# 虚拟立方体进行KinectFusion的三种策略结果测试报告

## 概述

本报告主要演示三种配准策略下，KinectFusion的重建结果。我们使用虚拟立方体，并修改程序，使得待配准的目标点云（vmap，nmap）改变，进而导致ICP求解相机外参的结果改变。

原KinectFusion算法的目标点云生成过程为：对每一帧，将其融入到TSDF模型后，利用当前相机外参，从TSDF模型用光线投射算法生成一个模拟帧（vmap，nmap）作为下一帧的配准目标。本次测试中，我们仅改变配准目标点云的数据源，所采用的三种配准策略具体分别为：

1. 策略1。在使用真实数据进行配准之前，将虚拟点云通过报告[1]附录A策略生成一个TSDF模型（记为***A***），我们称之为“影子模型”，此模型独立于待生成的真实TSDF模型（记为***B***）。对每一帧，使用当前帧的相机外参，对TSDF ***A***进行光线投射，得到一帧虚拟点云作为目标；
2. 策略2。与策略1不同，不生成TSDF ***A***，自己实现一个消隐算法，对给定的虚拟立方体点云，每一帧使用相机外参解得一张深度图（伪代码如表1所述），进而得到目标点云与法向图。

//策略2为2016-04-18 周一组会商定内容

1. 策略3。与策略1类似，但是是将虚拟立方体点云预先融合到TSDF ***B***中（而不是***A***）。对于每一帧，从***B***中光线投射出点云与法向图作为配准目标。其与策略1的本质区别是，策略3用于生成模拟帧（vmap，nmap）的TSDF是由虚拟立方体与真实深度数据融合产生，而策略1只用虚拟立方体生成TSDF做光线投射。

//策略3为报告[1]的改进，由仅手工将真实场景与虚拟立方体粗配准，改进为第0帧真实数据通过ICP与虚拟立方体产生的TSDF精配准。

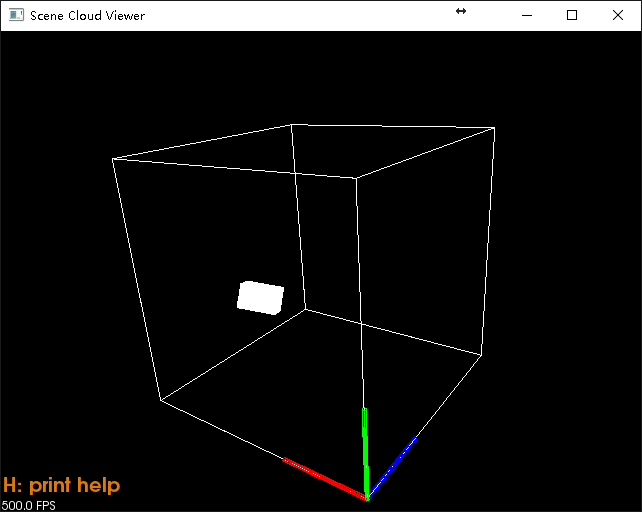
表 1 点云生成深度图的消隐算法流程伪代码

|  |
| --- |
| 对点云中每一个点 pt：  vcam = R’\*(pt-t) //利用相机外参{R,t}转到相机坐标系，物理量纲：（米）  //利用相机内参 intr(cx, cy, fx, fy)，计算图像坐标 (u,v)：  int u = int((vcam.x \* intr.fx) / vcam.z + intr.cx);  int v = int((vcam.y \* intr.fy) / vcam.z + intr.cy);  d = vcam.z \* 1000; // z值转换到毫米深度值  if dmat(u,v)!=0 && d < dmat(u,v) //若对应像素已初始化，且d比像素深度更小，则更新  dmat(u, v) = d; |

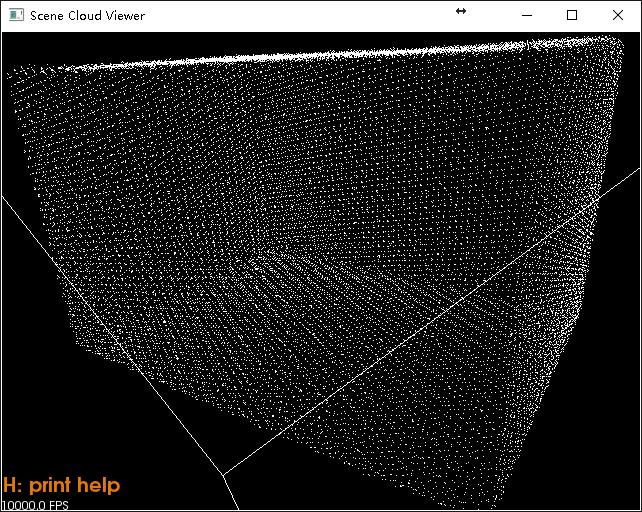
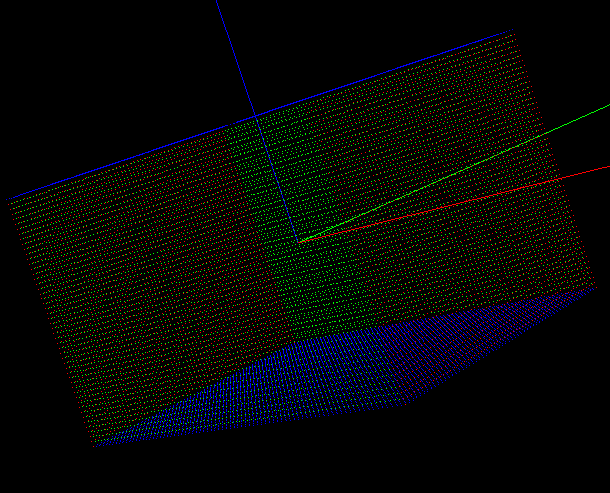
## 流程与结果

### TSDF模型&光线投射

下图1为虚拟立方体融合到TSDF模型中的结果 (来源: 参考报告[1]图2)，图2为TSDF重新光线投射到vmap, nmap 的结果（可视化仍要转为深度图）。

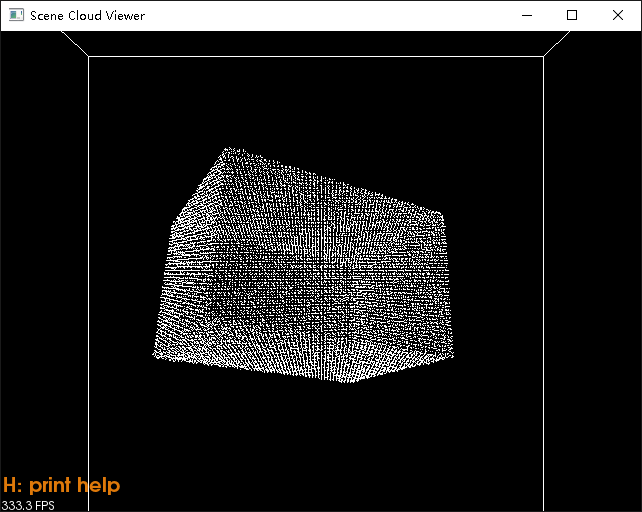
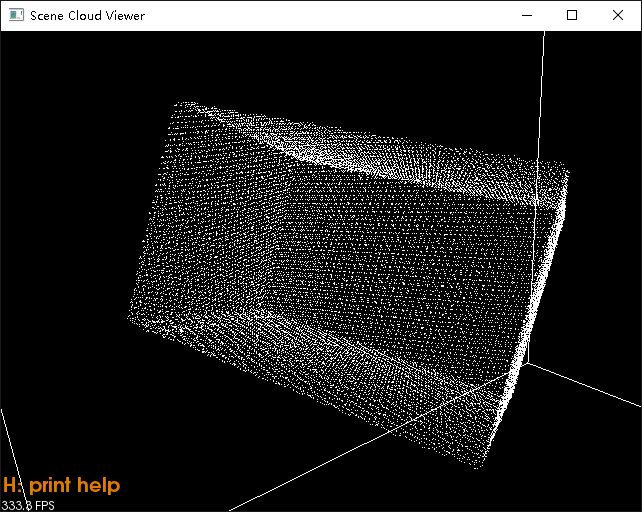
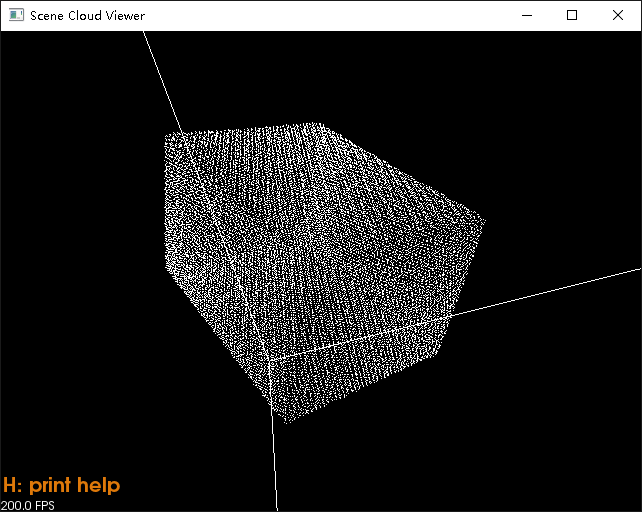
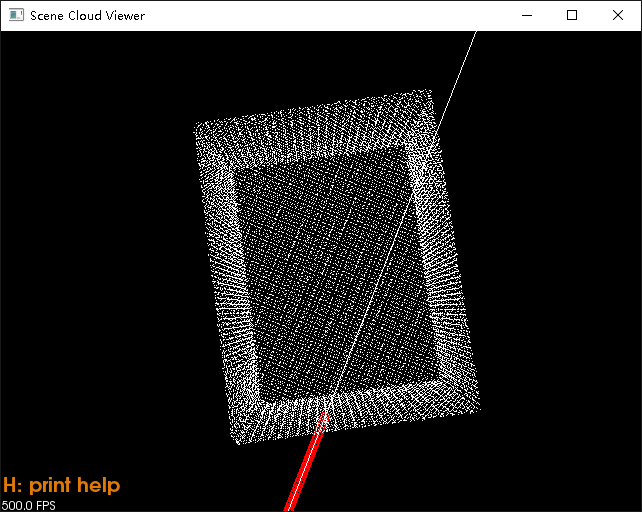
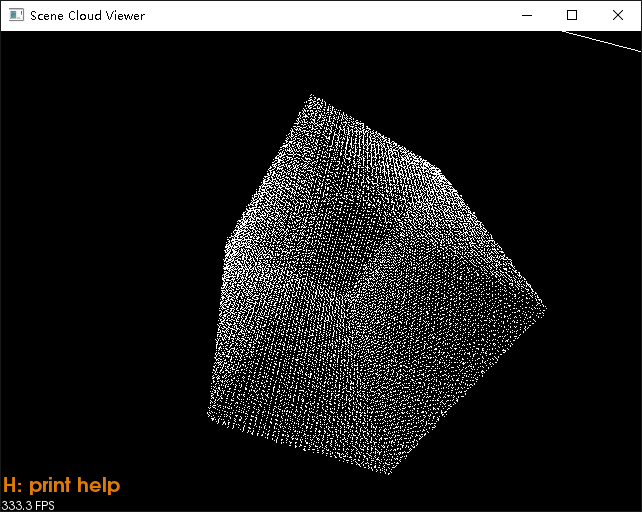


(a)

(b) (c)

图 1 (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

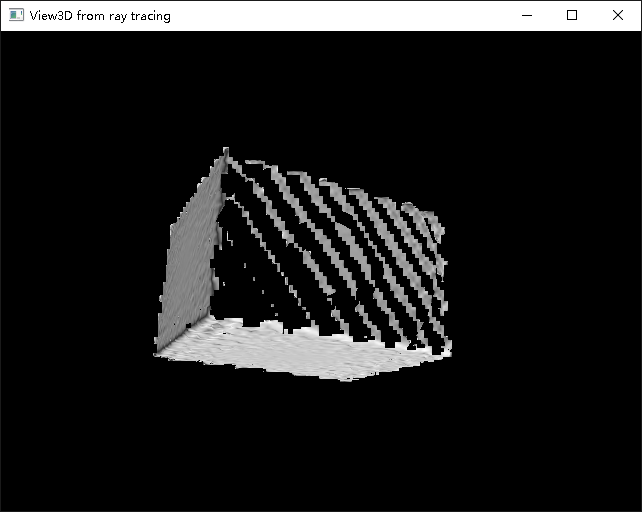
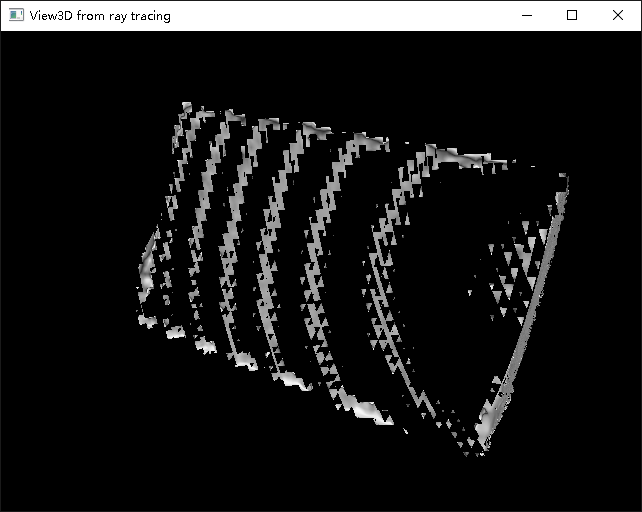
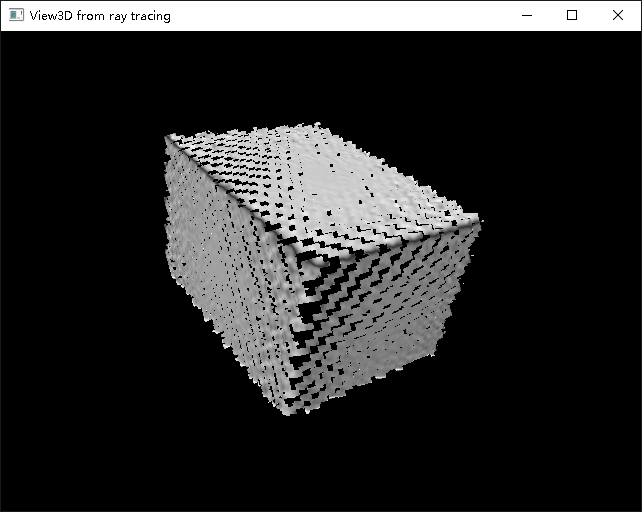
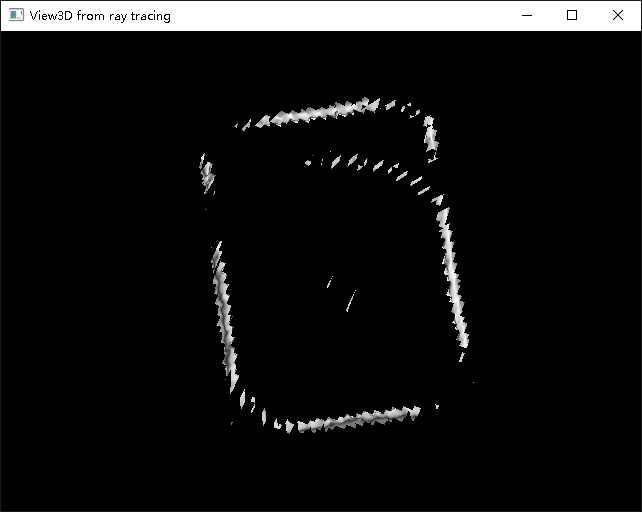
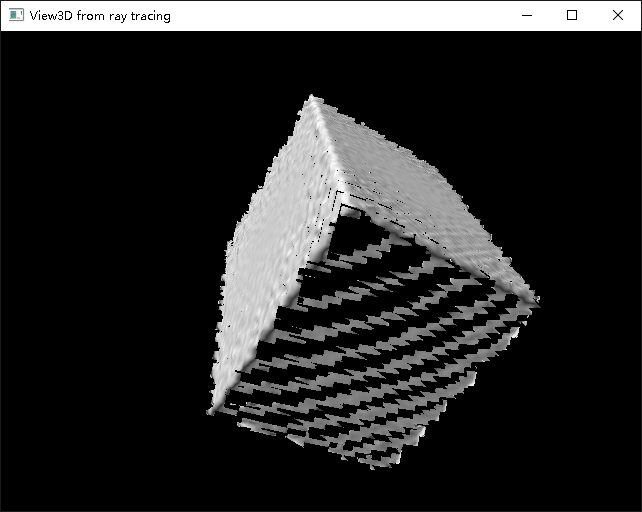
    

图 2 虚拟立方体融入TSDF（第一行）之后，重新光线投射得到vmap、nmap，采用冯氏着色法（phong shading）呈现出来（第二行）

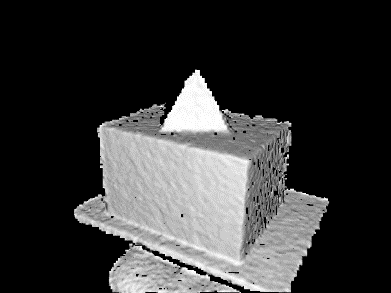
由图2看到，我们通过报告[1]附录A方法生成的TSDF ***A*** 应该正确，但是光线投射得到的 vmap, nmap 并不完美，当视线与曲面斜交时，生成的曲面尚可接受；但是当视线与曲面垂直时，vmap严重缺失。

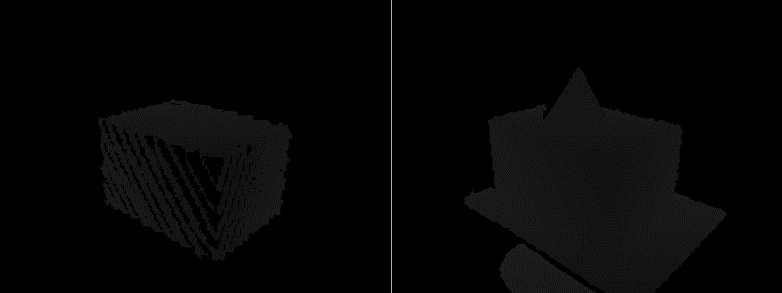
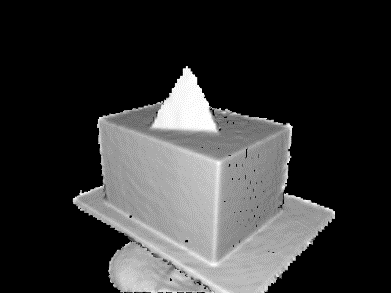
原因暂不确定，可能是生成TSDF的算法用到质心虚拟视点，与真实外部视点相互作用导致？

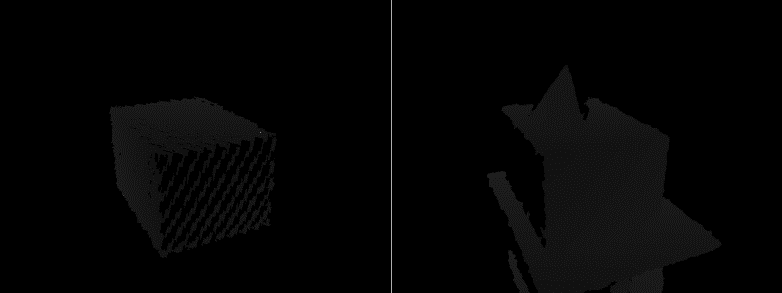
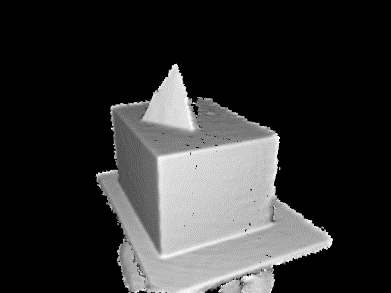
//2016-4-23 20:35:45

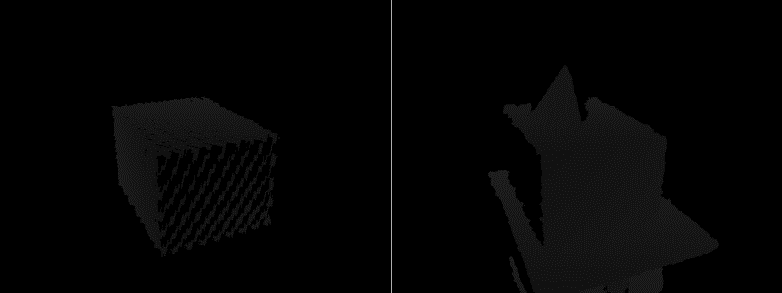
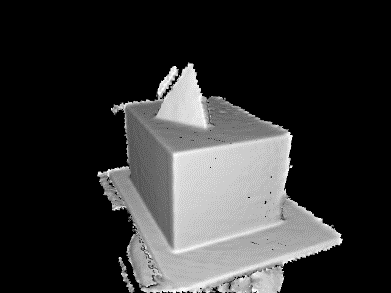
### 策略1结果

下图3为策略1在离线数据 \geo-cube399\00f80c.oni 上**第0, 24, 47, 48, 58帧**的运行结果，每一行从左到右依次是上一帧配准目标dst，当前帧待配准真实深度图src，TSDF模型结果。

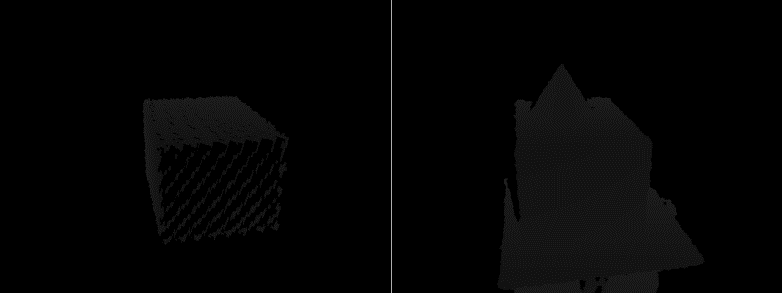
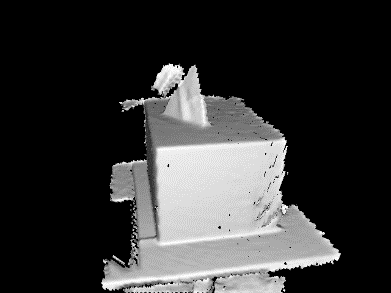
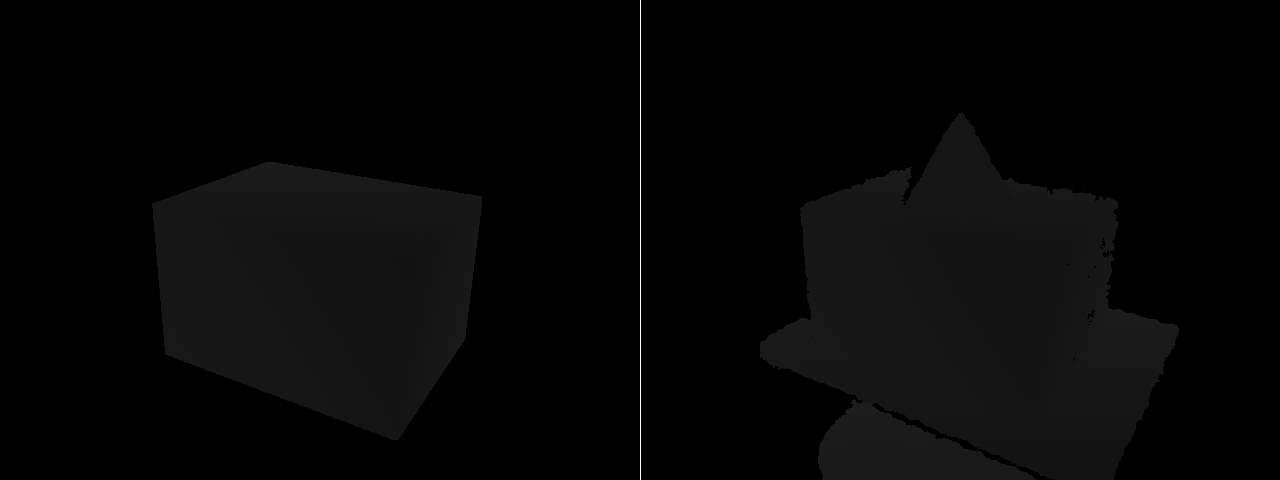
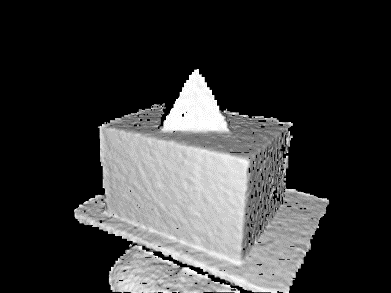
 

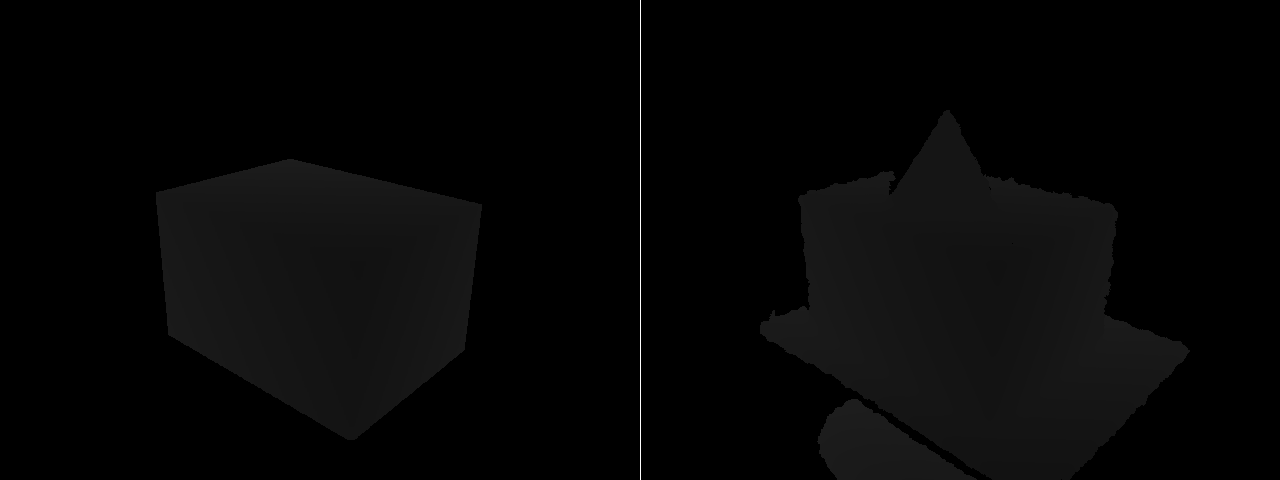
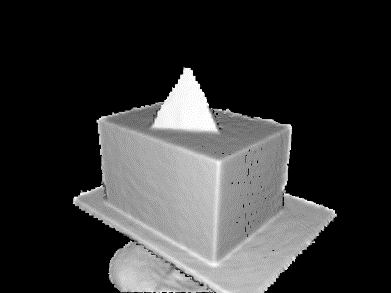
图 3 策略1结果。每一行从左到右依次是上一帧配准目标dst，当前帧待配准真实深度图src，TSDF模型结果

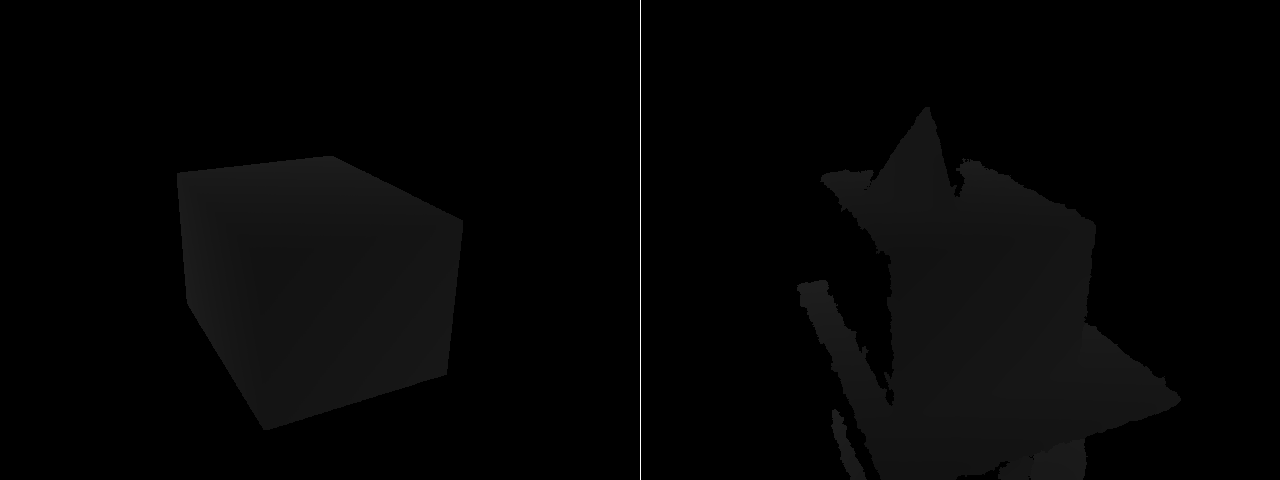
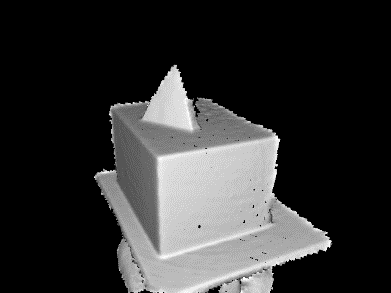
发现一开始外参估计比较准确，但是当立方体由三个可视面旋转到两个可视面时（47~48帧左右），相机立即开始出现漂移，这是因为我们的配准目标只有立方体，而立方体的深度特征在只有两个可视面时，缺乏一个自由度的约束。

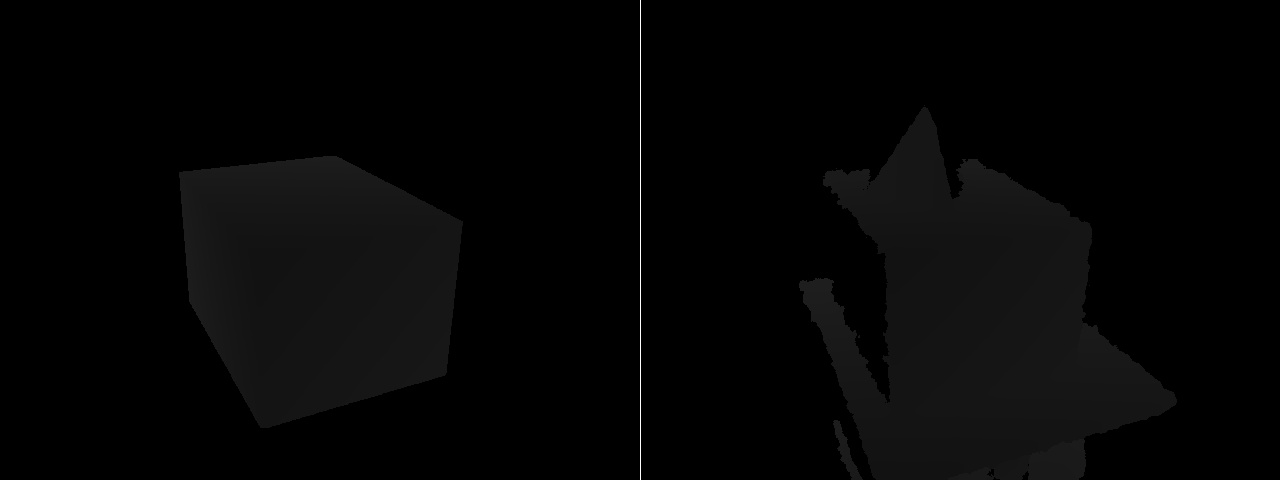
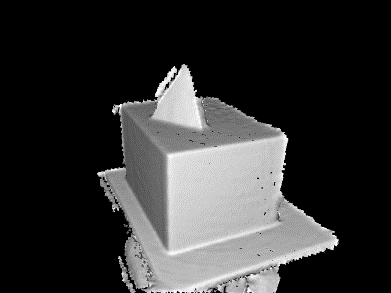
### 策略2结果

策略2采用表1所述的消隐算法，较好的保留了立方体面、边性质，结果如下图4所示（仍取第0, 24, 47, 48, 58帧进行演示）。

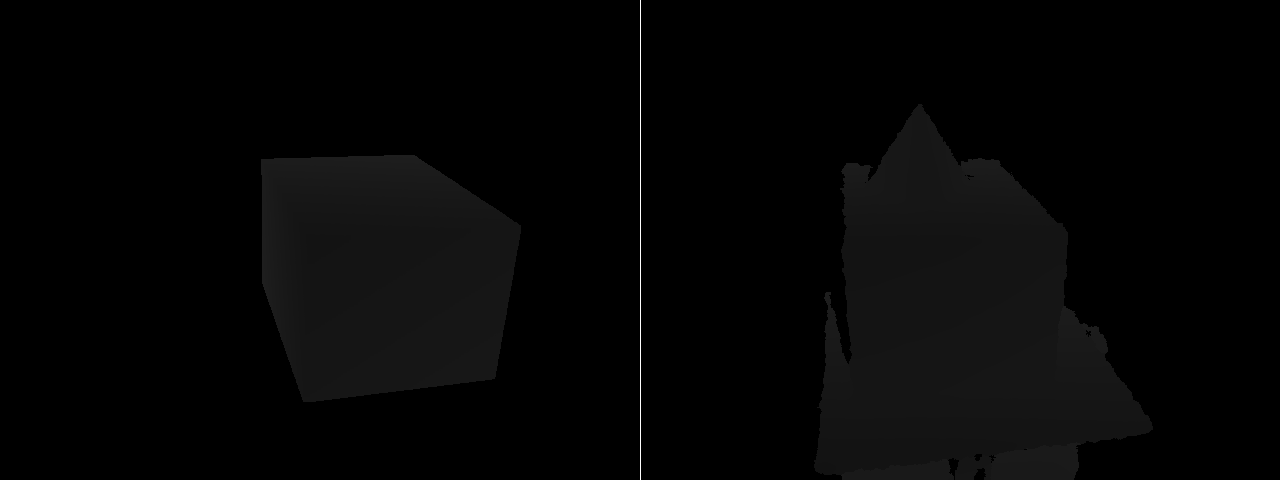
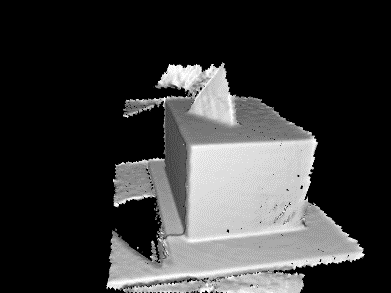
 

图 4 策略2在第0, 24, 47, 48, 58帧的结果。与策略1类似，同样在47~48帧开始漂移

发现尽管配准目标点云更加精确，但是漂移程度却增大，可能原因是，虚拟立方体越精确，表面越光滑，导致约束力越弱，漂移越大。

### 策略3结果

策略3预置虚拟立方体点云到TSDF模型中（非前述“影子模型”），因此存在一个可变量，权重w。预置权重w越大，则后来的真实数据（每帧权重w=1）融合到TSDF中时，对TSDF的改变越慢。

在报告[1]中，我们手动将虚拟立方体与真实数据中立方体对齐，当w=1时，在175帧漂移，但是之后有成功恢复；当w=17时，没有产生大幅漂移。

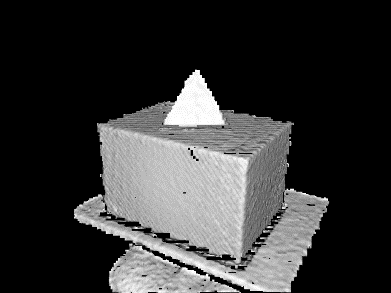
本报告中，在第0帧，将真实数据通过ICP进一步精细对齐到虚拟立方体上，发现之后无论权重w=1或w=17，配准均未在发生漂移。

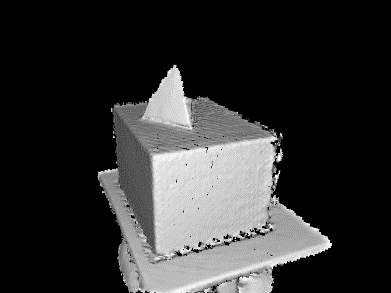
但是与报告[1]相似，初始第0帧TSDF生成结果中，立方体底边缘仍产生无效区域“黑线”，推测也和图2所示的光线投射算法有关，实际产生的点云并无此瑕疵。

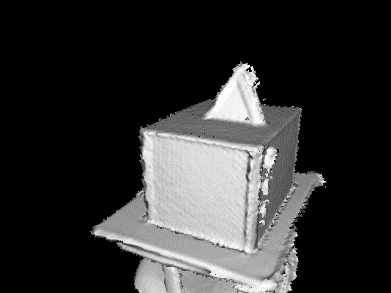
以下展示的是**第0, 48, 176, 177, 200 帧**结果。

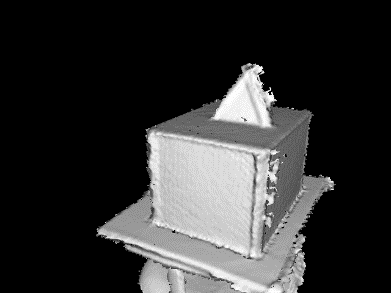
下图5、6中，第一行配准目标dst（最左）为虚拟立方体，之后几行dst为TSDF投射结果，符合我们的预期，说明实现应该是正确的。

#### 预置权重w=1

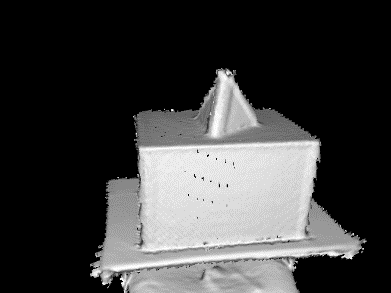
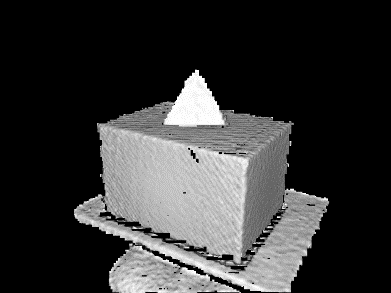
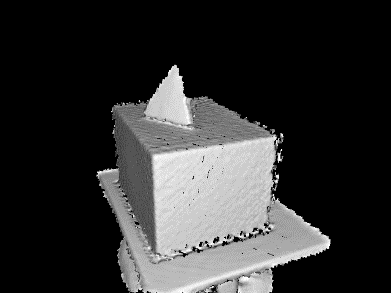
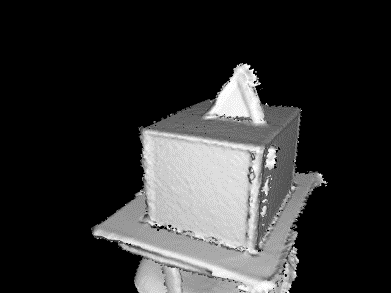
 

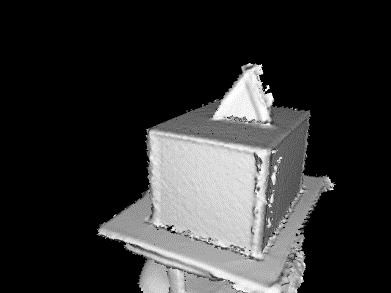
图 5 策略3设定w=1结果，由上到下为第0, 48, 176, 177, 200 帧

#### 预置权重w=17

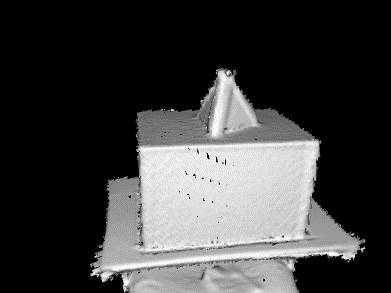
 

图 6 策略3设定w=17结果，由上到下为第0, 48, 176, 177, 200 帧

图5、6的对比可以发现，策略3对于预置虚拟立方体的权重w大小已经不敏感，即使w=1也没有漂移，与w=17结果差异细微。

## 参考文献

[1] 张琛, 《虚拟立方体作第0帧进行KinectFusion测试报告》.