DOI：10.3724/SP.J.1218.2014.00129

一种改进的 KinectFusion 三维重构算法

朱笑笑，曹其新，杨　扬，陈培华

（上海交通大学机器人研究所机械系统与振动国家重点实验室，上海　200240）

摘　要：对 KinectFusion 算法进行了两个方面的改进，一方面提出使用环境中的边线特征点匹配来提高其定位鲁棒性，另一方面在点云模型中预设一个地面点云来降低累积误差提高精度．在一个 RGB-D（颜色－深度）SLAM 验证数据集以及一个实验室的场景数据上进行了建模对比实验，结果显示，改进后的算法在鲁棒性和精度上均有明显提高，在建立一个尺度为 6 m × 3 m × 3 m 的环境时建模误差由 4.5%降低为 1.5%．虽然算法运行的效率有所下降但仍保持较高实时性，对建模时的用户体验没有明显影响．

关键词：KinectFusion 算法；同时定位与地图创建；Kinect 传感器；密集重建

中图分类号： TP391　　**文献标识码：**A　　**文章编号：**1002-0446(2014)-02-0129-08

An Improved KinectFusion 3D Reconstruction Algorithm

ZHU Xiaoxiao，CAO Qixin，YANG Yang，CHEN Peihua

(*State Key Lab of Mechanical System and Vibration*, *Research Institute of Robotics*, *Shanghai Jiao Tong University*, *Shanghai* 200240, *China*)

Abstract: Two improvements of KinectFusion algorithm are proposed. On one hand, the edge feature points in the environment are matched to improve the robustness, on the other hand, a ground plane point cloud in the model is preset to improve the accuracy. A standard RGB-D (RGB-Depth) SLAM (simultaneous localization and mapping) benchmark dataset and a data of the lab environment are modeled, and the comparison results show that, both the robustness and the accuracy are improved obviously after the improvement. The improved algorithm decreases the modeling error from 4.5% to 1.5% in a room of 6 m × 3 m × 3 m. Although the efficiency is influenced, the running speed of the algorithm is still very high, and the user experience during modeling is good.

Keywords: KinectFusion algorithm; SLAM (simultaneous localization and mapping); Kinect sensor; dense reconstruction

1　引言（Introduction）

Kinect 传感器是一种 RGB-D 传感器，即可以同时获得环境颜色值（RGB）和深度值（depth）的传感器．它的采集速度快，精度高，且价格不到 1000 元，使其迅速被运用到很多领域．机器人领域也开始了对 Kinect 传感器广泛的研究．

利用 Kinect 传感器对室内环境进行 3D 重构，获得环境的 3D 点云模型是研究热点之一．华盛顿大学与微软实验室 [1]，开发了基于 SIFT（尺度不变特征变换）特征匹配定位及 TORO（Tree-based netwORk Optimizer）优化算法的实时视觉 SLAM 系统来建立 3D 点云地图．德国 Freiburg 大学 [2] 提出了 RGBD-SLAM 算法，采用了与华盛顿大学类似的方法，但是为了提高实时性，使用了 Hogman（hierarchical optimization for pose graphs on manifolds）图优化算法，同时在相对位姿检测上采用了 SURF（加速鲁棒特征）特征进行对应点匹配．KinectFusion[3-4] 算法与这些算法不同，它仅使用深度信息，通过设计高效及高度并行的算法在 GPU（图形处理单元）上运行达到了非常高的实时性，在试验中，在配置 4000 元左右的电脑上运行速度达到了 18 帧/秒（在同样配置的计算机上前面两种算法仅达到 2 帧/秒），在进行场景建立时有良好的用户体验，甚至可以用来做一些人机交互方面的应用 [3-4]．同时 KinectFusion 采用了基于 TSDF（truncated signed distance function）模型的点云融合方法，构建的点云模型冗余点少．而前面的方法因为没有进行点云融合，所以在地图创建过程中，点云的容量将不断增加．

当然，KinectFusion 算法也存在一些问题，主要有以下几个：a) 由于受 GPU 内存限制，只能建

基金项目：国际热核聚变实验堆（ITER）计划专项资助项目（2011GB113005）．
通信作者：朱笑笑，ttl@sjtu.edu.cn　收稿/录用/修回：2013-04-23/2013-06-03/2013-11-26

立小规模环境；b) 算法注重效率的提升，在鲁棒性方面有不足之处，比如在一些环境中容易出现定位失效而无法正常建模；c) 累积误差问题．针对这些问题，研究者们提出了一些改进的方法．如文 [5-6] 设计了一种在 GPU 中可执行的 Octree（八叉树）数据结构代替原始算法的 3 维数组来表示环境 TSDF 模型，大大降低了对 GPU 内存的需求．文 [7] 使用动态内存管理方法来解决 GPU 内存限制的问题，同时采用视觉里程算法 FOVIS（Fast Odometry from VISion）来代替原始算法中的 ICP（迭代最近点算法）方法进行定位，解决原始算法定位经常失效的问题以及累积误差的问题．这种方法为了达到原始算法的相近的运行速度，需要配置性能高的 CPU.

本文针对 KinectFusion 定位失效和累积误差两个问题，提出不同于文 [7] 的改进方法．通过在深度数据中提取环境中的边线特征点，利用边线特征点对应点关系来辅助原始算法中的用投影法得到的对应点关系，提高其鲁棒性，同时通过在重构的模型中预设一个地面模型来降低累积误差，提高算法精度.

2 Kinect 传感器与 KinectFusion 算法介绍（Introduction of the Kinect sensor and the KinectFusion algorithm）

2.1 Kinect 传感器

Kinect 传感器如图 1[8]，它包括随机红外点云投射器、红外相机和彩色相机．红外点云投射器和红外相机构成了一个结构光 3D 成像系统 [8]．Kinect 可以同时采集到环境的深度信息数据和颜色信息数据，其中深度信息数据是一个 2 维矩阵，它的元素值表示环境中物体到相机中心的距离值，可以作为灰度图像进行处理．如图 2(a) 和 (b) 是从 Kinect 得到的颜色数据和相应的深度数据，而 (c) 是根据 Kinect 的标定参数将颜色信息和深度信息进行结合，得到的彩色 3D 点云．由于使用到了红外线结构光技术，Kinect 的应用也限于室内环境.

Kinect 的主要技术参数如下：检测距离 0.5 m ～ 7 m，检测角度水平方向 57°，竖直方向 43°，采集帧率为 30 帧/秒，误差在 4 m 远处为 1.4%.

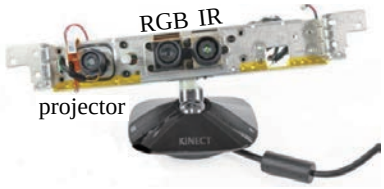
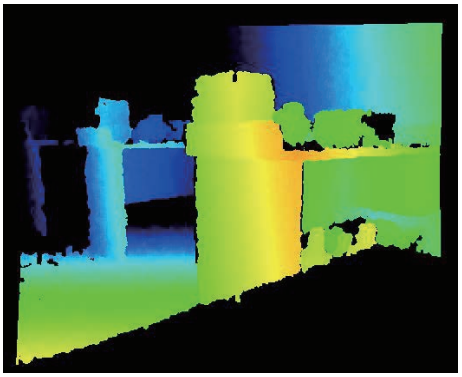


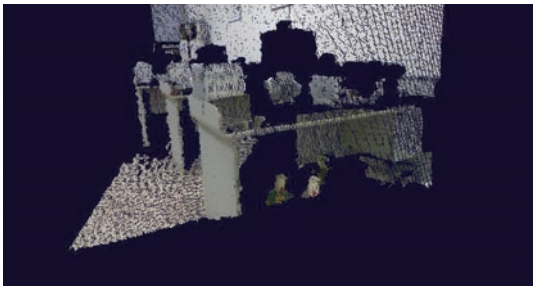
图 1 Kinect 成像系统结构图

Fig.1 Structure of Kinect's imaging system



(a) 用色温图来示意的深度数据，冷色温表示距离远

(b) 对应的颜色图像数据



(c) 通过相机标定参数将颜色信息和深度信息结合得到的 3D 颜色点云数据，图为改变视角后的效果

图 2 Kinect 传感器数据示意图

Fig.2 Illustration of the sensor data of Kinect

2.2 KinectFusion 算法

KinectFusion 算法通过将 Kinect 采集到的深度数据进行匹配定位与融合来实现 3D 场景重构．它的算法流程如图 3[3] 所示，主要由 4 个部分组成：a) 深度数据处理，是将传感器原始的深度数据转换成 3D 点云，得到点云中顶点的 3 维坐标和法向量；b) 相机跟踪，是将当前帧 3D 点云和由现有模型生成的预测的 3D 点云进行 ICP 匹配，计算得到当前帧相机的位姿；c) 点云融合，是根据所计算出的当前相机位姿，使用 TSDF 点云融合算法 [9] 将当前帧的 3D 点云融合到现有模型中；d) 场景渲染，是使用光线跟踪的方法，根据现有模型和当前相机位姿预测出当前相机观察到的环境点云，一方面用于反馈给用户，另一方面提供给 b) 进行 ICP 匹配.

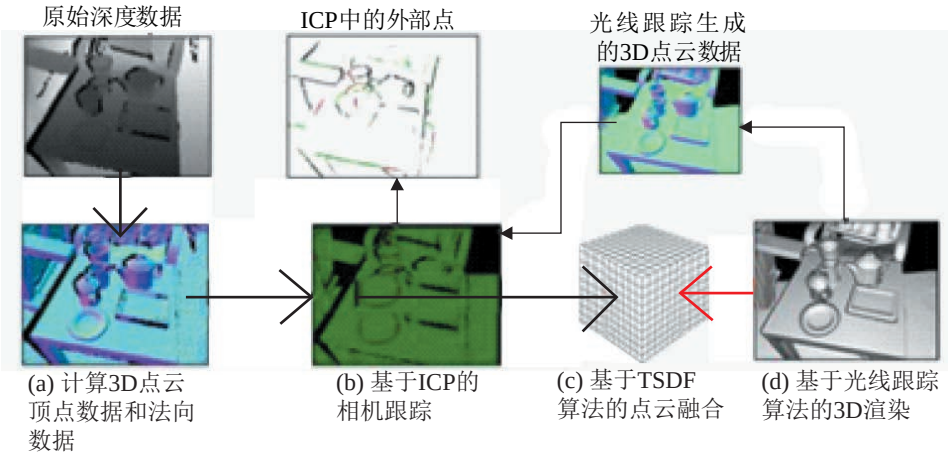


图 3　KinectFusion 算法流程图

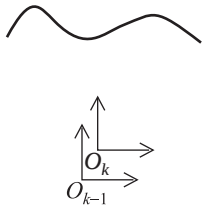
Fig.3　Flow chart of KinectFusion algorithm

因为本文主要对环节 b) 和 d) 进行改进，所以下面两小节对这两个部分作进一步的介绍.

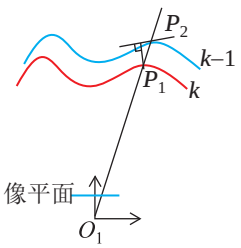
2.3　KinectFusion 中的 ICP 定位方法

ICP 定位环节将当前帧 3D 点云和预测得到的 3D 点云进行匹配时，由以下步骤来实现：

A) 利用投影法来确定对应点关系. 用一个 2 维 ICP 来表示其过程，如图 4 所示. 图 4(a) 中黑色曲线是环境中的物体，相机在连续两个位置分别对其进行采样和预测，O_k 和 O_{k-1} 分别是当前相机和前一帧相机的坐标系原点. 首先将 k 和 $k-1$ 时刻的两个点云都转换到当前帧 k 的相机坐标系下，然后将两个点云通过相机中心 O_k 向像平面上投影，两个点云中具有相同的像平面上的投影点的点即为对应点，如图 4(b) 中的 P_1 和 P_2 两点. 算法中还通过对应点间的欧氏距离和法方向夹角来对对应点进行筛选.



(a) 对同一个场景进行连续两次采样



(b) 投影法点云对应点查找与点到面误差机制示意图

图 4　2D 的 ICP 例子示意图

Fig.4　The schematic diagram of an example of 2D ICP

B) 利用点到平面的误差机制来衡量当前相对位姿的准确度. 如图 4(b) 所示，在 2 维情况下，P_1 和 P_2 间的误差为 P_1 到 P_2 点的切线的距离 d. 所有对应点间的总误差公式如下：

$$E(\boldsymbol{T}_{\mathrm{g},k}) = \sum_{\Omega(\boldsymbol{u})\neq\varnothing} \left\| \left(\boldsymbol{T}_{\mathrm{g},k} \dot{\boldsymbol{V}}_k(\boldsymbol{u}) - \hat{\boldsymbol{V}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}}) \right)^{\mathrm{T}} \hat{\boldsymbol{N}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}}) \right\|_2 \tag{1}$$

其中 $\Omega(\boldsymbol{u}) \neq \varnothing$ 表示当前点云中的一个点 $\boldsymbol{u}$ 存在对应点，$\boldsymbol{T}_{\mathrm{g},k}$ 是一个 4×4 的位姿矩阵，表示当前帧相机在世界坐标系下的绝对位姿，世界坐标系定义为第一帧的相机坐标系，$\dot{\boldsymbol{V}}_k(\boldsymbol{u})$ 为当前帧中 $\boldsymbol{u}$ 点的顶点坐标，$\hat{\boldsymbol{V}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}})$ 是 $\boldsymbol{u}$ 点在预测帧中对应点的顶点坐标，$\hat{\boldsymbol{N}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}})$ 为对应点的法向量.

C) 通过优化式 (1) 得到最佳的相对位姿 $\boldsymbol{T}_{\mathrm{g},k}$.

采用线性化的方法将优化问题转化为一个最小二乘优化，通过计算一个线性方程组如式 (2) 来计算最优解 $\boldsymbol{x}$.

$$\sum_{\Omega(\boldsymbol{u})\neq\varnothing} (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})\boldsymbol{x} = \sum \boldsymbol{A}^{\mathrm{T}}\boldsymbol{b} \tag{2}$$

其中

$$\begin{aligned}
\boldsymbol{x} &= (\beta,\gamma,\alpha,t_x,t_y,t_z)^{\mathrm{T}} \in \mathrm{R}^6,\\
\boldsymbol{G}(\boldsymbol{u}) &= [[\tilde{\boldsymbol{V}}^{\mathrm{g}}_k(\boldsymbol{u})]_\times \mid \boldsymbol{I}_{3\times3}],\\
\boldsymbol{A}^{\mathrm{T}} &= \boldsymbol{G}^{\mathrm{T}}(\boldsymbol{u})\hat{\boldsymbol{N}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}}),\\
\boldsymbol{b} &= \hat{\boldsymbol{N}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}})^{\mathrm{T}}(\hat{\boldsymbol{V}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}}) - \tilde{\boldsymbol{V}}^{\mathrm{g}}_k(\boldsymbol{u})),\\
\boldsymbol{T}_{\mathrm{g},k} &= \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix}
\end{aligned}$$

D) 迭代 A) ～ B)10 次.

2.4 KinectFusion 中的 TSDF 点云融合算法

TSDF 算法用一个立方体栅格来表示 3 维空间（如图 3 步骤 (c)），立方体中每一个栅格存放的是该栅格到物体模型表面的距离，同时使用正负来表示在表面被遮挡一侧和可见一侧，而过零点就是表面上的点，如图 5 中左侧的立方体中的一个物体模型．当有新的数据需要加入模型时会按照式 (3) 和 (4)，进行融合处理，式中 $i+1$ 表示当前点云对应的栅格距离值，i 表示原有的栅格距离值，同一个栅格的距离值通过一个权重 W 来进行融合，新的权重为两个权重之和，示意图如图 5 中右侧．在 KinectFusion 算法中当前点云的权重为 1.

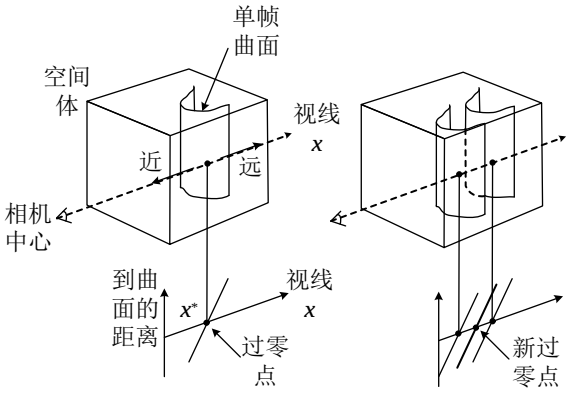


图 5 基于空间体的点云融合

Fig.5 Merging of two point clouds using volumetric method

$$D_{i+1}(x)=\frac{W_i(x)D_i(x)+w_{i+1}(x)d_{i+1}(x)}{W_i(x)+w_{i+1}(x)} \tag{3}$$

$$W_{i+1}(x)=W_i(x)+w_{i+1}(x) \tag{4}$$

这种方法是具有最小二乘优化性质的，同时使用了权重值来进行融合，对一些传感器的噪声具有一定的抑制作用.

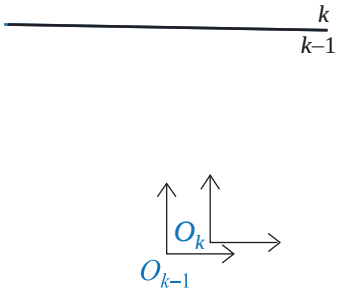
3 改进的 KinectFusion 算法（The improved KinectFusion algorithm）

3.1 KinectFusion 算法的两个问题分析

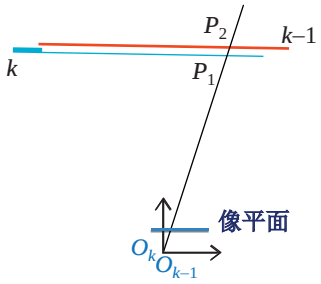
A) ICP 定位失效问题．当环境主要由平行平面构成时，在建模过程中经常会出现相机视野范围内的环境空间限制不全面，这时 KinectFusion 中的 ICP 算法会失效，进而导致整个建模过程的中断．如图 6 所示，当前视野中的主要部分是地面和墙面以及与墙面平行的平面，这种场景将导致相对位姿中平行于墙面和地面的运动分量无法正确估算．用一个 2 维的特例来进行说明．如图 7 所示，假设环境中仅有一条直线，相机移动一段距离进行了两次采样．根据 2.3 节中 ICP 的方法首先找到对应点如 P_1 和 P_2，然后计算点到切面的距离误差，这时所有对应点的误差均为 0，第一次迭代后 ICP 就结束．这样 ICP 定位就失效了.

图 6 一个空间限制不全面的场景

Fig.6 An incompetent space constraint scene



(a) 对一个场景进行连续两次采样，场景中仅包含一条直线



(b) 对投影法点云对应点示意图

图 7 一个 2 维 ICP 定位失效的特例

Fig.7 A special case of 2D ICP failure

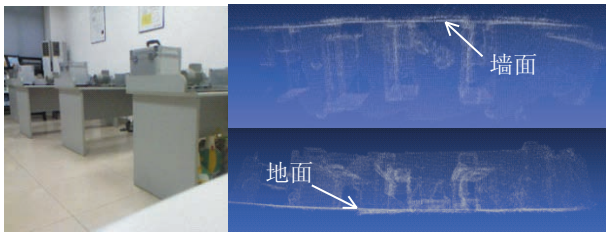


图 8 KinectFusion 算法累积误差

Fig.8 Accumulated error of KinectFusion algorithm

B) 累积误差问题．在不断创建新场景时累积误差比较严重．虽然 KinectFusion 算法将当前点云与模型中预测的点云进行匹配定位，所以在对同一个环境进行重复建模时不会出现累积误差无限增加的

情况，但是在创建新环境时累积误差依然存在．如图 8 所示，在创建的点云中可以看到创建得到的地面和墙面不是一个平面，有明显的累积误差．

对于一些人机交互应用来说，这不是一个很大问题，但是用于机器人定位导航中却是关键的问题．下面将针对这两个问题分别提出改进方法．

3.2　边线点对应点关系改进

从定位失效问题的分析中可以看到单一的对应点关系是其根本原因．文 [7] 提出利用颜色特征点进行辅助定位来显著提高鲁棒性和精度．但是该方法需要使用高性能的 CPU，否则程序运行效率将大幅下降．本文提出直接在深度数据中提取边线特征点作为辅助对应点关系来提高系统的鲁棒性．仍以 3.1 节的特例来说明，如图 9 所示，如果获得两对边线对应点，则可以解决 ICP 定位失效问题．

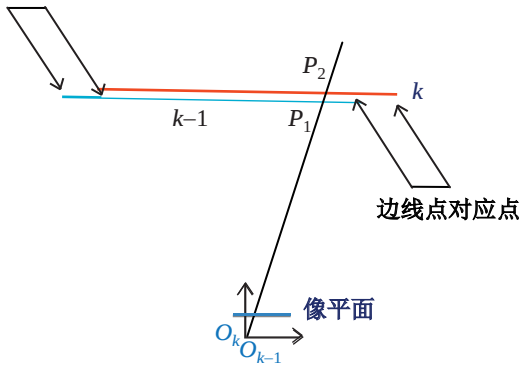


图 9　在 2 维空间上的特例上使用边线点对应点

Fig.9　Using correspondence of boundary points in a 2D ICP case

A) 边线点的计算．将深度数据作为一张灰度图进行处理，对每个像素作以下处理，当像素的 3×3 的邻域内没有距离值为 0 的点（未获得深度数据的点）时，利用 Sodel 算子计算该像素点的梯度，如果该点的梯度大于一定阈值，那么将邻域中距离值最小的点定义为边线点．这里需要取距离值最小点，如图 10 中 A 和 B 两点的梯度都满足要求，但是只有 A 点是边线点．

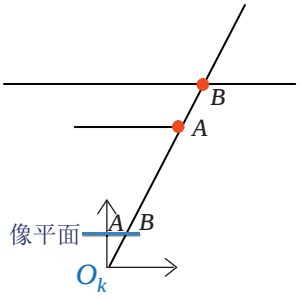
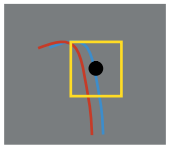


图 10　边线点检测示意图

Fig.10　Illustration of the boundary points detecting

B) 边线点对应关系确定．前后两帧深度数据中的边线点的对应关系通过邻域搜索最近点的方法来确定．首先将两个边线点统一到同一个图像坐标系中，如图 11(b) 所示，然后在当前边线点的一个邻域内的所有前一帧边线点中查找最近点．找到的对应点还需要使用式 (5) 进行滤波，即要求两个对应点间的距离要小于一定阈值 ε_{d}.

(a) 两帧数据中的边线点



(b) 邻域最近点法查找边线点对应点示意图

图 11　邻域最近点查找对应点示意图

Fig.11　Illustration of the neighborhood closest point finding for the corresponding points of boundary points

$$\psi(\boldsymbol{u})\neq\varnothing \quad \text{iff } M_k(\boldsymbol{u})=1 \text{ and } \|\boldsymbol{T}_{\mathrm{g},k}\dot{\boldsymbol{V}}_k(\boldsymbol{u})-\hat{\boldsymbol{V}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}})\|_2\leqslant\varepsilon_{\mathrm{d}} \tag{5}$$

C) 误差融合优化．加入边线点对应关系后总的误差方程为式 (6)；其中 E_1 为投影对应点误差，E_2 为边线点对应关系误差．α 为放大系数，因为试验中发现由于边线点要求苛刻，对应点数量远远小于投影对应点，导致边线点起不到作用，通过一个放大系数可以扩大边线点的影响．按实验经验取为 100.

$$E(\boldsymbol{T}_{\mathrm{g},k})=E_1+\alpha E_2 \tag{6}$$

其中：

$$E_1=\sum_{\Omega(\boldsymbol{u})\neq\varnothing}\|(\boldsymbol{T}_{\mathrm{g},k}\dot{\boldsymbol{V}}_k(\boldsymbol{u})-\hat{\boldsymbol{V}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}}))^{\mathrm{T}}\hat{\boldsymbol{N}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}})\|_2$$

$$E_2=\sum_{\psi(\boldsymbol{u})\neq\varnothing}\|(\boldsymbol{T}_{\mathrm{g},k}\dot{\boldsymbol{V}}_k(\boldsymbol{u})-\hat{\boldsymbol{V}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}}))^{\mathrm{T}}\|_2$$

同样需要对误差 E_2 进行线性化，最后可以得到总的线性方程如下：

$$\left(\sum_{\Omega(\boldsymbol{u})\neq\varnothing}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})+\alpha\sum_{\psi(\boldsymbol{u})\neq\varnothing}(\boldsymbol{A}_{\mathrm{e}}^{\mathrm{T}}\boldsymbol{A}_{\mathrm{e}})\right)\boldsymbol{x}=\sum\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}+\alpha\sum\boldsymbol{A}_{\mathrm{e}}^{\mathrm{T}}\boldsymbol{b}_{\mathrm{e}} \tag{7}$$

其中：$\boldsymbol{A}_{\mathrm{e}}^{\mathrm{T}}=\boldsymbol{G}^{\mathrm{T}}(\boldsymbol{u})$，$\boldsymbol{b}_{\mathrm{e}}=\hat{\boldsymbol{V}}^{\mathrm{g}}_{k-1}(\hat{\boldsymbol{u}})-\widetilde{\boldsymbol{V}}^{\mathrm{g}}_{k}(\boldsymbol{u})$.

3.3 预设地面点云改进

KinectFusion 算法一般都用于室内环境的建模，而在室内环境中地面是一个占面积很大的部分，地面的模型为简单的一个平面模型．所以如果将地面作为环境中已知的部分，预先载入到 TSDF 的空间立方中，可以消除部分的累积误差．

具体实施方法如图 12 所示，首先在模型中加入一个平面，在 TSDF 立方体中根据栅格到平面的距离对栅格进行赋值同时将权重值设置为最大权重，然后从 Kinect 中读取第一帧深度数据，利用 RANSAC（随机抽样一致算法）平面提取方法 [10]，对深度数据中的地面进行提取，得到地面的方程并计算出相机的位姿参数，最后将第一帧的点云通过 TSDF 融合算法融入到预设地面的 TSDF 模型中，同时将相机位姿作为初始位姿，供后续的相机跟踪算法使用．

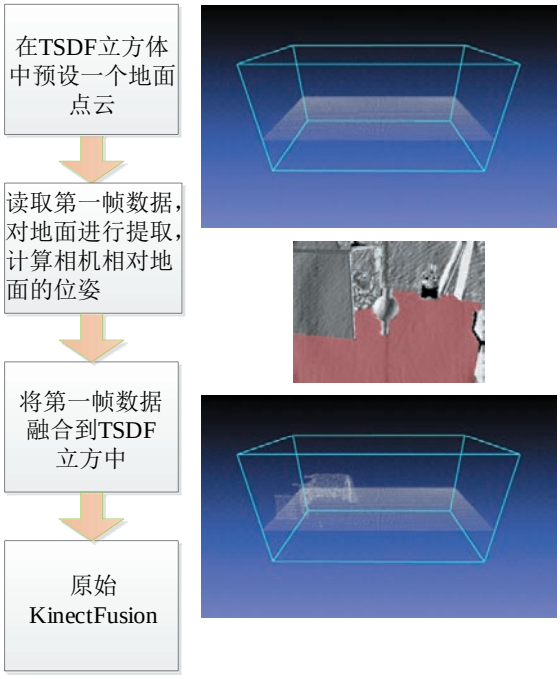


图 12 预设地面改进流程图

Fig.12 Flow chart of the preset ground method

4 实验（Experiment）

本节通过在两个数据集进行建模测试，将改进前和改进后的算法进行对比．第一个数据集是一个公共的 RGB-SLAM 验证数据集 [11]．第二个数据集是一个实验室的场景．

测试程序在 PCL[12] 库中 Kinfu 的基础上进行开发，运行电脑配置为 CPU Intel Core2 E7200 双核 2.53 GHz，内存 4 G，显卡 GeForce GTX 560 显存 2 G，操作系统为 Ubuntu 10.10.

实验中采用的 TSDF 立方体的空间尺寸为 6 m×3 m×3 m，像素尺寸为 1024×512×512，单元格分辨率为 5.86 mm.

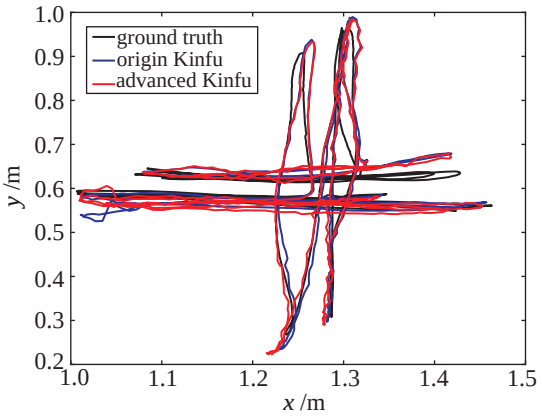
4.1 利用 RGB-D SLAM 验证数据集及实验室数据集对边线点对应关系进行改进测试

RGB-D SLAM 验证数据集主要用来为各种 SLAM 算法提供验证对比数据，每一个场景数据包中包含彩色图片文件夹、深度图片文件夹、加速度计数据文件、精确轨迹文件、彩色图片及深度图片列表文件，而且所提供的比较工具可以生成对比数据图表．

第一个场景数据包中的一帧如图 13(a) 所示，是一个桌面的环境，这是一个常规环境，原始的 KinectFusion 算法可以正常地运行．

为了方便观察，验证数据集进行轨迹对比时给出一个像 *x*-*y* 平面投影后的轨迹，如图 13(b) 所示，可以看到添加边线点对应关系后计算出的轨迹与原始算法轨迹是有差别的．在验证程序同时给出的误差统计值如表 1 所示，可以看到，边线点改进对原始算法的精度仅有很小的影响．而在运行速度方面，改进后算法运行的帧率由平均 18.5 帧下降为平均 16.3 帧．

(a) 实验场景图



(b) 轨迹对比图

图 13 常规环境下边线点改进前后结果对比

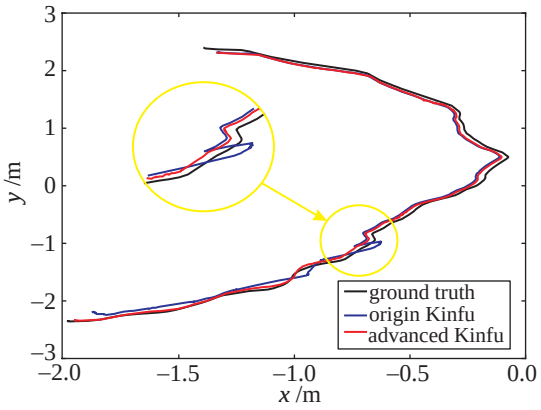
Fig.13 The comparison between the results before and after boundary points improvement in a normal environment

表 1　轨迹误差对比表

Tab.1　Trajectory errors

误差/mm	改进前与真实轨迹比较	改进后与真实轨迹比较	改进后与真进后比较
均方根误差	19.3	20	6.6
平均误差	16.7	17.7	4.5
标准方差	9.6	9.3	4.8
最小误差	1.3	1.4	0
最大误差	5.7	5.1	4.4

(a) 实验场景图　　(b) 轨迹对比图



(c) 出现定位失效时传感器采集到的场景

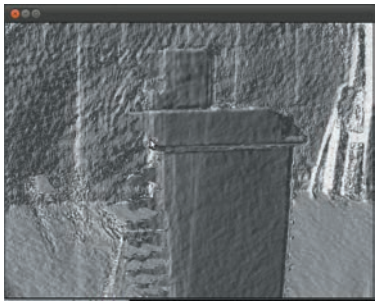
图 14　空间限制不全环境下边线点改进前后结果对比

Fig.14　The comparison between the results before and after boundary points improvement in an unconstrained environment

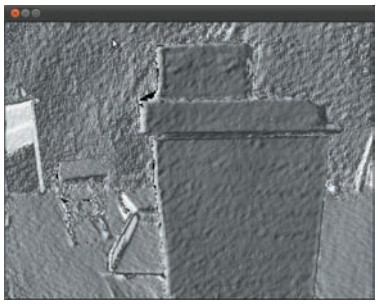
第二个场景是一个较为特殊的场景，主要由平行平面构成，如图 14(a) 所示，主要物体是一个纸板折成的台阶形状．运行后的轨迹对比结果如图 14(b) 所示，可以看到，原始算法在黄色圆圈处有了明显错误，最终导致后续建模的失败．改进后的算法则可以正常地运行，与真实轨迹吻合得非常好．此时对应的相机拍摄到的数据如图 14(c) 所示，场景中缺少一个方向的空间限制．在运行速度方面，改进后帧率由平均 17.2 帧下降为平均 15.4 帧．

从这组对比实验可以看出，对应点关系改进后的算法明显提高了算法的鲁棒性，同时对精度的影响非常小，虽然在帧率上有所下降，但是因为仍保持较高的帧率，所以在建模时对用户的体验并没有太大的影响．

(a) 实验场景图



(b) 原始算法创建地图结果



(c) 改进算法创建地图结果

图 15　实验室环境下边线点改进前后结果对比

Fig.15　The comparison between the results before and after boundary points improvement in a lab environmentt

为了进一步验证边线点改进对鲁棒性的提高，在实验室数据集上进行了对比验证，其场景如图 15(a) 所示，实验时手持 Kinect 沿黑线所示路径进行移动并保存 Kinect 采集的数据后分别使用边线点改进前和改进后的算法进行地图创建．对比创建的结果，从图 15(b) 和 (c) 可以看到，原始算法创建得到了错误的模型，没有能够成功地创建桌子的左侧，其原因在 3.1 节中已经分析过，是因为 ICP 算法对 Kinect 的水平方向的移动分量计算错误，进而导致融合生成地图模型时产生错误．而改进后的算法则可以正常创建得到完整的桌子模型，如图 15(c) 所示．

4.2　利用实验室数据集对预设地面方法测试

该组数据场景如图 15(a) 所示，根据 4.1 节中的实验结果，这个场景下原始算法无法正常运行，所以在对比的时候，一组是在原始算法中加入了边线点关系，另一组是同时加入了边线点关系和预设地

面改进.

两种方法建立的点云结果如图 16 所示，从比较来看，改进后的算法建立的点云模型更为精确. 为了定量地进行对比，在环境中已知位置放置了 4 个标志物. 对比结果如表 2 所示. 可以看到，预设地面的方法明显提高了点云模型的精度，在这个尺度为 6 m× 3 m× 3 m 的环境中建模误差小于 1.5%.

表 2 标志物距离误差对比表

Tab.2 Landmark distance errors

真实距离 /m	无地面预设误差 /m	有地面预设误差 /m
1.346	0.017	0.025
4.383	0.109	0.026
4.214	0.194	0.019
4.816	0.059	0.046
4.964	0.143	0.076
1.204	0.024	0.008

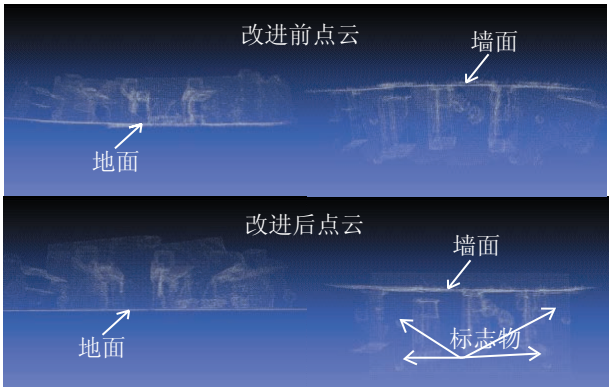


图 16 改进前和改进后的建模结果对比图

Fig.16 Modeling results before and after improvement

5 结论（Conclusions）

本文针对当前流行的 KinectFusion 算法进行了两个方面的改进，一是在定位方面添加边线点对应关系，二是在模型中进行地面预设. 实验证明这两个改进对算法的鲁棒性和精度都有明显的提升. 边线点改进使得模型精度和运行速度稍有下降，但都在可以接受范围，而预设地面法对环境中地面较多的情况下效果会更明显. 这两种改进都是在原始算法基础上开发，在实际使用过程中，用户可以实时地打开或者关闭改进功能. 改进方法可以直接运用到以 KinectFusion 为基础的方法中，如文 [5-7] 的方法，提高这些方法的鲁棒性和精度. 另外，本课题组正在研究室内环境分块创建拼接的方法，即首先利用本文提出的改进算法创建环境的一部分，然后通过拼接方法实现大规模场景的创建.

参考文献（References）

[1] Henry P, Krainin M, Herbst E, et al. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments[C]//RSS Workshop on RGB-D Cameras. 2010.

[2] Fioraio N, Konolige K. Realtime visual and point cloud SLAM[C]//RSS Workshop on RGB-D Cameras. 2011.

[3] Izadi S, Kim D, Hilliges O, et al. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera [C]//Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology. New York, USA: ACM, 2011: 559-568.

[4] Newcombe R A, Izadi S, Hilliges O, et al. KinectFusion: Real-time dense surface mapping and tracking[C]//10th IEEE International Symposium on Mixed and Augmented Reality. Piscataway, USA: IEEE, 2011: 127-136.

[5] Zeng M, Zhao F K, Zheng J X, et al. Octree-based fusion for realtime 3D reconstruction[J]. Graphical Models, 2013, 75(3): 126-136.

[6] Zeng M, Zhao F K, Zheng J X, et al. A memory-efficient KinectFusion using octree[C]//First International Conference Computational Visual Media. Berlin, Germany: Springer, 2012: 234-241.

[7] Whelan T, McDonald J, Kaess M, et al. Kintinuous: Spatially extended KinectFusion[C]//RGB-D Workshop at Robotics: Science and Systems. 2012.

[8] Herrera D C, Kannala J, Heikkila J. Joint depth and color camera calibration with distortion correction[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(10): 2058-2064.

[9] Curless B, Levoy M. A volumetric method for building complex models from range images[C]//Proceedings of ACM SIGGRAPH. New York, USA: ACM, 1996: 303-312.

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

[11] Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway, USA: IEEE, 2012: 573-580.

[12] PCL. PCL Library[OL] (2013-03-06) [2013-03-06]. http://www. Pointcloud s.org/.

作者简介：

朱笑笑（1982–），男，博士生. 研究领域：3 维环境建立，SLAM，结构化环境下机器人定位.

曹其新（1960–），男，博士，教授. 研究领域：机器视觉与模式识别，神经网络与智能控制，基于 Web 的智能维护系统，移动机器人，泛在机器人技术，农业机器人等.

杨 扬（1986–），男，博士生. 研究领域：机器视觉，手眼协调.