# 虚拟立方体作第0帧进行KinectFusion测试报告

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 变更人 | 变更说明 | 变更时间 |
| V1.0 | 张琛 | 初稿 | 2016/04/02 |

## 引言

本次测试，尝试使用虚拟的精确立方体点云模型[[1]](#footnote-1)，作为场景中初始基准点云，将后续真实深度图与之融合，生成新的重建结果，与之前无虚拟立方体做参考的重建结果对比精度。

具体的**测试目标**为：

1. 使用整个虚拟立方体点云做初始化，将重建结果与之前对比；
2. 分别提取立方体点云轮廓（精确），真实深度图点云轮廓（自动粗提），单独将两个轮廓建立对应关系，作为目标函数的优化项（带权重），如公式(1) 所示；

(1)

其中，K1为**整个点云**上的对应点对集合，K2为仅**轮廓**上对应点对集合，w为轮廓权重，按论文[2]设定默认值w=4。

1. 在真实数据点云上，改自动粗提轮廓为手动标记轮廓，与b对比重建结果；
2. 在真实数据点云，尝试拟合立方体各个面，得到立方体参数化模型顶点、棱边（轮廓）、面，与b、c对比；

【注】目前已实现a、b；尚未实现c、d //2016-4-3 16:23:08

## 测试步骤

### 合成虚拟立方体点云，并调整姿态

1. 在已采集的数据[2]上，使用我们实现的论文[1]程序，输出一个完整的重建点云A；

【注】目前使用的点云为**res-pcc-m2\00f80c-cloud.ply**，命令行参数为：//2016-4-2 15:37:23

|  |
| --- |
| -dbg1 --depth-intrinsics 579.267,585.016,311.056,242.254 -cc\_norm\_prev **01** -icp\_impl "sgf\_cpu" -png-dir "D:\Users\zhangxaochen\Documents\geo-woReg\00f80c-raw\_frames" |

1. 按我们采集数据的尺寸规格[2]，生成立方体点云B（30.5\*28.0\*25.0 ），点云密度为间隔**5mm**；
2. 加载点云A，以其为参照，**手动调整**点云B位姿 ，使其与点云A中真实立方体大致重合，如图1所示。

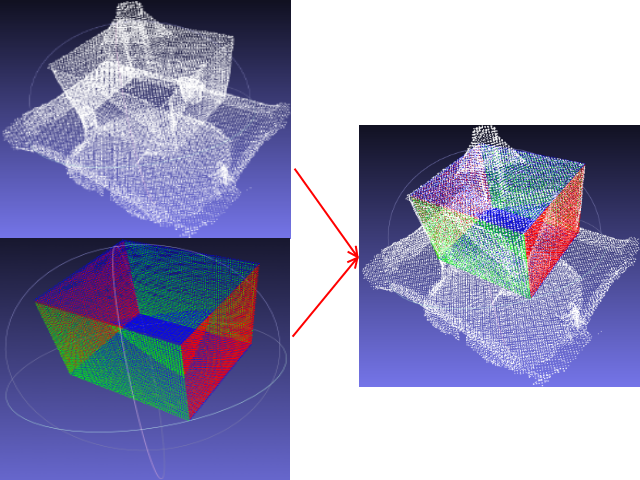


图 1 合成虚拟点云B并调整位姿，使其与点云A中真实立方体大致重合

### 修改pcl\_kinfu\_app，加载虚拟点云，初始化TSDF模型

* **具体实现步骤：**

1. 增加命令行参数 -synF0 <\*.pcd | \*.ply>, 预加载虚拟点云文件；
2. KinectFusion的TSDF模型生成算法，采用时序的移动平均（MA）策略。我们增加命令行参数 –synF0w <weight>, 设定预加载的点云转化为TSDF时的时间权重weight；
3. 点云生成TSDF模型，策略有两个：
   1. 使用立方体点云（PointCloud）生成，体素值求解步骤见附录A。

//目前暂采用的方案 //2016-4-2 22:45:23

优点: 无需遍历TSDF每个体素, 只关注点云每个点对应体素的26邻域(=33-1)，效率较高；若点云密度足够高，可能精度足够?

缺点: 精度受制于点云密度, 如果点云点间距大于体素尺寸, 生成的TSDF可能有孔洞;

* 1. 使用参数化立方体模型（六个参数面）生成，体素值求解步骤见附录B。

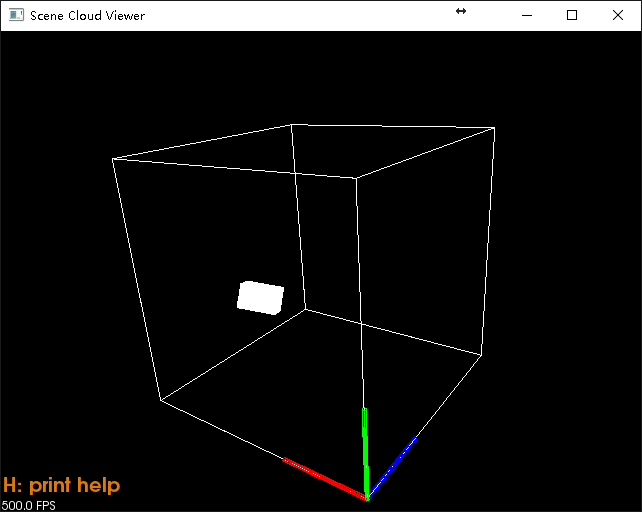
优点: 立方体TSDF精度（即分辨率）可以“无限”高（仅受制于浮点运算精度），体素尺寸可以很小；

缺点: 精度过高没用，因为后续真实深度图精度不够；需要遍历每个体素，效率较低；

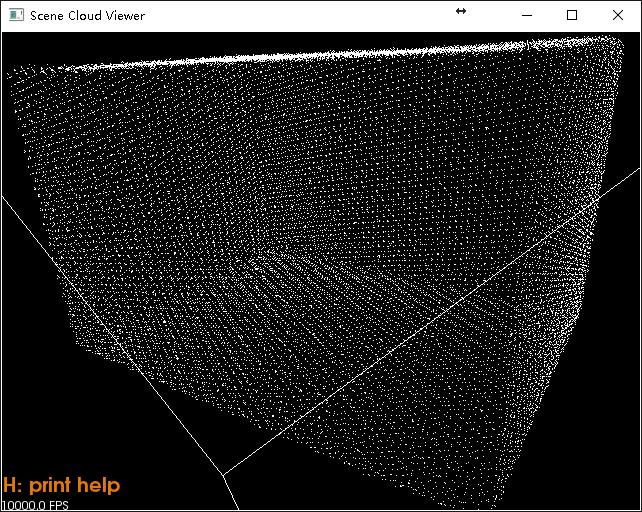
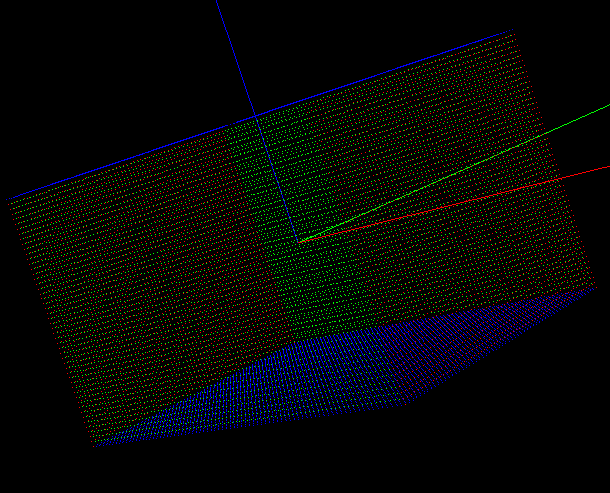
【注】两个策略都使用立方体质心做“虚拟视点”，视线为由内向外“看”。其产生的TSDF，与真实深度图（视点总在点云之外）生成的TSDF进行融合的误差大小暂无定量评估。 //2016-4-2 23:01:41

//目前想到粗评估方法：仅将立方体生成的TSDF重新转为点云（fetchCloud），观察立方体误差缺陷。 //2016-4-2 23:10:11

目前在GPU上采用策略①正确实现无组织点云整合到TSDF模型，效果如图2所示。注意到图2(a)(b)显示的立方体点云，是整合到TSDF模型，然后再重新“投射”（非透视）为点云的。“投射”得到的点云，与原始生成的合成点云相比，较为粗糙，噪声增大了。其原因在于转到TSDF模型的过程相当于对数据进行压缩，导致轻微的失真。 //**2016-4-16 22:50:13**



(a)

(b) (c)

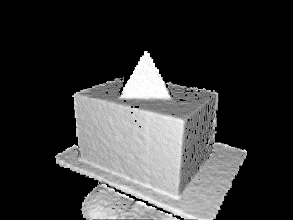
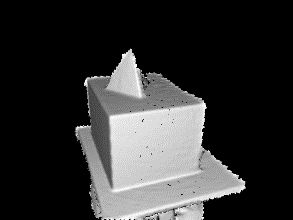
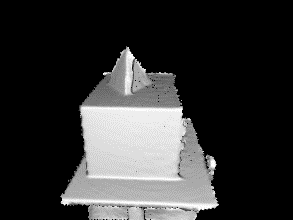
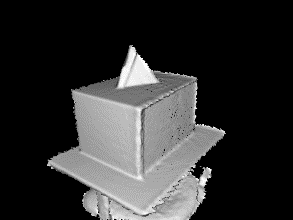
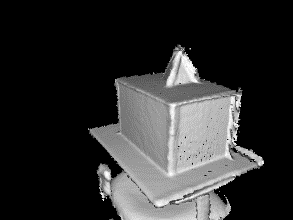
图 2 (a) 虚拟立方体在整个体空间中的姿态(按真实数据中立方体位姿摆放)；(b)TSDF重新投射得到的立方体点云；(c)原始虚拟立方体点云，与(b)相比，当视线位于一个平面内（如顶面）时，此平面视觉上变为一条单像素的线段，而图(b)因为TSDF精度不够，重新投射的立方体顶面在视野中变“粗”了

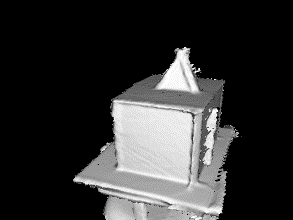
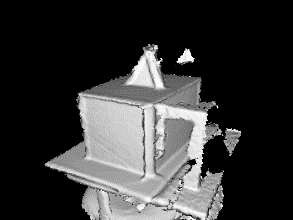
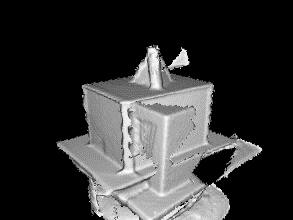
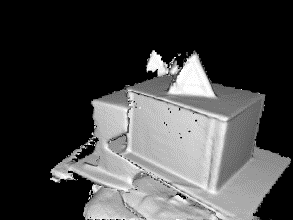
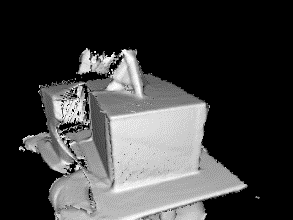
### 将真实数据与预置到TSDF的虚拟立方体进行配准

在真实数据配准之前，将虚拟立方体点云预置生成到TSDF模型中，作为后续配准的参照，发现对原来kinfu算法确实有积极影响。

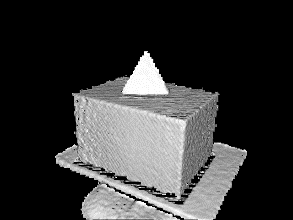
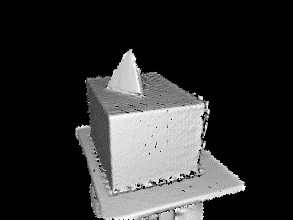
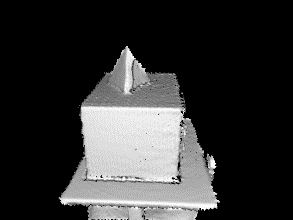
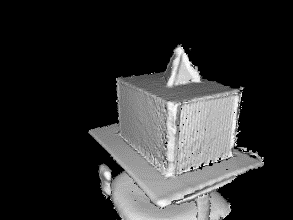
图3为在测试数据 00f80c399上，第0, 49, 67, 93, 151, 172, 175, 184, 211, 237共10帧的重建结果，具体为：

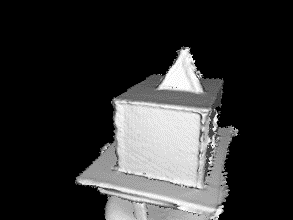
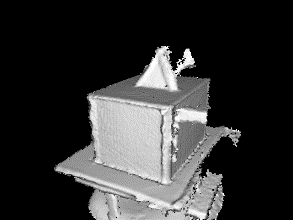
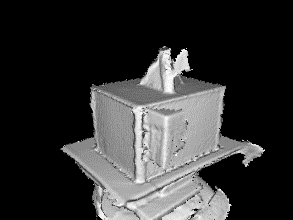
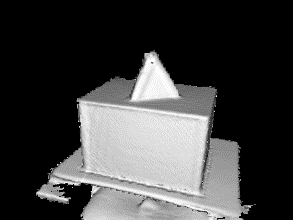
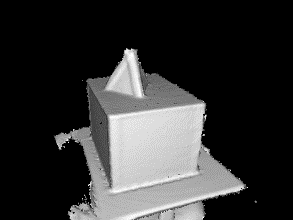
1. 使用原kinfu算法，在175帧产生严重漂移，且不可恢复；
2. 改用虚拟立方体生成预置TSDF，且设定权重w=1时，虽然在175帧也产生漂移（比(a)程度轻），但是在200帧之后又成功恢复轨迹；不过下面基座板上漂移误差的残余还在（左下角）；
3. 使用虚拟立方体生成预置TSDF，且设定权重w=17时，始终未发生严重漂移；

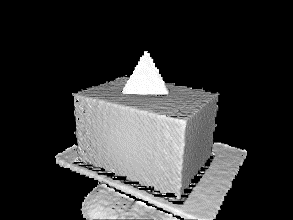
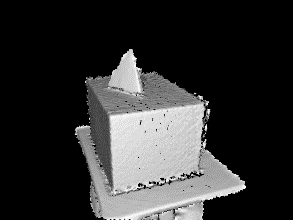
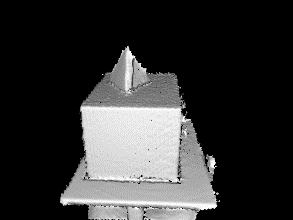
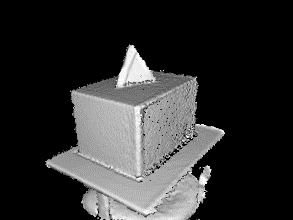
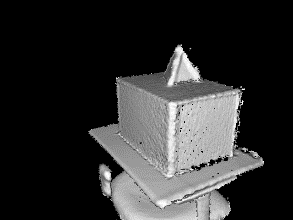
    

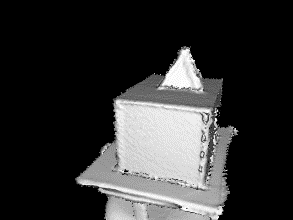
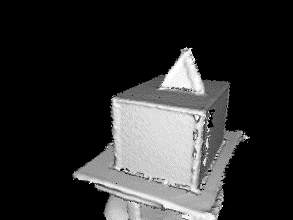
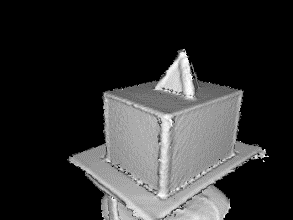
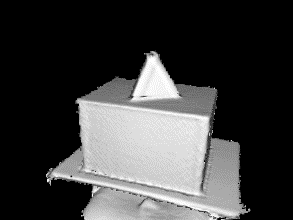
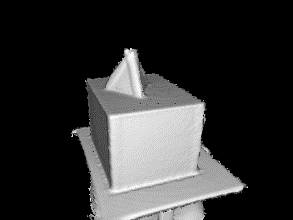
(a)

(b)

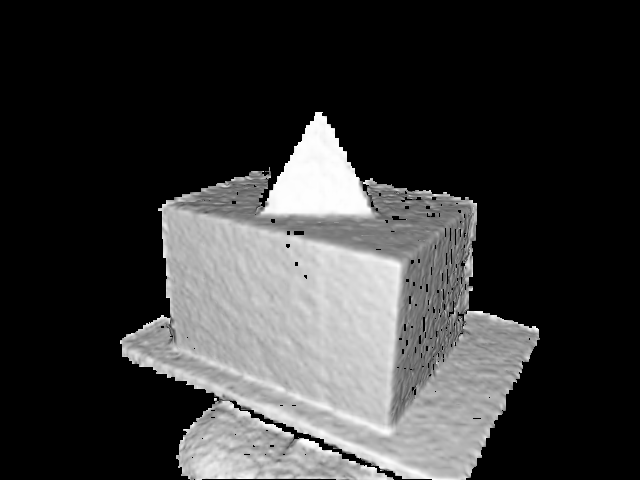
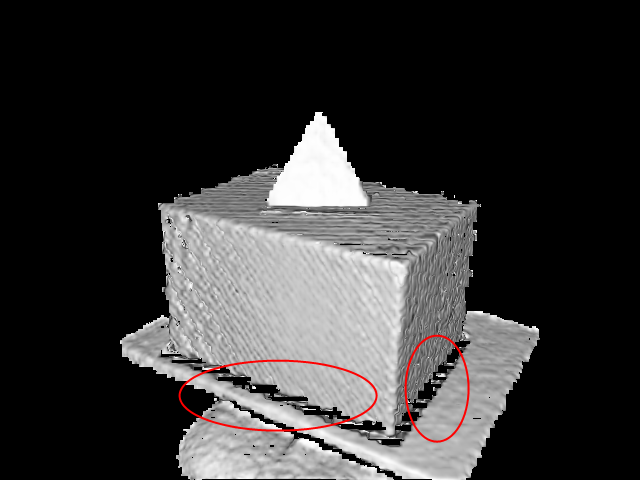
    

(c)

图 3 三种策略（kinfu, 虚拟点云预置到TSDF且权重w=1, 以及w=17）的重建结果

注意到图3(b)(c)中，第0帧与图3(a)相比，立方体下面平板之间存在缝隙（黑影区域），且平面区域也更凹凸不平，如图4所示，其原因是目前真实数据的第0帧与预置的TSDF并没有进一步配准调整，我们只是预处理时手动将虚拟立方体调整到与真实数据第0帧大致重合。

下一步要自动化将第0帧与预置TSDF进行配准，消除这种瑕疵。 //**2016-4-17 10:45:13**

(a)真实数据第0帧 (b)TSDF中预置虚拟立方体后第0帧结果

图 4 预置虚拟立方体到TSDF，且第0帧并不与其配准产生的瑕疵

## 参考文献

[1] Zhou, Qian-Yi, and Vladlen Koltun. "Depth camera tracking with contour cues." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

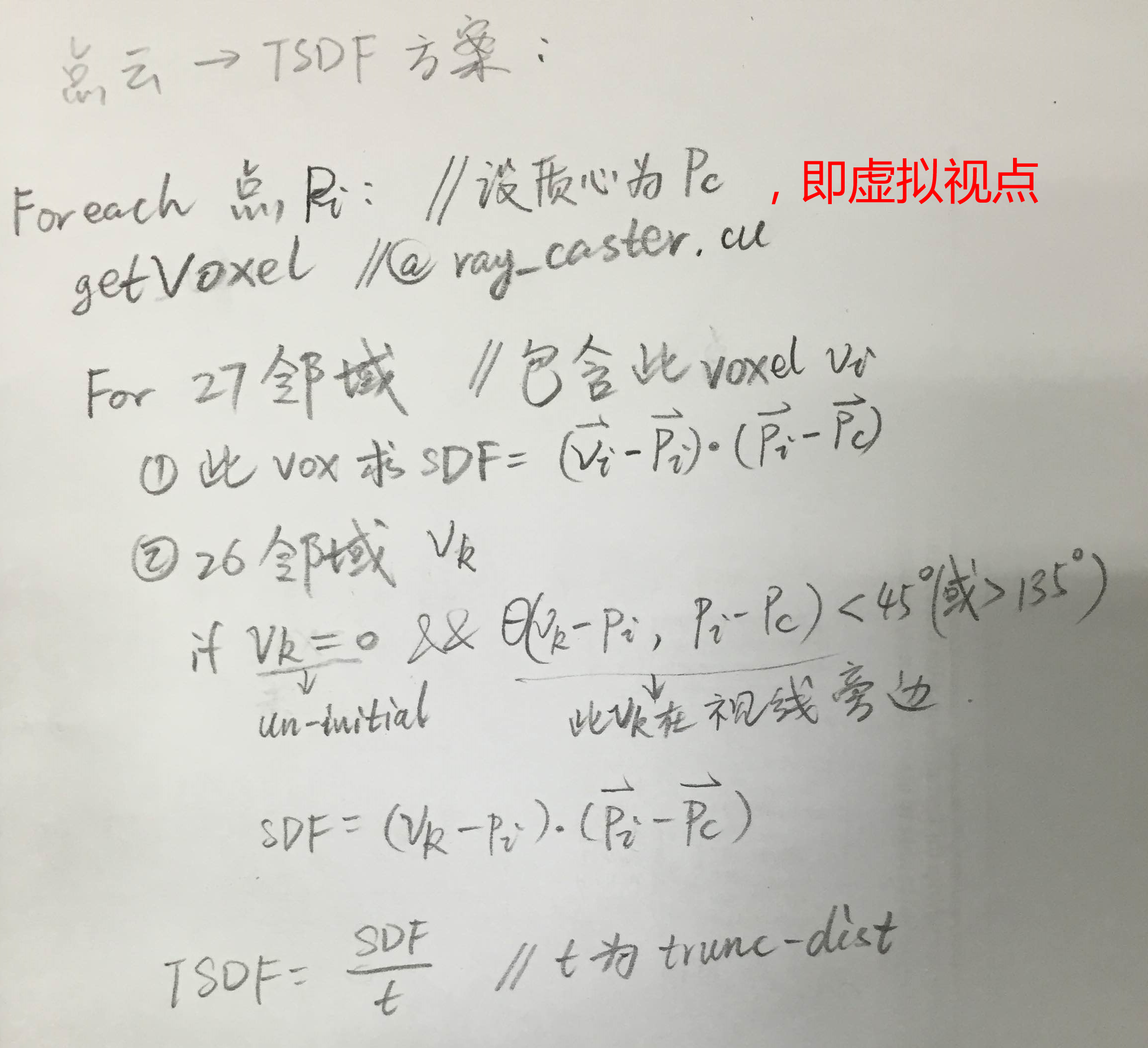
[2] 张琛.《带已知立方体模型的RGBD数据采集报告》

## 附录A 使用无结构(unorganized)点云生成TSDF步骤草稿

所谓PCL点云的“结构”，是指点云数据的 (width, height) 成员域。

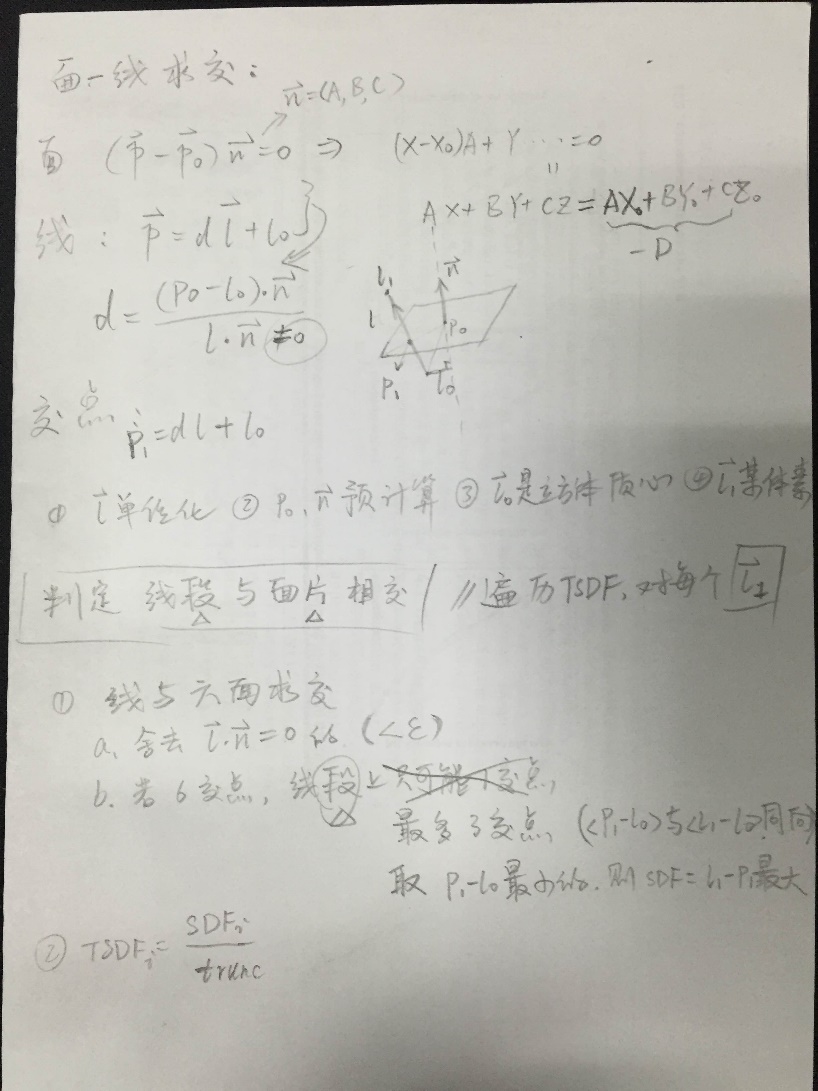
若其为图像宽高，则点云可能由深度图+相机内参生成（也可能是虚拟合成），它与一张深度图一一对应，可以直接转为深度图，若同时知道此点云对应的相机外参，可以直接生成TSDF模型。

若 width=cloudSize; height=1，则点云是无结构的，无法生成深度图（因为相机视角无法确定），不能用现有的办法生成TSDF。这里我使用立方体质心做“虚拟视点”，尝试生成TSDF，算法伪代码如下图：



## 附录B 使用参数化立方体生成TSDF步骤草稿

//暂未实现 //2016-4-2 23:10:57



1. 此“模型”为**点云模型**，并非立方体**参数模型**，其实际精度还受点云密度的影响，并非参数化的绝对精确。 [↑](#footnote-ref-1)