多 KINECT 实时室内动态场景重建技术

摘要

—摘要正文—

关键词：关键词 1，关键词 2，关键词 3

REAL-TIME DYNAMIC INDOOR SCENE RECONSTRUCTION BASED ON MULTIPLE KINECT

ABSTRACT

- Abstract Text –

Key words: Key word1, Key word2, Key word3

目录

第1章 绪论

近些年来增强现实一词逐渐走进了普通大众的视野。正如[1]中所述，有别于虚拟现实，增强现实（Augmented Reality）试图将（一般是由计算机所合成的）虚拟的物体、场景等信息嵌入到真实的环境中，以增强真实场景的信息。由于真实环境的复杂性和难以控制性[1]，增强现实技术的发展较虚拟现实技术要困难许多。即便在最近基于增强现实技术的游戏和应用已经逐渐丰富起来，增强现实技术本身仍然存在诸多有待克服的困难和限制。

本课题所研究的主题是场景重建技术（更一般地，可以称为三维重建技术（3D Reconstruction）），属于计算机视觉领域中的基础技术之一，在虚实结合和交互应用、人工智能机器人控制等方面有着重要作用。场景重建技术是利用传感器和计算机，通过对场景的纹理、深度等元数据进行软件分析和处理自动（或半自动）地构建场景的三维模型。

然而由于真实场景中普遍存在各类表面形状复杂的物件，再加上受限于传感器技术，场景重建一直存在诸多困难和取舍，例如效率、硬件成本和重建精度之间的相互制约。本课题以满足实时应用的效率约束为基础，以 Kinect 这一大众化商业传感器捕获场景数据，研究如何重建精度满足一般用户视觉要求的场景模型。

1.1 研究内容和意义

过去，场景重建技术在工业模拟、智能机器人等诸多方面都承担着十分重要的作用。然而，极长的计算时间和昂贵的设备极大地阻碍了这一技术的推广应用。因此，基于廉价设备的实时场景重建技术具有巨大的应用价值。

人们都想相信未来的办公室会是全数字化的，倘若依托数字设备，人们能够将异地的工作环境无缝融合，这将对人们的工作方式产生革命性的影响。这样一个基于远程协作的增强现实应用不但出现在各种科幻片中，也出现在过去数十年人们的研究主题里。在早些年就有[2]提出了一个对于未来办公室的设想，但是由于技术限制并没有予以实现。后来人们在显示、交互等方面的研究上取得进展，使得一些基本的设想得以实现，例如[3]中通过投影在现有模型表面来模拟远程用户的头部，[4]中展现了他们自己设计的一种三维显示屏幕，在远程会议中可以使得坐在不同位置的用户看到不同角度的画面。

而本课题所研究的实时动态场景重建技术可以在上述应用中发挥巨大作用。倘若将实时动态场景重建技术应用与我们的办公环境，通过对场景的建模并在异地三维展现，异地的工作环境的无缝融合将得以实现。另外，它将整个真实场景数字化，使得在这个场景中进行的一切交互都能得到计算机的反馈，从而实现一个全智能的虚实结合办公空间。

Kinect 设备于 2010 年 6 月正式发布，最初作为 Xbox360 游戏主机配套设备发售，用以提供对体感游戏的支持。Kinect 由于其成本低廉拥有广泛的应用潜力而在近两年备受人们关注。Kinect 包含一个深度传感器、一个彩色摄像头、另外还内置了麦克风阵列和用于追踪人体的步进电机。其深度传感器采用红外结构光技术，能够获得传感器前方 1.2 米至 3.5 米的有效深度数据[1]。Kinect 所具有的实时深度捕获能力使得实时场景重建技术的实现具备了硬件基础。

本课题最主要的研究重点在于实现基于 Kinect 满足实时交互要求的动态场景重建算法。在场景重建算法方面涉及图像去噪、可信点云构建以及基于点云的表面重建等多个问题；而

[1] Kinect 最大深度范围是 0.5 米至 9 米，在 1.2 米至 3.5 米之间的数据较为准确。

在实时计算方面涉及场景重建算法在 GPU 上的实现方案和快速近似算法的设计。为了得到较为完整的表面模型，本课题涉及同时使用多个 Kinect 设备，因此也涉及到对多组点云进行注册的研究。实时的动态场景重建使得当用户移动场景中的物体，做各种动作时，系统重建的模型能实时反应出这一变化，实现一些更加自然的交互手段。

1.2 相关工作和文献综述

由于三维重建技术的重要性，人们已经在这一方面研究了数十年，也提出了许多方法。这些方法各有优势和缺陷，适用于不同的应用场合。一般可以从原始场景数据的类型和重建结果的抽象等级两个方面对这些方法予以分类：

根据原始场景数据的类型主要分为两大类：基于深度数据的三维重建和基于纹理色彩数据的三维重建。大多数方法是基于深度数据的，例如[5-9]提出的重建方法都使用从特殊传感器（例如激光扫描仪等）获得的场景深度图重建三维模型的。另一类基于纹理色彩的重建有[10, 11]等，这些方法通过从图像中寻找特征点或特征区域来得到物体特征从而构建模型的。虽然基于纹理色彩的方法具有硬件成本低廉的优势（往往仅需要一个普通的摄像头），但物体表面的纹理色彩本身受环境光影响较大，很难设计一个鲁棒性高的算法，而且从本身就缺失第三维度信息的二维图像中反推三维信息的过程包含大量经验和假设，在一般情况下的重建精度往往不能满足要求，故基于深度数据的三维重建方法更为简单、稳定、易于推广应用。另外，还有部分工作是研究基于 RGB-D[1]的三维重建技术，如[12]提出了基于深度数据和纹理色彩数据分别重建三维模型然后相互结合的方法改进鲁棒性和精度。

根据[8]中所述，由于重建结果的应用不同，三维重建技术从重建模型的抽象等级角度可以分 3 个层次：高层次关注的是一种或者一类物体，结果可能并不会给出一个具体可见的模型，这一层次的建模主要用于物体识别，例如[13]；中层次使用诸如球、长方体等基本几何图元构建模型，这样构建出来的模型表面较为规整，有助于进行物体分割，例如[5]中所述就属于这一类；低层次则完全构建模型网格表面，不受任何假设的限制，能够重建出任意复杂表面的物体模型，用于对物体或场景精确建模以便重现，例如[6, 8, 9]中所提出的方法就属于这一类，他们通过从多个角度拍摄物体，获得大量深度图并结合在一起得到一个精细的三维模型。

近几年由于 GPGPU 的推广应用以及类似 Kinect 这样的具有实时深度获取的设备的出现，实时三维重建技术开始成为研究热点。[9, 14]中所使用的重建算法虽然都是几年前就提出来的，但他们利用 Kinect 设备和 GPU 运算达到了实时重建的效率，这在三维重建领域是一个重要突破。美中不足的是，它们仍旧不能对动态场景进行实时重建，当场景中的一个物体位置发生变化时，它需要用数秒的时间才能将这个物体在原来位置上的模型彻底从重建出的场景中消除，并在新的位置重建。本课题根据实际需求改进了上述算法，实现了对动态场景实时重建。

1.3 论文组织结构

从论文的第二章开始将详细介绍本课题所进行的工作和取得的成果，将按照先总体后局部的顺序详细阐述设计思想和算法原理，然后通过实验和测试给出本课题所取得的结果以及尚存的不足之处，最后总结全篇并对未来的工作进行思考。

第二章主要阐述了本课题所实现的实时动态场景重建系统采用的硬件设备及布局设计、使用的软件平台和框架、核心重建算法的主要步骤等。

第三章至第五章详细阐述了重建算法的每个步骤，包括思想和原理以及主要的公式推导

[1] 即 RGB & Depth，纹理色彩数据和深度数据的结合

等。

第六章给出了基于实时动态场景重建技术远程协作平台原型的设计和实现，并展现该平台的运行效果。

第七章给出了对系统进行的一系列实验以及结果，包括重建结果的精度、重建方法的鲁棒性以及性能等。

第八章总结。首先根据对第七章的实验结果的分析，总结本课题所取得的成果以及仍旧存在的问题。然后针对仍旧存在的问题提出对下一步工作内容的设想。

1.4 本章小结

本章主要介绍了本课题所进行的研究的基本情况。首先对研究领域进行了概要介绍，其次从场景重建技术目前存在的问题和实时动态场景重建技术所具有的意义引出本课题所研究的内容和主要目的。紧接着分类列举了过去二十年内人们在三维重建技术上进行的工作，它们的特点，并介绍了最新的研究进展和趋势，阐明选择 Kinect 设备和基于 GPU 的算法实现的原因。最后简要说明了本论文的组织结构和每章的主要内容。

第2章 总体系统设计

2.1 硬件系统设计

本系统采用 Kinect 设备作为输入设备，经由普通的个人计算机进行计算并得到场景的三维网格模型。本系统不需要其他额外的传感器，因此硬件平台的部署相对比较简单。

与[9, 14]中所采用的手持 Kinect 方式不同的是，本系统使用 3-4 个固定摆放在场景周围不同方位的 Kinect 设备（图 1），这是由应用环境所决定的。在远程协作的工作平台中，用户不可能手持 Kinect 进行工作，也不可能手持 Kinect 对着自己[1]。另外，固定 Kinect 设备能简化重建动态场景的算法，也使重建结果更加稳定。

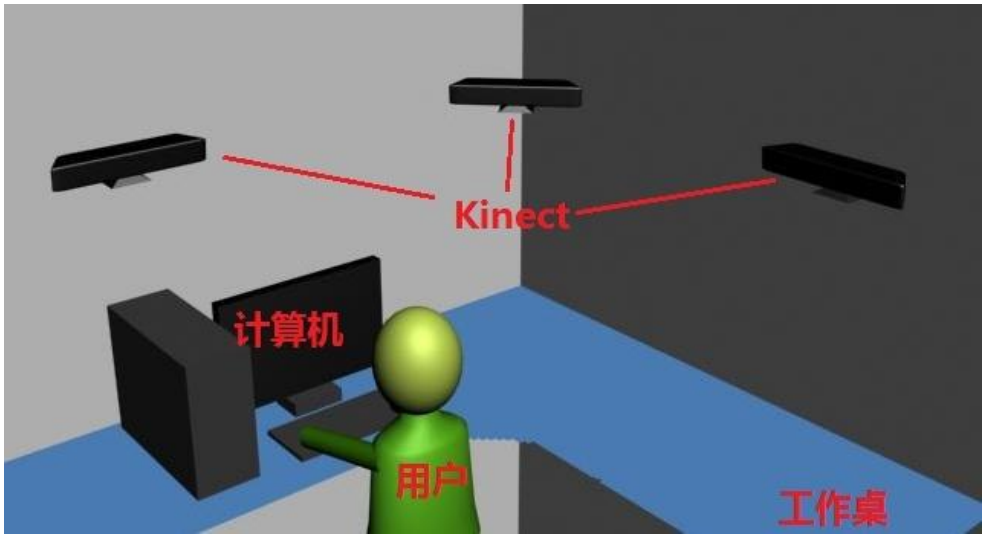

图 1 硬件系统布局示意图

本系统在部署完成后需要进行一次标定操作，包括对所有 Kinect 设备分别进行内参标定，对所有 Kinect 设备统一进行外参标定。采用棋盘格标定，具体阐述见下文 3.2 节。

2.2 软件平台和核心架构设计

本系统的设计运行平台为 Windows 7 操作系统，采用 Nokia QT 作为窗口显示框架、OpenNI 驱动 Kinect 设备并获取深度和纹理色彩数据，使用 DirectX 11 进行 GPU 运算和渲染。

虽然微软为 Kinect 设备发布了 Kinect SDK 开发库以及驱动程序，但该开发库更多地注重人体识别、追踪和语音识别等体感游戏需要用到的技术，在提供原始数据的接口和深度数据与纹理色彩数据的校准的方面不如 OpenNI，且在实际测试中较之 OpenNI 需要占用更多资源[2]，故本系统更适合使用 OpenNI 框架作为驱动 Kinect 的底层接口。

由于本系统的功能并不多，重点是在于核心功能——实时动态场景重建算法的实现，故本系统的架构设计也比较简单。图 2 描述了本系统的核心架构图，包含关键类和它们之间

[1] 由于 Kinect 有最近距离限制（约为 0.5 米），用户手持 Kinect 对着自己的距离过近以至于 Kinect 无法捕获那么近距离的表面深度信息。

[2] 在作者使用的计算机上（具体配置见下文第 7 章），使用 Kinect SDK 获取数据并做深度图—纹理色彩图校准的执行性能约为 15fps，而使用 OpenNI 执行相同操作则能够达到 25fps 左右。

的关系。从该图中也可以很容易地看出本系统的主要执行流程。

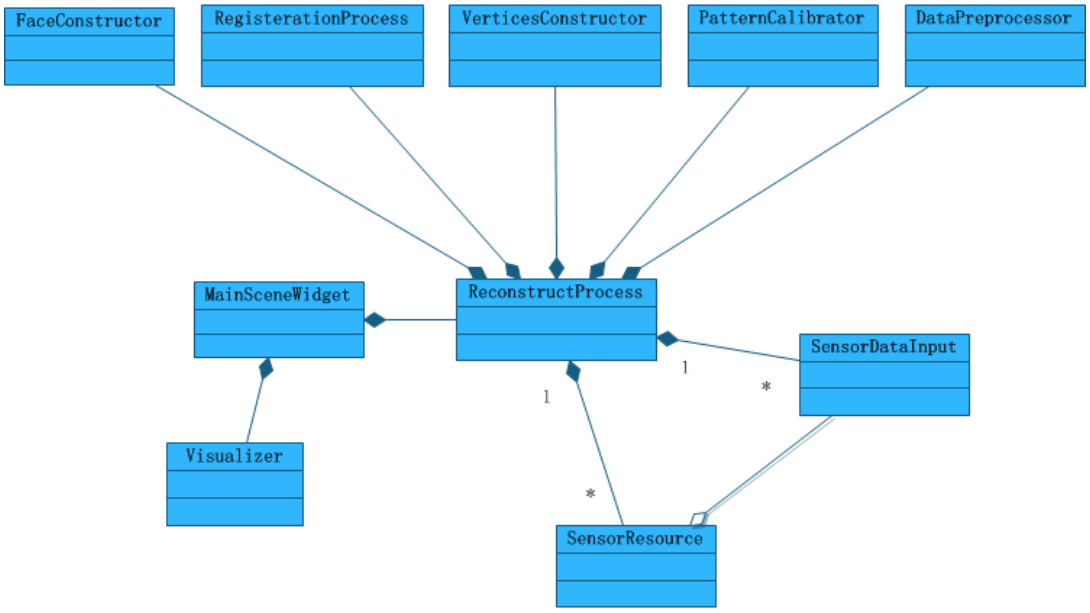

图 2 系统核心架构概览

2.3 算法流程概览

本节将概要性地介绍实时动态场景重建算法的执行步骤。从下一章开始，将对每个步骤进行详细阐述。上文第 1.2 节中提到了许许多多和三维重建算法有关的工作。虽然各论文中都提到了不同的方法，但凡是基于深度数据的三维重建算法，它们的主要流程都是一致的。一般基于深度数据的三维重建算法按先后顺序可以分为获取数据（深度数据和纹理色彩数据等）、对数据进行预处理、根据深度数据构建三维点云、对多组点云进行注册、从点云构建网格表面（图 3）。

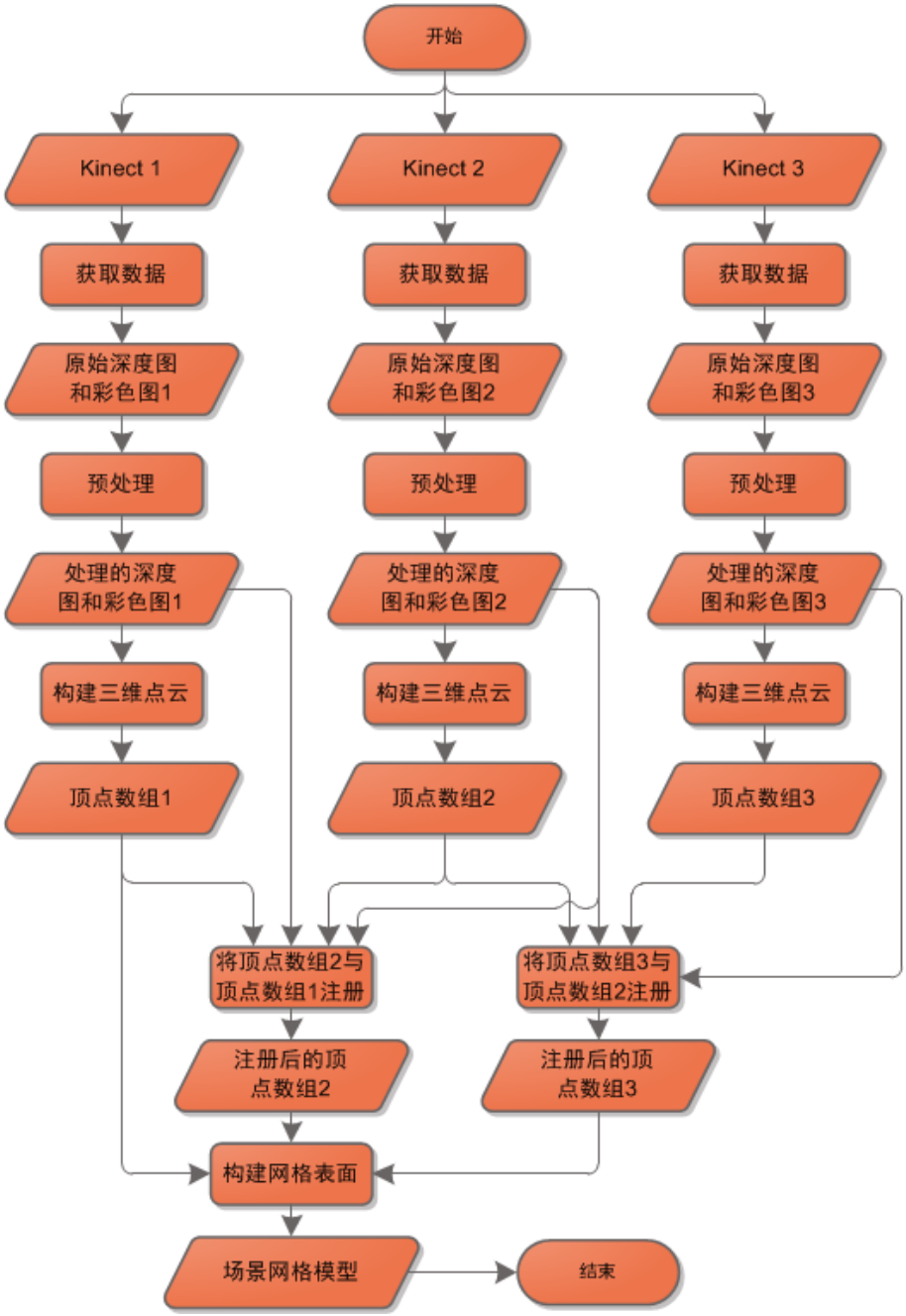

图 3 实时动态场景重建算法的主要流程（以使用 3 台 Kinect 为例）

对数据进行预处理：由于传感器的原理不同，获取的数据也存在各种不同的误差分布特性。例如在[6]中提到，光学扫描仪的误差往往与物体表面法向量和光轴所成的夹角有关，并且在物体的边缘获取的数据具有较强的不确定性。根据这些误差分布特性，需要对原始的深度数据进行特殊的预处理以降低噪声的干扰，从原始数据的层面减少误差。

根据深度数据构建三维点云：该步骤输入预处理后的深度数据和色彩纹理数据，主要执行以下操作：根据传感器的内参和外参信息计算深度数据中的某一像素对应在三维场景坐标下顶点位置；根据该像素对应的色彩纹理数据标示该顶点的颜色；根据该顶点及其周围的邻居顶点[1]计算该顶点的法向量。将几组传感器获得的数据中的所有像素均进行上述计算后，可以得到对应的几组三维点云。这时将他们可视化已经可以看出一定的重建效果，但是由于

[1] 在本文中，除特别说明外，一个顶点的邻居顶点都是指，在有效场景区域内，由其在深度图中对应像素的 8 邻域像素生成的顶点，且它们之间的距离小于某一阈值。

基于棋盘格的外参标定并不十分精确，从结构和纹理上可以看出多组点云并不能完全匹配。

对多组点云进行注册：该步骤旨在减小多组点云之间存在的变换偏差。注册即指将两组或两组以上的点云拼合在一起，使它们的重合区域匹配度最高。许多有关三维重建的论文（例如[9, 14]）都使用迭代最近点算法（Iterative Closest Point，以下简称 ICP）完成注册工作。ICP 算法最初是在[15]中被提出的。由于[14]中所提到的三维重建算法的输入中包含纹理色彩图，故其综合顶点的位置和颜色信息对 ICP 算法作了改进，增强了它的鲁棒性。本文中所使用的注册算法便是基于上述改进的 ICP 算法，同时提出了自己的改进方案以加快该算法的执行速度。

从点云构建网格表面：该步骤是重建算法的最后一步。虽然多组点云注册完成后，一个场景的重建基本完成，但是一些离散的点无论是用于可视化还是用于虚实物体接触检测都是远远不够的。该步骤即是从三维点云中构建较为完整的表面网格模型，具体涉及顶点合并、空洞填补和三角面片生成等算法。

2.4 本章小结

本章主要从总体层面阐述了本课题所研究的实时动态场景重建系统的实现方法。首先阐述了该系统所用到的硬件设备和布局设计，并给出了如此设计的理由。其次，简要介绍了系统所使用的软件框架、工具库等，并通过图示的方式描述了系统的主要工作流程。最后对本课题的主要研究内容—实时动态场景重建算法的执行步骤作了概要性的描述。

第3章 基于彩色图和深度图构建三维点云

3.1 彩色图和深度图的预处理

在上文 2.3 节中已经提到，由于传感器的原理不同，获取的数据会产生不同分布特征的误差。然而不管是何种传感器，获取的数据的误差是不可避免的。Kinect 上的深度传感器基于结构光技术，属于光学传感器的一种，故其获得的深度数据的误差符合[6]中所描述的光学传感器的误差分布特征。另外，由于硬件层面的原因，Kinect 获得的彩色图呈现出明显的条纹状噪声（图 4）。因此，在使用这些数据生成三维点云前，首先需要先进行二维图像层面的降噪处理。最后，预处理操作还根据 Kinect 的内参数据补偿彩色图和深度图的畸变，使后面的步骤可以不需要考虑畸变因子的影响，简化逆投影变换的运算。本节将针对彩色图和深度图分别阐述所使用的降噪算法。

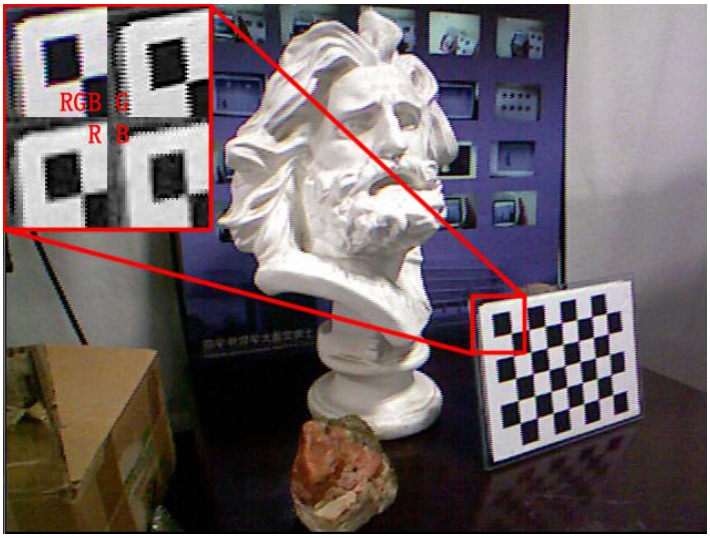

图 4 Kinect 获得的原始彩色图

从左上角的局部放大图中可以看出有明显的横向条纹状噪声

3.1.1 彩色图的预处理算法

根据上文所述，Kinect 获得的彩色图呈现出明显的条纹状噪声。虽然从总体上看这一噪声并不明显，但是它会影响到外参标定中的棋盘格角点提取的准确度和最近点查找算法的精度。

为了进一步分析条纹状噪声的特性，将彩色图的 RGB 三通道分别分开显示效果如图 4 右上角所显示的那样。从三通道显示的结果可以看出，G 通道的横向条纹较为明显，R 通道不太明显，而 G 通道没有横向条纹，但是存在轻微的纵向条纹。同时，这些条纹都呈现出各行分错的特性。

因为条纹状噪声特性简单，故降低这一噪声的方法也较为简单。对于原图像的三个通道，分别针对奇偶行（列）相对原图像做不同程度的偏移，可以明显降低条纹噪声（图 5）。

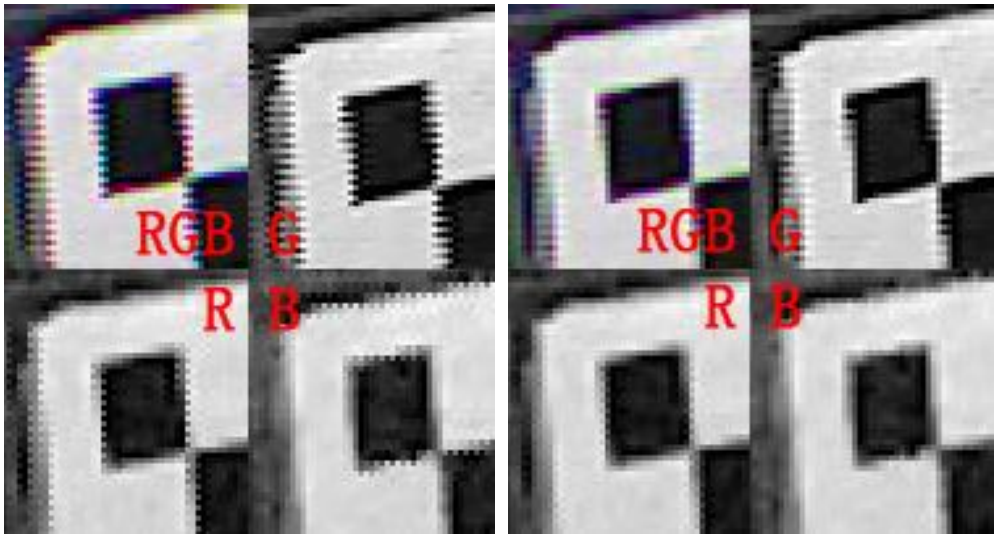

图 5 彩色图预处理的效果

左图：处理前，条纹明显；右图：处理后，条纹几不可见

3.1.2 深度图的预处理算法

Kinect 的深度传感器属于光学传感器中的一类。因此它所产生的误差收到多个因素的影响，例如光强度（对于 Kinect 来说是环境红外光场的强度），距离等。但是由于室内的环境红外光强一般都比较弱，而且在一定的距离范围内误差随距离的变化并不明显，因此本课题并不考虑深度误差受光强和距离的影响因素，而是着重考虑了和场景中物体表面结构有关的一些重要的影响因素。另外 Kinect 获得深度数据也具有较强的噪声。

在本节，将首先介绍深度图的降噪处理算法。在后文 3.4 节中将考虑其他的误差特性。

本课题中，降噪处理使用高斯滤波的变种。该处理算法主要步骤如算法 1 所示。

算法 1 深度图预处理算法

输入：原始深度图d_{in}，格式为 16 位单通道

输出：预处理后的深度图d_{out}，格式与输入相同

过程：

初始化输出图像，所有像素值归 0

对于图像中的每个像素点$\boldsymbol{p}$：

　　遍历以p为中心的5×5区域，对于每个像素点$\mathbf{q}$：

　　　　倘若$|d_{in}(\boldsymbol{p})-d_{in}(\boldsymbol{q})|>t_d$：$d_{out}(\boldsymbol{p})=\omega(\boldsymbol{q}-\boldsymbol{p})d_{in}(\boldsymbol{q})$；

　　　　否则：$d_{out}(\boldsymbol{p})=\omega(\boldsymbol{q}-\boldsymbol{p})d_{in}(\boldsymbol{p})$；

　　归一化$d_{out}(\boldsymbol{p})$；

结束

该变种在进行高斯模糊的基础上仍保留了强边缘的清晰。经过这一处理的深度图所构建的三维模型不会在边缘部分产生毛刺（图 6）。

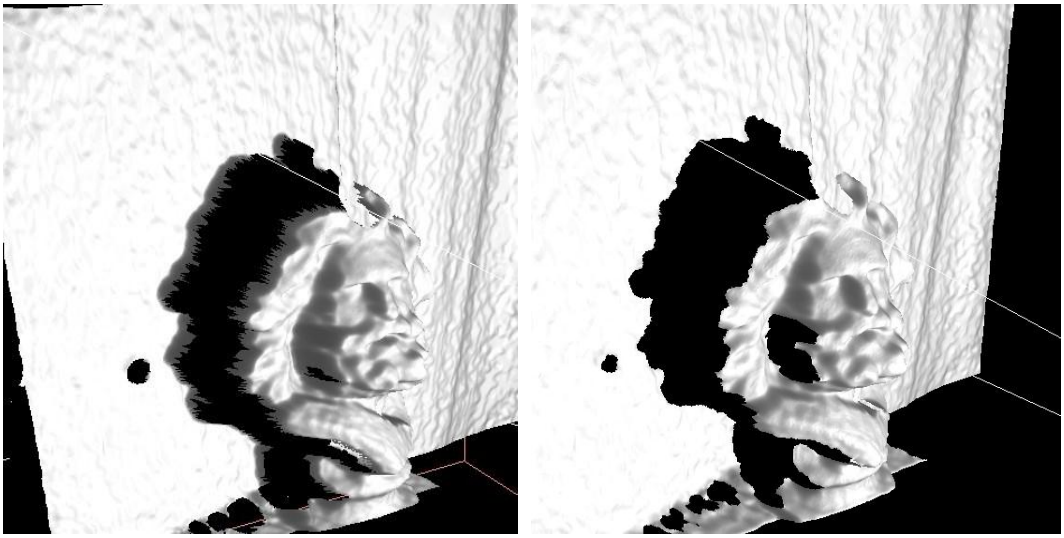

图 6 高斯滤波和本课题设计的处理算法对深度数据的处理效果对比
左图：基于使用高斯滤波后的深度图重建的人物雕像模型边缘毛刺明显；右图：基于使用本课题设计的处理算法处理后的深度图重建的人物雕像模型边缘没有毛刺

3.2 基于棋盘格的 Kinect 标定

在进行三维点云的构建前，首先需要知道 Kinect 设备的内参和外参数据。在计算机视觉领域中，内参是指光学传感器的内部参数，该参数与硬件有关，而与传感器在场景中所处的方位无关；外参则是用于表征传感器在场景中所处的方位的参数。

3.2.1 内参标定

根据计算机视觉领域常用的针孔摄像机模型（图 7），在场景中的某个点通过与摄像机光心的连线投影到摄像机平面，这之间的投影关系满足

$$s\boldsymbol{p}' = A\boldsymbol{P}' \tag{1}$$

其中$\boldsymbol{P}' = (X, Y, Z)^T$为一个点在摄像机坐标系下的三维坐标值，$\boldsymbol{p}' = (u, v, 1)^T$为该点投影到摄像机图像平面上的齐次坐标，$A = \begin{pmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{pmatrix}$为摄像机内参矩阵。

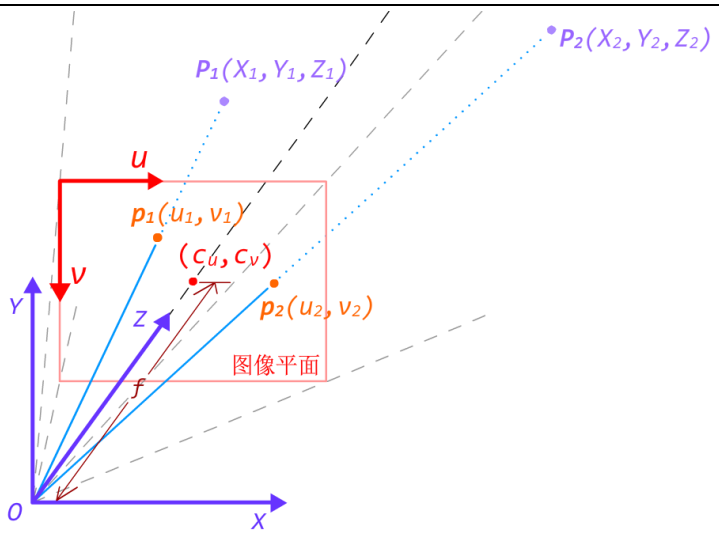

图 7 针孔摄像机模型

上述公式仅适用于理想摄像机。在实际中，考虑到实际结构与理想模型的差异和硬件制造工艺上的误差，上述投影变换关系还应加入畸变因子的影响。实际的投影变换公式如下：

$$\begin{cases} u = f_x\left(x'(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_1x'y' + p_2(r^2 + 2x'^2)\right) + c_x \\ v = f_y\left(y'(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_2x'y' + p_1(r^2 + 2y'^2)\right) + c_y \end{cases} \tag{2}$$

其中$x' = \frac{X}{Z}, y' = \frac{Y}{Z}$，$r^2 = x'^2 + y'^2$。

公式⑴中的A和公式⑵中的$(k_1, k_2, p_1, p_2, k_3)^T$合在一起被称为传感器的内参数据。由于内参数据仅和硬件本身有关，故仅需在系统部署前求得一次即可。本课题选用基于棋盘格的张正友标定法对使用的 Kinect 设备进行内参标定。张正友标定法是由张正友在[16]中所提出的摄像机内参标定方法。由于该方法使用起来简单快速，故在提出后即被广泛应用。OpenCV 函数库还实现了针对该方法的内参计算函数可供开发者直接调用。

3.2.2 外参标定

外参标定用以决定 Kinect 设备在场景中放置的方位。由于本课题使用多个 Kinect 设备重建场景，由不同 Kinect 设备所提供的数据中构建的多组点云必须在一个统一的三维坐标系下才能拼合在一起，所以预先需要得到所有 Kinect 设备相对某一三维坐标系的方位。该三维坐标系可以随意定义，故本课题中使用棋盘格标定板定义该三维坐标系，以下称之为场景坐标系（图 8）。

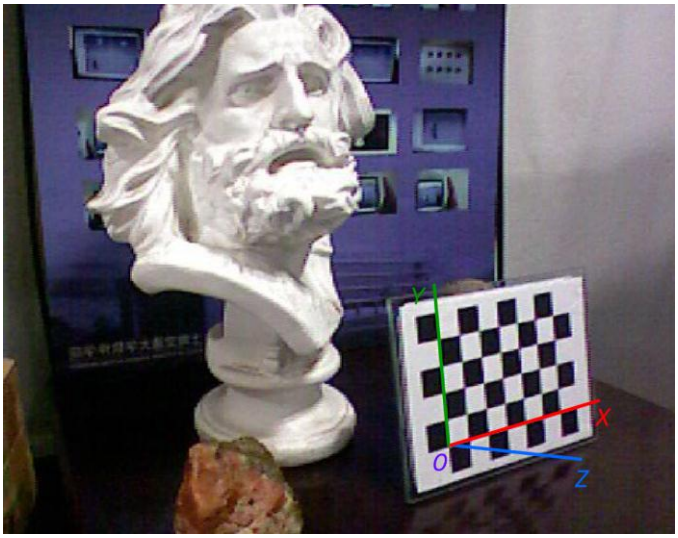

图 8 使用棋盘格标定板定义场景坐标系

外参可以用一个 4 阶矩阵M_4表示，它表示从场景坐标系到 Kinect 设备的坐标变换。传统的摄像机外参标定法可以在已知摄像机内参的条件下从棋盘格角点在场景坐标系中的坐标及它们投影在摄像机图像平面中的坐标之间的多组一一对应关系求得摄像机的外参矩阵。本课题中所采用的外参标定方法利用了 Kinect 设备的深度数据，其标定结果比传统方法更加精确。该方法较之传统方法的区别在于使用 Kinect 坐标系下的三维坐标而非图像平面内的二维坐标。由于 Kinect 内参已知，根据某个像素的深度值可求出其对应在 Kinect 坐标系下的三维坐标。利用 4 个（或 4 个以上）不共面的点在场景坐标系下的三维坐标和 Kinect 坐标系下的三维坐标可以列出线性方程组：

$$M_4(\boldsymbol{P_1} \quad \boldsymbol{P_2} \quad \dots \quad \boldsymbol{P_n}) = (\boldsymbol{P_1}' \quad \boldsymbol{P_2}' \quad \dots \quad \boldsymbol{P_n}') \tag{3}$$

其中$\boldsymbol{P_1}, \boldsymbol{P_2}, \dots, \boldsymbol{P_n}$为 n 个棋盘格角点和根据角点计算的不在 z 轴平面上的点（为了满足不共面约束）在场景坐标系下的坐标，$\boldsymbol{P_1}', \boldsymbol{P_2}', \dots, \boldsymbol{P_n}'$为它们在 Kinect 坐标系下的坐标。通过最小二乘法即可求解系数矩阵M_4。显示了外参标定过程中的角点检测和结果验证。

图 9 外参标定过程

上图：对彩色图像做角点检测识别棋盘格；下图：外参计算结果的验证

3.3 基本三维点云的构建

场景重建算法的第二步是从深度图和彩色图构建三维点云。对于每个 Kinect 设备，都会由该步骤构建一组对应的三维点云。该点云中的每个顶点包括位置、法向量和颜色、权重等信息。本节将阐述如何由深度图和彩色图生成上述信息中的位置、法向量和颜色，权重的生成将单独列入下一节阐述。

在该步骤中，并没有顶点的颜色做过多的处理，仅是将其设为对应的像素在彩色图中的颜色。

由深度图生成顶点位置涉及两个坐标变换。首先根据内参将深度图中的像素做逆投影变换，求得 Kinect 坐标系下的三维坐标。再根据外参的逆矩阵将该三维坐标变换到场景坐标系下，即得到对应顶点的位置。由于在预处理过程中已经根据内参中的畸变因子进行了畸变补偿，故此处的逆投影变换不需要再考虑畸变因子。整个变换求解过程如公式⑷所示。

$$\boldsymbol{P} = M_4'\boldsymbol{P}' = M_4'\begin{pmatrix} \frac{f_x}{d(u,v)}(u-c_x) \\ \frac{f_y}{d(u,v)}(v-c_y) \\ d(u,v) \\ 1 \end{pmatrix} \tag{4}$$

最后，顶点的法向量由该顶点和其周围的邻居顶点的位置计算而得。假设由该组点集生成的网格模型中，任意三个邻居顶点之间都会生成一个三角面片。根据一般网格模型的法向量计算方法，每个顶点的法向量等于以它为顶点的所有三角面片的法向量之和（图 10）。

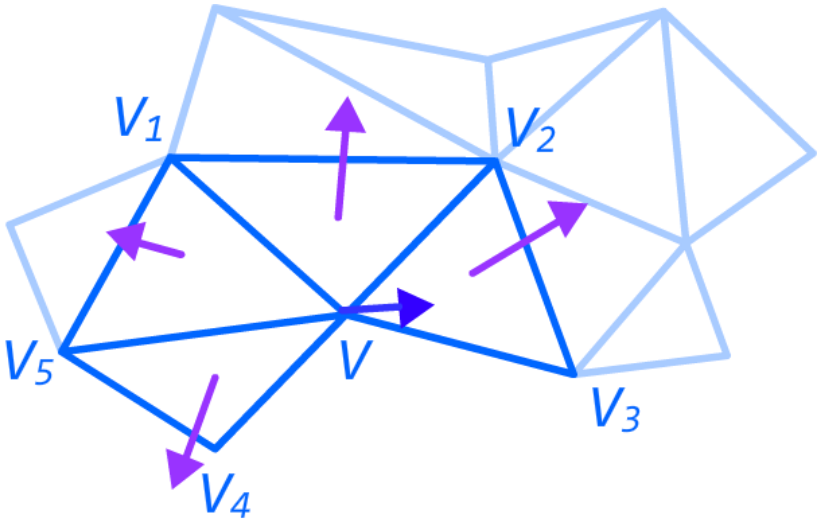

图 10 顶点法向量由所有以它为顶点的三角面片的法向量加和而得

整个构建算法如算法 2 所示。

算法 2 三维点云构建算法

输入：彩色图c（格式为 8 位三通道）和深度图d（格式为 16 位单通道）
输出：一组顶点元素，其中每个元素包含位置$\boldsymbol{P}$、法向量$\boldsymbol{N}$、颜色$\boldsymbol{C}$
过程：
对于图像平面中的每个像素$\boldsymbol{p}$：
　　添加一个新的顶点元素V；
　　倘若$d(\boldsymbol{p}) > d_{max}$：
　　　　$V.\boldsymbol{P} =$ 无效坐标；
　　否则：
　　　　由公式⑷计算$V.\boldsymbol{P}$；
　　　　$V.\boldsymbol{C} = c(\boldsymbol{p})$；
对于图像平面中除第一行和第一列外的每个像素$\boldsymbol{p} = (u, v)^T$（对应顶点$V$，记场景坐标为$\boldsymbol{P}$）：
　　倘若$V.\boldsymbol{P}$是有效坐标：
　　　　定义$\boldsymbol{p_1} = (u-1, v-1)^T, \boldsymbol{p_2} = (u, v-1)^T, \boldsymbol{p_3} = (u-1, v)^T$，
　　　　　　对应场景坐标$\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{P_3}$；
　　　　对于$\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{P_3}$中任意两个符合$\boldsymbol{P}$邻居顶点要求的顶点$\boldsymbol{P_a}, \boldsymbol{P_b}$($\boldsymbol{PP_aP_b}$逆时针排列)：
　　　　　　记$\boldsymbol{P_a}, \boldsymbol{P_b}$对应的顶点为$V_a, V_b$；
　　　　　　构建一个三角面片T；
　　　　　　$T.\boldsymbol{N} = (\boldsymbol{P_b} - \boldsymbol{P}) \times (\boldsymbol{P_a} - \boldsymbol{P})$；
　　　　　　$V.\boldsymbol{N} = V.\boldsymbol{N} + \frac{T.\boldsymbol{N}}{|T.\boldsymbol{N}|}$，$V_a.\boldsymbol{N} = V_a.\boldsymbol{N} + \frac{T.\boldsymbol{N}}{|T.\boldsymbol{N}|}$，$V_b.\boldsymbol{N} = V_b.\boldsymbol{N} + \frac{T.\boldsymbol{N}}{|T.\boldsymbol{N}|}$；
结束

3.4 点的权重和三维点云中的散点去除

由于 Kinect 深度传感器内在的误差特性，其获得的深度图中每个像素的可信度都不一样。在重建过程中将该可信度考虑进来可以使得重建结果更加精确。因此，每个顶点都额外包含一个权重属性。在构建点云的过程中，根据 Kinect 深度传感器的误差特性计算每个顶点的权值。

B. Curless 等人在[6]中总结他人工作并指出基于光学的深度传感器往往具有如下特性：

（1）误差和表面法向量与传感器光轴的夹角有关，误差随夹角增大而增大。

（2）靠近物体边缘的深度数据具有很强的不稳定性。

Kinect 深度传感器基于结构光技术，因此其特性也满足上述两点。本课题在对 Kinect 获得的深度图进行分析后，得出与和上述两点相对应的结论：

（1）深度数据可信度与该深度数据所生成顶点的法向量与 Kinect 光轴夹角的余弦成正比。即

$$\omega(\boldsymbol{P}) = \cos\frac{\boldsymbol{N}(\boldsymbol{P}) \cdot \boldsymbol{v_0}}{|\boldsymbol{N}(\boldsymbol{P})||\boldsymbol{v_0}|} \tag{5}$$

（2）在物体边缘顶点[1]所对应像素的7×7邻域内，深度数据是不可信的。即

$$\omega(\boldsymbol{P}) = 0，\ \boldsymbol{P} \in \{\boldsymbol{P_i} | \boldsymbol{p_i}\text{的 } 7 \times 7 \text{ 邻域内有至少一个边缘顶点，} i = 1 \ldots n\} \tag{6}$$

[1] 边缘顶点是指那些邻居顶点不足 8 个的顶点。即倘若某个顶点邻居顶点满 8 个，则它是一个内部顶点。

根据以上两点结论，在求得所有顶点的位置和法向量后，即可计算它们的权重。加入权重计算后的完整三维点云构建算法如算法 3 所示。

算法 3 加入顶点权重计算的三维点云构建算法

输入：彩色图c（格式为 8 位三通道）和深度图d（格式为 16 位单通道）
输出：一组顶点元素，其中每个元素包含位置$\boldsymbol{P}$、法向量$\boldsymbol{N}$、颜色$\boldsymbol{C}$、邻居个数n和权值ω
过程：
对于图像平面中的每个像素$\boldsymbol{p}$：
　添加一个新的顶点元素V；
　倘若$d(\boldsymbol{p}) > d_{max}$：
　　$V.\boldsymbol{P} =$ 无效坐标；
　否则：
　　由公式⑷计算$V.\boldsymbol{P}$；
　　$V.\boldsymbol{C} = c(\boldsymbol{p})$；
对于图像平面中除第一行和第一列外的每个像素$\boldsymbol{p} = (u, v)^T$（对应顶点$V$，记场景坐标为$\boldsymbol{P}$）：
　倘若$V.\boldsymbol{P}$是有效坐标：
　　定义$\boldsymbol{p_1} = (u-1, v-1)^T, \boldsymbol{p_2} = (u, v-1)^T, \boldsymbol{p_3} = (u-1, v)^T$，
　　对应场景坐标$\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{P_3}$；
　　对于$\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{P_3}$中任意两个符合$\boldsymbol{P}$邻居顶点要求的顶点$\boldsymbol{P_a}, \boldsymbol{P_b}$($\boldsymbol{P}\boldsymbol{P_a}\boldsymbol{P_b}$逆时针排列)：
　　记$\boldsymbol{P_a}, \boldsymbol{P_b}$对应的顶点为$V_a, V_b$；
　　构建一个三角面片T；
　　$T.\boldsymbol{N} = (\boldsymbol{P_b} - \boldsymbol{P}) \times (\boldsymbol{P_a} - \boldsymbol{P})$；
　　$V.\boldsymbol{N} = V.\boldsymbol{N} + \frac{T.\boldsymbol{N}}{|T.\boldsymbol{N}|}$，$V_a.\boldsymbol{N} = V_a.\boldsymbol{N} + \frac{T.\boldsymbol{N}}{|T.\boldsymbol{N}|}$，$V_b.\boldsymbol{N} = V_b.\boldsymbol{N} + \frac{T.\boldsymbol{N}}{|T.\boldsymbol{N}|}$；
　　$V.n = V.n + 1$，$V_a.n = V_a.n + 1$，$V_b.n = V_b.n + 1$；
对于图像平面中的每个像素$\boldsymbol{p}$（对应顶点V）：
　倘若$V.\boldsymbol{P} =$ 无效坐标或$\boldsymbol{p}$的7×7邻域内存在某个像素，其对应顶点的邻居个数不等于 8：
　　$V.\omega = 0$；
　否则：
　　根据公式⑸计算$V.\omega$；
结束

需要注明的是，所有权值为 0 的顶点被视为无效顶点，也即不再参与到在后续的计算中。同时可以发现，经过上述算法计算后，所有小于等于8×8的独立点集将被移除，这样也就去除了三维点云中由于各种外界因素或内部噪声所造成的散点集。

3.5 本章小结

本章详细阐述了本课题所设计的场景重建算法的第一个步骤。该步骤的输入为从 Kinect 接收到的原始深度图和彩色图，输出为包含位置、法向量、颜色和权重信息的顶点数组。该步骤主要流程为：原始数据的预处理和三维点云的构建。对于从每个 Kinect 设备接收到的数据，都要分别经过这一步骤的运算。所得到的多组点云将作为场景重建算法的下一个步骤的输入。该步骤的重点在于通过预处理和权重计算降低由于各种外在或内在因素而产生的误差的影响，提高最后的重建结果的精度。另外该步骤中所需要的 Kinect 内参和外参数据的求取方法也在本章中予以阐述。

第4章 基于改进的 ICP 算法注册多组三维点云

虽然事先已经对所有的 Kinect 设备进行了外参标定，使得对于每个 Kinect 设备数据分别构建出的点云都在统一的场景坐标系下，但是外参标定过程中的微小误差会在构建三维点云时的逆投影变换放大。在实践中发现，几组点云在同一坐标系下被可视化后可以很轻易地发现它们之间存在的偏差，离场景原点越远偏差越大。这使得离场景原点较远的那些区域的重建结果十分地不精确。因此仅通过外参标定所得到数据统一这些点云的坐标系是不够的。

常规的三维重建算法，为了重建一个完整的模型，也都会涉及到将多个传感器数据组合在一起的操作。这需要找到一个三维变换，使得一组点云（源）在经过变换后和另一组点云（目标）重合部分的偏差最小。ICP 算法即是用来求这个三维变换的算法。它最初是在[15]中被提出的，而后被许多三维重建工作引用。例如[9, 14]中都提到了使用该算法注册点云。本课题所设计的注册算法也是基于 ICP 算法思想，结合了[14]中提出的改进思想和本课题所提出的改进思想。

4.1 ICP 算法的基本原理

ICP 算法的基本思想其实很简单。它是一个迭代改进算法，每个迭代可以分为三个步骤：

（1） 为源中的每个点在目标中寻找离它最近的点，形成一组点对；

（2） 求一个三维变换，使得源中的每个点经过该变换后与其在目标中对应的那个点的距离平方和最小；

（3） 对源应用该三维变换。

迭代终止条件为

（1） 源和目标之间的偏差小于预先设定的阈值。算法报告说这两组点是可重合的；

或者

（2） 迭代次数达到了设定的最大迭代次数。算法报告说这两组点是不可重合的。

作为每个迭代的第一个步骤，寻找点对的过程事实上是一个最近邻搜素算法。虽然最近邻搜索算法已经经过了数十年的研究，也已经存在了许多成熟的算法，比如 K-d 树、八叉树等，但它们都是基于串行思想的。在 GPGPU 时代到来后人们发现，在 GPU 上并行执行的简单蛮力搜索其效率甚至是相比这些经过精心设计的算法都极具竞争性[17]。由于并行算法需要考虑许多诸如数据依赖、数据冲突等问题才能充分利用 GPU 的并行处理能力，传统的基于串行思想而设计的算法并不适用于在 GPU 上进行并发执行。近几年，人们也针对这一问题进行了许多研究，包括用并行思想重新设计传统的 K-d 树算法（例如[18]）以及在简单蛮力搜索基础上设计更为高效的蛮力搜索算法（例如[17]）。本课题同样在蛮力搜索算法的基础上，依据本课题中三维点云都是从高度结构化的二维深度图构建的这一前提条件，提出了更有针对性且更为高效的最近邻搜索算法。具体细节见下文 4.3 节。

在第二步中，求解三维变换的过程是一个求解最小误差平方和算法。一个三维变换可以分为平移、旋转和缩放三个部分组成。即

$$\boldsymbol{P}' = sR\boldsymbol{P} + \boldsymbol{r_0} \tag{7}$$

其中s为缩放标量，R为3×3旋转矩阵，$\boldsymbol{r_0}$为三维平移向量。于是第二步骤的问题即求一组$s, R, \boldsymbol{r_0}$满足

$$\min \sum_{i=0}^{n} \left(\boldsymbol{r}_{r,i} - sR\boldsymbol{r}_{l,i} - \boldsymbol{r}_{\boldsymbol{0}} \right)^2 \tag{8}$$

其中$\boldsymbol{r}_{r,i}$是目标中的一个点，$\boldsymbol{r}_{l,i}$是源中的一个点。该算法由 B. K. P. Horn 在[19]中提出。该算法用单位四元数来表示旋转。B. K. P. Horn 通过严格的数学推导后给出了如下结论：

（1） 最佳的旋转四元数是与一个对称矩阵的最大正特征值相关的特征向量。

（2） 最佳的缩放标量s为两组点云到它们的质心距离的均方根的比例。

（3） 最佳的平移向量$\boldsymbol{r}_{\boldsymbol{0}}$为目标的质心与经过旋转和缩放后的源的质心的差。

附录 A1 中将给出该结论的简要推导过程。详细推导过程请参阅[19]。

4.2 基于位置-色彩距离的 ICP 算法改进

如上文 4.1 中所述，ICP 算法的第一个步骤是要在目标中为每个源中的顶点搜索最近的顶点。传统的 ICP 算法是基于最一般的距离概念定义最近邻。基于这样的概念得到的的点对关系可能与正确的点对关系相差很远（图 11）。而如果结合色彩信息对距离进行新的定义，能大大增加最近邻搜索的鲁棒性。基于 RGB-D 的最近邻搜索能够在结构特征不足的区域（如平坦的区域）依靠纹理色彩特征弥补，而在纹理色彩特征缺乏的区域（如纯色的区域）依靠结构特征弥补，从而提高了最近邻搜索的正确率[14]。

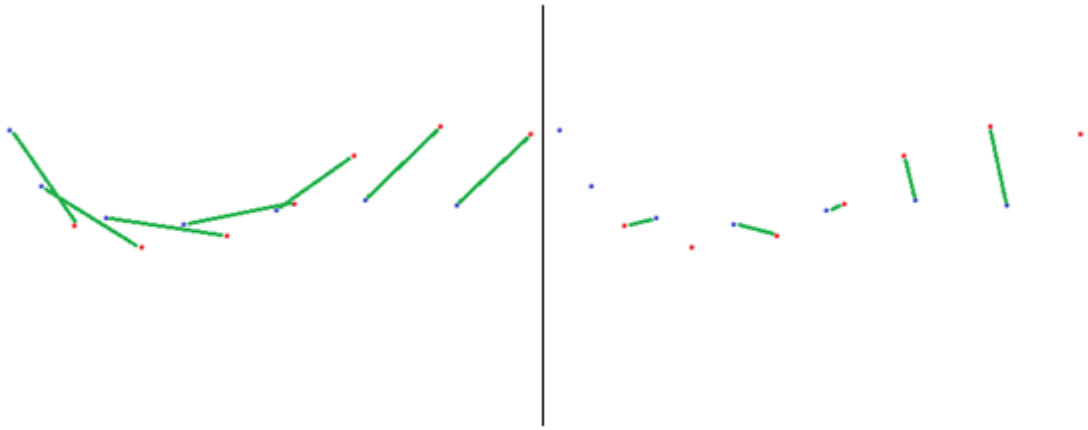

图 11 一种基于一般距离进行最近邻搜索产生不正确结果的情况
左图：正确的点对关系；右图：执行最近邻搜索后产生的错误的点对关系

新的距离定义由位置空间下的欧几里得距离和 RGB 空间下的欧几里得距离相加而得。加入权重因子后可以调节两者对最终距离的影响程度。

$$\begin{cases} d = \omega_p |v_1.\boldsymbol{p} - v_2.\boldsymbol{p}| + \omega_c |v_1.\boldsymbol{c} - v_2.\boldsymbol{c}| \\ \omega_p + \omega_c = 1 \end{cases} \tag{9}$$

其中d为两顶点间的位置-色彩距离，$v_i.\boldsymbol{p}$为某顶点的位置坐标，$v_i.\boldsymbol{c}$为某顶点的 RGB 色彩值。ω_p, ω_c分别为位置和色彩对最终结果的影响权值。

4.3 基于投影关系的快速最近邻搜索算法

有了距离定义，就可以通过最近邻搜索算法构建点对。上文 4.1 节中提到，基于 GPU 并行执行的蛮力搜索算法执行效率可以和精心设计的串行最近邻搜索算法媲美。但是其效率仍不能满足实时计算的要求。为此许多人都在蛮力搜索算法的基础上研究了更高效的 GPU

最近邻搜索算法。

本课题使用了作者自己设计的 GPU 快速最近邻搜索算法也是基于蛮力搜索的基础，通过尽可能减少搜索范围来达到提高效率的目的。该算法不是一个通用的最近邻搜索算法，它依靠一个前提条件：每组点云都是从二维深度图中生成的，即他们中的每个点与某个深度图中的像素一一对应，形成已知的投影关系。由于这种投影关系的存在，每组点云都是高度结构化的，因此搜索范围可以极度地缩小。

基于投影关系的快速最近邻搜索算法的主要根据是在深度图中排列的像素通过逆投影生成的三维顶点也具有一定的排列关系。也即两个距离相近的顶点，其在深度图上的投影之间的距离也很可能很近（只要这两个点深度值不是很小）。因此若假设某个顶点v_1的最近邻顶点是v_2，v_2的投影$\boldsymbol{p_2}$有很大的概率落在v_1的投影$\boldsymbol{p_1}$的小邻域内。

根据以上思想设计的基于投影关系的快速最近邻搜索算法如算法 4 所示。

算法 4 基于投影关系的快速最近邻搜索算法

输入：源$\{V_s\}$，目标$\{V_t\}$，目标深度图d_t，目标内外参数$param_t$
输出：点对记录$\{Rec_{s,t}\}$
过程：
对于$\{V_s\}$中的每个元素$V_s(i)$：
　$\boldsymbol{p_s'} = \text{project}(V_s(i).\boldsymbol{P}, param_t)$；
　$d_{min} =$ 无穷大$, index_{min} = -1$；
　对于以$\boldsymbol{p_s}'$为中心的$n \times n$矩形区域中的每个像素$\boldsymbol{p_t}'$：
　　找到在目标中的对应点$V_t(j)$；
　　倘若（由公式⑼计算得）$d\left(V_t(j), V_s(i)\right) < d_{min}$则：
　　$d_{min} =$ 无穷大$, index_{min} = j$；
　$Rec_{s,t}(i) = j$；
结束

在具体实现过程中，对上述算法进行了针对 GPU 架构的修改和优化。

另外，由于该算法给出的是目标中距离源的某个点距离最小的点，因此点对中会产生源中的多个点对应到目标中的一个点的情况。为了消除这一现象，本课题中该算法被执行两遍，其中一遍是从源到目标，另一边是从目标到源，然后判断双方的最近点是否是对方（即是否满足$Rec_{t,s}\left(Rec_{s,t}(i)\right) = i$）。只有满足该条件的点对会被加入最终结果。

4.4 ICP 算法的 GPU 实现

4.5 本章小结

第5章 从三维点云生成表面网格模型

~~5.1 这一章的内容还没有明确想法~~

5.2 本章小结

第6章 基于实时动态场景重建的远程交互应用设计

6.1 本地场景的远程展现~~（这一章可能最后来不及完成了）~~

6.2 本章小结

第7章 实验结果和性能

7.1 对人物雕像的重建实验结果

7.2 对办公环境的重建实验结果

7.3 性能分析和调优

7.4 本章小结

第8章 总结

8.1 研究成果和问题

8.2 下一步工作

附录

A1. 最优三维变换求解公式简要推导

参考文献

[1] O. Bimber and R. Raskar, *Spatial Augmented Reality: Merging Real and Virtual Worlds*: A. K. Peters, Ltd., 2005.

[2] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs, "The office of the future: a unified approach to image-based modeling and spatially immersive displays," presented at the Proceedings of the 25th annual conference on Computer graphics and interactive techniques, 1998.

[3] P. Lincoln, G. Welch, A. Nashel, A. Ilie, A. State, and H. Fuchs, "Animatronic Shader Lamps Avatars," in *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, 2009, pp. 27-33.

[4] P. Lincoln, A. Nashel, A. Ilie, H. Towles, G. Welch, and H. Fuchs, "Multi-view lenticular display for group teleconferencing," presented at the Proceedings of the 2nd International Conference on Immersive Telecommunications, Berkeley, California, 2009.

[5] S. Deng and B. Yuan, "A method for 3D surface reconstruction from range images," in *Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN '94., 1994 International Symposium on*, 1994, pp. 429-432 vol.2.

[6] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," presented at the Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996.

[7] C. Schutz, T. Jost, and H. Hugli, "Free-form 3D object reconstruction from range images," in *Virtual Systems and MultiMedia, 1997. VSMM '97. Proceedings., International Conference on*, 1997, pp. 69-70.

[8] R. T. Whitaker, "A Level-Set Approach to 3D Reconstruction from Range Data," *Int. J. Comput. Vision*, vol. 29, pp. 203-231, 1998.

[9] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. J. Davison, and A. Fitzgibbon, "KinectFusion: real-time dynamic 3D surface reconstruction and interaction," presented at the ACM SIGGRAPH 2011 Talks, Vancouver, British Columbia, Canada, 2011.

[10] 杨敏 and 沈春林, "基于场景几何约束未标定两视图的三维模型重建," *中国图象图形学报 A 辑*, vol. 8, p. 5, 2003.

[11] F. Remondino, S. F. El-Hakim, A. Gruen, and Z. Li, "Turning images into 3-D models," *Signal Processing Magazine, IEEE*, vol. 25, pp. 55-65, 2008.

[12] J. Wei and L. Jian, "Panoramic 3D Reconstruction by Fusing Color Intensity and Laser Range Data," in *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, 2006, pp. 947-953.

[13] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *ACM Computing Surveys (CSUR)*, vol. 17, pp. 75-145, 1985.

[14] D. Neumann, F. Lugauer, S. Bauer, J. Wasza, and J. Hornegger, "Real-time RGB-D mapping and 3-D modeling on the GPU using the random ball cover data structure," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp.

1161-1167.

[15] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 239-256, 1992.

[16] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1330-1334, 2000.

[17] L. Cayton, "A nearest neighbor data structure for graphics hardware," in *International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures*, 2010.

[18] T. Foley and J. Sugerman, "KD-tree acceleration structures for a GPU raytracer," presented at the Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, Los Angeles, California, 2005.

[19] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, pp. 629-642, 1987.

谢辞

—正文文本—

ENGLISH LARGE ABSTRACT TITLE

- Text -