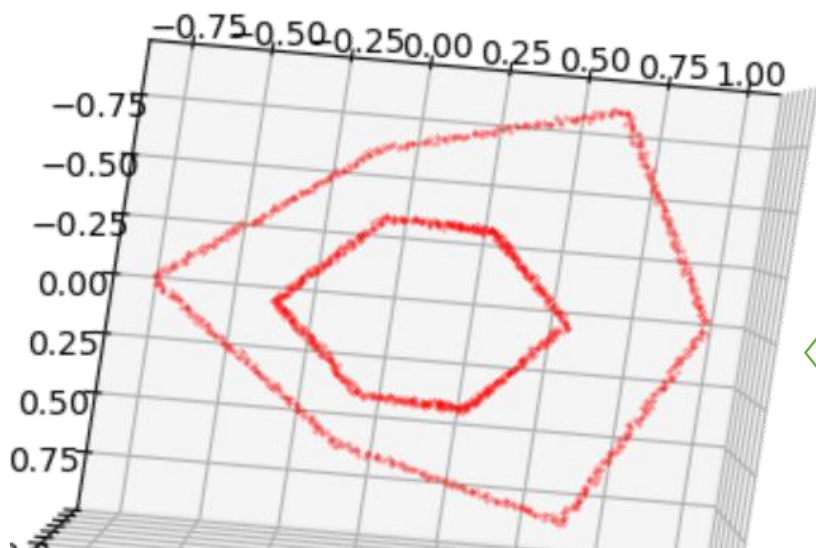
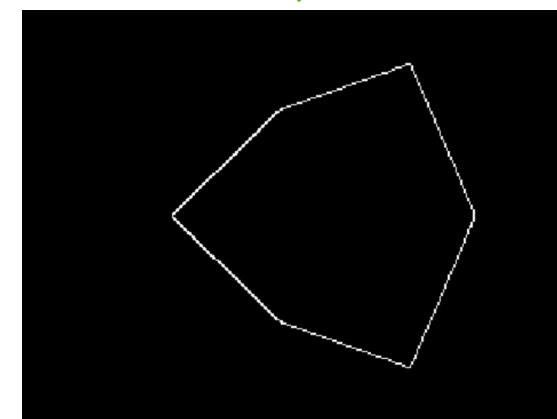
- 把深度图当成强度图直接处理
- 边沿检测的方法
 - Canny/Sobel算子
 - 局部方差计算（在之前滤除飞散点噪声时用过）

例子——求得深度图中立方体的边界



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

用边沿检测算子检测边沿



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

注意：图中外圈多边形时立方体遮挡在成的点云“空洞”的边界

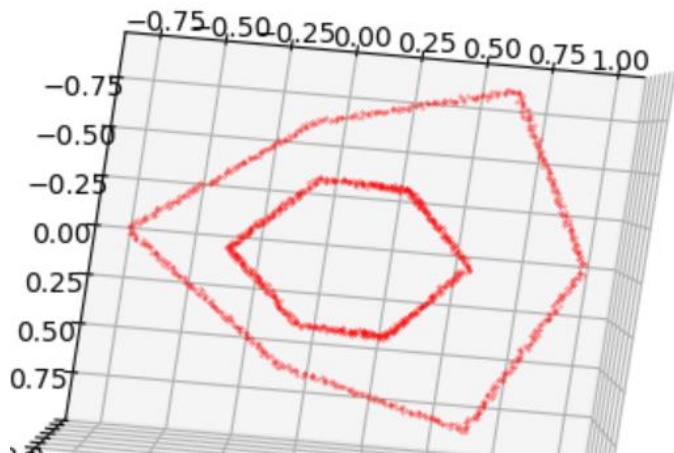
边沿检测——基于深度图的边沿检测

代码实现

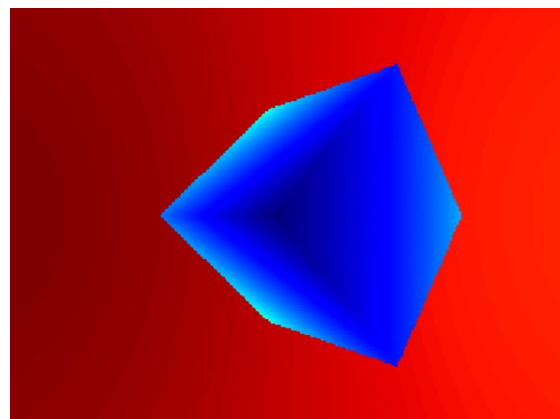
```

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
    
```

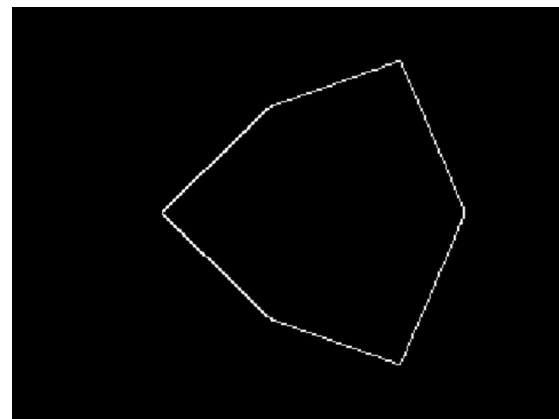
注意：上面代码返回边沿“强度”，需要将他 和门限比较，来确定属于边界的点



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

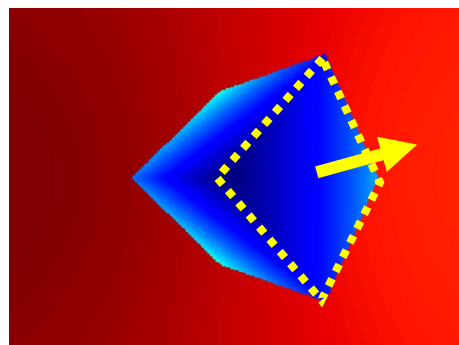


用边沿检测算
子检测边沿

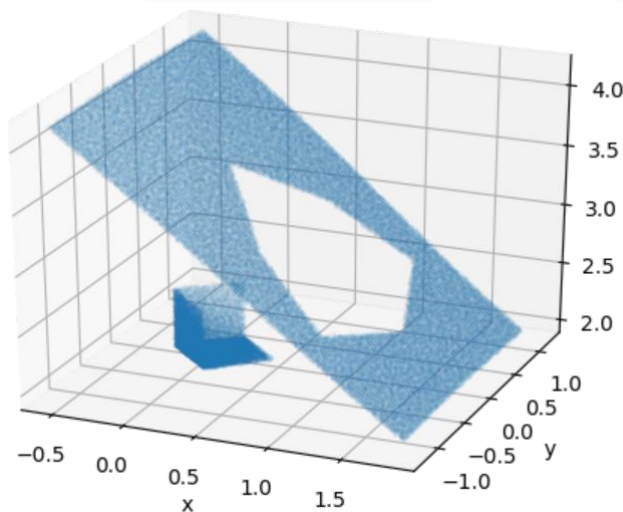


边沿检测——基于深度图的边沿检测

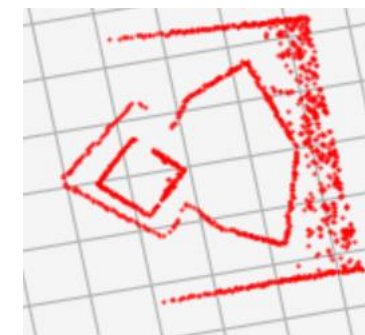
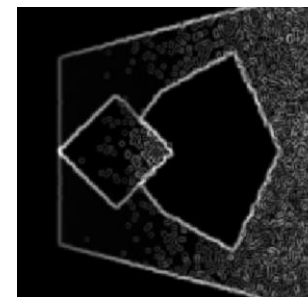
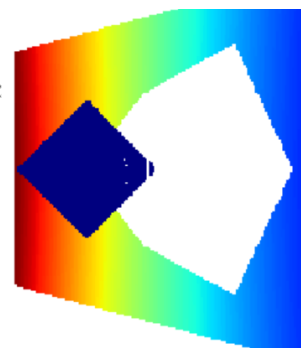
- ☹️ • 前后景边界被检出，但立方体棱边如何检测
- 😊 • 可以通过平面旋转提高边沿的“深度”对比度，使得棱边容易通过边沿运算检出



期望检出上面立方体右侧平面的边界



上图中底面和屏幕平行

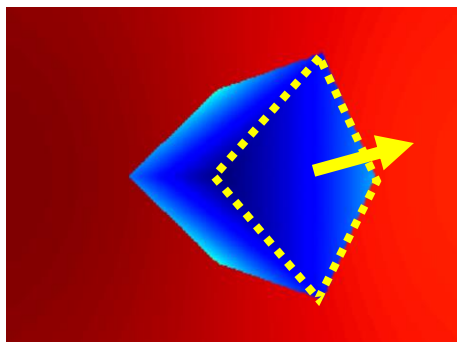


注意：图中外圈多边形时立方体遮挡在成的点云“空洞”的边界，有两个“空洞”，分别来自原始点云和重投影点云

边沿检测——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？

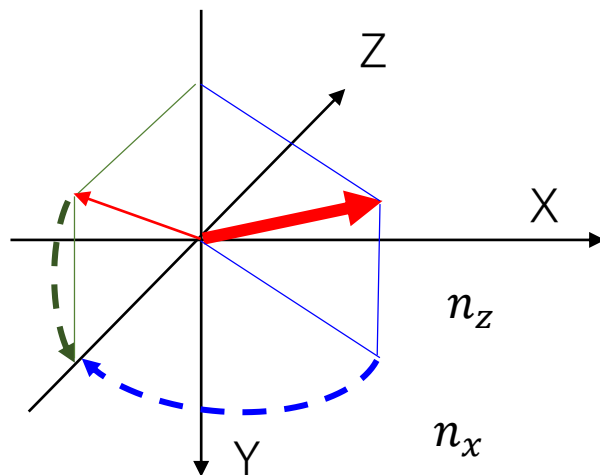
😊 • 根据坐标轴夹角旋转即可



- 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



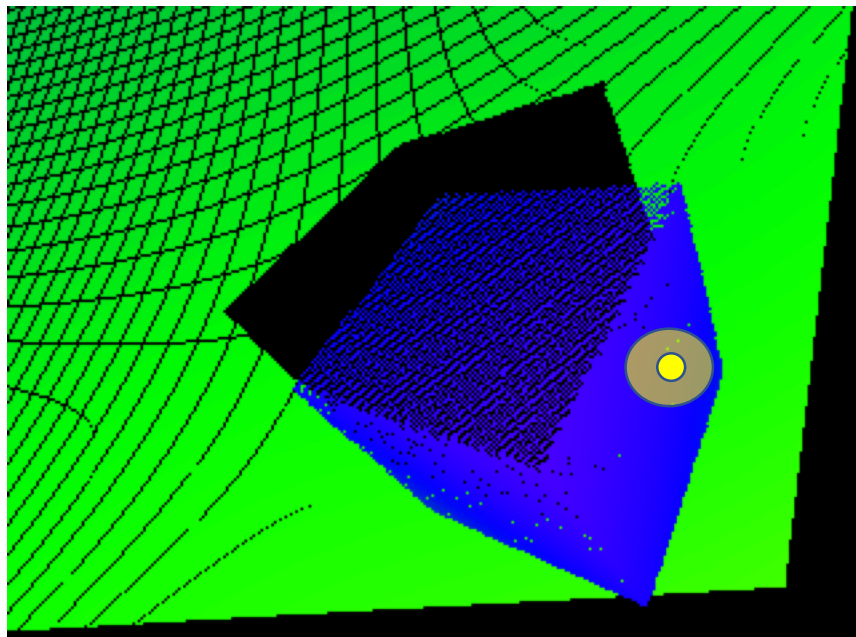
生成旋转矩阵的代码（注意，这里针对的点云运算使用行向量表示坐标）

```
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)
```


内容概要

- 特征提取与检测
 - 基于深度图的方法
 - 基于点云的方法
- 前后景分离
- 几何形状识别

边沿检测——计算特征值得到点云局部特征



- 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)
```



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

边沿检测——点云局部特征

• $\sigma_1, \sigma_2, \sigma_3$ 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：

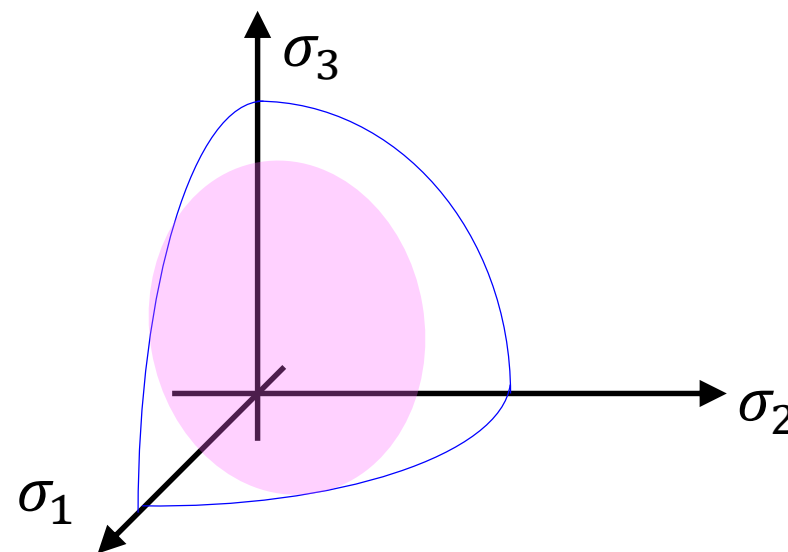
1. 排序，即： $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$

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

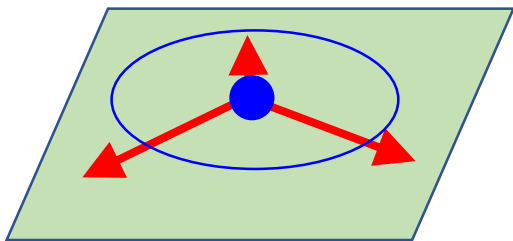


对于不同物体表面的点云，归一化了的局部特征值 $\sigma_1, \sigma_2, \sigma_3$ 能告诉我们什么？

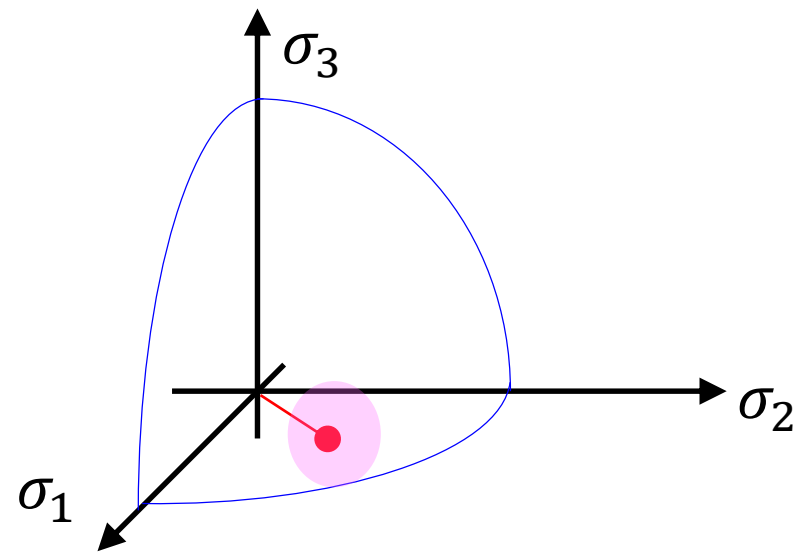
- 以 $\sigma_1, \sigma_2, \sigma_3$ 为坐标轴，
- 将每组 $\sigma_1, \sigma_2, \sigma_3$ 显示在这个坐标系下，
- 他们落在1/8的单位球面上
- 主要落在图中1/8球面的前下方



边沿检测——点云局部特征

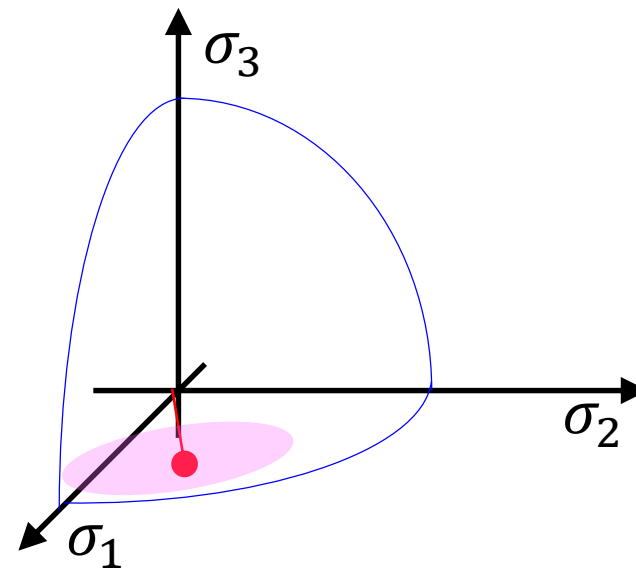
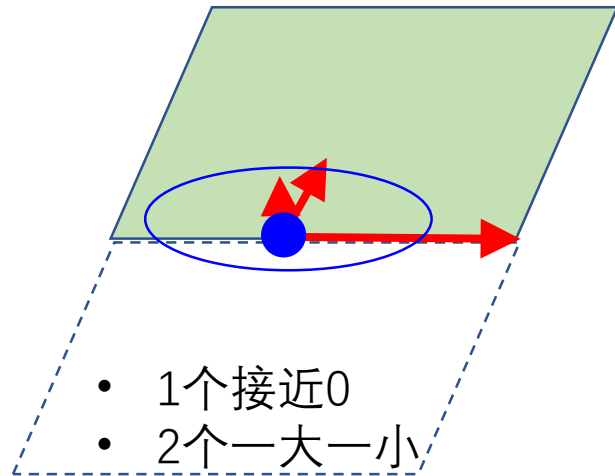


- 1个接近0
- 2个相同值



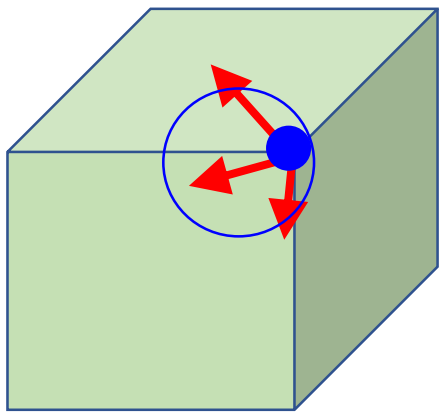
45度角前下部分区域

边沿检测——点云局部特征

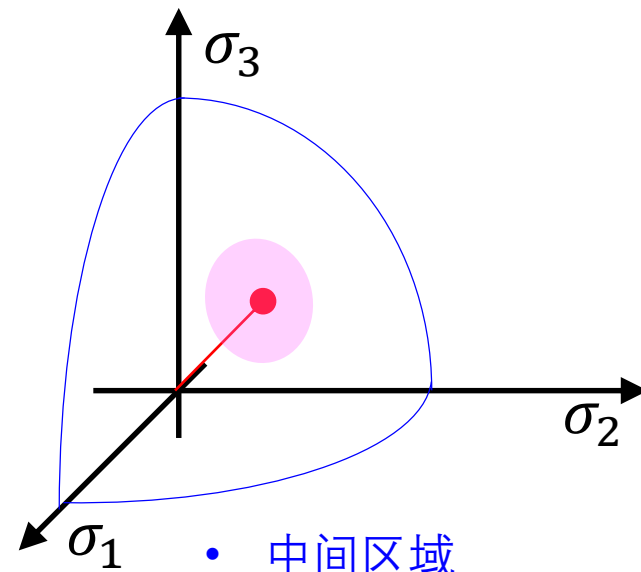


- 下方扁长区域，更靠近 σ_1 轴

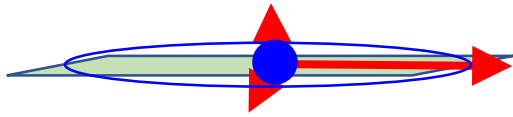
边沿检测——点云局部特征



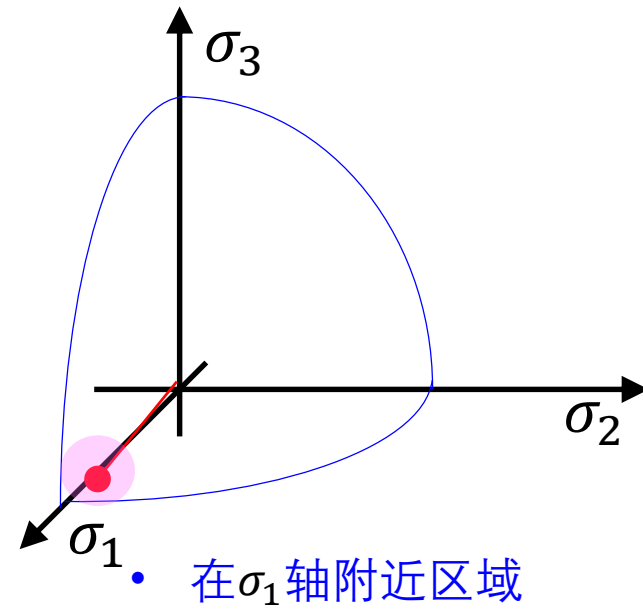
- 3个非零



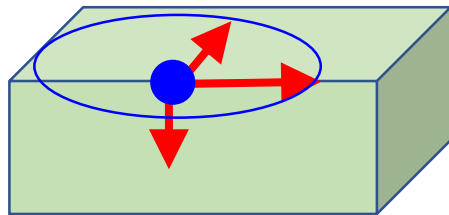
边沿检测——点云局部特征



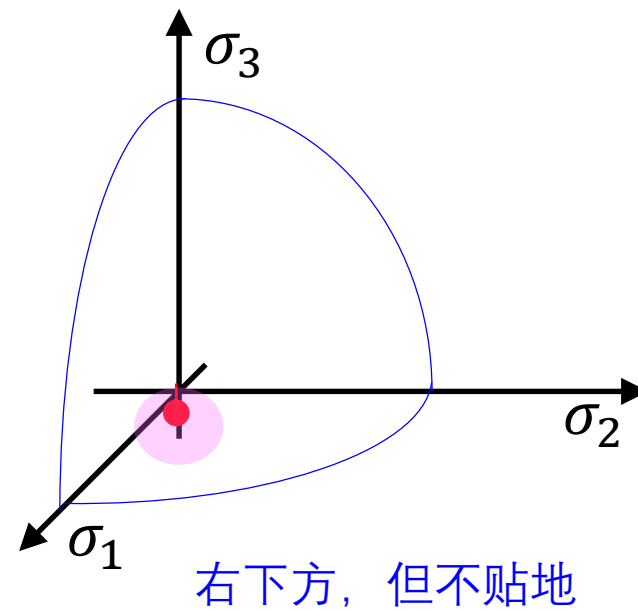
- 2个接近0
- 1个大特征值



边沿检测——点云局部特征

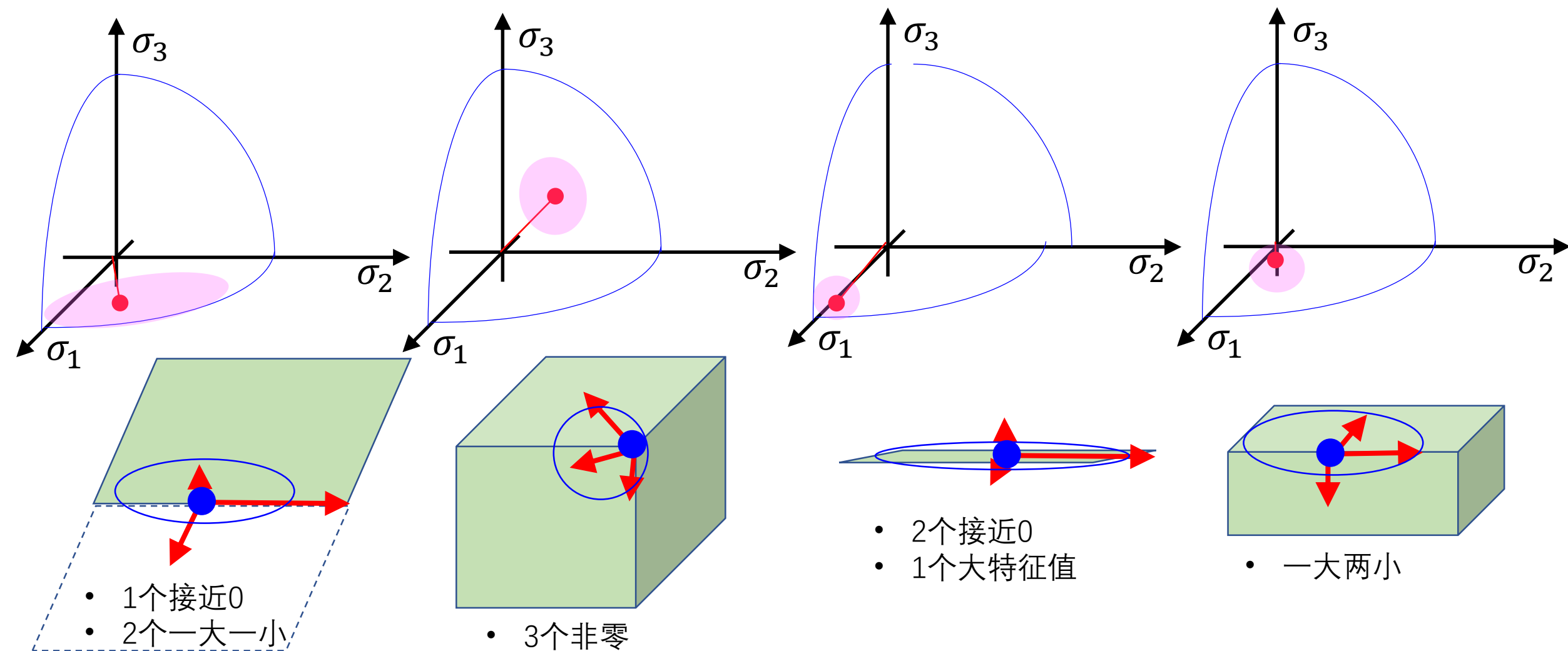


- 一大两小

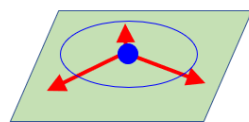


边沿检测——点云局部特征

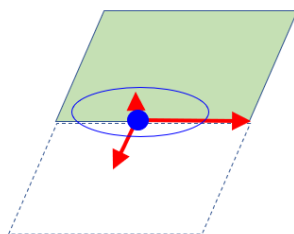
- 设计的算法通过将球面分为若干区域，可以区分这几种情况，



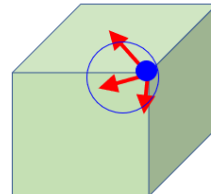
点云局部特征——特征值



- 1个接近0
- 2个相同值



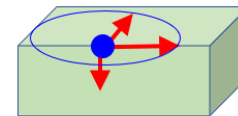
- 1个接近0
- 2个一大一小



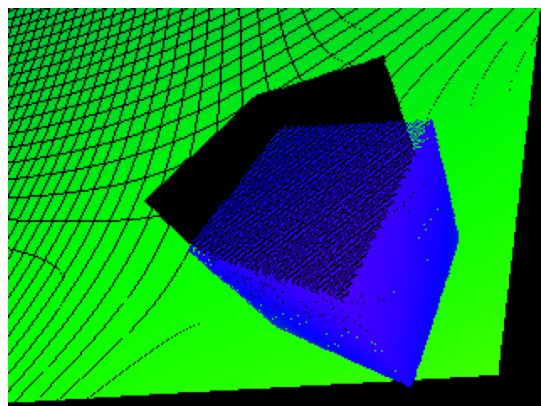
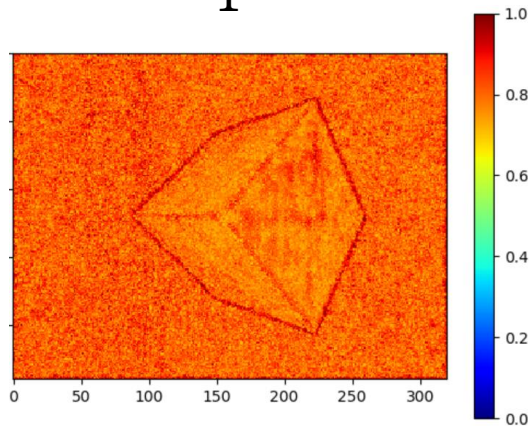
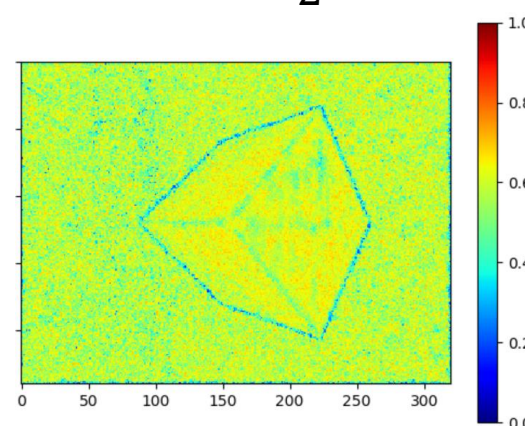
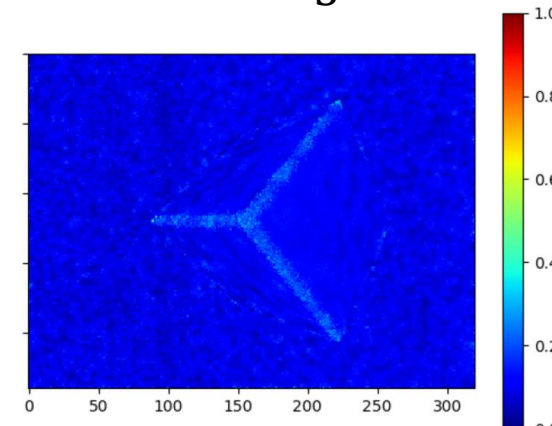
- 3个非零



- 2个接近0
- 1个大特征值

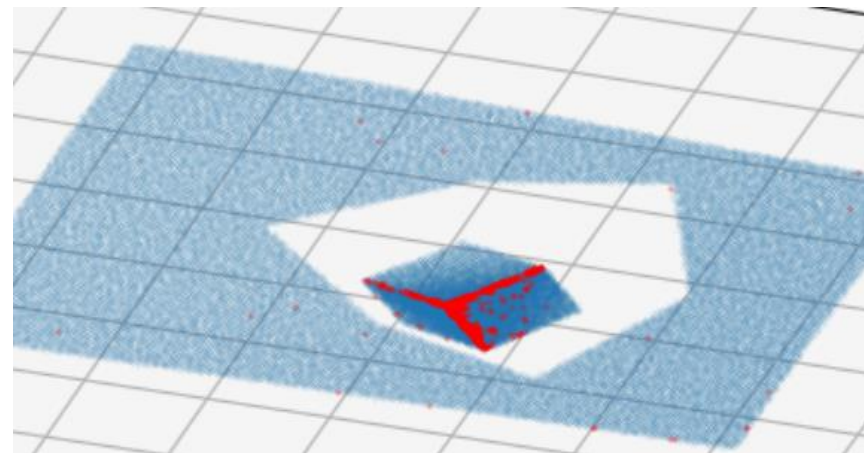
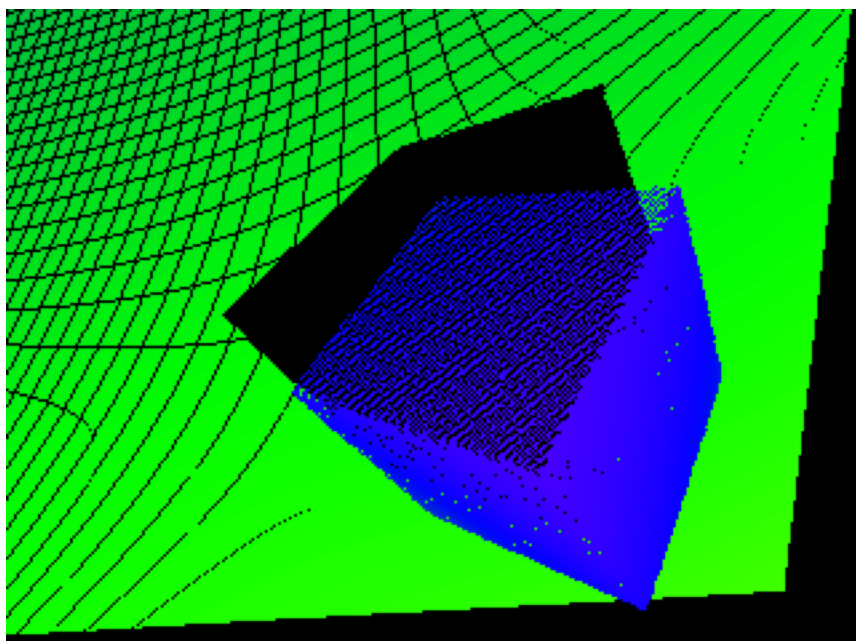


- 一大两小


 σ_1

 σ_2

 σ_3


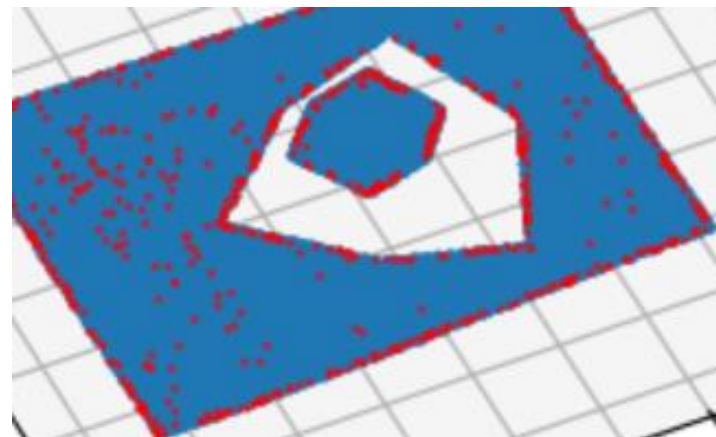
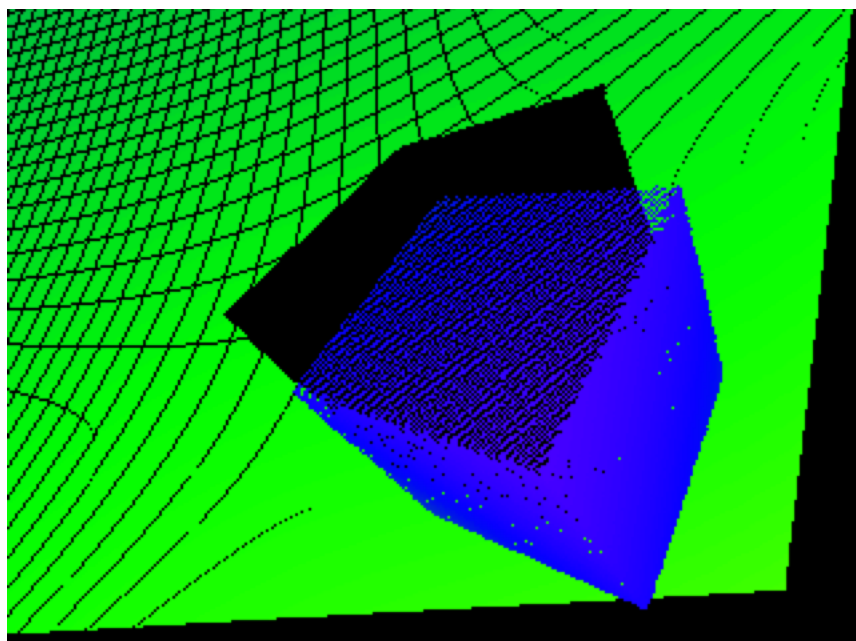
由于噪声，实际情况并不是那么理想

边界检测——棱边检测，基于手工选择的门限



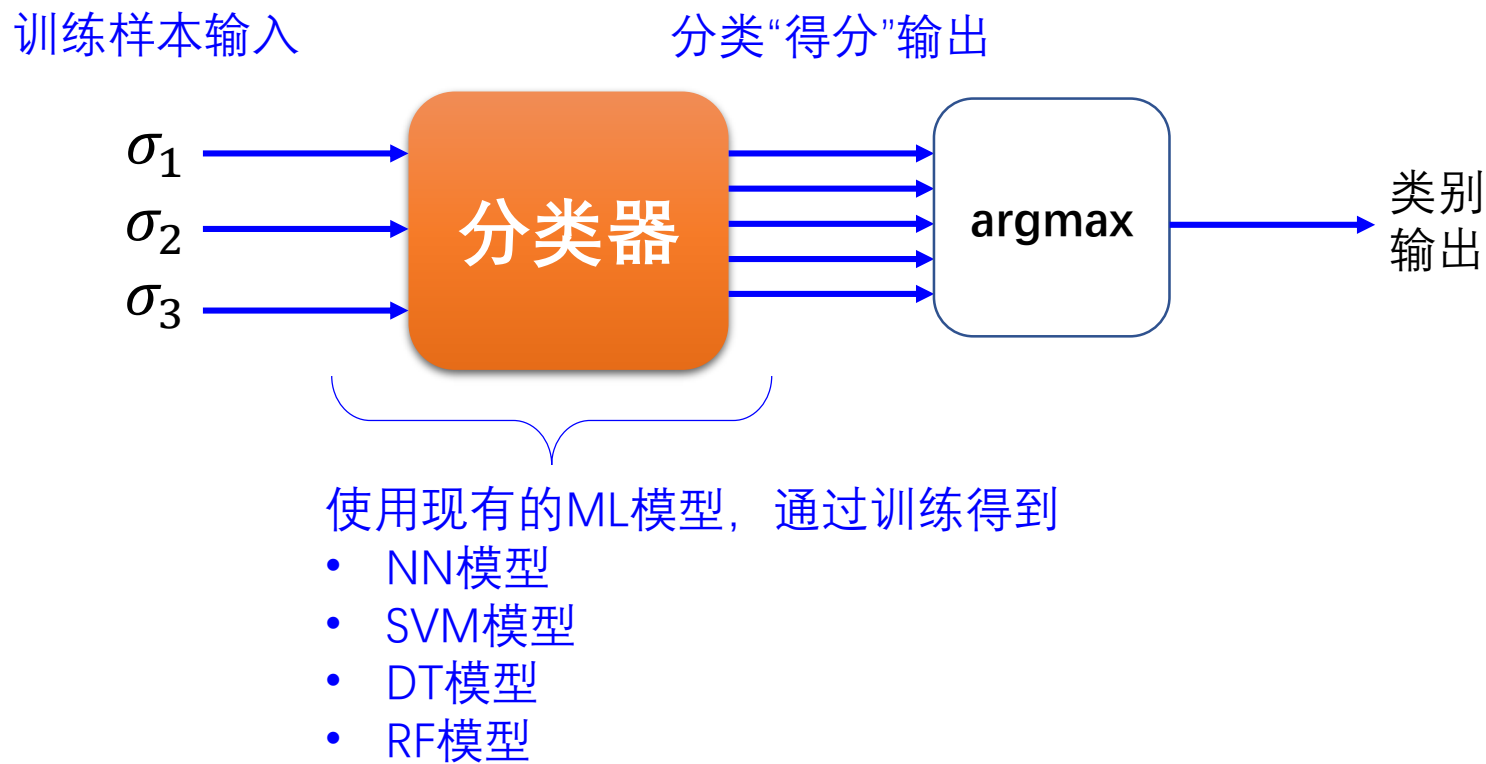
- 领域半径0.03
- $\sigma_3 > 0.22$

边界检测——平面边沿，基于手工选择的门限



- 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

基于ML的方案

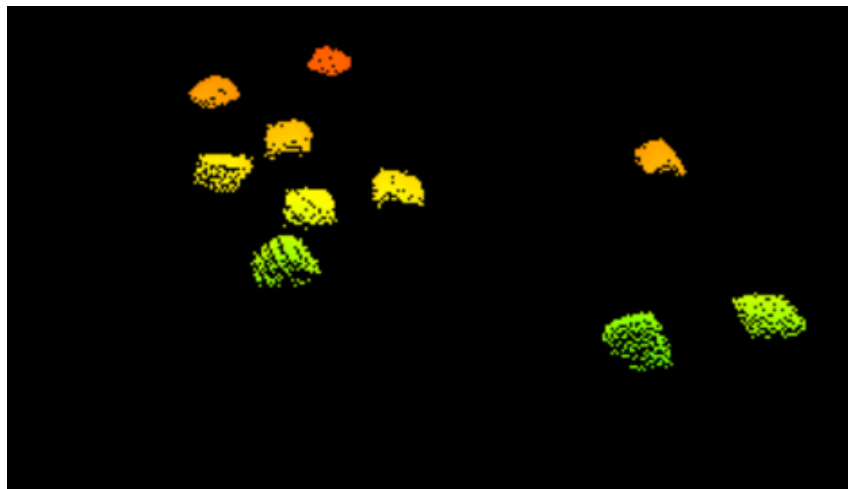
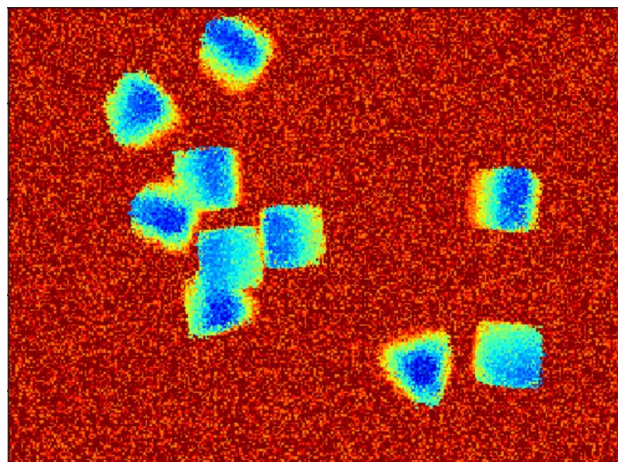
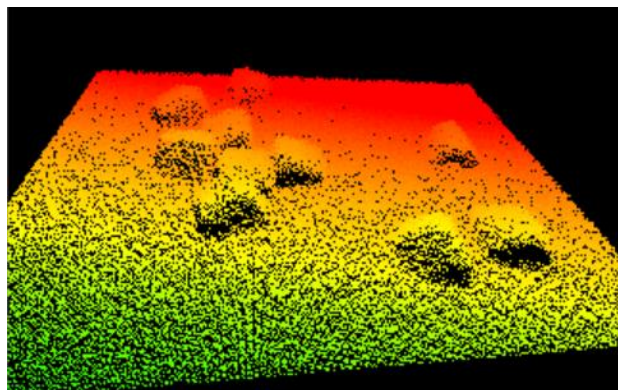


内容概要

- 特征提取与检测
- 前后景分离
 - 基于模型匹配的方法
 - 基于背景扣除的方法
 - 基于统计分类的方法
 - 背景自适应更新
- 几何形状识别

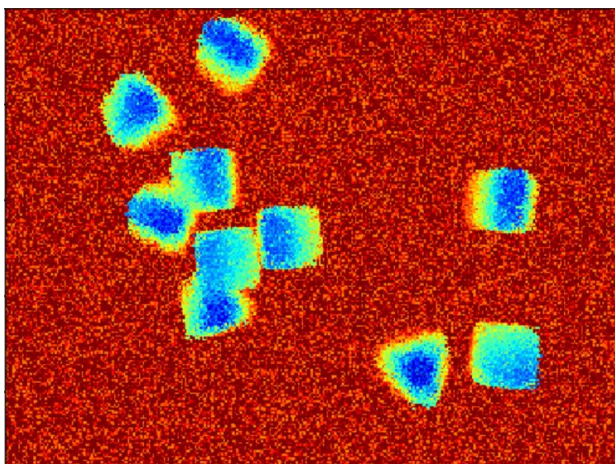
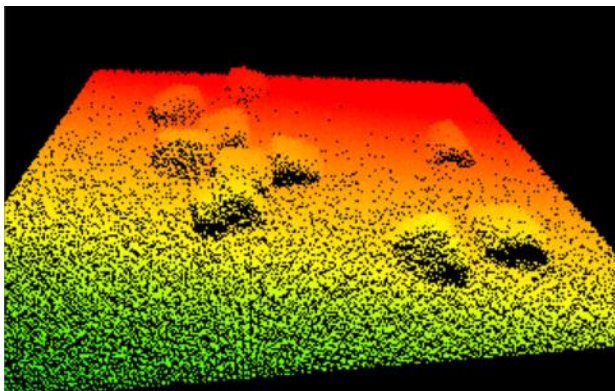
地面物体检测——地平面检测和扣除

- 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

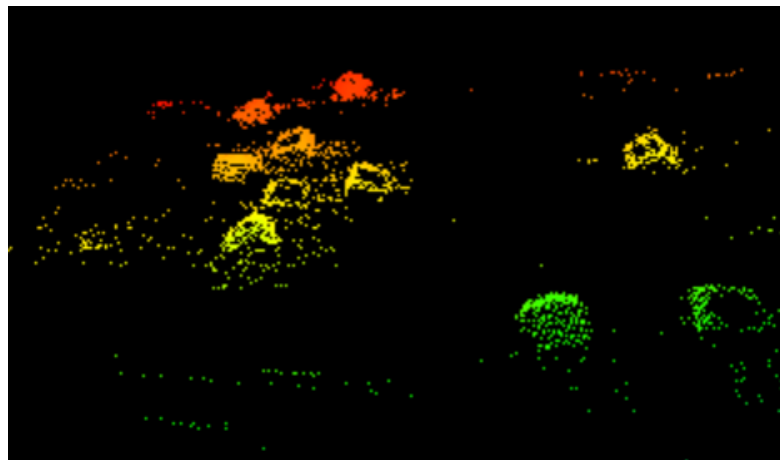


```
n,D=plane_det_pc_ransac(pc,r=0.2,k=0.4,th=0.06,it=200, it1=3, verbose=True)
# 选出平面上的点
dist=pc_to_plane_dist(n,D,pc)
# 扣除了平面上的点
TH=0.02
pc_view(pc[dist>TH],CAM_FX,CAM_FY,CAM_CX,CAM_CY,CAM_WID,CAM_HGT,dmin=0.0,dmax=3.0)
```

地面物体检测——PCA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：

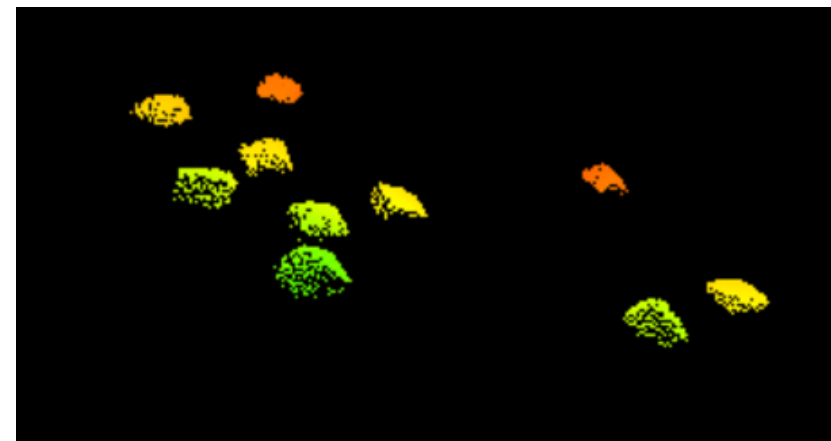
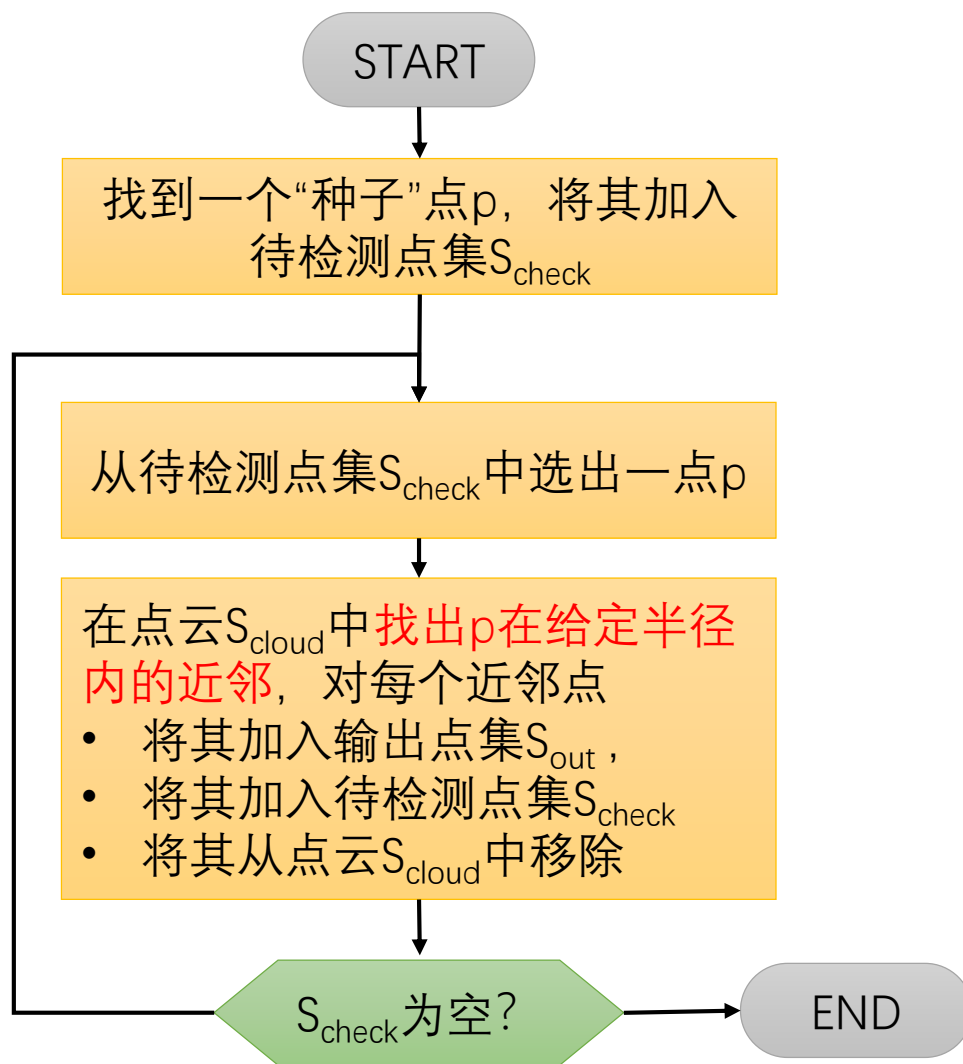
- 对于有较大平面的物体表面出现空洞
- 噪声带来“飞散点”



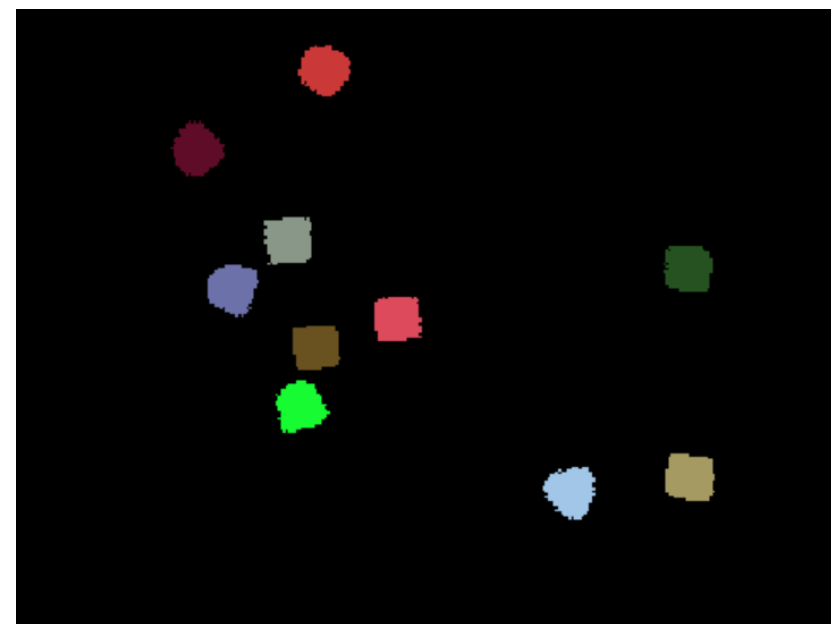
- 用点云的过滤算法可以清除

区域生长法表面点云提取

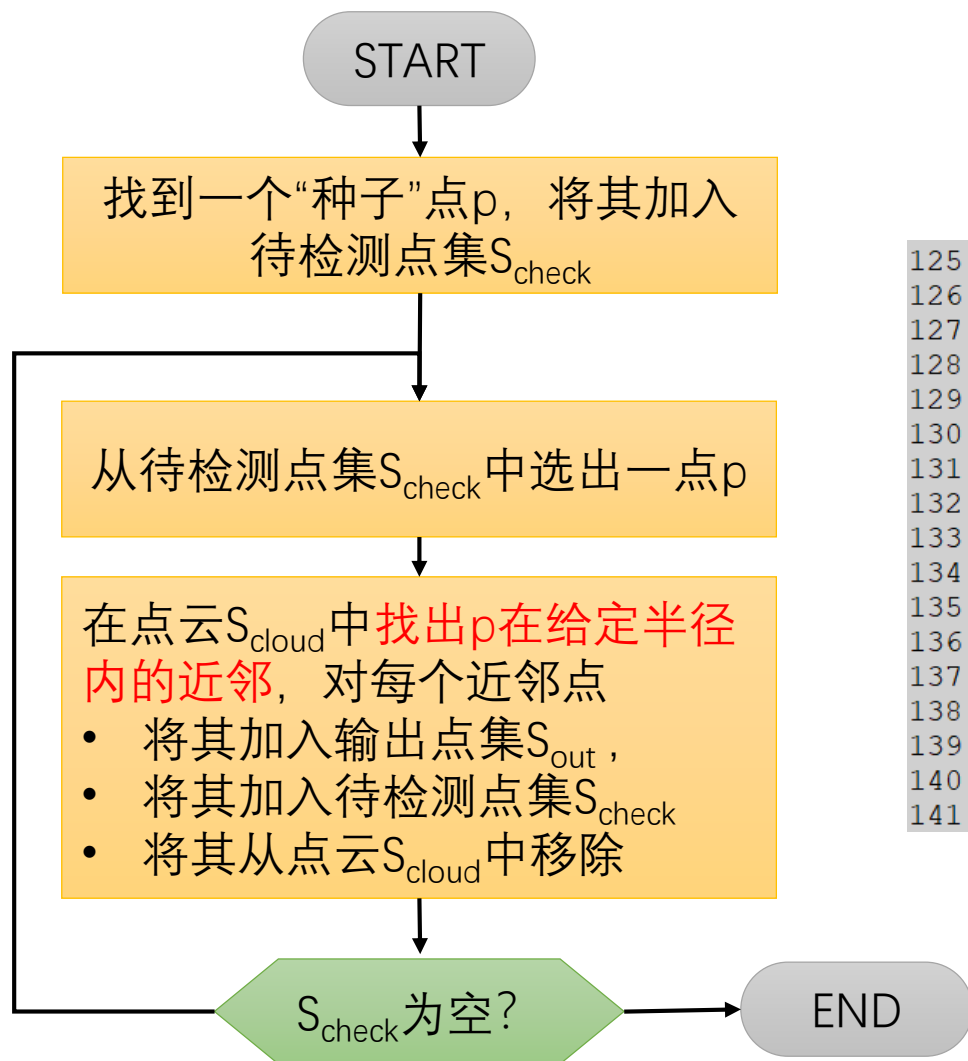
- 目的——将提取的点云按物体划分
- 物体分割是物体测量的前提



分离出每
“团”点云



区域生长法表面点云提取



```

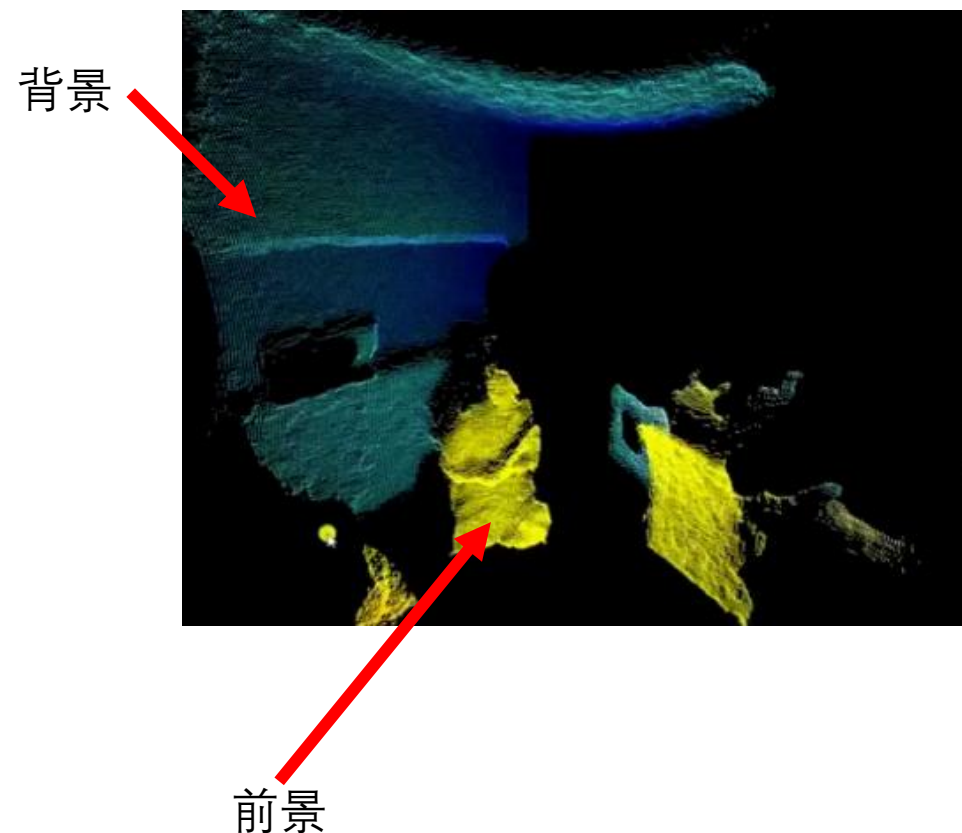
125 def pc_obj_merge(pc, p0, k, r, mask=None, ret_idx=False):
126     flann = cv2.FlannBasedMatcher(dict(algorithm=1, trees=5), dict(checks=50))
127
128     # 指示尚未分割的点云点 (这里用copy() 是为了防止修改输入的mask对象)
129     mask = np.ones(len(pc), dtype=bool) if mask is None else mask.copy()
130     # 通过mask指示哪些点从Scloud中移除了
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  # 新增的点云序号加入输出集合
139         pc_chk += [pc[i] for i in idx_nn]  # 新增点云加入待检测集合
140         mask[idx_nn] = False  # 标注已被处理的点 从Scloud中移除
141     return (pc[idx_out], idx_out) if ret_idx else pc[idx_out]
  
```

内容概要

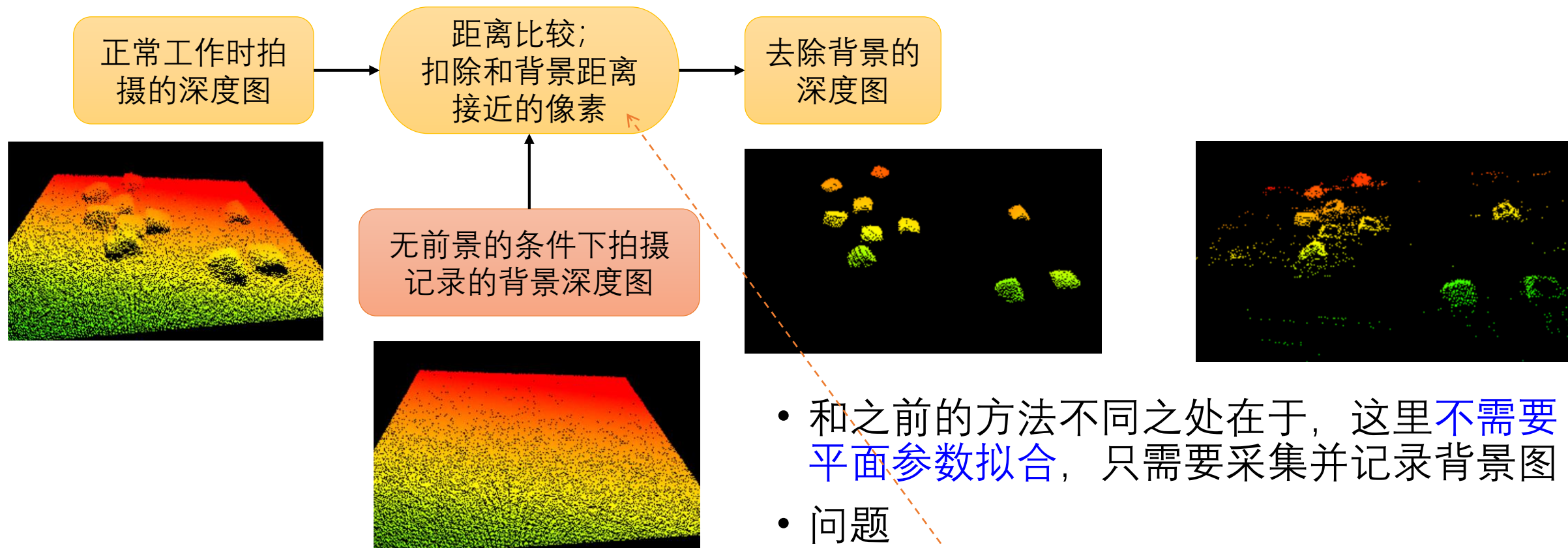
- 特征提取与检测
- 前后景分离
 - 基于模型匹配的方法
 - 基于背景扣除的方法
 - 基于统计分类的方法
 - 背景自适应更新
- 几何形状识别

深度图的前背景分离

- 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



深度图的前背景分离——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



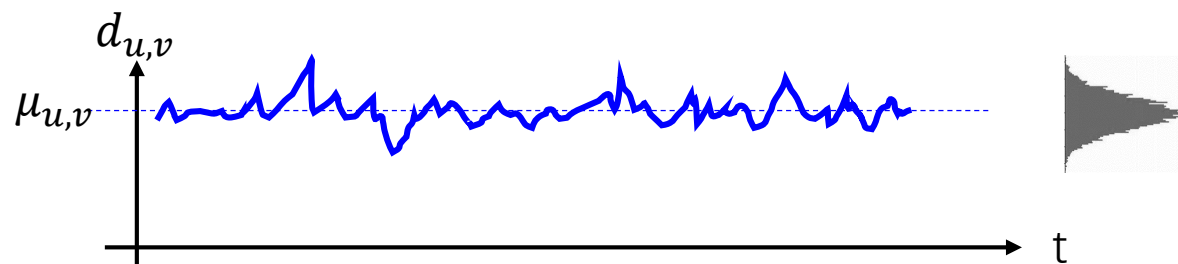
- 这个距离门限到底为多少？
- 噪声使得背景去除不彻底
- 相机或者背景改变，被错认为前景

内容概要

- 特征提取与检测
- 前后景分离
 - 基于模型匹配的方法
 - 基于背景扣除的方法
 - 基于统计分类的方法
 - 背景自适应更新
- 几何形状识别

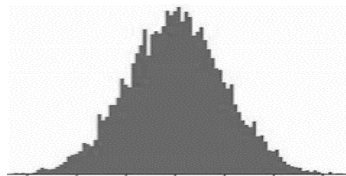
深度图的前背景分离——统计建模前背景分离方法

- 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



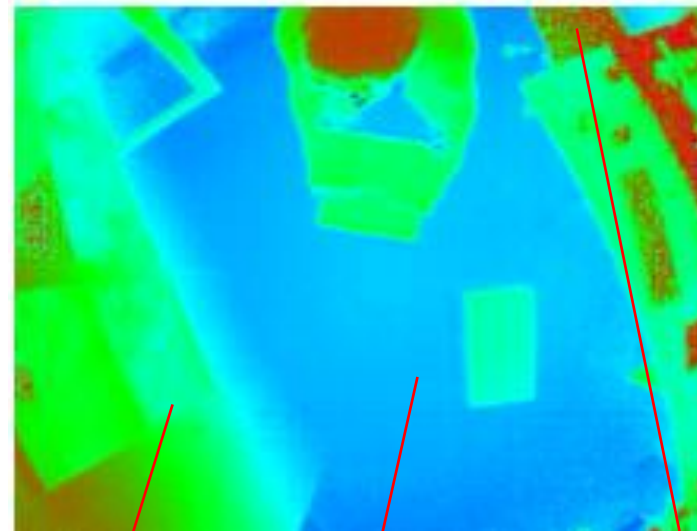
高斯型状的描述：

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

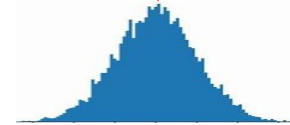
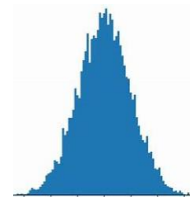


- 背景不同地方有不同的均值的方差

顶视相机，拍摄在屋子里摆放物件



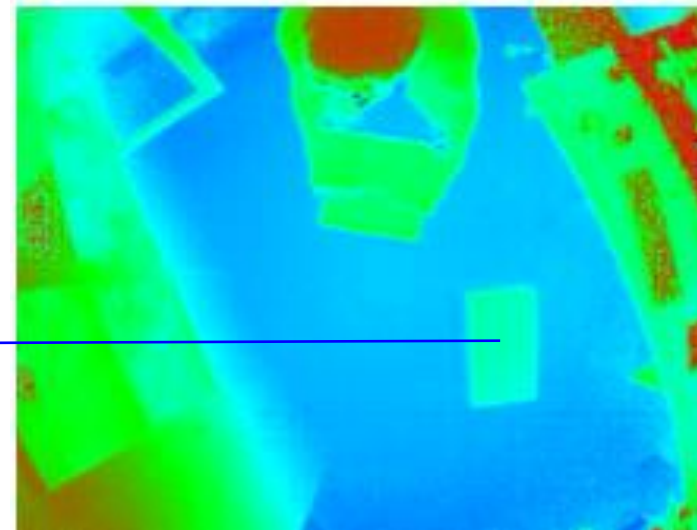
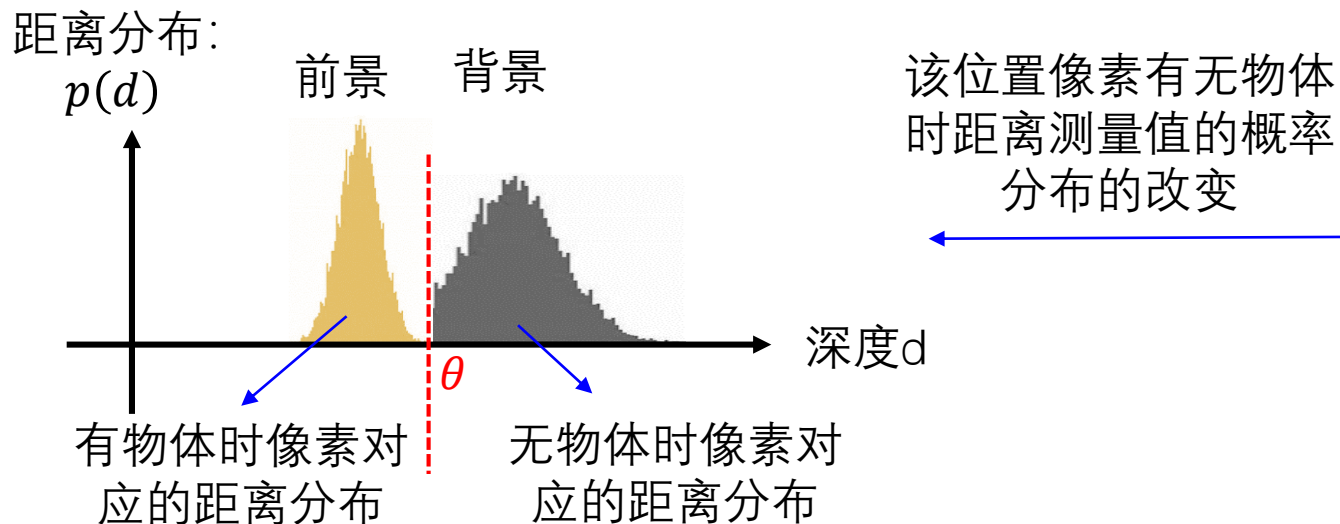
近距离，测量
噪声方差小



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

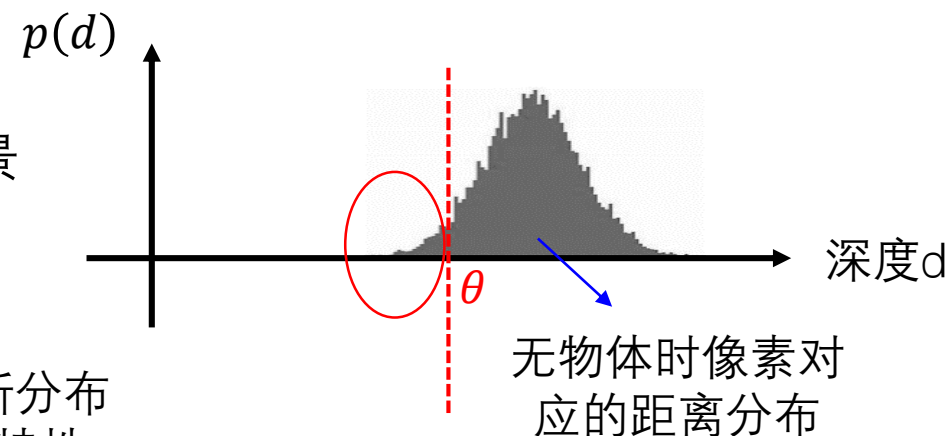


- 通过设定门限 θ 来确定当前像素是否是前景
- 测量值小于门限 θ 表示前景物体出现，大于 θ 表示目前只有背景
- 背景被误认为前景的错误率——右图红色圈中的面积占比

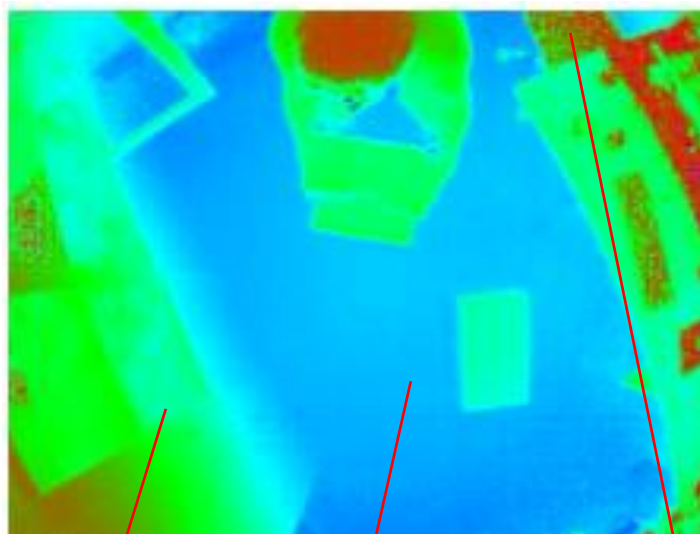
😐 如何选择门限 θ

- 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

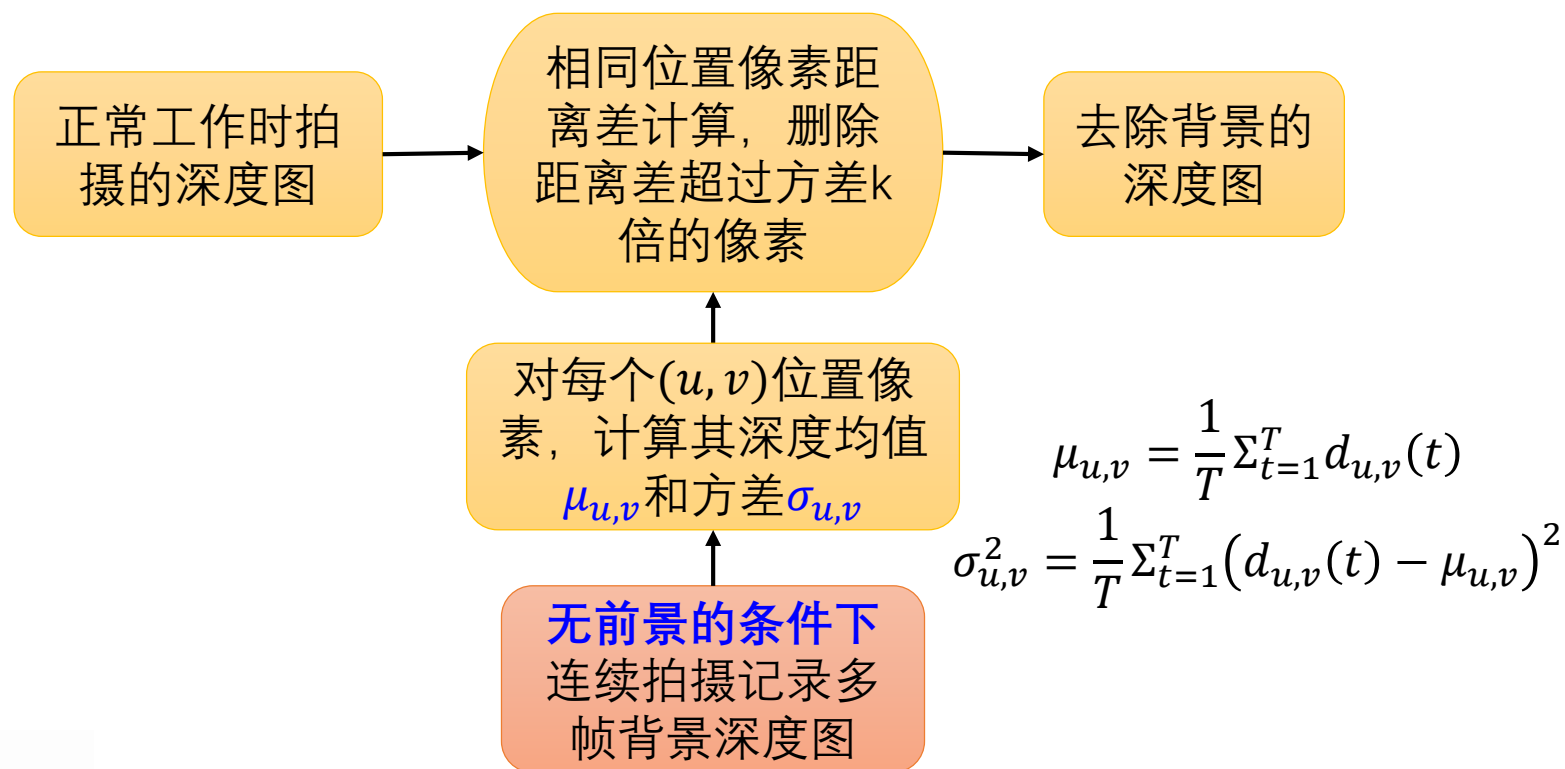


深度图的前背景分离——统计建模前背景分离方法



近距离，测量
噪声方差小

远距离，测量
噪声方差大



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

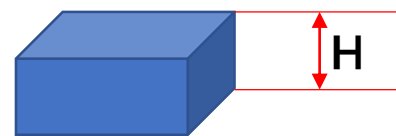
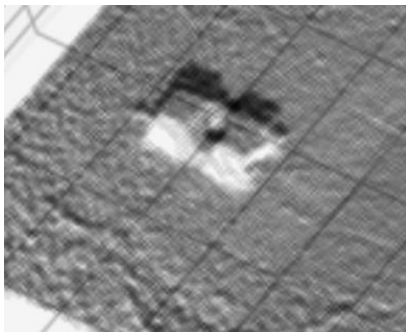
应用例子——检测叠放在地面的砖块位置

图中显示了在地面上的砖块，2层，叠成#字形

深度图



灰度图



- 以 $0.5H$ 为门限，
- 将高出地面 $0.5H$ 的点云分离



点云图



问题

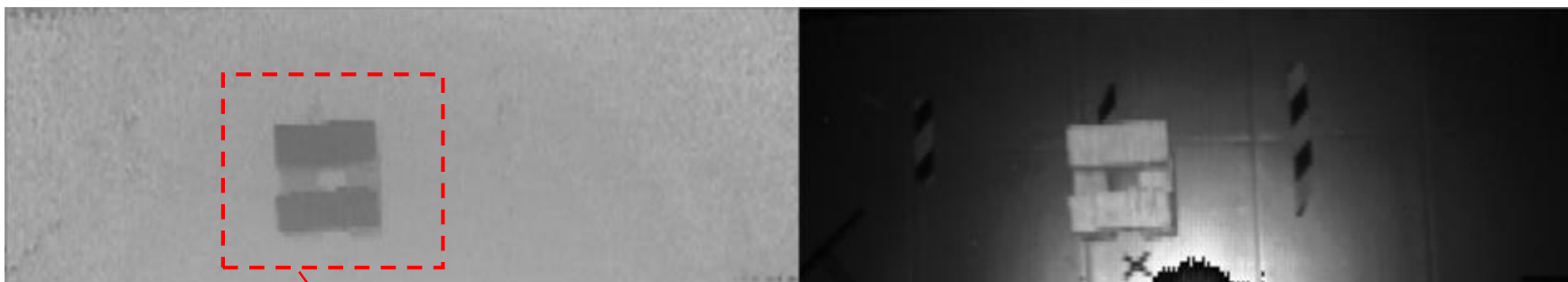
- 上述门限是否合理？
- 不同材质物体测量值波动是否一致
 - 比如地面测量结果波动很大
- 上面摆放的前景物体测量波动很小
- 此时，为降低地面误判为前景物体，期望提高分割门限，比如设为 $0.75H$
- 到底设多少合适？

深度图的前背景分离——统计建模前背景分离方法

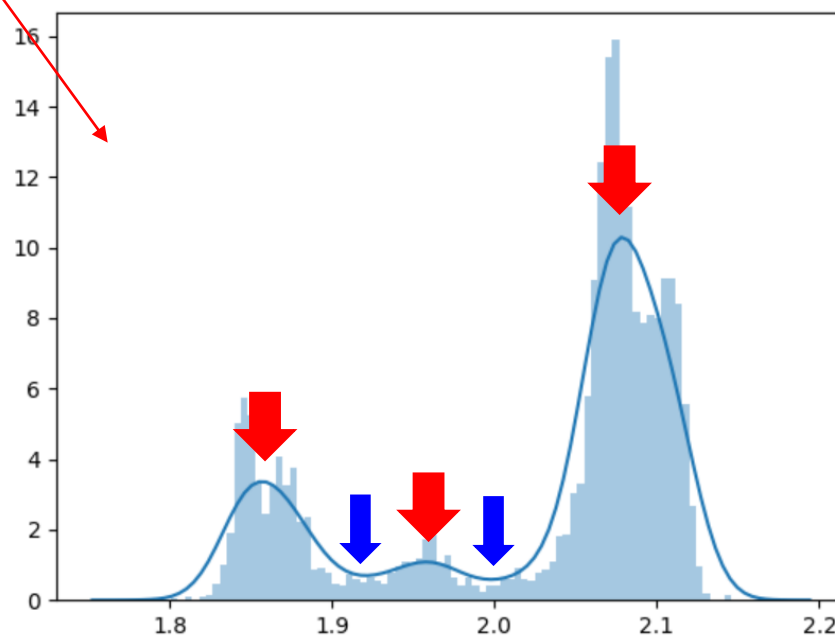
使用空间区域内的直方图分析来找到分割门限

深度图

灰度图



分析区域内
像素距离的
直方图

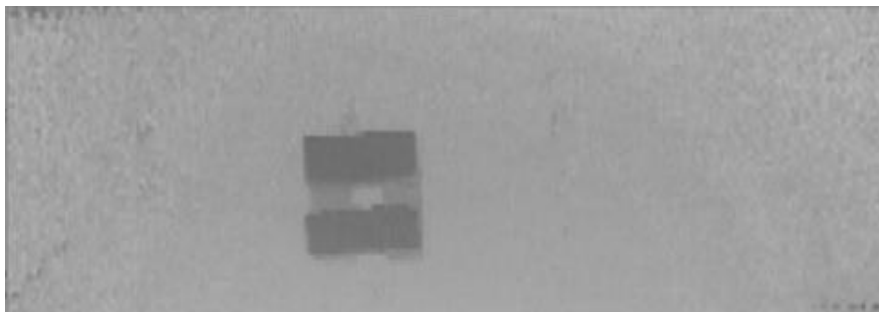


- 深度图直方图分析，显示有3个“高斯”型
- 三个高斯距离分别对应：
 - 地面
 - 第一层砖块
 - 第二层砖块的平面
- 通过每个高斯分布的均值能够知道砖块平面到相机的距离
- 通过高斯和高斯之间的“谷底”能够得到平面切割门限

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

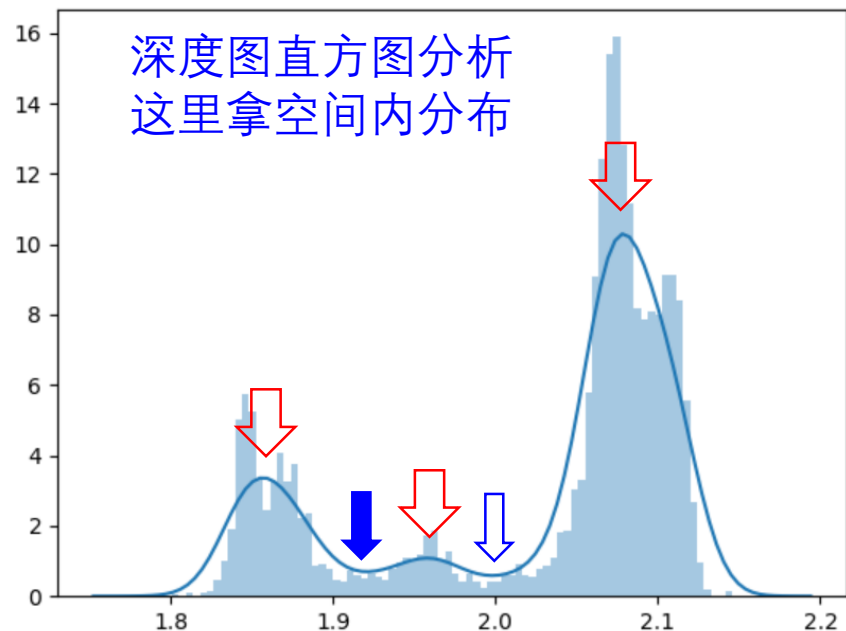
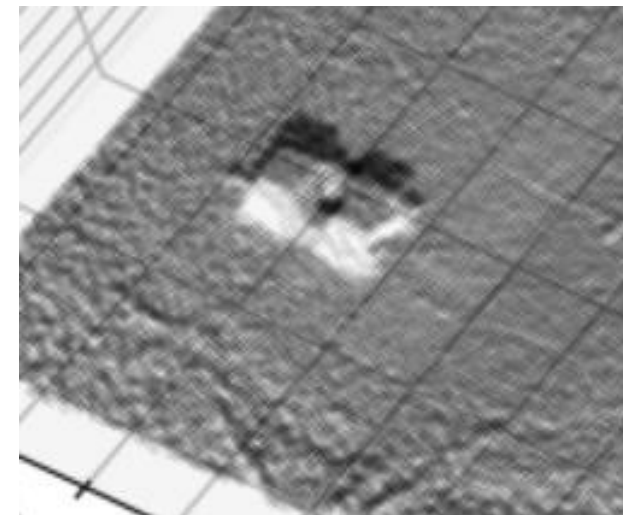
深度图



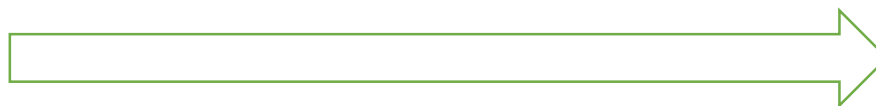
灰度图



点云图



通过第一个“谷底”为门限
分离出来的砖块顶端



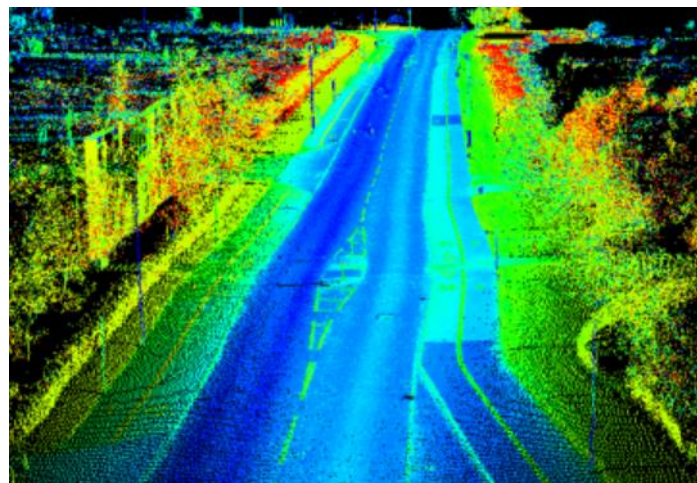
内容概要

- 特征提取与检测
- 前后景分离
 - 基于模型匹配的方法
 - 基于背景扣除的方法
 - 基于统计分类的方法
 - 背景的自适应更新
- 几何形状识别

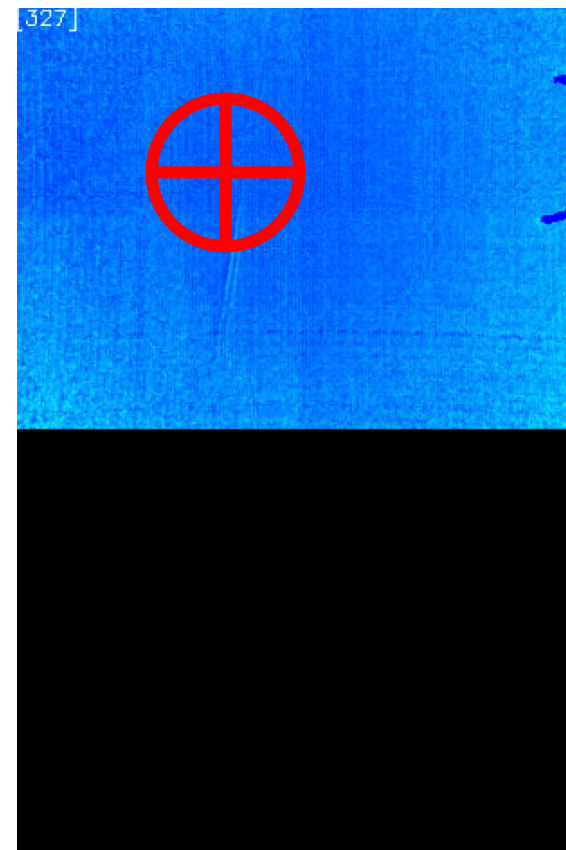
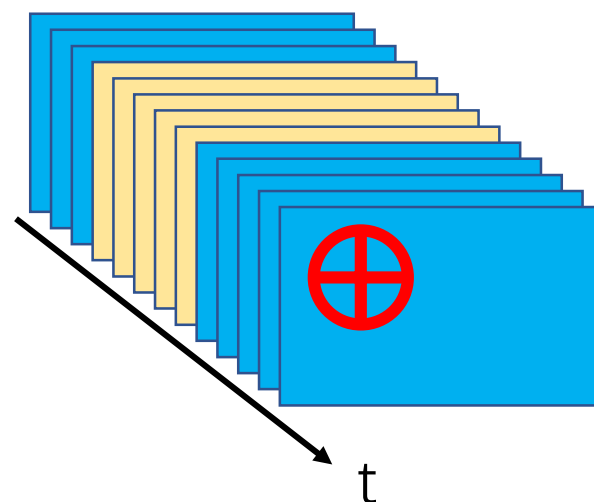
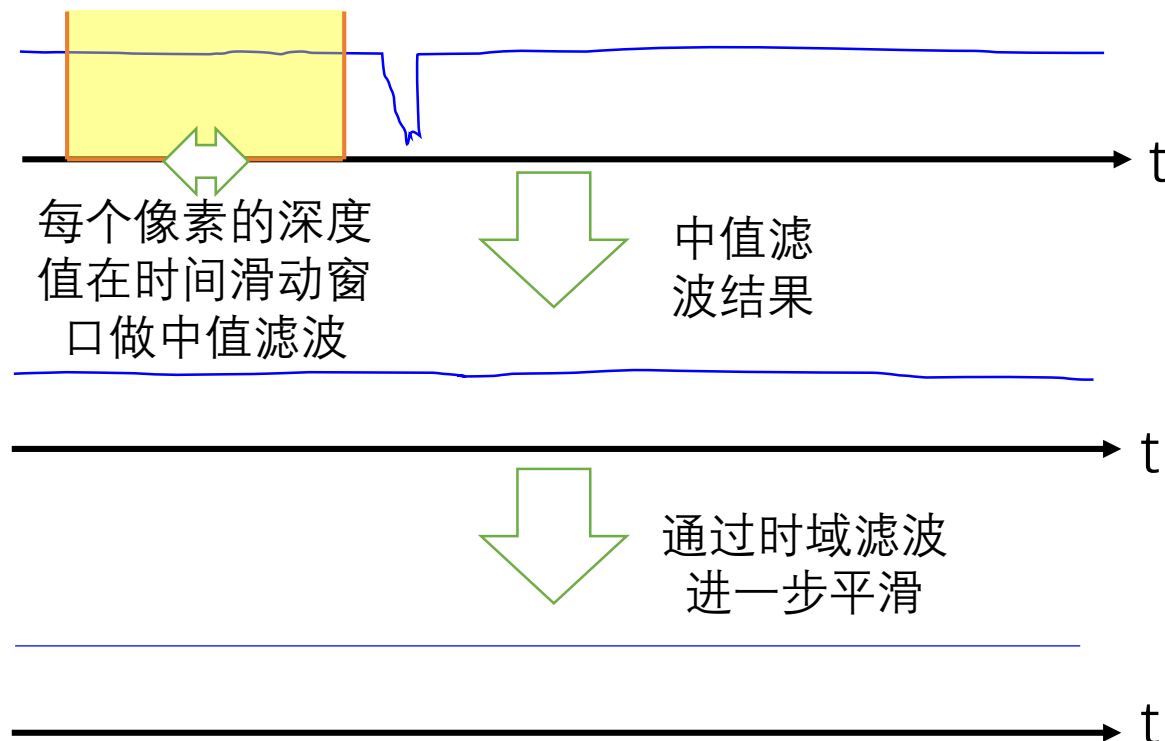
背景的自适应更新

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

内容概要

- 特征提取与检测
- 前后景分离
 - 基于模型匹配的方法
 - 基于背景扣除的方法
 - 基于统计分类的方法
 - 背景的自适应更新
- 几何形状识别

几何形状识别

- 基于“剪影”轮廓的方法
- 基于特征匹配的方法
- 神经网络

基于“剪影”轮廓的几何体识别

——通过不同视角的剪影识别几何体



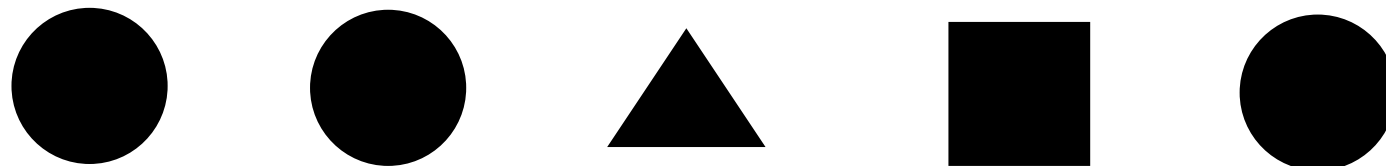
• 为什么不直接在3D数据上识别？



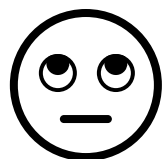
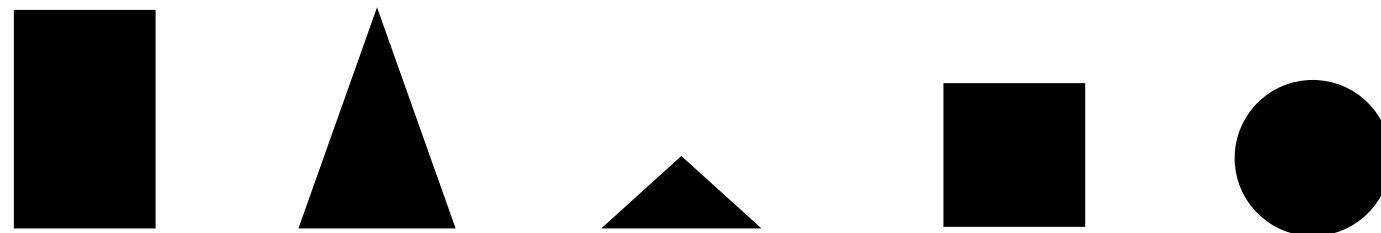
• 因为运算简单

- 当我们有了3D数据后，就能够方便地生成各个角度的投影
- 不同的几何体，在不同方向的投影有一定特点，可以用于识别

俯视图

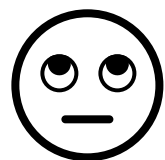


正视图



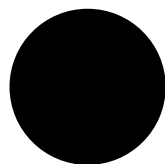
能够识别出以上投影对应的他们的3D几何体吗？

通过不同视角的剪影识别

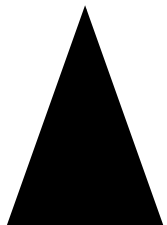
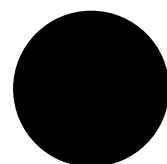


俯视图

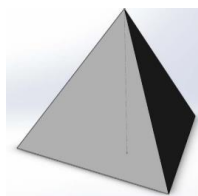
正视图



圆柱体



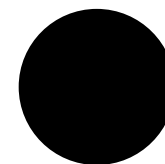
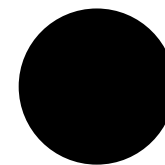
圆锥体



四面体



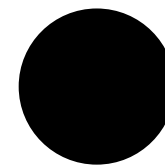
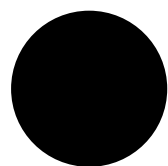
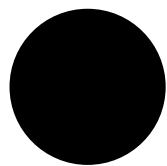
立方体



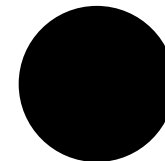
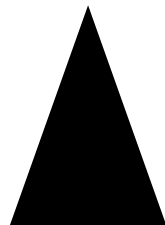
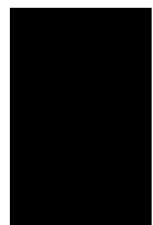
球

通过不同视角的剪影识别

俯视图



正视图



圆柱体

圆锥体

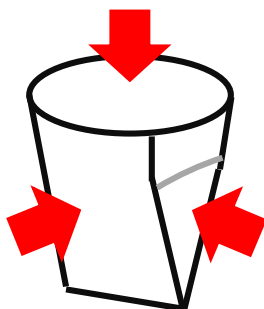
四面体

立方体

球

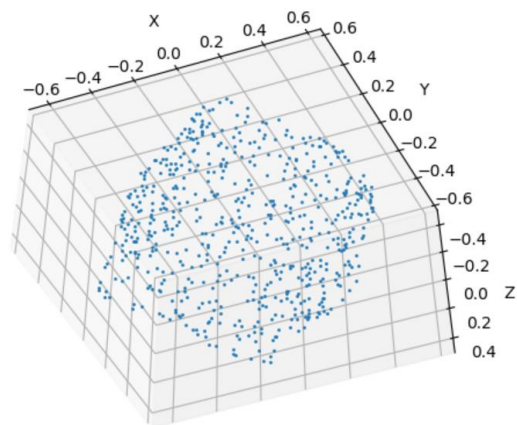


对形状有
多大把握?



视角越多，分类的把握越大

通过不同视角的剪影识别



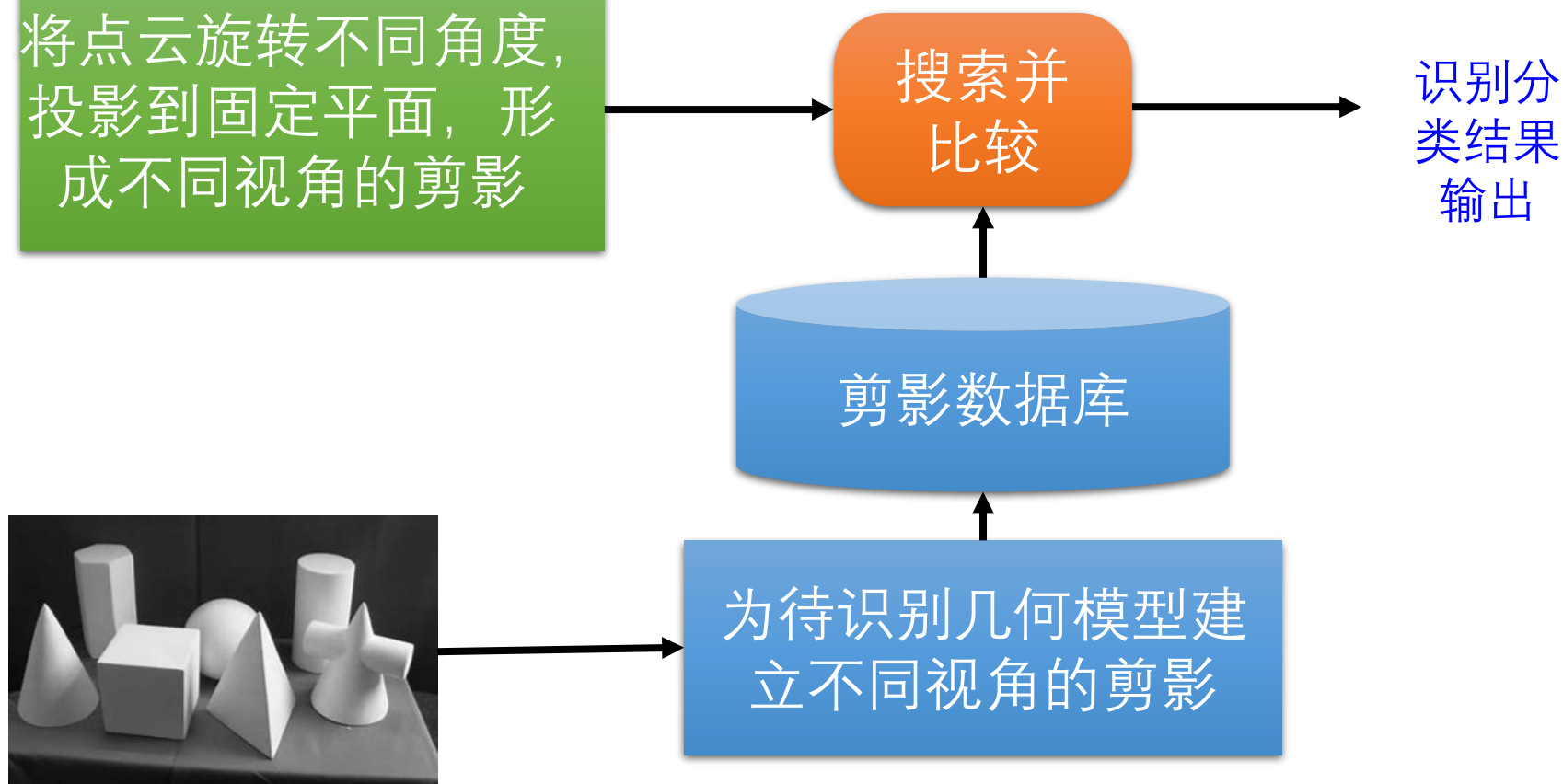
将点云旋转不同角度，
投影到固定平面，形
成不同视角的剪影



• 如何搜索？



• 后面通过手势识别的简单例子介绍
整个步骤，以及其中的算法优化



待识别分类的几何体模型数据

应用实例——静态手势识别

- 识别手部动作



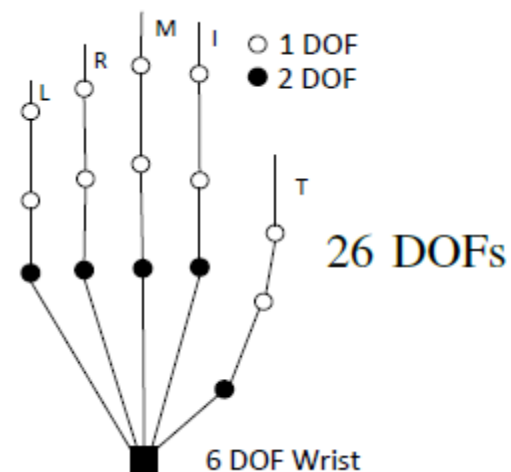
静态手势识别——技术途径

• 基于骨架模型

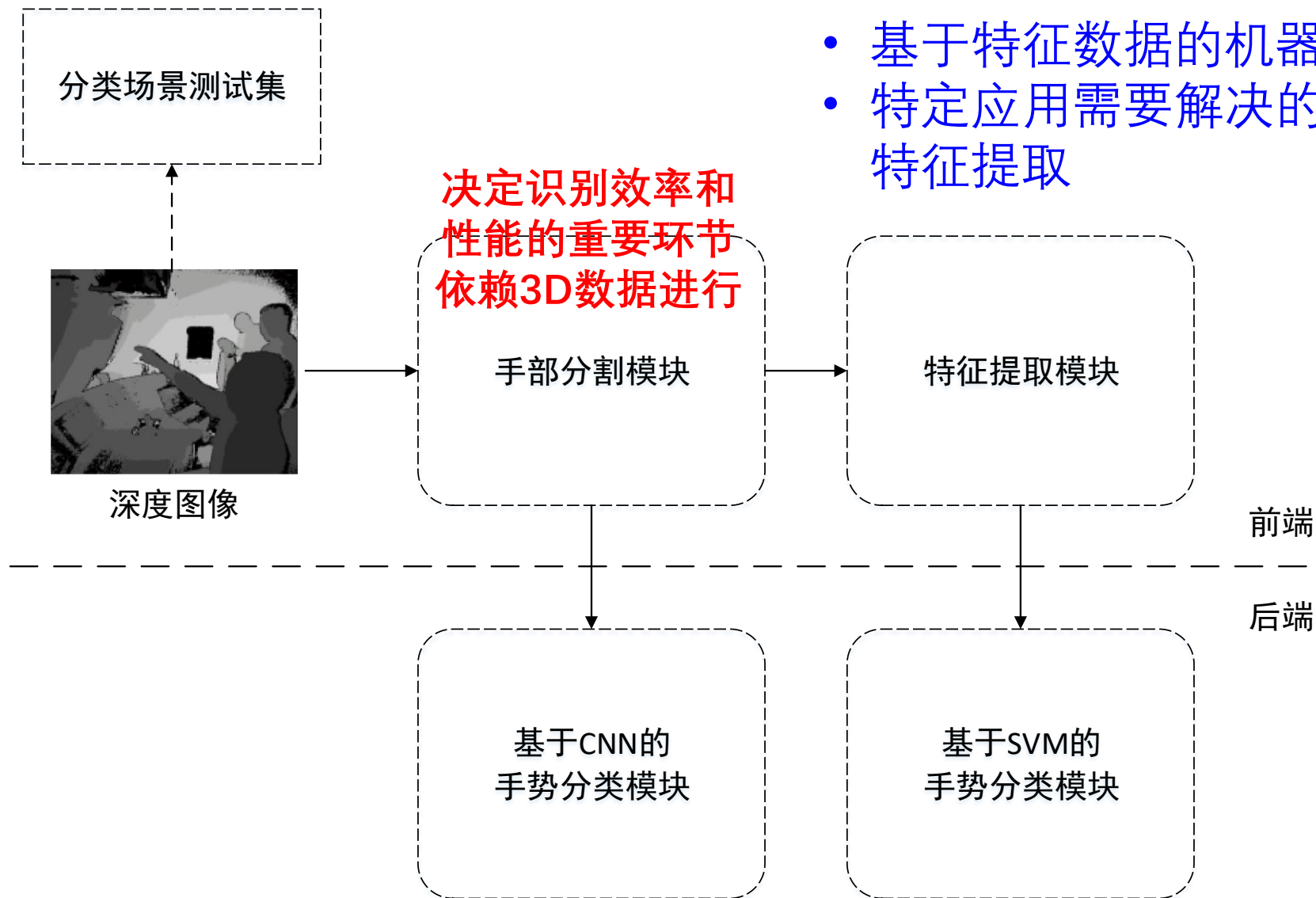
- 分两步进行：
 - 1) 识别手部关节骨架模型参数
 - 2) 识别手部动作分类
- 能够识别精细动作
- 效率低，运算量大，受自遮挡影响

• 基于剪影轮廓

- 分两步
 - 手部分割，获取剪影
 - 剪影形状分类
- 识别明显的手势动作
- 效率高，运算量小
- 受自遮挡影响，精细动作区分困难



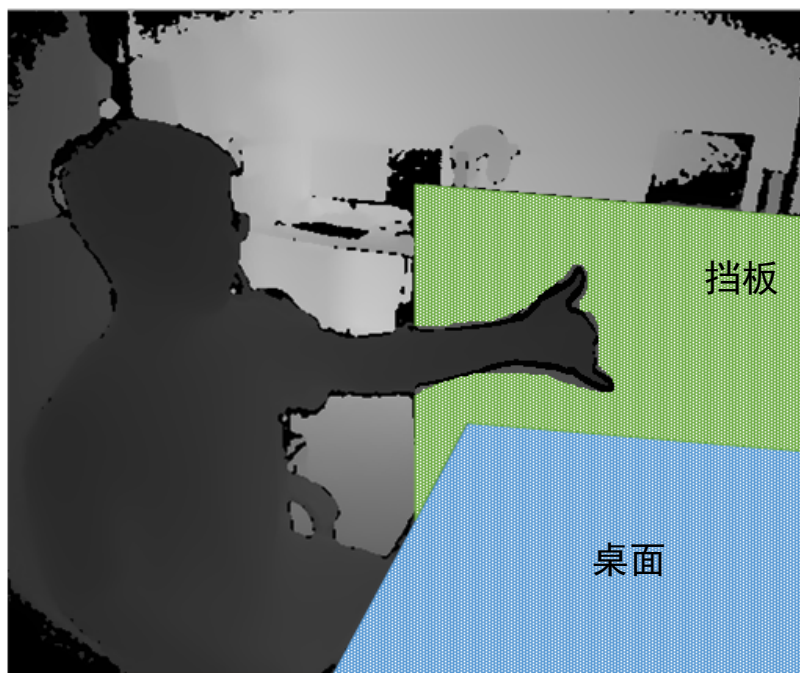
静态手势识别——算法流程(轮廓剪影)



- 基于特征数据的机器学习算法相对成熟
- 特定应用需要解决的困难是目标分割和特征提取

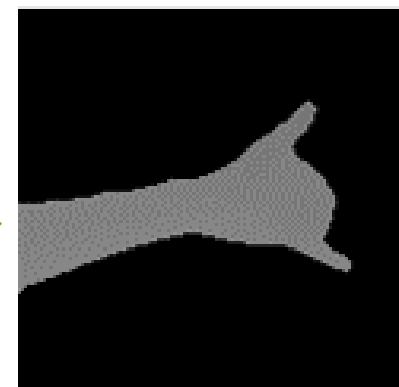
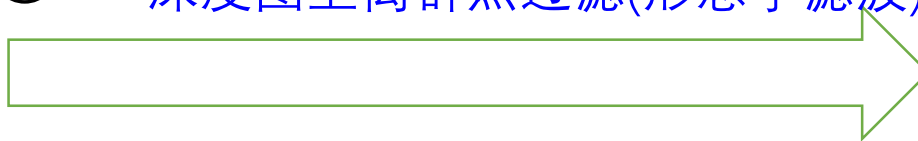
静态手势识别——前景提取

- 从深度图中提取手部像素

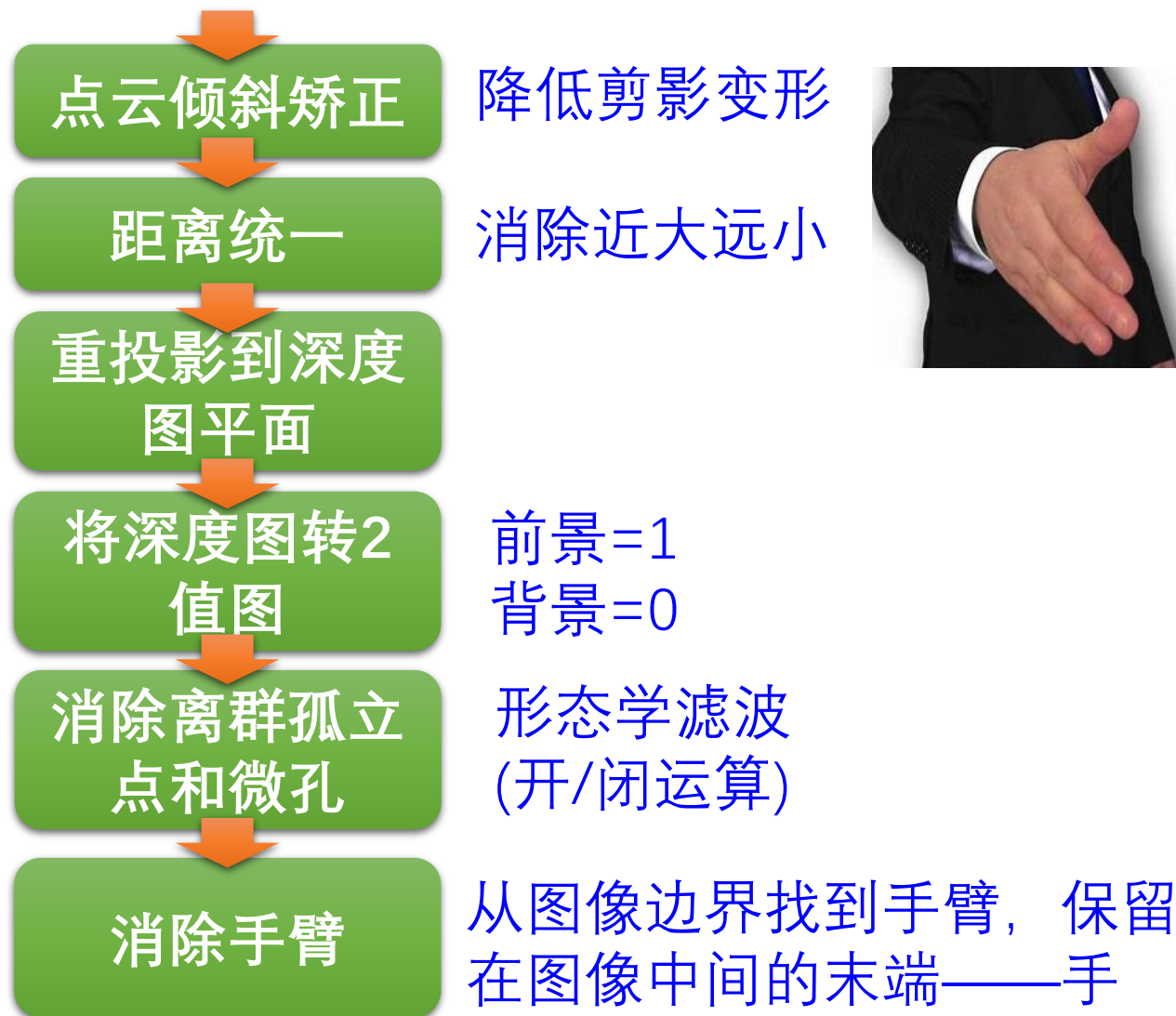


采取多种措施滤除背景

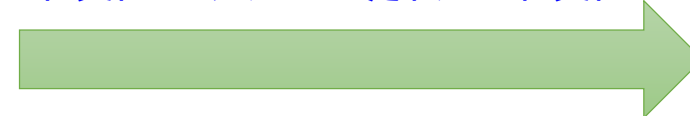
- 😊 • 基于距离的点云过滤
近距离的桌面怎么办? 😞
- 😊 • 基于背景扣除的点云过滤
背景的噪声怎么办? 😞
- 😊 • 基于统计建模的背景去除
残留的噪点怎么办? 😞
- 😊 • 深度图上离群点过滤(形态学滤波)



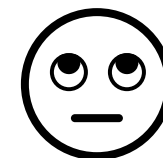
静态手势识别——图像剪影



深度图→点云→旋转→深度图



- 如何获得旋转量？
- 把连着手臂的点云看成椭球体的话，希望他的长轴旋转到平行于相机传感器平面



静态手势识别——图像剪影



- ☹️ • 如何将剪影变成可以比较的数字?
- 😊 • Hu矩——将剪影转成7维向量
 - 旋转不变
 - 平移不变
 - 缩放不变

图像的Hu不变矩特征

下面依次给出几个数学概念和运算方法

- 矩(Moment): 概率与统计中的概念，是随机变量的一种数字特征

$$m_{pq} = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} x^p y^q f(x, y) \quad p, q = 0, 1, 2 \dots$$

$f(x, y)$ 是图像在位置 (x, y) 的(灰度)值
 (N, M) 是图像尺寸
 对二值图 $f(x, y)=0$ 或1

构造7个不变矩，
具有平移、旋转和尺度不变特性

- 图像中心(重心)

$$\bar{x} = m_{10}/m_{00}$$

$$\bar{y} = m_{01}/m_{00}$$

把 $f(x, y)$ 看作像素重量的话，它给出了图像“重心”

- 中心矩

$$\mu_{pq} = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} (x - \bar{x})^p (y - \bar{y})^q f(x, y) \quad p, q = 0, 1, 2 \dots$$

- 归一化中心矩

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}$$

$$\gamma = \frac{p+q}{2} + 1, \quad p+q = 2, 3, \dots$$

$$\begin{aligned} h_0 &= \eta_{20} + \eta_{02} \\ h_1 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\ h_2 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\ h_3 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\ h_4 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\ &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\ h_5 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\ &\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\ h_6 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\ &\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \end{aligned}$$

图像剪影的Hu不变矩特征

7个Hu不变矩

$$\begin{aligned}
 h_0 &= \eta_{20} + \eta_{02} \\
 h_1 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
 h_2 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
 h_3 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
 h_4 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
 &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
 h_5 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
 &\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
 h_6 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
 &\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)
 \end{aligned}$$

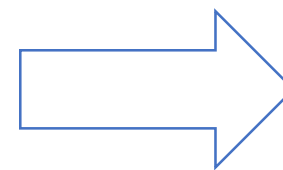


对数变换将其转
到接近的数量级

$$H_i = -\text{sign}(h_i) \log |h_i|$$

$$\begin{aligned}
 h_0 &= 0.00162663 \\
 h_1 &= 3.11619 \times 10^{-07} \\
 h_2 &= 3.61005 \times 10^{-10} \\
 h_3 &= 1.44485 \times 10^{-10} \\
 h_4 &= -2.55279 \times 10^{-20} \\
 h_5 &= -7.57625 \times 10^{-14} \\
 h_6 &= 2.09098 \times 10^{-20}
 \end{aligned}$$

数量级差别太大，
不利于特征比较








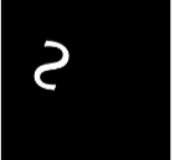
$$\begin{aligned}
 H_0 &= 2.78871 \\
 H_1 &= 6.50638 \\
 H_2 &= 9.44249 \\
 H_3 &= 9.84018 \\
 H_4 &= -19.593 \\
 H_5 &= -13.1205 \\
 H_6 &= 19.6797
 \end{aligned}$$

代码实现：

```

11 m = cv2.moments(img)
12 h = cv2.HuMoments(m)
13 H = -np.sign(h) * np.log10(np.abs(h))
    
```

图像剪影的Hu不变矩特征

id	Image	H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]
K0		2.78871	6.50638	9.44249	9.84018	-19.593	-13.1205	19.6797
S0		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S1		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S2		2.65884	5.7358	9.66822	10.7427	-20.9914	-13.8694	21.3202
S3		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	21.8214
S4		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	-21.8214

- 不同的S图像有相同的H值
- 具有平移、旋转和尺度不变特性

静态手势识别——图像剪影的Hu不变矩识别



- 为标准手势（包括不同的变形）建立Hu不变矩模板（训练集），作为样本库保存
- 对输入手势经过前处理，形成尺寸归一化的剪影
- 计算其Hu不变矩
- 和样本库的Hu不变矩比较，找到最接近的输出

比较两个不变矩 $H_i^{(a)}$ 和 $H_i^{(b)}$ 的
多种方式(不止列出的这几种)

$$D(a, b) = \sum_{i=0}^6 |H_i^{(a)} - H_i^{(b)}|$$

$$D(a, b) = \sum_{i=0}^6 \left| \frac{1}{H_i^{(a)}} - \frac{1}{H_i^{(b)}} \right|$$

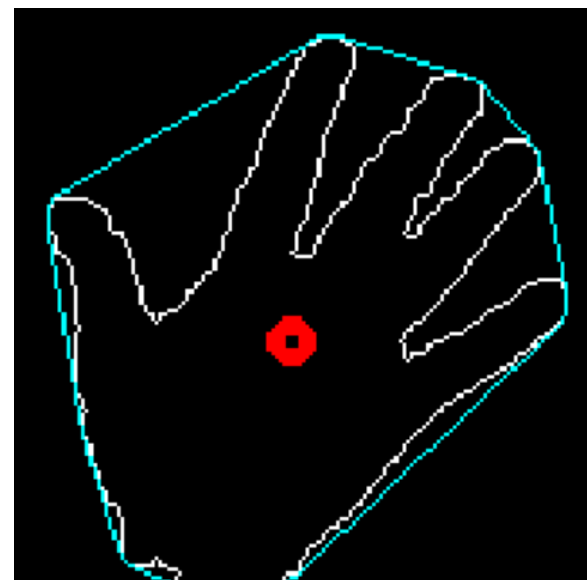
$$D(a, b) = \sum_{i=0}^6 \frac{|H_i^{(a)} - H_i^{(b)}|}{|H_i^{(a)}|}$$

$$D(a, b) = \sqrt{\sum_{i=0}^6 |H_i^{(a)} - H_i^{(b)}|^2}$$

静态手势识别——图像剪影的Hu不变矩识别

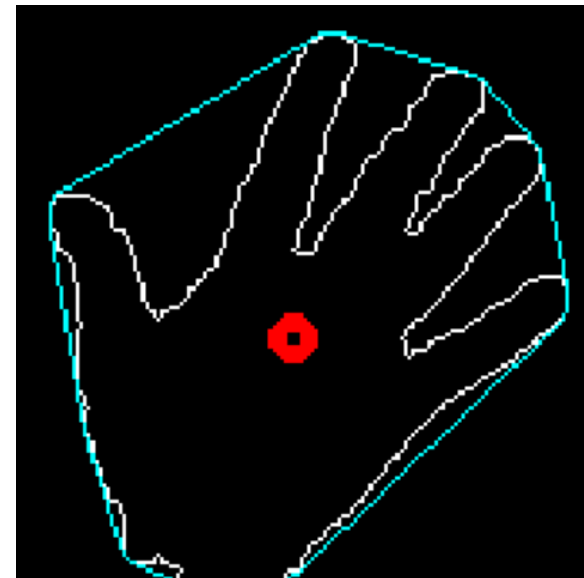
☹️ • 手形的变化、手指夹角改变、手掌切割误差、手指弯曲和手掌倾斜导致Hu不变矩的改变

- 😊 • 根据实际数据的算法调整
- 使用剪影的轮廓替代剪影计算Hu不变矩
 - 使用剪影的凸多边形轮廓替代剪影计算Hu不变矩
- 近邻检索的更新
- 替换成SVM分类器
 - 替换成神经网络分类器



静态手势识别——代码提示

- 深度图二值化: `cv2.threshold()`
- 找到外轮廓凸包: `cv2.convexHull()`
- 从二值图查找轮廓: `cv2.findContours()`
- 数值形态学滤波: `cv2.morphologyEx()`
- 形状比较: `cv2.matchShapes ()`



END

