清 华 大 学

综 合 论 文 训 练

题目：交互式点云分割算法研究

系　　别：计算机科学与技术系

专　　业：计算机科学与技术

姓　　名：徐捷

指导教师：张松海副教授

2016 年 6 月 19 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：徐捷　导师签名：张导海　日　期：2016.6.17

中文摘要

点云分割是计算机图形学和计算机视觉领域非常重要的一个问题，在物体检测，物体识别以及场景理解等方面都有着广泛的应用。很多前人的工作尝试用全自动、零交互的方法去对三维点云场景进行语义分割。然而，由于现实场景的复杂性，以及目前市场上的三维扫描仪在点云质量上的局限性，交互式的点云分割是目前能处理各种类型点云的唯一方法。

本文借鉴前人的工作，提出了一个全新的交互式点云分割系统，从而提高分割结果的精确性以及大大减少进行点云分割的时间。为了完成这一目的，本文的主要工作有：

1. 设计并实现了一个全新的交互式点云分割系统；
2. 设计并实现了一种有效的特征面片构建算法；
3. 提出了特征面片间的一种支撑关系，并给出了其计算方法；
4. 在经典的图割算法中加入了对支撑关系的考虑，显著地优化了点云分割结果；
5. 设计并实现了一个轻量级的相似物体搜索算法，从而将已有的分割结果在场景中进行传播，有效地减少了用户的交互数量。

实验结果表明我们的系统能在保证交互的实时性的前提下帮助用户正确且快速地分割各种类型的场景。

关键词：点云分割，交互式系统，语义分割，支撑关系

ABSTRACT

Point cloud segmentation is a fundamental problem in both computer graphics and computer vision, which is important for many subsequent tasks like object detection and recognition, scene understanding, etc. Most previous work attempted to achieve semantic segmentation of 3D point clouds with little or even no user intervention. However, due to the complexity of real-world scenes and the limitations of 3D scanners, interactive segmentation is currently the only way to cope with all kinds of point clouds.

This paper aims to present a novel interactive system for segmenting point cloud scenes, which leads to a precise segmentation result and largely reduces users' time on the segmentation task. To achieve this, this paper has the following main contributions:

1. we design and implement a novel system for interactive segmentation of point cloud,
2. we design and implement an effective representative patch construction algorithm,
3. we introduce a support relations between patches and give a simple method to calculate the probability of support relations,
4. we propose to incorporate support relations into a standard graph-cut based segmentation frame work, which significantly optimize the segmentation result,
5. we design and implement a light-weight method for finding similar objects for segmentation propagation.

The experiments show that our segmentation technique can help the users correctly and quickly segment various types of scenes while guaranteeing the real-time interaction.

Keywords: Point cloud segmentation; Interactive system; Semantic segmentation; Support relations

目 录

主要符号对照表

CAD	计算机辅助设计 (Computer Aided Design)
RANSAC	随机抽样一致 (RANdom Sample Consensus)
ICP	迭代最近点算法 (Iterative Closest Point)
CRFs	条件随机场 (Conditional Random Fields)
SfM	从运动信息中恢复三维场景结构 (Structure from Motion)
χ^2	卡方距离函数

第 1 章　引言

随着相对廉价的深度探测仪 (Microsoft Kinect, Microsoft Kinect 2) 的日益流行，扫描探测自己周围的环境变得相当寻常与简单。伴随着这一现象随之而来的，便是计算机图形学、计算机视觉领域对这些深度探测仪扫描得到的输出结果——RGB-D 图像及点云的更深一步的研究。

本章将首先介绍三维点云的基本概念，然后详细介绍本文研究的重点——三维点云分割 (3D point cloud segmentation) 的基本概念及研究背景，最后介绍本文的主要贡献。

1.1　点云的基本概念

点云是物体 (或场景) 的一种离散表示。从定义上来讲，点云是在某一特定参考坐标系下的一堆离散数据点的集合。

在计算机图形学中，点云通常是通过三维深度扫描仪来获取的，因此我们通常把参考系设置为三维直角坐标系，即点云中的每一个离散数据点都由 X、Y、Z 三个维度来表示。三维深度扫描仪不同于普通的彩色相机，它不仅可以得到当前视角下的彩色图片（红绿蓝三通道值），它还能通过深度传感器得到当前视角下每一个像素的深度信息，从而构造出当前帧的点云。通过把三维深度扫描仪在不同角度下生成的点云进行注册[1,2]，我们便可获得一个较为完整的点云。这些由深度扫描仪得到的点云往往表示了物体以及场景外表面的点集合。

坐标信息是点云中每一个离散点包含的最基本的信息。在此基础上，还可以通过记录每个点的颜色来记录点云的颜色信息。这些通过扫描得到的三维点云有着非常广泛的用途，例如零部件的 CAD 模型的建立以及物体场景的动画、渲染等虚拟可视化应用。

虽然点云能被直接进行渲染[3]，但由于点云存在密度、零体积等问题，点云模型 (见图 1.1(a)) 经常会被转化成其他的一些三维模型的表示，例如多边形网格模型（三角网格模型）(见图 1.1(c)) 以及体素化模型 (见图 1.1(b))。有很多算法致力于解决各类三维模型间的转化，比如用 Delaunay triangulation 可以完成从点云向三角网格模型的转化。

经典的点云相关操作的算法库为 PCL（Point Cloud Library）[4]，该算法库于 2011 年建立，其中收录了各类点云处理的最新算法。

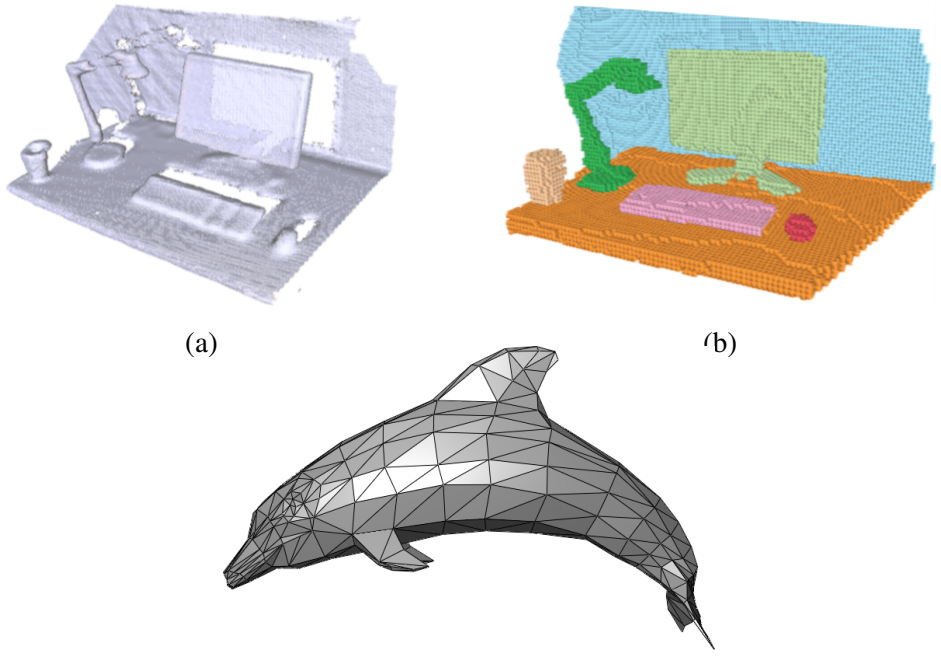


图 1.1　不同的三维模型表示: (a) 点云模型 (b) 体素化模型 (c) 多边形网格模型，摘自 [5]

1.2　点云分割的基本概念

点云分割是点云处理中极为重要的一个步骤，其目的是把原始点云分割成若干个互不相交的集合，并且每一个集合中的点都代表了相同语义信息。对于点云分割一般性的表述为：输入点云 $\mathcal{P}$，把 $\mathcal{P}$ 拆成若干个子集合 $\mathcal{P}_i$，使得每个 $\mathcal{P}_i$ 都代表了一个语义块，且 $\mathcal{P} = \bigcup_{i=1..k} \mathcal{P}_i$，以及 $\mathcal{P}_i \bigcap \mathcal{P}_j = \varnothing (i \neq j)$。(见图 1.2，图 1.3)

现在已经有很多算法来实现点云分割，A. Nguyen 等人 [6] 对各类算法进行了一个总结。目前的点云分割算法主要集中在以下几个类别：基于边缘的方法 [7]，基于区域的方法 [8]，基于特征的方法 [9]，基于模型的方法 [10]，基于图的方法 [11]。下面简单介绍一下这一些方法。

边缘信息往往刻画了一个物体的形状，因此就有了基于边缘的方法。此类方法的核心在于找到不同语义区域的边界点，然后根据检测出来的边缘来对点

云进行分割。

基于区域的方法是通过一些类似于聚类的方法，把具有相似局部性质的联通点聚到同一个集合中。这类算法主要有两个分支，一类是基于种子点来扩展出区域的（自下而上），另一类是通过不断分裂大区域来得到最终的分割结果（自上而下）。这类算法往往比基于边缘的方法对于噪声有更强的鲁棒性，但是常常会由于无法合理地对相似性质进行定义而导致了欠分割（under segmentation）或者过分割（over segmentation）的情况。例如最经典的区域生长算法（Region Growing）[8]，其能把点云以平面为标准分割成若干个点集，每个点集都代表了一个平面。

一个比较类似的方法是基于特征的方法，它首先会计算每一个点的特征描述符，然后根据特征描述符来对点云进行聚类，从而得到分割结果。这类方法不同于基于区域的方法，它不要求同一类中的点的连通性，并且它往往会把场景中所有相似的物体找出来，因为相似的物体常常具有详细的特征描述符。近年一篇由 M. Oliver 提出的论文中[9] 的方法就是属于基于特征的算法。这类算法的难点在于如何能设计出一个能普适大部分物体的特征描述符，以及如何计算特征空间中两点的距离。这类方法的分割结果见图 1.3(a)。

现在网络上三维模型库的逐渐健全，这也大大增加了研究者对基于模型的方法的兴趣。基于模型的方法使用一些简单的集合形状或者已有的三维模型，对点云中的物体进行检测、定位，以及分割。这类方法中最经典的算法就是 RANSAC 算法[10]，RANSAC 在检测直线、平面、圆等基本图形的任务重有着非常鲁棒的结果。目前很多的新算法也都是基于 RANSAC 算法衍生而来的。近年来也有很多基于模型库的点云分割算法，例如 Y. Li 提出的算法[12]。这类算法的优点就是在分割点云的同时能根据模型库的信息而得到每个集合的语义信息，缺点是这类算法会很大程度上受模型库的局限性的影响。这类方法的分割结果见图 1.2。

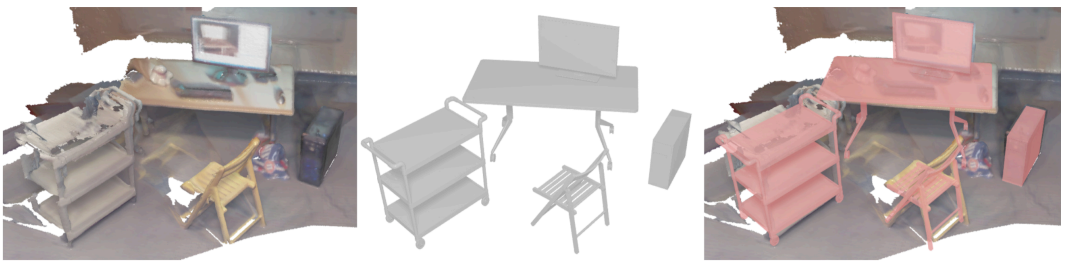
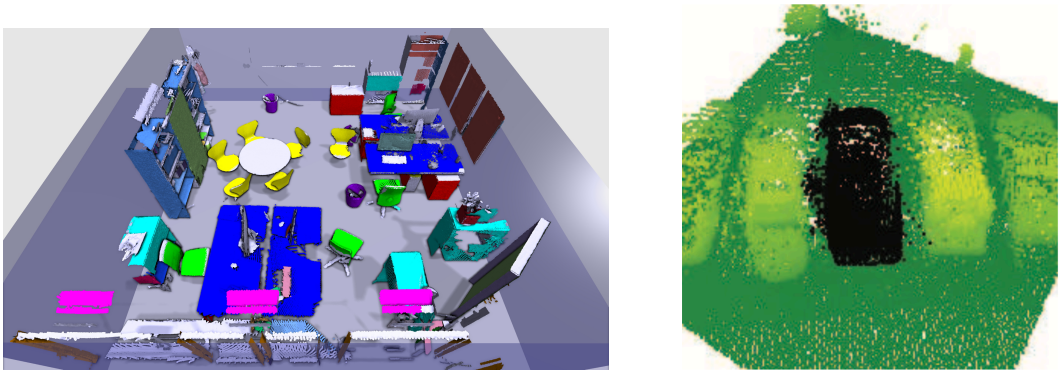


图 1.2　基于模型的方法分割结果，摘自 [12]

基于图的方法把整个点云当做是一个图，从而把点云的分割问题转化成了图的分割问题。这类问题最大的挑战性在于如何对于点云进行连边、赋权值，从而构建出图。对于图的分割问题一个经典的方法就是图割算法（Graph-Cut）[13,14]，之后会在3.3节中进行详细的介绍。最简单的一个构图方法就是对每一个点找空间中最近的 K 个点进行连接，例如 A. Golovinskiy 提出的方法[11]。这类方法的分割结果见图 1.3(b)。



(a) (b)

图 1.3 (a) 基于特征的方法分割结果, (b) 基于图的方法分割结果，摘自 [9,11]

1.3 点云分割的应用

对点云进行分割对点云的语义理解及分析和修改都有着重要的作用，因而在很多场合都有应用，例如物体检测与识别、场景分析、点云修复、点云编辑以及模型库补充等方面。

通过对场景的点云进行分割，可以更容易地让计算机对场景进行理解，例如检测场景中物体，并进行物体识别，以及分析场景中物体的位置关系从而推测出每个物体的功能和作用。当我们把点云中的物体都分离出来以后，可以很容易地使用计算机视觉、计算机图形学中的一些方法对分离出的物体进行分类或是在模型库中进行检索。

除此之外，点云修复与编辑也是点云分割的一个重要的应用。因为大多数三维深度探测仪捕捉的点云质量非常低，再加上在全局地采用特征点匹配算法，注册多帧的点云时出现的误差以及在拍摄时相机内部参数的细微变化形成的相机畸变，都会导致最终得到的点云无法满足进一步处理的需求。其次，三维深度

探测仪往往无法拍摄到物体所有角度的点云信息，因此会导致最终的点云的残缺与不完整。介于以上两种原因，对于原始点云的修复与编辑操作就变得非常重要。对于原始点云的修复，最经典的方法是采用 ICP（Iterative Closest Point）[15] 迭代最近点算法来优化每一帧点云在作刚体变换（Rigid Transformation）时的变换矩阵，从而点云更加符合真实场景。然而由于相机的一些原因，可能会导致得到的每一帧的点云并不能完全通过刚体变换转换到同一世界坐标系中，因此需要对点云进行非刚体注册（Non-Rigid Registration）[16]，而我们把每一个物体都分割出来，并对每一个物体进行 ICP 算法来进行优化，便是对非刚体变换的一种近似过程，它能帮助我们得到更高质量的点云。对于点云的编辑操作，也是一项非常有意义的工作，通过点云的分割，我们可以单独的得到每一个物体，能更有效地计算出每一个物体的几何信息，从而使得用户能够更方便、更准确地对每一个物体进行补全与修改，也可以对整个场景中物体进行重新排列与替换，从而合成出更多不同的场景，这样的合成操作有着非常广泛的应用和研究价值。

最后，对于点云进行分割可以帮助我们进一步的补充三维模型库。虽然随着数字设备的发展，网络上的二维图片数量已经以指数的速度进行爆炸式地增长，然而三维模型的数量相对而言却少之又少，其归根到底是因为三维模型采集过程复杂、合成复杂度高，并且对于采集得到的场景分割及标注工作繁琐和缓慢。因此，通过提高点云分割的效率和准确度，可以有效的减少对于三维数据的标注工作，从而更容易地扩充当前的三维模型库。

1.4 本文的主要贡献

基于点云分割的重要意义以及上述现有的点云分割算法在鲁棒性、准确性和交互性上的问题，本文设计并实现了一个全新的交互式点云分割系统，该系统能更好地满足用户对复杂的点云的分割需求。本文的主要贡献如下：

- 设计并实现了一个全新的交互式点云分割系统，给系统能在保证交互实时性的基础上对各类场景进行正确且快速的分割。
- 基于前人的工作，本文提出了一种改进的特征面片构建算法，该算法结合了点云中的几何信息以及颜色信息，在尽量不造成欠分割 (under segmentation) 的情况的前提下减少特征面片的数量，从而在实现对原始点云实现过分割 (over segmentation) 的同时为之后的实时分割算法中的面片数量提供

复杂度保证。

- 本文引入了一种特征面片间的支撑关系，并提出了两个特征面片间存在支撑关系的概率的计算方法，此方法对现实生活中常见的几类支撑关系均非常有效。
- 本文在经典的 Graph-Cut 算法中加入了对特征面片间支撑关系的考虑，显著地优化了 Graph-Cut 算法的分割结果，并且大大减少了用户的交互数量。
- 本文设计了一个基于已有分割结果的相似物体搜索算法，该算法非常轻量级地实现了对已有的分割结果在整个场景中有效的语义传播，从而有效避免了用户在相似物体上重复的交互工作。

第 2 章　相关工作

三维点云与 RGB-D 图片的语义分割问题是计算机图形学、计算机视觉以及机器人领域中一个被广泛研究的课题。在1.2节中，我们已经根据分割的方法将目前的算法分成了五大类。为了更进一步的讨论点云分割中交互的重要性，本节中，我们将把已有的点云分割工作分成三大类：有监督的算法，无监督的算法，以及交互式的算法。

2.1　有监督的算法

随着免费的 3D 数据库逐渐出现 (例如 NYU Depth Dataset[17]，SUN3D dataset[18] 和 ShapeNet[19])，基于监督性学习的点云分割算法常常能从有标注的数据集中学习到非常精确、可靠的语义信息，并利用这些学习到的知识去帮助系统从复杂的场景中检测、识别和分割出包含物体的区域。它们的训练数据主要来自于两种形式：CAD 模型和 RGB-D 图片。高质量的 CAD 模型是最理想的训练数据，因为它们提供了关于物体完整的三维几何形状[20,21] 或者提供完整的上下文关系信息[22]，这些完整的几何、语义信息能够帮助算法达到非常精确的分割效果。然而，高质量的 CAD 模型的数量和种类远不及现实场景中的物体，从而大大局限了这类算法的应用场景。与此同时，制作高质量 CAD 模型的成本和代价非常大，因此短时间内并不会出现一个涵盖大多数物体的 CAD 模型库。相比于高质量的 CAD 模型，RGB-D 图片的采集和获取就显得容易许多，目前很多算法[5,23,24] 尝试从带标注的 RGB-D 图像中学习各种物体的特征信息。然热，尽管这类数据非常容易采集，但是 RGB-D 图片提供的仅仅是 2.5D 信息，它表示的物体缺少了从三维空间中一些角度下的信息，从而影响特征的提取准确性和可行性。除此之外，对 RGB-D 图片中物体的标注工作依旧是一个的体力劳动，如何提高对这类数据进行标注的效率也成为了这类算法的一个瓶颈。本文提出的交互式系统可以对点云场景进行分割，从而实现对点云的高质量标注。

2.2 无监督的算法

无监督的语义分割算法常常依赖于从输入的数据本身观察得到的一些常见的结构特征 (如对称性、重复性)，这类算法尤其适用于分割室外的建筑物[25,26]。然而室内场景的结构往往比室外场景复杂得多。为了简化室内场景分割问题，之前的工作[9,27] 着眼于那些很大的室内场景，然而这些场景中仅仅包含了有限种类的物体 (如办公桌、办公椅、显示器等等)，并且每种物体都重复很多次。然而，现实生活中的场景往往会包含各种各样的物品，并且不一定会重复出现。因此，在点云分割中加入适当的人为干预来优化这些自动分割好的结果依然是必不可少的。

2.3 交互式点云分割算法

由于全自动化的点云分割算法还远远没有达到理想的效果，在实践中，大家大多还是采用了交互式的方法来分割室内场景的点云[28] 和室外场景的点云[29]。最常见的交互方式是让用户分别指定一些属于前景和背景的区域，然后通过构建一个概率模型 (例如，条件随机场 CRFs) 并用 Graph Cut 对函数进行优化[11,30]，或简单地使用这些指定的区域作为初始种子执行区域增长算法 (Region Growing)[31,32]。然而这些方法的交互方式及构建的概率模型并没有达到理想的状态。例如 [29] 是直接在 RGB-D 帧上进行区域的标识，用户无法自行选择适当的视角，并且由于画图是在二维上，而结果显示是在三维上，用户会在两个维度的切换上产生不便，从而使得用户在交互上花费过多的时间和精力。又例如 [11]，该方法只适用于对室外场景里中大型物体的分割，并且该算法的概率模型只考虑水平方向上的距离，因此它只能对在竖直方向上没有和其他物体相交的物体进行分割。

第 3 章　预备知识

本章将介绍一些之后会在本文中涉及到的基本算法，同时也是点云处理领域非常著名的一些经典算法。准确地了解这些经典算法对更好地理解本文的主要内容有着非常大的帮助，故在本章对这些预备知识进行一定的详述。这些算法依次是基于 SfM 的点云配准及注册算法，用于三维点云分割的 Region Rrow 算法，求解能量函数最小化的图割 (Graph Cut) 算法。

3.1　基于 SfM 的点云配准及注册算法

SfM(Structure from Motion) 是计算机图形学和计算机视觉领域一类重要技术，即从二维的运动信息中恢复三维场景的结构，从而重建出三维场景以及重构出相机的内部和外部参数。目前有很多不同的算法用来从 RGB-D 图像序列恢复出场景的原始点云，但其总体的算法流程都是相类似的。具体的来讲，这类算法的核心就是计算出相机位置在每一帧图像中相对统一的世界坐标系的变换矩阵，一般都是先对整体的场景进行一个大致的配准 (Alignment)，即把每一帧都大致对应到统一的世界坐标系中，然后用 ICP 或者 Loop-Closure 这类算法对点云进行准确的注册。各个算法对场景的配准流程大致也都比较统一，大都是先计算出每一帧的特征描述子 (feature descriptor)，然后利用特征描述子间的度量函数得到相邻帧的特征对应点 (feature correspondence)，从而计算出相邻两帧间的相机运动关系。不同的算法一般会在特征描述子或者度量函数上有一些不同。在配准方面，这里将介绍的点云配准方法是 J. Xiao 提出 SUN3D 数据库[18] 时采用的算法。在点云的精确注册上，这里讲介绍经典的 ICP 迭代最近点算法。

3.1.1　单帧点云生成

本节假定采用了 Kinect 作为深度采集仪器，而 Kinect 采集得到的是每一个角度下的 RGB-D 单帧图像，我们需要通过 RGB-D 图像生成在当前相机坐标系下的单帧点云。这里涉及到图像坐标系和相机坐标系之间的转换，图 3.1展示了二维图像坐标系和三维的相机坐标系之间的关系。

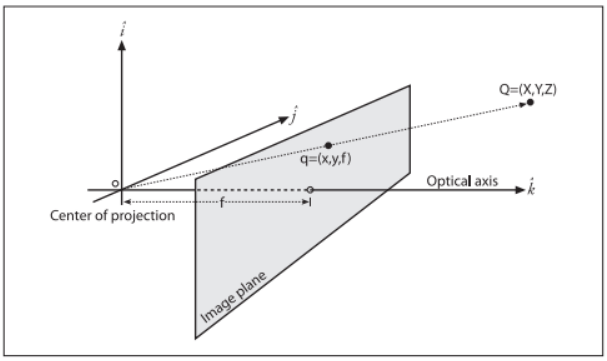


图 3.1　图像坐标系与相机坐标系的转换

因为这里只涉及到单帧的操作，我们仅需要考虑的相机的内部参数，通过相机参数，我们可以容易地得到以下坐标转换关系:

$$\begin{bmatrix} x \\ y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3-1}$$

其中 x, y 是图像坐标系下的坐标，X, Y, Z 是该点在相机坐标系下的坐标；c_x, c_y 是相机的成像面的中心在 x, y 轴方向上相对于光轴的偏移量，单位是像素；而 f_x, f_y 分别是相机相对于像素在 x, y 方向上的相对焦距，即 $f_x = F\delta_x, f_y = F\delta_y$，$F$ 是相机的焦距，δ_x, δ_y 是相机每一个像素的 x, y 方向上的长度。

由公式 3-1，便可以很容易的通过下式得到相机坐标系下的单帧点云

$$X = \frac{(x - c_x) \cdot Z}{f_x}, \quad Y = \frac{(y - c_y) \cdot Z}{f_y} \tag{3-2}$$

3.1.2　SIFT 特征描述子

SIFT(Scale-invariant feature transform) 特征描述子[33] 是一种能很好的反应图片中局部特征的描述符。从 SIFT 的全称可以看到，SIFT 特征描述符具有尺度不变性，除此之外它还对旋转有一定的鲁棒性，只要在图像平面上旋转角度不超过 60°，它就能正确地计算特征描述子，并且在当光照条件变化较大时，SIFT 特征点也有很强的可靠性。

简单的来说，在计算 SIFT 特征描述子时，首先对图像中的每一个像素点，计算其梯度方向。然后将每一个像素点周围 16×16 的像素划分成 16 个 4×4 的

区域，每个区域统计区域中的像素点的梯度方向，将这些梯度方向离散到 8 个梯度主方向，把所有区域的梯度主方向拼接成一个 128 维的梯度直方图，该梯度直方图即为该像素点的特征描述子。具体实现中的细节参见 [33]。

3.1.3 RANSAC 求解变换矩阵

在求解出相邻两帧 P, Q 的 SIFT 特征描述子，便可以通过 SIFT 特征描述子的相似度筛选出可能的对应点集合 $\mathcal{S} = \{(s_p, s_q)|s_p \in P, s_q \in Q\}$。接下来就需要利用 $\mathcal{S}$ 来计算出相邻两帧间相机位置的变换矩阵 $[R|t]$，$R \in \mathbb{R}^{3\times3}, t \in \mathbb{R}^3$ 分别为旋转矩阵和平移矩阵，都属于相机的外部参数。式 3-3表示了旋转矩阵对坐标的变换关系。

$$\begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{3,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{2,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_q \\ Y_q \\ Z_q \\ 1 \end{bmatrix} \tag{3-3}$$

我们仅需找到三组正确的对应点就能解得变换矩阵，而 RANSAC 算法可以非常有效地帮助我们找到内应点 (inlier points)。RANSAC 算法是一个重复迭代的过程，每一轮随机抽取三组点，并由该三组点计算出变换矩阵，然后将变换矩阵应用于 $\mathcal{S}$ 中所有的点对，找出所有的内应点对，若所有轮数中内应点对数目最多的一次迭代中的内应点数目超过一个阈值，则取该轮计算出来的内应点为假定正确的对应点对，并由此计算出两帧之间的变换矩阵。

得到相邻帧之间相机位置的变换矩阵后，我们就可以利用变换矩阵的乘法性质，以其中任一帧的相机坐标系为世界坐标系，并计算其余所有帧相对于该帧的变换矩阵，从而把所有点云进行一个初始的校准。

3.1.4 点云注册算法 ICP

由于计算相邻两帧间的变换矩阵会存在一定误差，而计算每一帧到统一的世界坐标系的变换矩阵又是相邻帧的变换矩阵的乘积，因此这之间的累积误差会非常大。为了能获得比较精准的点云以便后续处理，我们需要在当前的模糊校准的前提下进行一个精确的点云注册算法。ICP(Interative Closest Points，迭代

最近点算法)[15] 便是一个非常经典的方法，它能对两帧间校准后粗糙的变换矩阵做优化。

给定两个点集 $\mathcal{P} = \{p_i\}$ 和点集 $\mathcal{Q} = \{q_i\}$，它们各有 N_p, N_q 个点。ICP 算法如下:

- 令 $\hat{p}_i = C(p_i, \mathcal{Q}) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \|p_i - q\|_2$ 为 p_i 在二范数下载点集 $\mathcal{Q}$ 中的投影，即点集 $\mathcal{Q}$ 中距离 p_i 最近的点，并定义最近点集合 $\hat{\mathcal{P}} = \{\hat{p}_i\}$
- 在上述的定义下，对下列步骤做迭代循环:
  1 用 Kd-Tree 计算出当前的最近点集合 $\hat{\mathcal{P}}$。
  2 对 $\hat{\mathcal{P}}$ 进行筛选，删去距离超过均值 K 倍的点对。
  3 构造误差函数 $E(T) = \sum_{p_i \in \hat{\mathcal{P}}} \|T(p_i) - \hat{p}_i\|_2^2$，$T$ 为需要估计的变换矩阵，$T(p_i)$ 是点 p_i 在变换矩阵 T 后位置。
  4 求误差函数 $E(T)$ 的局部最小解 $\hat{T}$，并用此刚体变换更新点集 $\mathcal{P}$。
  5 如果相邻两次的误差值不超过阈值 τ，则退出迭代，否则回到第 1 步。
- 最后一次得到的变换矩阵 $\hat{T}$ 即为优化后的解。

上述的 ICP 算法的优点在于收敛速度快，但是其缺点是它假定了任意两帧之间都可以通过刚体变换得到统一的世界坐标系下的点云，而此前提往往并不能完全满足，因此基于刚体变换的 ICP 算法也会存在一定的瑕疵。如果把 ICP 算法中的变换函数变成非刚体变换 (Non-rigid tranformation)，则算法的运行速度会受到很大的影响，其换来的是更准确的点云注册结果。

3.2 Region Growing 区域生长算法

Region Growing 是一个非常简单但却经典的图像分割方法，它同样也可以适用于三维点云的分割上。如同之前所说，Region Growing 属于基于区域的分割方法，利用了每个点的周围点的信息去扩充出一个具有相似性质的联通点集。在本文中，我们将会用到改进后的 Region Growing 算法对原始点云进行基于平面的过分割，即把原始点云分割成一个个平面。因此，本节就以 P. J. Besl[8] 对基于平面分割的 Region Growing 算法进行介绍。

该算法主要有两个步骤:

- 确定初始种子点，每次选取表面曲率最小的点作为种子点 s，令当前点集为 $\mathcal{S} = \{s\}$。

- 每次找到一个与当前点集 $\mathcal{S}$ 相邻的点 p，若该点表面的法向与种子点 s 表面的法向相似，则将 p 放入点集 $\mathcal{S}$，从而实现区域增长。

Region Growing 算法的伪代码如 Algorithm 1所示。

Algorithm 1 Region Growing Algorithm

```
procedure RegionGrow
    G is K-nearest graph of all points
    Normal is stored normal of all points
    rest ← {0, 1, 2, ..., N}
    P ← {}
    while rest ≠ ∅ do
        select s ∈ rest with the minimal curvature
        S ← {s}
        for all p ∈ N(S) do
            if Normal_p · Normal_s ≤ τ then
                S ← S ∪ {p}
            end if
        end for
        P = P ∪ {S}
    end while
end procedure
```

3.3 Graph Cut 算法

Graph Cut(图割) 算法最初是图像处理领域非常流行的能量优化算法，它以及它的变种算法 Grab Cut 被广泛应用于图像的前景分割中。事实上，Graph Cut 能够处理一类特定的能量优化问题[14]，当优化函数的变量为二值变量时，Graph Cut 能够快速地在多项式复杂度内给出全局最优解；而当带估计的变量取值是在多个特定值中选取时，Graph Cut 依然能快速地在多项式时间复杂度内给出近似全局最优解。

顾名思义，Graph Cut 算法的核心思想是利用图论中经典的最大流最小割原理 (max-flow min-cut theorem) 去解决一系列的优化问题。此类能量优化问题常常能被转化成求一张图中的最小割集问题，比如图像的前景分割问题，就可以被转化成把整个图中的像素分成两个集合，使得在某种先验知识下，所求的集合分割方式花费最小。

下面本节会先简单介绍图论中的最大流最小割原理，接着会详细介绍 Graph-Cut 是如何求解能量最小化问题的。

3.3.1 最大流最小割原理

Max-flow min-cut theorem(最大流最小割原理) 描述的是在一个网络流模型中，从源节点 s 到汇节点 t 的最大流等于该网络流图的最小割的值。

定义 3.1： **流网络**是一个带有源节点 s 和汇节点 t 的有向图 $G = (V, E)$。每条边 $\langle u, v\rangle \in E$ 具有一个容量 c_{uv}。

定义 3.2： 流网络 G 的**最小割** C 是指一个容量和最小的边集合，使得在网络中去掉该集合中的边后源节点 s 和汇节点 t 不连通。

定义 3.3： 流网络的**流**是指一个映射 $f : E \to \mathbb{R}^+$，用 f_{uv} 表示边 $\langle u, v\rangle$ 上的流量，且满足每条边的容量限制及流量平衡条件。

定义 3.4： 一个流的**大小**

$$|f| = \sum_{\langle s,v\rangle} f_{sv}$$

一个流网络 G 的最大流问题就是求出该流网络下最大的流 $\hat{f}$

定理 3.1： 流网络 G 的最大流的值等于该网络的最小割。

定理 3.1的证明可以见 [34]。求得最大流 $\hat{f}$ 后可以非常容易地构造出其对应的最小割集。

由定理 3.1我们可以通过求解图 G 的最大流来得到该网络的最小割。目前比较流行的求解最大流问题的有 Dinic 算法 ($O(V^2E)$)[35] 和 J. B. Orline 提出的算法 ($O(VE)$)[36]。

3.3.2 Graph Cut 求解能量函数近似解

Y. Boykov 等人在 2001 年提出了用 Graph Cut 来解决一类在计算机视觉领域经常遇到的能量最小化问题[14]，这类问题的目标是找到一个标号方案 f，即给点集 $\mathcal{P}$ 中的每个点 p 都分配一个标号集合 $\mathcal{L}$ 中的标号 f_p，使得式3-4最小化：

$$E(f) = E_{smooth}(f) + \lambda E_{data}(f), \quad f_p \in \mathcal{L} \ for \ \forall p \in \mathcal{P} \tag{3-4}$$

其中，$E_{data}(f)$ 需要满足:

$$E_{data} = \sum_{p \in \mathcal{P}} D_p(f_p), \tag{3-5}$$

数据项 $E_{data}(f)$ 衡量了在给定的先验知识下，把元素标号成某一个类的代价，由式3-5可以发现每一个元素标号的代价 $D_p(f_p)$ 完全由这个元素及它的标号决定。

与此同时，$E_{smooth}(f)$ 需要满足:

$$E_{smooth} = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q), \tag{3-6}$$

$E_{smooth}(f)$ 衡量了标号方法对两个元素的影响程度，因此叫做平滑项，即最小化这一项能使得所有元素二元组之间的标号在先验知识下过渡地尽量平滑。$\mathcal{N}$ 是所有需要定义平滑关系的二元组的集合，V 是定义在标号空间 $\mathcal{L}$ 上的一个度量函数，它需要至少满足:

- 正定性：$\forall\{\alpha, \beta\} \in \mathcal{N}$，$V(\alpha, \beta) \geq 0$ 且 $V(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta$
- 对称性：$V(\alpha, \beta) = V(\beta, \alpha)$

由式3-4可知，能量函数 $E(f)$ 是 E_{data} 和 E_{smooth} 两个能量函数的双目标优化函数 (bi-criterion optimization problem)，并且这类能量函数的每一项最多只会涉及到二元组之间的计算，而不会受更多元素的影响。

在给定一个初始解 f_0 的前提下，Y. Boykov 提出可以通过两种基本的操作来分别实现对于能量函数 $E(f)$ 的优化，这两类操作分别是 $\alpha - \beta$ 交换和 α 扩展。可以找到在使用这两种操作中的任一种的情况下的局部最优解，并且收敛到的结果不一定会在初始解附近，因此该方法对初始解的没有太大的要求。

通过一个标号方案 f，我们可以得到原先点集 $\mathcal{P}$ 的一个划分 P，在此基础上给出 $\alpha-\beta$ 交换和 α 扩展的定义。

定义 3.5： 给定两个标号 α, β, 当在原标号方案 f^n(划分为 P^n) 的基础上，仅通过将标号为 α 和标号为 β 的元素进行标号的交换，从而变换到一个新的标号方案 f^{n+1}(划分为 P^{n+1})，则该操称为一次 $\alpha-\beta$ **交换**，即对于任意的标号 $l \neq \alpha, \beta$，都有 $P_l^n = P_l^{n+1}$。

定义 3.6： 给定一个标号 α，当在原标号方案 f^n(划分为 P^n) 的基础上，仅通过增加标号为 α 的元素，从而变换到一个新的标号方案 f^{n+1}(划分为 P^{n+1})，则该操称为一次 α **扩展**，即对于任意的标号 $l \neq \alpha$，均有 $P_l^{n+1} \subset P_l^n$，且 $P_\alpha^n \subset P_\alpha^{n+1}$。

Graph Cut 解决能量优化问题的伪代码如 Algorithm 2所示。

Algorithm 2 Graph Cut solving energy minimization

```
设置初始标号方案 f
while TRUE do
    success ← 0
    for all (α, β) 或 α(两种不同的交换策略不同) do
        计算在一次 α − β 交换或 α 扩展操作下最优的标号方案 f̂
        if E(f̂) < E(f) then
            f ← f̂
            success ← 1
        end if
    end for
    if success = 0 then
        break
    end if
end while
```

定义 $\mathcal{N}$ 为具有平滑项的二元组集合。接下来开始介绍对于两种操作，如何构建流网络 $G = (V, E)$，从而可以使用最大流算法解得最优解。

对于 $\alpha-\beta$ 交换，在给定 α,β 后，先建立网络 G_0 的源节点 s 和汇节点 t，s 表示 α 集合，t 表示 β 集合，然后令 $V=\{s,t\}\cup P_\alpha\cup P_\beta$，边集 E 以及每条边的权值如表3.1所示。

表 3.1　$\alpha-\beta$ 交换边权值表，摘自 [14]

顶点 u	**顶点** v	(u,v) **权值**
$u=s$	$v\in P_\alpha\cup P_\beta$	$D_v(\alpha)+\sum_{q\in\mathcal{N}_v,q\notin P_\alpha\cup P_\beta}V_{\{v,q\}}(\alpha,f_q)$
$u\in P_\alpha\cup P_\beta$	$v=t$	$D_u(\beta)+\sum_{q\in\mathcal{N}_u,q\notin P_\alpha\cup P_\beta}V_{\{u,q\}}(\beta,f_q)$
$u\in P_\alpha\cup P_\beta$	$v\in P_\alpha\cup P_\beta$	$V_{\{u,v\}}(\alpha,\beta)$

求出该流网络 G_0 的最小割 C 后，可以构建出对应的新标号方案 f'

$$f'_p=\begin{cases}\alpha, & p\in P_\alpha\cup P_\beta \text{ 且 } (s,p)\in C\\ \beta, & p\in P_\alpha\cup P_\beta \text{ 且 } (p,\beta)\in C\\ f_p, & p\notin P_\alpha\cup P_\beta\end{cases}$$

对于 α 扩展，也可以用类似的方法构建图，但这种情况下，需要把除了集合 P_α 外的点都加入图中，并且源和汇分别代表属于 α 和不属于 α 两种情况。因为在本文中，我们会使用 $\alpha-\beta$ 交换作为求解能量函数的基本操作，因此不再对 α 扩展做详细的介绍。

第 4 章　基于 Graph Cut 的交互式点云分割算法

4.1　算法总体流程

本文的主要贡献是建立了一个完整、高效的交互式点云分割系统，并通过对点云场景中支撑关系的推测，来提高交互效率，节省交互时间。如图4.1所示，我们的交互系统主要由五个部分构成：预处理、支撑关系的计算、基于 GraphCut 的交互分割，后处理和相似物体的搜索。

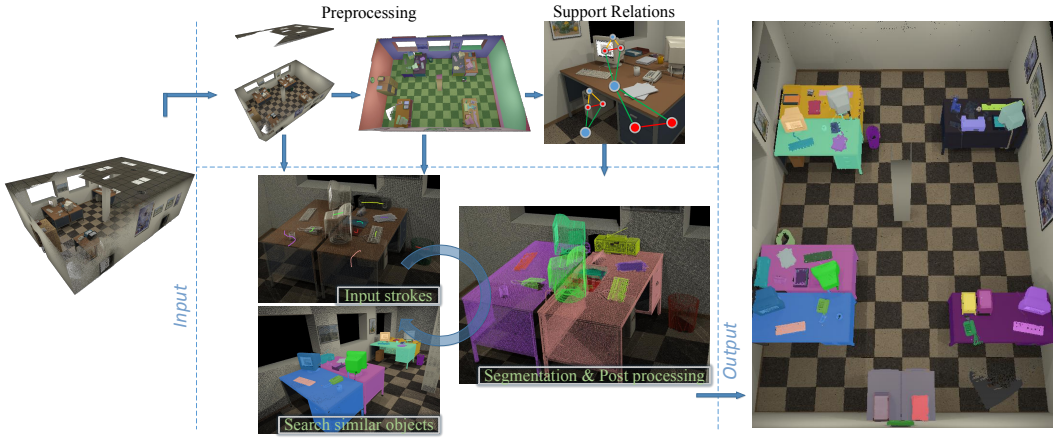


图 4.1　系统流程图

在预处理阶段 (4.2节)，系统自动地把一个原始点云与坐标轴进行对齐，提取点云中不同的楼层，最后把点云进行过分割，即把所有点聚类成若干个语义块，这些块在后续分割中作为原子单元，不可再被细分。

接下来 (4.3节)，系统根据各个语义块的位置关系，自动计算出语义块间的支撑关系，这些支撑关系在后续的分割及相似物体的搜索中有着极为重要的作用。

之后整个系统就进入了交互式分割阶段 (4.4节)，每一次交互，用户可以方便地自行拖拽和推进视角至一个理想的区域，然后在那些待分割或修改分割的物体 (包括未分割、过分割、欠分割) 上用画笔给出一些信息。不同颜色的画笔代表了不同的物体，被画笔所触及到的语义块将作为 Graph Cut 算法的初始块 (seed patch)。根据这些初始块以及语义块的几何、颜色和支撑关系，可以计算出

Graph Cut 算法中每一条边对应的权值，从而使得 Graph Cut 算法正确运行，从而计算出一个当前的最优分割方案。

为了进一步地优化 Graph Cut 得到的结果，使得结果更加合理，在每一次 Graph Cut 算法结束后，都会对结果进行后处理 (4.5节)，修正一些明显不符合语义的分割结果，从而减少用户进一步在这些错误结果上的交互量。

同时，考虑到室内场景中具有大量的重复物体，为了避免这些重复物体所产生的巨大交互量，系统在最后提出了一种轻量级的相似物体搜索算法 (4.6节)，用来将用户已经分割完成结果扩散到整个场景中。

4.2 预处理

本节我们讲介绍系统的预处理部分，这部分工作为后续的操作奠定了重要的基础。

首先，为了证明方法的通用性，我们的实验数据来自于三种不同的深度探测仪，包括 Kinect、Kinect 2 以及 LiDAR，同时我们的数据也包括了合成场景。由于 LiDAR 本身的拍摄精度非常高，以及在原始数据中就记录下来相机在每一帧的变换矩阵参数，因此对于 LiDAR 数据，我们很容易就能将原始的多帧数据注册到统一的世界坐标系下。对于 Kinect 和 Kinect 2 拍摄的 RGB-D 图片流，我们采用了3.1节中介绍的基于 SfM 的点云配准及注册算法，从而得到了完整的点云数据。

对于一个注册后的点云场景，其点的数目是百万甚至千万级别的，不便于后续的处理，因此我们先对其进行降采样，然后对降采样后的点云估计每个点表面的法向，并且将点云变换到曼哈顿坐标系 (Manhattan coordinate)[37]，且使得 Z 轴朝向正上方。接着使用 RANSAC 提取出水平的大平面作为地面和房顶。如图4.2所示，我们会通过去掉每一个楼层的天花板，将一个多楼层的场景分解成多个单层的场景。

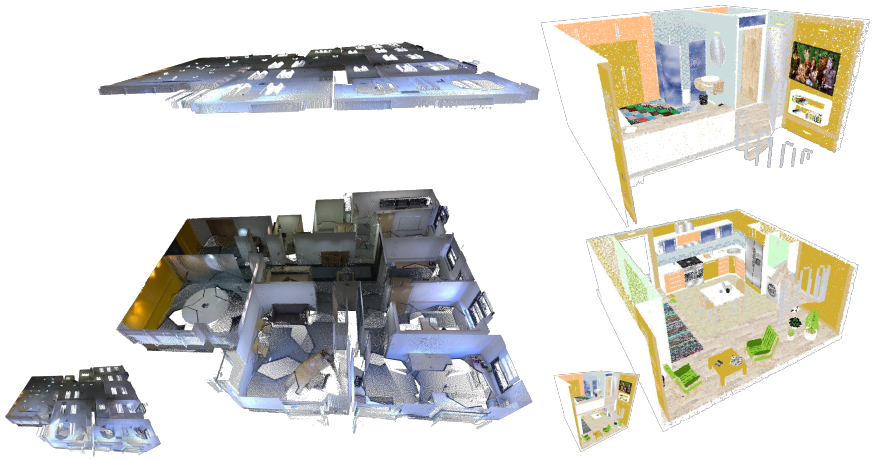


图 4.2　通过提取房顶和地面将一层楼的场景 (左) 和两层楼的场景 (右) 进行分解

4.2.1　构建特征面片

对原始点云进行过分割 (over-segmentation) 处理被作为预处理工作广泛地应用在各种点云分割算法中[9,11,29]。在本文中，我们希望通过过分割把原始场景分割成一个个语义特征面片，而不是像 [11] 中仅仅根据距离把点云分割成若干个大小相似的却毫无语义信息的块，图4.3(b) 展示了这种方法的问题。在我们的算法中，这种语义块指的就是属于同一个物体的近似平面块。这种语义代表快不但能帮助我们大大减少后续算法的计算复杂度，还能利用它的语义整体性去对整个场景进行更好的分析 (例如支撑关系)。

我们的算法是在 O. Mattausch 等人在论文[9] 中提出的的 Region Growing 算法的基础上的一个扩展。由于最初的算法不会考虑任何关于点云颜色的信息，它并不能很好地把表面较贴合的不同物体区别出来，图4.3(c) 中桌面上的键盘等和桌面被分到了同一个块中，导致后续算法无法分割出这些物体。如果仅仅在原始方法中加入颜色信息的考虑，因为不同帧的光照条件不同，再加上低端深度相机 (Kinect) 本身采集的点云质量就不高，很难通过一轮 Region Growing 就得到理想的过分割结果，其会严重地受点云的噪声的影响，从而形成很多小块 (尤其是当数据采集工具是 Kinect 的时候)。这些过小的块因为它们包含的点太少，因而并没有统计学上的语义，并且过少的点数也容易受噪声的影响，为了解决这一问题，我们在原先算法的基础上又加了一轮新的 Region Growing 过程。原先的第一轮是以点云中的点基本单位进行 Region Growing，而第二轮是以第一轮得到的初始语义块为基本单位进行 Region Growing。这样一来，使用第一轮的

初分割结果能大大增加第二轮 Region Growing 的抗噪声能力，从而得到更加有意义的过分割结果。

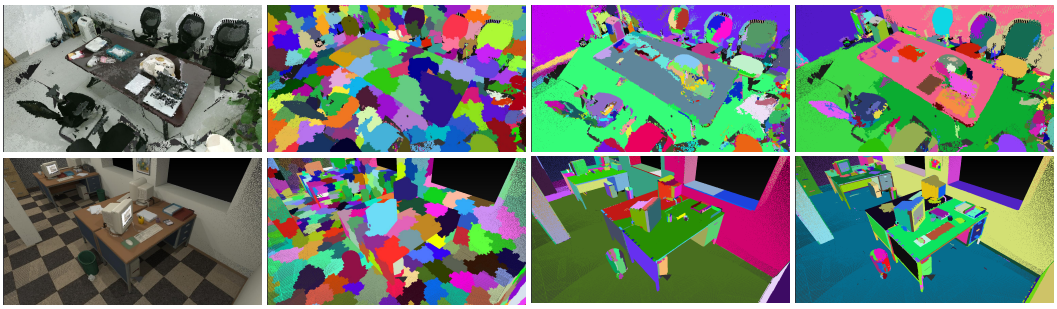


图 4.3　构建特征面片结果。(a) 两个场景进行降采样后的点云，(b) 用 A. Golovinskiy 等人采用的方法进行构建，(c) 用 O. Mattausch 等人采用的方法进行构建，(d) 本文的方法

本文的 Region Growing 算法的第一阶段是对点云中的所有点进行的，与 [9] 中的方法类似。具体的，令 $G_0 = \langle V_0, E_0 \rangle$ 表示一个图结构，其中 V_0 是点云中所有的点集合，E_0 中的边是对所有点通过 k 近邻方法构建的 (在实现中采用了 $k = 15$)。因为具有相似颜色的点更有可能属于同一个物体，而相反的，有时候两个点虽然近似位于同一平面上，但它们的颜色不同，则它们和有可能属于两个物体（例如桌上放了一本书），从而违背过分割的初衷。因此，我们在 [9] 考虑最基本的法向与位置信息的限制下，由增加了对颜色的限制：

$$\begin{aligned} N_p \cdot N_s &> t_0 \\ (p - s) \cdot N_s &< t_1 \\ \max(\|R_p^{HS} - R_s^{HS}\|,\ |C_p^V - C_s^V|) &< t_2 \end{aligned} \tag{4-1}$$

前两项是对法向和位置的考虑，而第三项是对颜色的限制。其中 p, s 分别为待加入点的坐标和初始点的坐标。N_p, N_s 分别问待加入点和初始点的法向，C_i^H, C_i^S, C_i^V 分别表示归一化后点 i 的 HSV 颜色值 ($H \in [0, 2\pi]$,　$S, V \in [0, 1]$)。$R_i^{HS} = (C_i^S \cos C_i^H, C_i^S \sin C_i^H)$ 代表点 i 的 H(ue)-S(aturation) 色盘上的 2D 坐标。直观上地，如果两个点的颜色在 H-S 色盘上的距离与他们的明度值差均在一个非常小的阈值 t_2 内时，我们认为这两个点可以被合并到一个块中。这里三个阈值 t_0, t_1, t_2 都非常小，因为需要考虑到单个点的误差较大，导致比较时的置信度比较低。在所有的实验中，我们均设定了 $t_0 = 0.8, t_1 = 0.005, t_2 = 0.05$ 作为参数。

第一步的算法过后，我们可以得到很多合并后的语义块。然而，在实验中，我们发现当点云质量较低时，会产生很多细碎的块，这些块事实上可以很容易地被合并到其他块中，然而由于噪声产生的色差使得他们无法在第一阶段被合并成较大的语义块。为了解决这个问题，我们需要执行第二步的算法。在这一步中，我们将第一步中得到的语义块作为最小单位，对这些块用一组比较宽松的阈值继续进行 Region Growing 算法。与 G_0 类似的，我们对所有的块建立 $G_1 = \langle V_1, E_1 \rangle$。每一次，选择未被点数最多且未被合并的块作为初始块。对每一块，我们都计算出它的最佳拟合平面以及其质心，用对拟合平面的法向和质心坐标的限制来替代第一步中对法向和位置的限制 (对应两个阈值参数 t_3, t_4)。在这一步中，颜色的距离稍微有些不同，我们用一个块中所有点的 HSV 颜色分布直方图来表示一个块的颜色，并使用卡方距离 (χ-squared distance)[38] 来作为两个块的颜色距离。

$$\chi^2(I_i, I_s) < t_5 \tag{4-2}$$

I_i, I_s 分别表示了归一化后待加入块和初始块的 HSV 颜色直方图。特别的，在直方图统计时，我们把 H,S,V 通道分别离散成 16, 16, 8 个组。我们通过把 V 通道离散到较少的组来减少不同的光照条件带来的影响。同时，这一步中我们采用了比较宽松的阈值，这是因为此时考虑的是整个块的平均特征，因此比较时置信度会变高。此外，为了能尽量把所有细碎的块都合并到较大的块中，我们迭代地执行第二步中的算法，每迭代一轮，便把所有阈值都放宽到原先的 20%。在所有的实验中，我们均设定了 $t_3 = 0.75, t_4 = 0.2, t_5 = 1.6$ 作为第一轮迭代时的参数。

4.3 支撑关系的计算

此前的基于 Graph Cut 的点云分割算法主要注重于物体的底层的几何信息 (如位置、向量、颜色)，并用这些几何信息将分割的信息从初始点传播到整个场景中。由于人造的物体常常会出现支撑结构 (比如盒子状)，而不同的支撑面在几何上会有很大的变化，因此用户为了完整分割出这些物体往往需要画很多笔 (每个面一笔)。如图4.4中间那幅图中的桌子或者显示器就需要更多的画笔才能分割完全。

为了减少用户的交互量，我们在 Graph Cut 中引入支撑关系，把两个语义块

可能产生支撑关系的概率作为 Graph Cut 中衡量两个语义块间距离的一部分。支撑关系的引入对整个系统的分割能力有着非常大的帮助，如图4.4右边的图所示，在采用了支撑关系后，用户只需要在桌子和显示器的顶面分别画一笔就可以完成对这两个物体的分割。

图 4.4　Graph Cut 结果，左图: 原始场景及输入的画笔；中图：不采用支撑关系；右图：采用支撑关系

由于重力的影响，在我们的身边支撑关系无处不在。一般我们认为的支撑关系有两类，第一类是物体间的支撑关系 (桌子支撑电脑)，另一类是同一物体中不同部分间的支撑关系 (笔记本电脑的底座支撑屏幕)。若能准确推测两类支撑关系中的任一种，显然对 Graph Cut 中的数据项和平滑项都有显著的作用。虽然已经有相关工作能对场景中中的第一类支撑关系进行分析[5,17]，然而由于第一类支撑关系的计算很大程度上依赖于对物体的分割结果，因此与本文的目标相矛盾。在本文中，我们关注第二类支撑关系来帮助 Graph Cut 更精确地分割点云。

我们以4.2.1节的结果 P(语义块集合，下面简称为面片) 作为输入。首先我们依据每个面片的拟合平面与水平面所成的角度把所有的面片分成两类 (以 45° 为分界)：准竖直面 Q_v(quasi-vertical) 和准水平面 Q_h(quasi-horizontal)。接着，我们从所有的相邻面片二元组 $(P_i, P_j) \in \mathcal{N}$ 中提取出可能存在支撑关系的二元组，具体地说，就是以下两类相邻面片二元组 (P_i, P_j)：

- $P_i \in Q_v, P_j \in Q_h$，且 P_j 中的点全都位于大部分 P_i 中的点的上方。这种情况下 P_i 可能会支撑 P_j，下文中称为 Q_{vh} 关系。
- $P_i \in Q_v, P_j \in Q_h$，且 P_j 中的点全都位于大部分 P_i 中的点的下方。这种情况下 P_i 可能会被 P_j 支撑，下文中称为 Q_{hv} 关系。

除此之外，我们还考虑所有的面片二元组 $\{(P_i, P_j) | P_i, P_j \in Q_v\}$，这种情况下 P_i, P_j 可能同时支撑一个水平面片或同时被一个水平面片支撑，下文中称为 Q_{vv} 关系。

显而易见，存在上述三种关系中的任一种都能直接推导出 P_i, P_j 属于同一个物体。图4.5中很好的展示了这三类支撑关系。

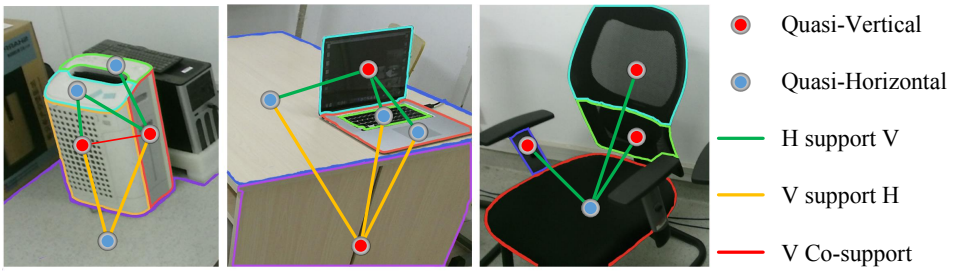


图 4.5　支撑关系检测示意图

对于每一对 $(P_i, P_j) \in Q_{vh}$，面片 P_i 确实支撑面片 P_j 的概率如下：

$$Q_{vh}\langle P_i, P_j\rangle = 1 - \lambda_Q^{U(P_i,P_j)} \tag{4-3}$$

$U(P_i, P_j)$ 表示有多少准竖直面与 P_i 相邻 (包括 P_i) 且与 P_j 构成 Q_{vh} 关系，即 $U(P_i, P_j) = |\{P_k | P_k \in \mathcal{N}_{P_i} \cup \{P_i\}, (P_k, P_j) \in Q_{vh}\}|$。$\lambda_Q$ 是一个固定的参数。非常直观的，有越多的准竖直面支撑着同一个准水平面，那么确实存在这种 Q_{vh} 支撑关系的概率就越高。

对于每一对 $(P_i, P_j) \in Q_{hv}$，面片 P_i 确实被面 P_j 支撑的概率如下：

$$Q_{hv}\langle P_i, P_j\rangle = \begin{cases} 0, & U(P_i, P_k) = 0, P_k \in Q_h \\ \frac{\min(W_i, W_j)}{\max(W_i, W_j)}, & otherwise \end{cases} \tag{4-4}$$

W_i 是面片 P_i 中所有的点在拟合平面上构成的凸包的面积。对于一个准竖直面 P_i，仅当它不存在 Q_{vh} 关系时，我们才会考虑与它有 Q_{hv} 关系的面片。在图4.5中间的图中，虽然笔记本的显示屏同时与桌面、笔记本底座以及笔记本的键盘这三个面片存在 Q_{hv} 关系，然而式4-4保证了显示屏不会被桌面所支撑，因为它们面积相差过大。

得到 Q_{vh} 和 Q_{hv} 后，我们就可以开始计算两个面片 P_i, P_j 具有支撑关系的概率 $\mathcal{T}_s(P_i, P_j)$，这个概率表明了通过对两个面片的支撑关系的角度推测两个面片

属于同一个物体的可能性。计算方法如下：

$$\mathcal{T}_S(P_i, P_j) = \begin{cases} Q_x\langle P_i, P_j\rangle, & P_i \in Q_v, P_j \in Q_h \\ Q_x\langle P_j, P_i\rangle, & P_i \in Q_h, P_j \in Q_v \\ \max\limits_k \left(Q_x\langle P_i, P_k\rangle \cdot Q_x\langle P_j, P_k\rangle\right), & P_i, P_j \in Q_v, P_k \in Q_h \end{cases} \tag{4-5}$$

前两行的式子事实上就是把前面计算出来的 Q_{vh} 和 Q_{hv} 直接赋值给 $\mathcal{T}$。第三行的式子是用来计算 Q_{vv} 关系的，两个准竖直面存在 Q_{vv} 关系当且仅当这两个面能同时与一个水平面有相同的支撑关系 (Q_x)，因此是把两个概率做乘积。

4.4 Graph Cut 分割点云

当用户在一个视角下给出一些简笔画后，我们会将那些画笔所穿过的面片作为初始块，并由这些初始块通过 Graph Cut 分割点云。求解画笔所穿过的面片时，我们需要考虑到点云的零体积特点，对于画笔上的每一个像素，如果我们仅仅找出点云中距离该像素构成的射线最近的点来作为此画笔穿过的点，会使得我们求解出的点可能事实上穿过了一些面片，从而导致选择错误的初始块。因此我们需要解决防止射线穿过面片的情况，为此我们先对整个点云建立一个三角网格模型，计算射线遇到的第一个三角网格，并用该三角网格对应的面片作为一个初始面片。

在利用 Graph Cut 求解分割时，我们的算法既考虑了点云的一些底层的几何特征 (颜色、位置、法向)，同时也考虑了上层的语义信息 (支撑关系)。与式3-4类似的，我们给出如下具体的能量函数：

$$E = \sum_{P_i} E_D(P_i, v_i) + \lambda \sum_{P_i, P_j} E_S(P_i, v_i, P_j, v_j) \tag{4-6}$$

E_D 是数据相关项，用来衡量把每个面片 P_i 归类成标号 v_i 的代价，E_S 是平滑项，用来衡量将每一对相邻的面片二元组 $(P_i, P_j) \in E_1$ 归类成两个不同的标号 v_i, v_j 时需要的代价。在实验中，我们设置 $\lambda = 1$。

对于数据相关项 E_D，我们考虑以下的一些特征 $\mathcal{D}_n$。为方便起见，下文中定义 $\mathcal{U}_i$ 为标号 v_i 的初始块集合。数据项的各个部分如下：

1. 距离项 **(Distance)**　我们定义距离项 $\mathcal{D}_D = \exp \frac{D'(P_i, \mathcal{U}_i)}{2\sigma_D^2}$，$D'(P_i, \mathcal{U}_i)$ 是面片 P_i 的质心与 $\mathcal{U}_i$ 中所有初始块的质心之间最短的欧几里得距离。
2. 颜色项 **(Color)**　我们通过两个块之间的颜色分布的距离来定义颜色项 $\mathcal{D}_C = -\log \mathcal{T}_C(P_i, \mathcal{U}_i)$。这里，$\mathcal{T}_C$ 被定义成 $\mathcal{T}_C(i, j) = \exp -\frac{\chi^2(I_i, I_j)}{2\sigma_C^2}$，$\chi^2(I_i, I_j)$ 是面片 P_i, P_j 标准化后的 HSV 颜色直方图的卡方距离。
3. 平面项 **(Plane)**　我们通过比较两个面片的拟合平面之间的差别来定义平面项 $\mathcal{D}_P = -\log\left(\max_{P_j \in U_i} \mathcal{T}_P(P_i, P_j)\right)$。其中，$\mathcal{T}_P(i, j) = \min(n_i \cdot n_j, \exp -\frac{\rho(P_i, P_j)}{2\sigma_P^2}$，$\rho(P_i, P_j)$ 是两个面片的质心分别在另一个面片的法向上的投影长度的平方，并取较大值赋给 $\rho(P_i, P_j)$。
4. 支撑项 **(Support)**　我们通过4.3节中计算的支撑关系的概率 $\mathcal{T}_S(P_i, P_j)$ 来计算支撑项 $\mathcal{D}_S = -\log(\max_{P_j \in U_i} \mathcal{T}_S(P_i, P_j))$。

数据相关项 E_D 通过 $E_D = \sum_n \lambda_n D_n$ 来计算。对于所有的场景，我们使用统一的参数，如下：$\lambda_D = 0.15, \lambda_C = 0.15, \lambda_P = 0.3, \lambda_S = 0.4, \sigma_D = 1.0, \sigma_C = 0.7, \sigma_P = 0.1$。

对于平滑项，我们考虑相邻面片二元组 $(P_i, P_j) \in E_1$ 间的颜色 $(\mathcal{T}_C)$，平面 $(\mathcal{T}_P)$ 和支撑关系 $(\mathcal{T}_S)$。同样的，平滑项 E_S 也通过 $E_S = \sum_n \lambda'_n \mathcal{T}_n$ 来计算，并对于所有的场景，我们也使用了统一的参数:$\lambda'_C = 0.3, \lambda'_P = 0.4, \lambda'_S = 0.3$。

在利用 Graph Cut 进行分割前，我们并不是把所有的面片都放入待分割集合中，我们会先通过所有面片和初始面片间的距离、颜色、平面和支撑关系，将所有的面片进行一个筛选（计算方法与数据项的计算方法类似，不再赘述），只挑选出其中与初始面片在这些方面比较相似的面片放入 Graph Cut 算法中进行求解。这样做不但能降低 Graph Cut 算法的时间复杂度，又能便于更精确地对点云进行分割。

4.5 后处理

因为 Graph Cut 是通过优化能量函数来解决分割问题，它所关注的只是最小化能量函数，因此可能会出现一些不符合常识的结果，此类不符合常识的结果主要分为以下两种：

1. 连通性问题：所求得的标号为 v_i 的面片可能不与任何标号为 v_i 的初始面片相连通，这样就导致了物体的不连通问题。
2. 重力问题：有些细碎的面片可能再也不可能被其他面片支撑（包括第一类

支撑和第二类支撑，见4.3节），但是没有被分到任何一个标号集合中，例如某些情况下，椅子的背和底座被成功标号，但是椅子的把手没有被分类，显而易见把手不会被其他东西支撑，因此它一定是属于椅子的一部分。此类问题出现的原因是由于这些面片面积过小导致的支撑性关系的无法被正确计算。然而这些面片由于面积非常小，用户如果需要进一步在这些面片上画简笔画会较困难，因此对这些面片的自动识别及分类能大大减少用户的交互量。

对应于上述两种问题，在进行完 Graph Cut 算法后，我们的算法会进行两步后处理操作。

第一步后处理操作是把以下集合中所有面片的标号置为 -1(即未标号状态):

$$\{P_i | v_i \neq -1, C(P_i, \mathcal{U}_i) = \text{false}\}$$

$C(P_i, \mathcal{U}_i)$ 是返回面片 P_i 与面片集合 $\mathcal{U}_i$ 是否连通。

第二步操作是为了来解决上述的重力问题，我们需要找到寻到那些没有得到标号，但明显能通过一定启发式的方法得到他的标号的面片，我们把这种面片称为可确定面片，令可确定面片集合为 $\mathcal{Z}$。首先，我们只在那些相邻面片中只存在恰好一种标号的面片中考虑其是否为可确定面片，令该唯一的标号为 $\hat{v}$。如果面片的面积大于 τ_{area}，那么我们也不考虑这些面片。接着，我们对面片 $P_i \in Q_v$ 和 $P_i \in Q_h$ 分别作考虑。

对于准水平面 ($P_i \in Q_h$)，若满足以下两个条件中的任一个则该面片可能被赋予除了 $\hat{v}$ 以外的标号，因此 $P_i \notin \mathcal{Z}$:

- 在它的相邻块中有准水平面 P_j 且位于 P_i 下方，这种情况下，P_i 可能与 P_j 有第一类支撑关系，即 P_i 可能是放在水平面 P_j 上的一个物体;
- 在它的相邻块中有准竖直面 P_j 位于 P_i 下方且 $(P_j, P_i) \in Q_{vh}$，并且 $v_j = -1$，在这种情况下，P_i 可能会和 P_j 共同属于一个其他的物体。

对于准竖直面 ($P_i \in Q_v$)，当存在另一个准竖直面 P_j，能与 P_i 共同支撑准水平面 P_k 或被 P_k 支撑，且 $v_j = -1$，则该面片可能被赋予除了 $\hat{v}$ 以外的标号，因此 $P_i \notin \mathcal{Z}$。

因为可确定面片一定与至少一块已被确认标号的面片相邻，因此我们可以从已被标号的面片入手开始遍历。伪代码如下:

Algorithm 3 The second step of post processing

$\mathcal{Z} \leftarrow \varnothing$
$Q \leftarrow \{P_i | v_i \neq -1\}$
while $Q \neq \varnothing$ **do**
    $P_u \leftarrow \text{pop}(Q)$
    for all $P_i \in \mathcal{N}_{P_u}$ **do**
        flag $\leftarrow$ true
        if area$(P_i) > \tau_{area}$ **then**
            flag $\leftarrow$ false
        end if
        找到一个 $\hat{v}$, 使得 $(\exists P_j \in \mathcal{N}_{P_i})\ v_j = \hat{v}$ 且 $(\forall P_j \in \mathcal{N}_{P_i})\ v_j = \hat{v}$ or $v_j = -1$
        if area$(P_i) > \tau_{area}$ or $\hat{v} = -1$ **then**
            flag $\leftarrow$ false
        end if
        if $P_i \in Q_h$ **then**
            if $(\exists P_j \in \mathcal{N}_{P_i})(P_j$ lower than P_i and $P_j \in Q_h)$ **then**
                flag $\leftarrow$ false
            end if
            if $(\exists P_j \in \mathcal{N}_{P_i})(v_j \neq \hat{v}$ and $(P_j, P_i) \in Q_{vh})$ **then**
                flag $\leftarrow$ false
            end if
        else
            if $\exists (P_j \in Q_v$ and $P_k \in Q_h)(v_j \neq \hat{v}$ and $(P_i, P_k), (P_j, P_k) \in Q_x)$ **then**
                flag $\leftarrow$ false
            end if
        end if
        if flag **then**
            $\mathcal{Z} = \mathcal{Z} \cup \{P_i\}$
            $Q = Q \cup \{P_i\}$
        end if
    end for
end while

4.6 相似物体搜索

在一个场景中常常会有重复的物体，如果用户需要对每一个物体都进行标注，那么这显而易见是费时费力的。在我们的系统中，我们设计了一个简单的算法去根据已经分割好的物体自动地搜索场景中与该物体相似的物体，并对直接把那些物体上的面片进行标号。当然，因为一些算法的局限性，我们的算法可能会出现一些错误的搜索 (6.1节)，但是我们的整个交互系统保证了容错性，用户可以非常方便地修正、删除那些错误标号的物体。

前人在点云场景的相似物体搜索这个问题上已经提出了一些有效的算法，例如 Kim 等人提出了一种算法通过学习一些常见物体的模型，从而实现实时的搜索和匹配[27]。O. Mattausch 等人也提出了根据面片相似度和空间一致性通过聚类的方法把场景中相似的物体进行聚类[9]，然而他们的方法对于大场景并不能达到实时的需求。

在本文中，我们希望设计一个简单而有效的算法，能实时地进行相似物体搜索，并且不需要预先对模型进行学习。我们的方法同样也是基于4.3节中提出的支撑关系的，算法的基本假设是：相似的物体中的面片应该有相似的支撑结构。这个假设让我们非常直观地就找到了定义相似物体的方法——通过面片的相似度以及面片间的支撑关系的比较。

具体地，我们首先定义两个同类型 (同为 Q_v 或 Q_h) 面片 P_i, P_j 的相似度 $\mathcal{A}_{i,j}$，此处我们对 O. Mattausch 等人在论文[9] 中提出的相似度计算方法做了少许改变，设计了一个适用于本系统的相似度计算方法。我们使用以下几个特征 (ξ_n) 来衡量两个面片间的相似度：

1. 颜色直方图 ξ_{ch} 是两个面片标准化后的 HSV 颜色直方图的卡方距离。
2. 法向 设 ϑ 是两个面片的拟合平面所成的角度，则定义法向相似度 $\xi_{nl} = \exp -\frac{\vartheta^2}{2\sigma_\vartheta^2}$。
3. 高度 这一项衡量了两个面片的位置相似度。若两个面片均为准竖直面，我们直接比较它们在 Z 方向的范围 ψ；若两个面片均为准水平面，那么我们比较它们质心间的垂直高度差 δ_h。高度项 ξ_{vl} 如下计算:

$$\xi_{vl} = \begin{cases} \exp -\frac{\delta_h^2}{2\sigma_h^2} & P_i, P_j \in Q_h \\ 1 - \frac{|\delta_h|}{\min(\psi_i, \psi_j)} & P_i, P_j \in Q_v \end{cases}$$

4. 覆盖面积　我们用 W 表示面片在拟合平面上所构成的凸包的面积，覆盖面积项 ξ_{ad} 由 $\xi_{ad} = 1 - (1 - \frac{\min(W_i, W_j)}{\max(W_i, W_j)})^2$ 计算得到。

有了上述用来衡量面片相似度的几个特征，我们便可以计算两个面片的相似度 $\mathcal{A}_{i,j} = \prod_n \xi_n$。并且对于所有场景，我们都使用参数 $\sigma_\vartheta = 10°, \sigma_h = 0.15$。

随后，给定一个已经分割好的物体 C_0，我们首先从未标号面片中找出所有的备选面片，这些面片至少和 C_0 中的一个面片有较高的相似度 $\mathcal{A}$(阈值 $t_7 = 0.3$)。

接着，我们根据所有备选面片的连通性将它们分成若干个面片群 C_i。对于每一个面片群 $C_i, i = 1, 2, ..., k$，我们对其中的每个相邻面片二元组 $(P_i, P_j) \in C_n$ 寻找一个对应的二元组 $(P_k, P_l) \in C_0$，满足这两个二元组有相同的支撑关系 (Q_{vh}, Q_{hv}或Q_{vv})。给予我们的前提假设，我们通过式4-7来计算面片群 C_n 与模型 C_0 之间的相似度 $Sim(C_n, C_0)$：

$$Sim(C_n, C_0) = 1 - \prod_{P_i, P_j \in C_n} (1 - \mathcal{I}_{i,j}) \tag{4-7}$$

$\mathcal{I}_{i,j}$ 是 (P_i, P_j) 在 C_0 中的支撑关系的同构概率 (isomorphic possibility)，由下式计算：

$$\mathcal{I}_{i,j} = \min_{k,l} \mathcal{A}_{i,k} \cdot \mathcal{A}_{j,l} \cdot \mathcal{T}_S(P_i, P_j) \cdot \mathcal{T}_S(P_k, P_l) \tag{4-8}$$

如果面片群 C_n 和模型 C_0 间的相似度 $Sim(C_n, C_0)$ 超过一个阈值 ($t_8 = 0.25$)，我们认为面片群 C_n 中的面片可以构成一个和模型 C_0 相似的物体。我们把那些找到支撑关系同构的面片 (即得到的 $\mathcal{I}_{i,j}$ 超过一个阈值) 作为新的一轮 Graph Cut 的初始面片，并由此进行相似物体的分割。

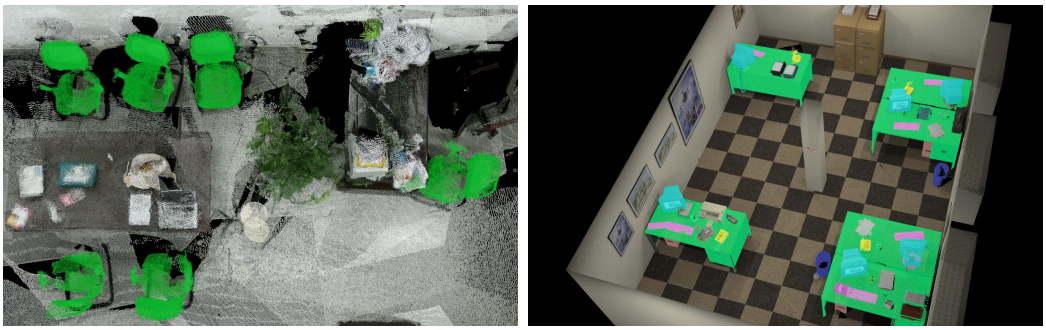


图 4.6　相似物体搜索结果，相同颜色表示相同物体

图4.6中，我们把相似物体检测应用于两个具有较多相似物体的场景中。在第一个场景中，因为点云的噪声，构成每把椅子的面片数量都是不同的，但是我们的算法依然能够寻找到这些面片中相似的支撑结构并执行 Graph Cut 算法，从而通过一把椅子找到其他的所有椅子。第二个例子展示了我们的相似物体搜索算法适用于各类物品，而不是仅限于椅子，在这个场景中，桌子 (绿色)，显示器 (青色)，电话机 (黄色) 以及键盘 (紫色) 都能通过我们的算法找出来。同时也可以看到有一些键盘附近的物体也会被相似物体的算法错选出来，此时用户可以通过简单的几笔来对这些粗糙的结果作一些修正。

第 5 章　实验结果与分析

实验阶段，我们在各种类型的场景中对系统经行了测试，这些场景来自于不同的数据集，从而更能证明我们的系统的普适性。对于所有的场景，我们均采用了同一套参数，这也表明了我们的参数的鲁棒性与对数据的不敏感性。对五个从其他数据集中搜集得到的场景的分割结果如图5.1、图5.2所示。

5.1　数据集

为了测试我们系统的点云分割能力，我们选择了一些具有较多物体的场景进行实验。其中，我们使用了 UZH 数据集[9] 中三个场景中的两个 (office2, office3)，使用了 Floored Panorama 数据集[39] 中的一个场景 (office1)，还使用了两个来自 ICL-NUIM 数据集[40] 中的两个合成场景。其中，ICL-NIUM 数据我们通过在其中的三维网格模型的表面上进行采样得到我们所需要的点云数据。除此之外，我们还用 Microsoft Kinect2 手动采集了两个身边的场景。

5.2　鲁棒性和效率

5.2.1　参数

在所有的实验中，我们都选取了同一套参数设定。在第4章中，我们已经介绍了部分参数的取值，表5.1中对一些重要的参数或上文中没有提到的参数进行描述即给出在实验中的取值。

5.2.2　特征面片提取算法

特征面片提取算法的有效性在4.2.1节中已经做了一定的描述，图4.3也很好地展示了本文采用的算法相对于前人提出的算法的优势。A. Golovinskiy 等人所提出的算法完全根据点与点之间的距离作为块的构建方法，这种方法类似于根据距离聚类的效果，最后得到的结果只是满足空间一致性，并不包含任何的语义信息，会将很多不是同一个物体的点合到同一个面片中，从而直接导致后续分割结果的错误。而 O. Mattausch 等人采用了 Region Growing 的算法对原始点云

表 5.1　部分参数定义与参数设定

参数	描述	值
k	建立 G_0 时每个点选择的最近的点数	15
λ_Q	求 Q_{vh} 支撑关系时指数项的底数	0.2
λ	能量函数 E 中平滑项的权重	1.0
σ_D	Graph Cut 中距离项标准化参数	1.0
σ_C	Graph Cut 中颜色项标准化参数	0.7
σ_P	Graph Cut 中平面项标准化参数	0.1
τ_{area}	后处理第二步时考虑需要考虑的面片的最大面积	0.2
σ_ϑ	相似物体搜索时法向项标准化参数	10°

中的平面进行提取，并把处于同一个平面上的点归为同一个特征面片，这种方法简洁且包含语义含义，但是也正是由于它没有考虑颜色信息，无法正确区分两个位于相同平面的东西，例如桌上的键盘和书本，都会和桌面混到一起。而本文提出的算法通过对颜色信息的考虑以及对初始特征面片的二次 Region Growing，提供了一个更鲁棒的语义面片提取算法。

5.2.3　支撑关系的有效性

本文通过把支撑关系加入到 Graph Cut 的考虑因素中，提高了点云分割的正确性和效率，从而大大减少了用户的交互量。支撑关系的作用在4.3中已经做了一定的阐述，图4.4也很好地对采用支撑关系的分割结果与不采用支撑关系的分割结果进行了对比，在输入同样的简笔画的情况下，具有支撑关系的 Graph Cut 算法能够正确地分割出显示器、桌子等具有明显支撑结构的物体，而不具有支撑关系的 Graph Cut 则无法分割出这些不同面之间几何、颜色都差距较大的物体。

5.2.4　时间效率

所有的实验我们都是在相同的计算机上进行的，配置为：处理器为 Core i7 2.10GHz, 显卡为 GTX 660Ti，内存为 16GB RAM。

对于每个场景 (点云中点数约为 500 万 1000 万)，预处理均能在十分钟内完成，其中预处理阶段中大部分时间是花在法向估计和 k 近邻搜索这两个步骤中，预处理中其余步骤均能在 1 分钟内完成。并且预处理中各个步骤都均被并行化，因此预处理的时间可以说是非常快的。

在交互阶段，每一轮当用户给定简笔画后，大约只需要 100ms 的时间做 Graph Cut 点云分割；给定一个特定标号的物体，其相似物体的搜索也只需要 200ms 就能完成。综上，交互阶段的处理时间能完全满足一个交互时系统的响应速度需求。

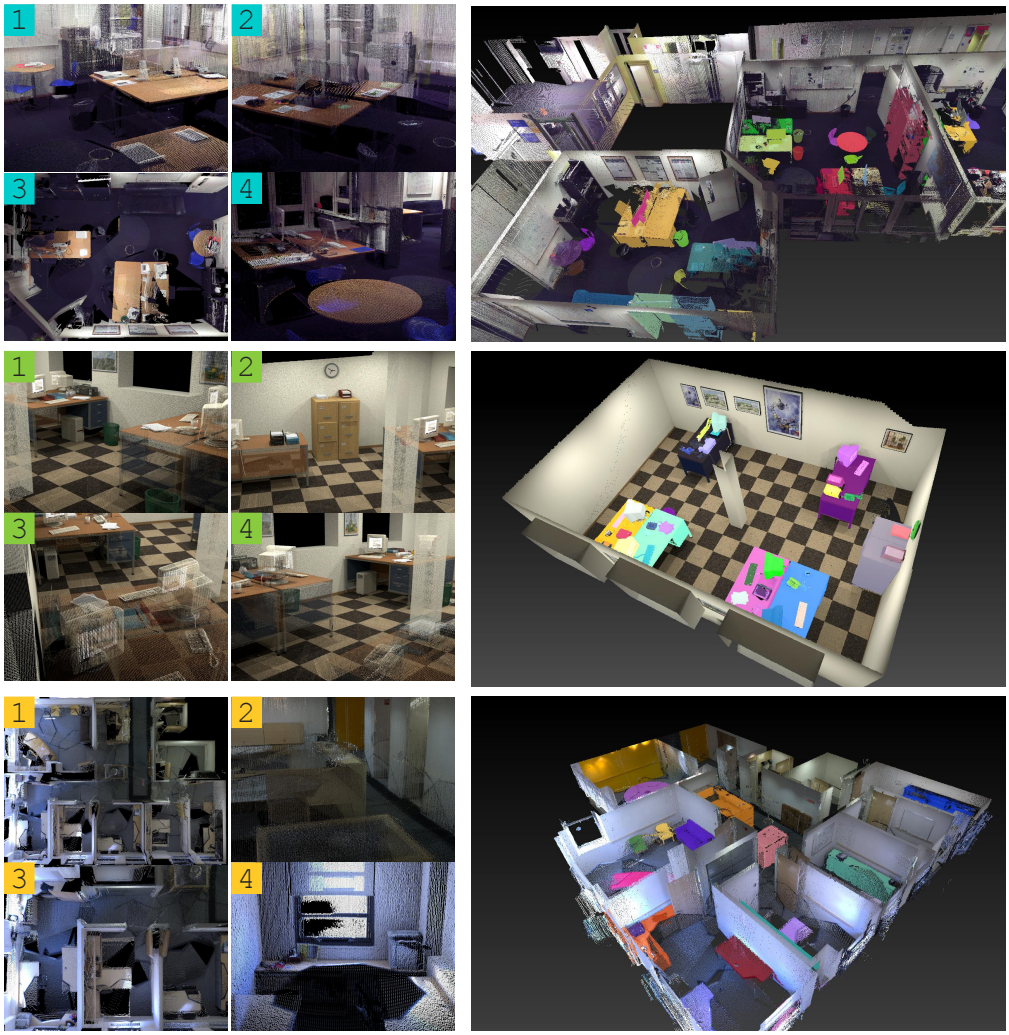


图 5.1　点云分割结果 (1)，每一行是一个场景，左侧是其中四个选择画简笔画的视角，右侧为分割结果，不同的颜色表示不同的物体

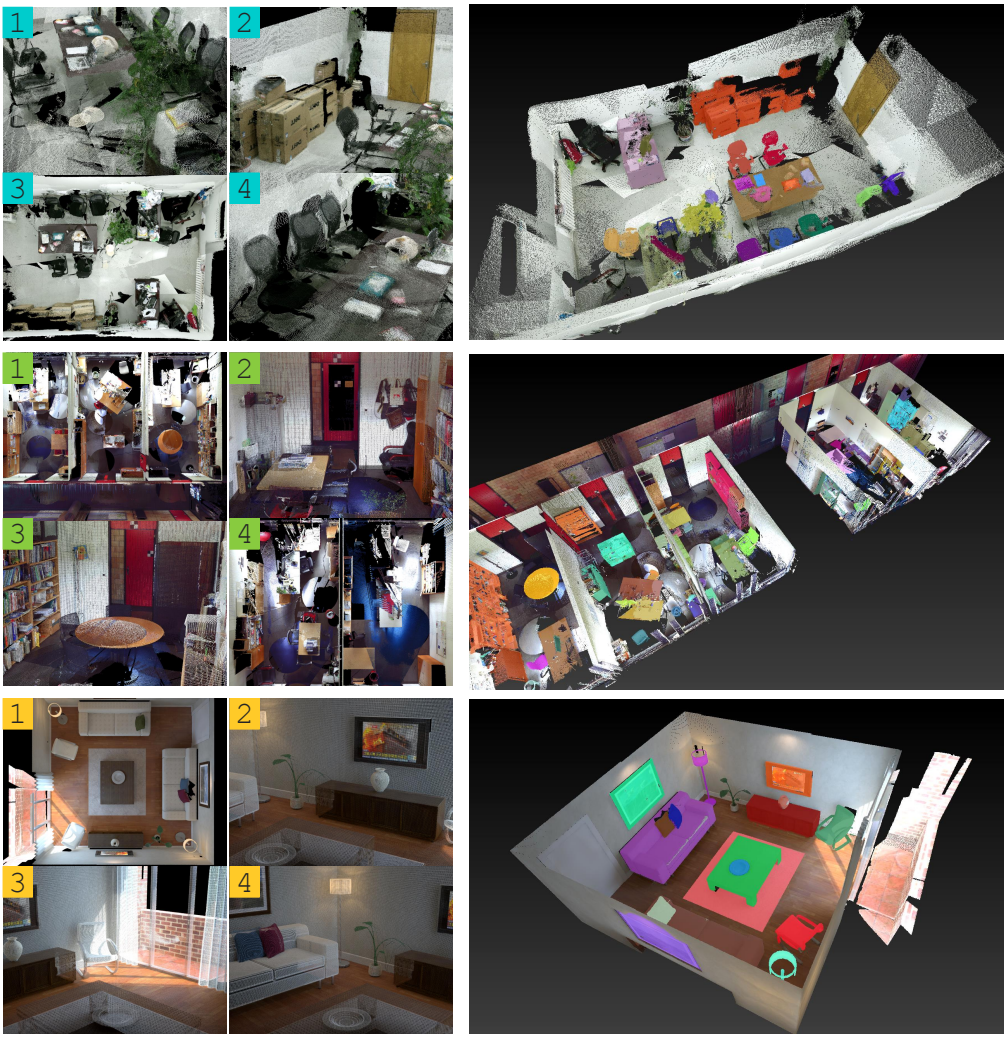


图 5.2　点云分割结果 (2)，每一行是一个场景，左侧是其中四个选择画筒笔画的视角，右侧为分割结果，不同的颜色表示不同的物体

第 6 章　总结与展望

6.1　本文工作的总结

点云分割问题即是把原始点云分割成几个具有语义信息的部分，每个部分代表了一个物体，它在计算机视觉、计算机图形学领域中都扮演着极其重要的作用，是很多后续算法得以进一步实现的重要基础。本文借鉴与前人的工作，提出了一个全新的交互式点云分割系统，该系统能做到实时交互的需求。

本文的主要贡献在于首先通过总结前人的工作，提出了一个更具鲁棒性的特征面片构建算法，该算法不但考虑了点云的几何特征，同时也很好地利用了点云的颜色信息，并且该算法采用了一个两层的 Region Growing 算法，从而使得构建的面片更符合真实的情况。

其次，作者根据对物体的支撑结构的观察，提出了一种新的物体内部的支撑关系，并给出了一个简单的面片间的支撑关系概率的计算公式，由此可以正确计算并识别出真实点云场景中绝大部分的支撑关系。

最后，本文提出了一个轻量级的相似物体搜索算法，该算法能很好的适用于本文提出的交互式分割系统，能在交互过程中实时地搜索并分割场景中相似的物体。

我们的系统通过进一步地优化用户界面、增加用户交互的多样性，从而对所有分割算法提供了容错性，用户能通过我们的系统对分割结果做任意程度的修改以保证最终分割结果的正确性。

同时，本文提出的算法也存在一定的局限性：

- 本文提出的支撑关系计算方法仅能适用于一些简单物体的支撑关系，例如沙发、椅子、桌子等等，因为本文对于支撑关系计算的公式假定了一个侧面最多只会支撑一个平面，或者只会被一个平面所支撑这个前提条件。然而对于一些具有复杂支撑结构的物体这个假设并不成立，比如书架。对于书架，它的几个侧面共同支撑了书架上的很多个平面，因此不能用本文提出的计算方法来计算这些面之间的支撑概率。因此，对于书架这类物体，用户就需要画较多的笔画才能成功地分割出整个物体。
- 本文提出的相似物体搜索中也有一个潜在的假设前提，那就是同一个物体

是连通的，这个假设在现实生活中满足，然而在低质量的点云中却并不一定满足。由于物体的材质 (如金属等强反光的材料) 或是物体的某一部分过于细小，会导致一些深度探测设备无法检测到物体所有部分的深度，从而产生点云的缺失。在这种情况下，我们的相似物体搜索便会遇到困难，会容易产生错误的搜索，从而增加用户为了修改这些错误搜索而额外产生的交互量。如图6.1，红色的椅子是需要搜索相似物体的模型，由于在点云中，椅子的背部和底座产生了分离，因此搜索出来的相似物体都是只有底座或者只有椅背。

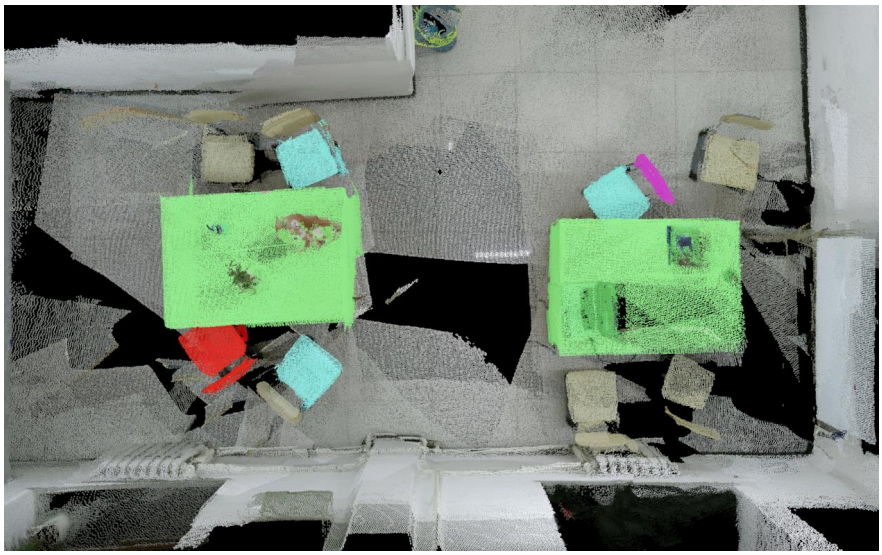


图 6.1　产生错误的相似物体搜索

6.2　未来工作展望

首先是上述提出的算法的局限性，在之后的工作中希望能得以解决，能设计出更具有普适性的支撑关系计算方法，并能提出更鲁邦的相似物体搜索算法用来解决点云中出现的各种情况，一种想法是可以我们可以优化原先的支撑关系计算方法，让图6.1这种情况出现时，我们能依然计算出椅子背部和椅子底部的支撑关系，这个支撑关系其实是很直观的，因为椅子背如果没有东西支撑，那么它就会掉下来，因此可以朝着这个方向去尝试用计算机的方式去解决这个问题。

除此之外，我们还可以继续优化此系统中的用户交互方式，我们发现对于

大场景，由用户亲自通过拖拽场景与点击小窗口进行视角的选择是一件费时费力的事情，一方面当场景非常大时，用户难以对整个场景有一个全局的把握，甚至连下一个需要分割的视角、房间在什么地方都毫无头绪，从而产生被遗漏的物体，另外一方面当场景中的点数量非常大时，场景的渲染速度会降低，场景的拖拽会变得非常缓慢，这会大大影响用户的交互体验，增加分割的时长。基于此，我们希望在未来能提出一个视角推荐系统，能自动地向用户推荐下一个需要分割的视角。

更进一步的，因为点云分割算法是很多后续算法的基础，因此我们还希望能在未来基于此系统的基础上设计更多的后续应用，比如提出墙面以后我们可以构建出出整个楼层的楼层设计图，再比如展开对于点云编辑、修复相关的工作。

插图索引

表格索引

参考文献

[1] Newcombe R A, Izadi S, Hilliges O, et al. Kinectfusion: Real-time dense surface mapping and tracking. Proc. of IEEE International Symposium on Mixed and Augmented Reality, 2011. 127–136

[2] Choi S, Zhou Q Y, Koltun V. Robust reconstruction of indoor scenes. Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2015. 5556–5565

[3] Rusinkiewicz S, Levoy M. Qsplat: A multiresolution point rendering system for large meshes. Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 2000. 343–352

[4] Rusu R B, Cousins S. 3D is here: Point Cloud Library (PCL). Proc. of IEEE International Conference on Robotics and Automation, 2011. 1–4

[5] Zheng B, Zhao Y, Yu J C, et al. Beyond point clouds: Scene understanding by reasoning geometry and physics. Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2013. 3127–3134

[6] Nguyen A, Le B. 3d point cloud segmentation: A survey. Proc. of IEEE Conference on Robotics, Automation and Mechatronics, 2013. 225–230

[7] Bhanu B, Lee S, Ho C, et al. Range data processing: Representation of surfaces by edges. Proc. of int. Pattern recognition conf, 1896

[8] Besl P J, Jain R C. Segmentation through variable-order surface fitting. IEEE Trans. Pattern Anal. Mach. Intell., 1988, 10(2):167–192

[9] Mattausch O, Panozzo D, Mura C, et al. Object Detection and Classification from Large-Scale Cluttered Indoor Scans. Computer Graphics Forum, 2014, 33(2):11–21

[10] Fischler M A, Bolles R C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM, 1981, 24(6):381–395

[11] Golovinskiy A, Funkhouser T. Min-cut based segmentation of point clouds. Proc. of IEEE Workshop on Search in 3D and Video (S3DV) at ICCV, 2009

[12] Li Y, Dai A, Guibas L, et al. Database-assisted object retrieval for real-time 3d reconstruction. Computer Graphics Forum, 2015, 34(2):435–446

[13] Boykov Y, Funka-Lea G. Graph cuts and efficient n-d image segmentation. Int. J. Comput. Vision, 2006, 70(2):109–131

[14] Boykov Y, Veksler O, Zabih R. Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell., 2001, 23(11):1222–1239

[15] Besl P J, McKay H D. A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell., 1992, 14(2):239–256

[16] Li H, Adams B, Guibas L J, et al. Robust single-view geometry and motion reconstruction. ACM Trans. Graph., 2009, 28(5):175:1–175:10

[17] Silberman N, Hoiem D, Kohli P, et al. Indoor segmentation and support inference from rgbd images. Proc. of European Conference on Computer Vision, 2012. 746–760

[18] Xiao J, Owens A, Torralba A. Sun3d: A database of big spaces reconstructed using sfm and object labels. Proc. of IEEE International Conference on Computer Vision, 2013. 1625–1632

[19] Chang A X, Funkhouser T, Guibas L, et al. ShapeNet: An Information-Rich 3D Model Repository. 2015, (arXiv:1512.03012 [cs.GR])

[20] Nan L, Xie K, Sharf A. A search-classify approach for cluttered indoor scene understanding. ACM Trans. Graph., 2012, 31(6):137:1–137:10

[21] Hinterstoisser S, Lepetit V, Ilic S, et al. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. Proc. of Asian Conference on Computer Vision, 2012. 548–562

[22] Chen K, Lai Y K, Wu Y X, et al. Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. ACM Trans. Graph., 2014, 33(6):208:1–208:12

[23] Lai K, Bo L, Ren X, et al. Detection-based object labeling in 3d scenes. Proc. of IEEE International Conference on Robotics and Automation, 2012. 1330–1337

[24] Silberman N, Sontag D, Fergus R. Instance segmentation of indoor scenes using a coverage loss. Proc. of European Conference on Computer Vision, 2014. 616–631

[25] Shen C H, Huang S S, Fu H, et al. Adaptive partitioning of urban facades. ACM Trans. Graph., 2011, 30(6):184:1–184:10

[26] Zhang H, Xu K, Jiang W, et al. Layered analysis of irregular facades via symmetry maximization. ACM Trans. Graph., 2013, 32(4):121:1–121:13

[27] Kim Y M, Mitra N J, Yan D M, et al. Acquiring 3d indoor environments with variability and repetition. ACM Trans. Graph., 2012, 31(6):138:1–138:11

[28] Shao T, Xu W, Zhou K, et al. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. ACM Trans. Graph., 2012, 31(6):136:1–136:11

[29] Yuan X, Xu H, Nguyen M X, et al. Sketch-based segmentation of scanned outdoor environment models. Sketch Based Interfaces and Modeling, 2005. 19–26

[30] Sedlacek D, Zara J. Graph cut based point-cloud segmentation for polygonal reconstruction. Proc. of International Symposium on Advances in Visual Computing, 2009. 218–227

[31] Rabbani T, Heuvel F, Vosselmann G. Segmentation of point clouds using smoothness constraint. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2006, 36(5):248–253

[32] Holz D, Behnke S. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. Proc. of International Conference on Intelligent Autonomous Systems, 2013. 61–73

[33] Lowe D G. Object recognition from local scale-invariant features. Proc. of IEEE International Conference on Computer Vision, volume 2, 1999. 1150–1157 vol.2

[34] Papadimitriou C H, Steiglitz K. The max-flow, min-cut theorem. Combinatorial Optimization: Algorithms and Complexity, 1998. 120–128

[35] Dinitz Y. Algorithm for solution of a problem of maximum flow in a network with power estimation. Doklady Akademii nauk SSSR, 1970. 1277–1280

[36] Orlin J B. Max flows in o(nm) time, or better. Proc. of the annual ACM symposium on Theory of computing, 2013. 765–774

[37] Furukawa Y, Curless B, Seitz S M, et al. Manhattan-world stereo. Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2009. 1422–1429

[38] Asha V, Bhajantri N U, Nagabhushan P. Glcm-based chi-square histogram distance for automatic detection of defects on patterned textures. Int. J. Comput. Vision Robot., 2011, 2(4):302–313

[39] Ikehata S, Yang H, Furukawa Y. Structured indoor modeling. Proc. of IEEE International Conference on Computer Vision, 2015. 1323–1331

[40] Handa A, Whelan T, McDonald J, et al. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. Proc. of IEEE International Conference on Robotics and Automation, 2014. 1524–1531

致　谢

感谢张松海副教授在本次毕业设计中对我的精心指导。

感谢胡事民教授自我进入计算机图形学与几何计算实验室以来对我的学习和研究工作上耐心详细的指导。

感谢杨晟师兄对本次毕业设计中部分算法的讨论和指导。

感谢曹炎培学长一直以来对我在计算机图形学领域中的指导与帮助。

感谢实验室中其他老师和同学对我的帮助。

感谢计 25 的同学们对我在学习和生活各方面上的鼓励和帮助。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：张捷　　日　期：2016.6.17