文章编号: 1001-0920(2019)12-2585-09 DOI: 10.13195/j.kzyjc.2018.0239

基于快速不变卡尔曼滤波的视觉惯性里程计

黄伟杰, 张国山†

(天津大学 电气自动化与信息工程学院，天津 300072)

摘 要: 针对相机定位问题,设计基于深度相机和惯性传感器的视觉惯性里程计,里程计包含定位部分和重定位部分. 定位部分使用不变卡尔曼滤波融合多层迭代最近点(ICP)的估计值和惯性传感器的测量值来获得精确的相机位姿,其中ICP的估计误差使用费舍尔信息矩阵进行量化. 由于需要使用海量的点云作为输入,采用GPU并行计算以快速实现ICP估计和误差量化的过程. 当视觉惯性里程计出现定位失败时,结合惯性传感器数据建立恒速模型,并基于此模型改进随机蕨定位方法,实现视觉惯性里程计的重定位. 实验结果表明,所设计的视觉惯性里程计可以获得准确追踪相机且可以进行有效的重定位.

关键词: 视觉惯性里程计；不变卡尔曼滤波；多层迭代最近点；随机蕨；GPU并行计算；惯性传感器

中图分类号: TP24 **文献标志码:** A

Visual-inertial odometry based on fast invariant Kalman filter

HUANG Wei-jie, ZHANG Guo-shan†

(School of Electrical and Information Engineering，Tianjin University，Tianjin 300072，China)

Abstract: This paper designs a visual-inertial odometry based on a depth camera and an inertial sensor for localizing the camera. The odometry contains camera localization and camera relocalization. Camera localization uses the invariant extended Kalman filter (IEKF) to fuse multilevel iteration closest point (ICP) estimates with the measurements from the inertial sensor for obtaining an accurate camera pose, in which, the estimated error of the multilevel ICP is quantized by the fisher information matrix. Because massive points are set as the inputs, GPU parallel computing is used to fast implement multilevel ICP estimation and its error quantization. When the odometry tracks camera failed, a constant velocity model is constructed with the data from the inertial sensor, and the random fern method is improved based on the velocity model to relocate the odometry. The experiment results show that the designed odometry can track the camera accurately and relocalize the camera effectively.

Keywords: visual-inertial odometry；invariant extended Kalman filter；multilevel iteration closest point；random fern；GPU parallel computing；inertial sensor

0 引 言

深度相机(如以色列的primesense)是一种新型的相机设备,使用低成本的摄像头感知周围环境,可以同时捕获彩色视频序列和深度视频序列. 视觉里程计[1]通过配准技术估计视频序列中每帧图像对应的相机位姿,根据估计的位姿将所有局部图像信息对齐到一个坐标系生成三维模型[2],这些模型可以用于虚拟现实和增强现实等领域.

为了提高位姿估计的精度,一般增加惯性传感器到视觉里程计中来辅助定位,组成视觉惯性里程计[3]. 起初惯性传感器的数据作为初值输入到视觉配准算法(例如ICP)中,但是由于惯性传感器易受噪声的影响,会导致配准算法估计位姿时产生误差,长时间运行还会导致误差的累积. 减少该误差的一个有效途径是使用卡尔曼滤波器,研究者提出使用各种形式的卡尔曼滤波器进行视觉惯性定位. Cheviron等[4]使用标准扩展卡尔曼滤波器自适应估计陀螺仪和加速度计中的偏置,为空中机器人提供姿态和速度估计. Huang等[5]提出了基于雅可比矩阵的卡尔曼滤波器,这种滤波器在结果一致性方面表现得更好,因为雅可比矩阵使用状态变量首次可用的估计进行计算时,误差状态系统与底层系统有相同维度的可观测子空间. Martin等[6]提出了广义乘性扩展卡尔曼滤波器获得飞行器的刚体变换,并证明所提出的滤波器在估

收稿日期: 2018-03-02；**修回日期:** 2018-06-28.
基金项目: 国家自然科学基金项目(61473202).
责任编委: 方勇纯.
†通讯作者. E-mail: zhanggs@tju.edu.cn.

计位姿方面有更好的收敛性. Bonnabel等[7]提出了不变卡尔曼滤波器,基于不变的输出误差获得几何自适应校正项,使增益和协方差在出现估计偏差时得到校正. 随着GPU并行计算的发展,基于稠密点定位方法的研究非常活跃,不变卡尔曼滤波被应用到基于稠密点的定位方法中. Hervier等[8]使用费舍尔信息矩阵估计点到点单层ICP的协方差,并估算ICP估计误差,进而结合不变卡尔曼滤波融合惯性传感器减少该误差. Barczyk等[9]将文献[8]方法扩展到点到平面单层ICP中,在状态观测量较差的情况下仍然表现出很好的鲁棒性. Zhang等[10]证明了不变卡尔曼滤波方法在收敛性和消除估计误差上比其他形式的卡尔曼滤波方法有更好的效果.

视觉惯性里程计在扫描场景的过程中不可避免地会出现跟踪相机丢失的情况,需要对相机进行重定位. Glocker等[11]提出随机蕨算法解决深度相机的重定位问题,实现了关键帧编码、相似度度量、哈希表存储以及位姿恢复,但是对相机运行速度比较敏感,不能适用于相机运行较快的场景.

本文基于深度相机和惯性传感器设计一种快速视觉惯性里程计,如图1所示. 左边是基于多层ICP的不变卡尔曼滤波的定位部分,右边是基于随机蕨和惯性传感器的重定位部分. 在定位部分中,惯性传感器获得的位姿数据用于ICP的初始值估计相机位姿,同时估算ICP的协方差矩阵,进而使用不变卡尔曼滤波融合估算的相机位姿和从惯性传感器获得的位姿. 在重定位部分中,编码图像数据,并判断是否将其加入关键帧集合,出现定位失败时,当前帧与集合中的每一帧进行相似度度量,进行重定位检测,检测到的匹配关键帧与使用惯性传感器数据构建的速度模型相结合来重定位相机. 实验结果表明,所设计的视觉惯性里程计可以获得准确追踪相机且可以进行有效的重定位.

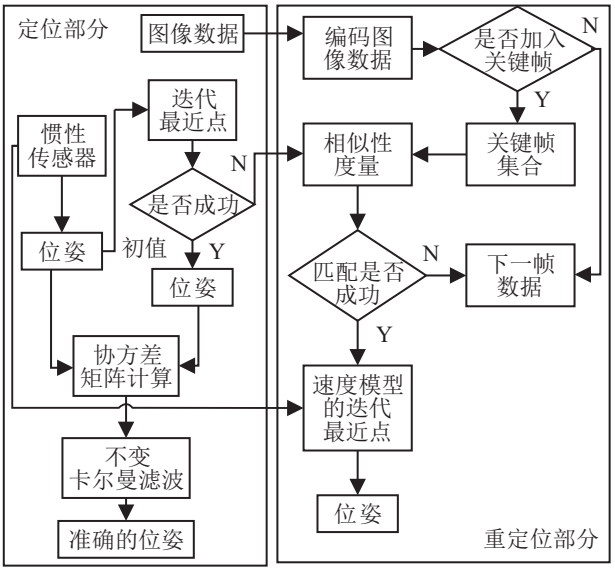


图1 视觉惯性里程计整体框图

1 不变卡尔曼滤波

本节描述原始不变卡尔曼滤波[8],以及单层点到平面ICP[9]协方差矩阵的估计方法.

1.1 相关定义

定义刚体变换 $\boldsymbol{X}=\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ 0^{\mathrm{T}} & 1\end{bmatrix}$. 其中: $\boldsymbol{R}$ 为旋转矩阵, $\boldsymbol{t}$ 为平移向量, 0^{T} 为 3×1 零向量的转置. 旋转与平移使用六自由度表示为 $\boldsymbol{\xi}=[\upsilon_1\ \ \upsilon_2\ \ \upsilon_3\ \ \omega_1\ \ \omega_2\ \ \omega_3]^{\mathrm{T}}\in\boldsymbol{R}^6$, 其中 $\upsilon_1,\upsilon_2,\upsilon_3$ 对应平移线速度, $\omega_1,\omega_2,\omega_3$ 对应旋转角速度. 转换为李代数形式是一个反对称矩阵

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}^{\wedge}=\begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix}.$$

ICP的目标是优化一个能量函数求解运动参数,其点到平面的形式为 $\sum\|(\boldsymbol{X}\boldsymbol{V}_k-\hat{\boldsymbol{V}}_{k-1})\hat{\boldsymbol{N}}_{k-1}\|_2$. 其中: $\boldsymbol{V}_k$ 为当前帧的顶点, $\hat{\boldsymbol{V}}_{k-1},\hat{\boldsymbol{N}}_{k-1}$ 为上一帧估算的顶点和法向量. 求解该能量函数需要计算对应的雅可比矩阵 $\boldsymbol{J}$, 由高斯牛顿法解方程 $\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}\xi=\boldsymbol{J}^{\mathrm{T}}\boldsymbol{r}$ 得到 ξ, 其中 $\boldsymbol{r}$ 为残差. 根据求得的 ξ 更新相关的运动参数.

1.2 不变卡尔曼滤波

文献[8]提出的不变卡尔曼滤波的形式为

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{\boldsymbol{X}}=\hat{\boldsymbol{X}}\boldsymbol{\Omega}_m+\boldsymbol{E}\hat{\boldsymbol{X}},\ \boldsymbol{E}=\begin{bmatrix}\boldsymbol{e}_\omega^{\wedge} & \boldsymbol{e}_\upsilon\\ 0^{\mathrm{T}} & 0\end{bmatrix}.\tag{1}$$

其中: $\hat{\boldsymbol{X}}$ 为状态变量,用矩阵的形式表示刚体变换的估计量; $\boldsymbol{\Omega}_m$ 为带噪声的先验速度信息,可以从惯性传感器中获得; $\boldsymbol{E}$ 为不变卡尔曼滤波的输出误差, $\boldsymbol{E}_\nu,\boldsymbol{E}_\omega$ 对应 $\boldsymbol{E}$ 的一个六自由度的向量. 根据文献[8], $\boldsymbol{E}$ 使用下式进行计算:

$$\boldsymbol{E}=\boldsymbol{g}(e)=\boldsymbol{g}(\boldsymbol{K}\boldsymbol{g}^{-1}[\phi(\boldsymbol{X}\hat{\boldsymbol{X}}^{-1}-\boldsymbol{I})]).\tag{2}$$

其中: $\boldsymbol{g}$ 为从向量到李代数的转化; $\boldsymbol{g}^{-1}$ 为 $\boldsymbol{g}$ 的逆;卡尔曼滤波估计值与真实值之间的误差计算为 $\boldsymbol{X}-\hat{\boldsymbol{X}}^{-1}$, 矩阵之间的变换不能通过矩阵相减获得,所以相应的形式修改为 $\boldsymbol{X}\hat{\boldsymbol{X}}^{-1}-\boldsymbol{I}$; ϕ 为一个投影,将矩阵的一般形式投影成反对称形式

$$\phi\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ 0^{\mathrm{T}} & 1\end{bmatrix}=\begin{bmatrix}\dfrac{\boldsymbol{R}-\boldsymbol{R}^{\mathrm{T}}}{2} & \boldsymbol{t}\\ 0^{\mathrm{T}} & 0\end{bmatrix};$$

$\boldsymbol{K}$ 为卡尔曼增益,其离散形式为

$$\boldsymbol{K}=\frac{\boldsymbol{P}_k(\boldsymbol{P}_k+\boldsymbol{N})^{-1}}{\Delta\boldsymbol{t}},\tag{3}$$

$\boldsymbol{P}$为卡尔曼滤波的另一个参数,其离散形式为

$$\boldsymbol{P}_{k+1}=\boldsymbol{M}\Delta\boldsymbol{t}+\boldsymbol{P}_k-\boldsymbol{P}_k(\boldsymbol{P}_k+\boldsymbol{N})^{-1}\boldsymbol{P}_k. \quad (4)$$

$\boldsymbol{M}$和$\boldsymbol{N}$为惯性传感器和ICP算法的协方差矩阵,文献[9]给出这两个参数的估计方法

$$\boldsymbol{M}=\begin{bmatrix}\mathrm{cov}(\omega_R) & 0_{33}\\ 0_{33} & \mathrm{cov}(\omega_t)\end{bmatrix}.$$

0_{33}是一个3×3的零矩阵,$\mathrm{cov}(\omega_R)$和$\mathrm{cov}(\omega_t)$为对角矩阵. 根据文献[9]所述,这两个对角矩阵与惯性传感器在线速度和角速度上的噪声的协方差有关: 角速度上相关参数为

$$\mathrm{cov}(\omega_R)=0.1\mathrm{diag}[(\sigma_{\omega_1})^2\quad(\sigma_{\omega_2})^2\quad(\sigma_{\omega_3})^2];$$

线速度上相关参数为

$$\mathrm{cov}(\omega_t)=0.1\mathrm{diag}[(\sigma_{v_1})^2\quad(\sigma_{v_2})^2\quad(\sigma_{v_3})^2].$$

本文使用的惯性传感器实测得到

$$\mathrm{cov}(\omega_R)=$$
$$0.1\mathrm{diag}[(0.006\,9)^2\quad(0.008\,2)^2\quad(0.008\,5)^2],$$
$$\mathrm{cov}(\omega_t)=$$
$$0.1\mathrm{diag}[(0.016\,6)^2\quad(0.039\,2)^2\quad(0.041\,6)^2].$$

ICP的协方差矩阵通过下式获得:

$$\boldsymbol{N}=\sigma^2\sum_{\mu_i\in\boldsymbol{\Omega}}\begin{bmatrix}(\boldsymbol{a}_i\times\boldsymbol{n}_i)(\boldsymbol{a}_i\times\boldsymbol{n}_i)^{\mathrm{T}} & (\boldsymbol{a}_i\times\boldsymbol{n}_i)\boldsymbol{n}_i{}^{\mathrm{T}}\\ \boldsymbol{n}_i(\boldsymbol{a}_i\times\boldsymbol{n}_i)^{\mathrm{T}} & \boldsymbol{n}_i\boldsymbol{n}_i{}^{\mathrm{T}}\end{bmatrix}. \quad (5)$$

其中: μ为深度图像中的一个像素; $\boldsymbol{\Omega}$为深度图像中有效像素点的集合; μ使用反投影到相机坐标系得到一个顶点,再将此相机坐标系下的顶点通过相机位姿矩阵的逆矩阵变换到全局坐标系下,得到在全局坐标系下的表示$\boldsymbol{a}_i,\boldsymbol{n}_i$是$\boldsymbol{a}_i$的法向量; σ为深度相机的噪声,通过文献[12] 获得. 为了表示方便,σ^2和脚标i在后续的公式中被省略.

2　基于多层ICP的不变卡尔曼滤波

本节详述如何使用不变卡尔曼滤波融合多层ICP估算出的位姿和惯性传感器中获取的位姿. 首先给出多层金字塔构建的方法; 然后给出多层ICP协方差的量化方法; 最后给出通过CPU和GPU实现算法的细节.

2.1　多层金字塔的构建

由文献[13]构建了一种由粗到细的多层深度图金字塔,层数设置为$L=4$,原始深度图(640×480)设为金字塔最底层($l=1$),第$l+1$层由第l层基于块平均降采样出来,所以$l+1$层是第l层分辨率的1/4. 产生的4层金字塔对应4层可用像素集合$[\boldsymbol{\Omega}_1\quad\boldsymbol{\Omega}_2\quad\boldsymbol{\Omega}_3\quad\boldsymbol{\Omega}_4]$. 产生的图像金字塔的每层图像都会按照单层形式估算出一个刚体变换$\boldsymbol{X}_{\mathrm{inc}}$,上面两层的分辨率较低,只估算旋转量的变化$\boldsymbol{X}_{\mathrm{inc}}=\begin{bmatrix}\boldsymbol{R} & 0\\ 0^{\mathrm{T}} & 1\end{bmatrix}$; 下面两层分辨率较高; 旋转分量和平移分量都需要进行估计$\boldsymbol{X}_{\mathrm{inc}}=\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ 0^{\mathrm{T}} & 1\end{bmatrix}$. 得到$\boldsymbol{X}_{\mathrm{inc}}$后,每层位姿更新为$\boldsymbol{X}=\boldsymbol{X}_{\mathrm{inc}}\boldsymbol{X}$.

2.2　多层ICP协方差量化

由式(5)计算各层协方差,该协方差由顶点和法向量构成,层与层之间存在重复的顶点和法向量,所以计算多层ICP的协方差不能直接相加各层的协方差,这样会导致得到的协方差包含许多重复信息. 第2.1节中构建的图像金字塔会产生重复的顶点和法向量,上一层图像信息经过下一层图像信息降采样而来,通过图像像素进行透视投影产生顶点和法向量时,两层会产生重复的顶点和法向量. 由于图像金字塔是逐层产生的,每一层的顶点和法向量在后面的层都会观测到,这些重复的顶点和法向量会生成相应的协方差矩阵,是要去除的重复信息. 多层ICP的每一层都会执行一次单层ICP优化能量函数求解运动参数,得到一个$\boldsymbol{X}_{\mathrm{inc}}$,所以每一层重复的顶点和法向量之间相差一个$\boldsymbol{X}_{\mathrm{inc}}$,将顶点与法向量乘以相关的刚体变换逆矩阵即可得到重复的顶点和法向量生成的协方差矩阵. 下面详述每一层重复信息产生对应协方差矩阵的方法.

$l=4$层的协方差N_4由式(5)计算,并且估算出一个相机旋转矩阵$\boldsymbol{R}_4$,点云对应的变换即为$\boldsymbol{R}'_4=\boldsymbol{R}_4^{-1}$,$\boldsymbol{N}_4$中使用的顶点和法向量在$l=3$层中存在重复信息,这些信息可以计算出一个相应的协方差矩阵,表示为$\boldsymbol{N}_{43}$,计算如下:

$$\boldsymbol{N}_{43}=$$
$$\sum_{\mu\in\boldsymbol{\Omega}_4}\begin{bmatrix}\boldsymbol{R}'_4(\boldsymbol{a}\times\boldsymbol{n})(\boldsymbol{R}'_4(\boldsymbol{a}\times\boldsymbol{n}))^{\mathrm{T}} & \rightarrow\\ \boldsymbol{R}'_4\boldsymbol{n}(\boldsymbol{R}'_4(\boldsymbol{a}\times\boldsymbol{n}))^{\mathrm{T}} & \\ \leftarrow & \boldsymbol{R}'_4(\boldsymbol{a}\times\boldsymbol{n})(\boldsymbol{R}'_4\boldsymbol{n})^{\mathrm{T}}\\ & \boldsymbol{R}'_4\boldsymbol{n}(\boldsymbol{R}'_4\boldsymbol{n})^{\mathrm{T}}\end{bmatrix}=$$
$$\sum_{\mu\in\boldsymbol{\Omega}_4}\begin{bmatrix}\boldsymbol{R}'_4(\boldsymbol{a}\times\boldsymbol{n})(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}}(\boldsymbol{R}'_4)^{\mathrm{T}} & \rightarrow\\ \boldsymbol{R}'_4\boldsymbol{n}(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}}(\boldsymbol{R}'_4)^{\mathrm{T}} & \\ \leftarrow & \boldsymbol{R}'_4(\boldsymbol{a}\times\boldsymbol{n})\boldsymbol{n}^{\mathrm{T}}(\boldsymbol{R}'_4)^{\mathrm{T}}\\ & \boldsymbol{R}'_4\boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}(\boldsymbol{R}'_4)^{\mathrm{T}}\end{bmatrix}=$$
$$\sum_{\mu\in\boldsymbol{\Omega}_4}\begin{bmatrix}\boldsymbol{R}'_4 & 0_{33}\\ 0_{33} & \boldsymbol{R}'_4\end{bmatrix}\begin{bmatrix}(\boldsymbol{a}\times\boldsymbol{n})(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}} & (\boldsymbol{a}\times\boldsymbol{n})\boldsymbol{n}^{\mathrm{T}}\\ \boldsymbol{n}(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}} & \boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}\end{bmatrix}\times$$

$$\begin{bmatrix}(\boldsymbol{R}_4')^{\mathrm{T}} & 0_{33}\\ 0_{33} & (\boldsymbol{R}_4')^{\mathrm{T}}\end{bmatrix}=$$

$$\begin{bmatrix}\boldsymbol{R}_4' & 0_{33}\\ 0_{33} & \boldsymbol{R}_4'\end{bmatrix}\sum_{\mu\in\boldsymbol{\Omega}_4}\begin{bmatrix}(\boldsymbol{a}\times\boldsymbol{n})(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}} & (\boldsymbol{a}\times\boldsymbol{n})\boldsymbol{n}^{\mathrm{T}}\\ \boldsymbol{n}(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}} & \boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}\end{bmatrix}\times$$

$$\begin{bmatrix}(\boldsymbol{R}_4')^{\mathrm{T}} & 0_{33}\\ 0_{33} & (\boldsymbol{R}_4')^{\mathrm{T}}\end{bmatrix}. \tag{6}$$

$l=3$ 层的协方差 $\boldsymbol{N}_3$ 由式(5)计算,并且可以估计出一个相机变换 $\boldsymbol{R}_3$,点云对应的变换即为 $\boldsymbol{R}_3'=\boldsymbol{R}_3^{-1}$. $\boldsymbol{N}_3$ 中使用的顶点和法向量在 $l=2$ 层中存在重复信息,由这些信息可以计算出一个相应的协方差矩阵,表示为 $\boldsymbol{N}_{32}$,执行与 $\boldsymbol{N}_{43}$ 同样的操作,计算如下:

$$\boldsymbol{N}_{32}=$$

$$\begin{bmatrix}\boldsymbol{R}_3' & 0_{33}\\ 0_{33} & \boldsymbol{R}_3'\end{bmatrix}\sum_{\mu\in\boldsymbol{\Omega}_3}\begin{bmatrix}(\boldsymbol{a}\times\boldsymbol{n})(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}} & (\boldsymbol{a}\times\boldsymbol{n})\boldsymbol{n}^{\mathrm{T}}\\ \boldsymbol{n}(\boldsymbol{a}\times\boldsymbol{n})^{\mathrm{T}} & \boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}\end{bmatrix}\times$$

$$\begin{bmatrix}(\boldsymbol{R}_3')^{\mathrm{T}} & 0_{33}\\ 0_{33} & (\boldsymbol{R}_3')^{\mathrm{T}}\end{bmatrix}. \tag{7}$$

$\boldsymbol{N}_4$ 中的顶点和法向量在 $l=2$ 层中存在重复信息,由这些信息可以计算出一个相应的协方差矩阵,表示为 $\boldsymbol{N}_{42}$,计算如下:

$$\boldsymbol{N}_{42}=\begin{bmatrix}\boldsymbol{R}_3' & 0_{33}\\ 0_{33} & \boldsymbol{R}_3'\end{bmatrix}\boldsymbol{N}_{43}\begin{bmatrix}(\boldsymbol{R}_3')^{\mathrm{T}} & 0_{33}\\ 0_{33} & (\boldsymbol{R}_3')^{\mathrm{T}}\end{bmatrix}. \tag{8}$$

$l=2$ 层的协方差矩阵 $\boldsymbol{N}_2$ 由式(5)计算,并且可以估算出一个变换 $\boldsymbol{R}_2$ 和 $\boldsymbol{t}_2$,点云对应的变换即为 $\boldsymbol{R}_2'=\boldsymbol{R}_2{}^{-1},\boldsymbol{t}_2'=-\boldsymbol{t}_2$. $\boldsymbol{N}_2$ 中使用的顶点和法向量在 $l=1$ 中存在重复信息,由这些信息可以计算出一个相应的协方差矩阵,表示为 $\boldsymbol{N}_{21}$,执行与 $\boldsymbol{N}_{43}$ 同样的操作可以得到 $\boldsymbol{N}_{21}$,计算如下:

$$\boldsymbol{N}_{21}=$$

$$\sum_{\mu\in\boldsymbol{\Omega}_2}\begin{bmatrix}\boldsymbol{R}_2' & \boldsymbol{t}_2' & 0_{33} & 0\\ 0_{33} & 0 & \boldsymbol{R}_2' & \boldsymbol{t}_2'\end{bmatrix}\times$$

$$\begin{bmatrix}(\overline{\boldsymbol{a}\times\boldsymbol{n}})(\overline{\boldsymbol{a}\times\boldsymbol{n}})^{\mathrm{T}} & (\overline{\boldsymbol{a}\times\boldsymbol{n}})\overline{\boldsymbol{n}}^{\mathrm{T}}\\ \overline{\boldsymbol{n}}(\overline{\boldsymbol{a}\times\boldsymbol{n}})^{\mathrm{T}} & \overline{\boldsymbol{n}\boldsymbol{n}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}(\boldsymbol{R}_2')^{\mathrm{T}} & 0_{33}\\ (\boldsymbol{t}_2')^{\mathrm{T}} & 0^{\mathrm{T}}\\ 0_{33} & (\boldsymbol{R}_2')^{\mathrm{T}}\\ 0^{\mathrm{T}} & (\boldsymbol{t}_2')^{\mathrm{T}}\end{bmatrix}=$$

$$\begin{bmatrix}\boldsymbol{R}_2' & \boldsymbol{t}_2' & 0_{33} & 0\\ 0_{33} & 0 & \boldsymbol{R}_2' & \boldsymbol{t}_2'\end{bmatrix}\overline{\boldsymbol{N}}_2\begin{bmatrix}(\boldsymbol{R}_2')^{\mathrm{T}} & 0_{33}\\ (\boldsymbol{t}_2')^{\mathrm{T}} & 0^{\mathrm{T}}\\ 0_{33} & (\boldsymbol{R}_2')^{\mathrm{T}}\\ 0^{\mathrm{T}} & (\boldsymbol{t}_2')^{\mathrm{T}}\end{bmatrix}. \tag{9}$$

为了构造 6×6 的协方差矩阵和适用 $\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ 0^{\mathrm{T}} & 1\end{bmatrix}$ 的形式,法向量使用齐次形式 $\overline{\boldsymbol{n}}=\begin{bmatrix}\boldsymbol{n}\\ 1\end{bmatrix}$,$(\overline{\boldsymbol{a}\times\boldsymbol{n}})$ 是叉乘 $\boldsymbol{a}\times\boldsymbol{n}$ 的齐次形式.

$\boldsymbol{N}_3$ 中使用的顶点和法向量在 $l=1$ 中存在重复信息,由这些信息可以计算出一个相应的协方差矩阵,表示为 $\boldsymbol{N}_{31}$,执行与 $\boldsymbol{N}_{21}$ 相同的操作

$$\boldsymbol{N}_{31}=\begin{bmatrix}\boldsymbol{R}_2' & \boldsymbol{t}_2' & 0_{33} & 0\\ 0_{33} & 0 & \boldsymbol{R}_2' & \boldsymbol{t}_2'\end{bmatrix}\overline{\boldsymbol{N}}_{32}\begin{bmatrix}(\boldsymbol{R}_2')^{\mathrm{T}} & 0_{33}\\ (\boldsymbol{t}_2')^{\mathrm{T}} & 0^{\mathrm{T}}\\ 0_{33} & (\boldsymbol{R}_2')^{\mathrm{T}}\\ 0^{\mathrm{T}} & (\boldsymbol{t}_2')^{\mathrm{T}}\end{bmatrix}. \tag{10}$$

其中 $\overline{\boldsymbol{N}}_{32}$ 是由式(7)计算出 $\boldsymbol{N}_{32}$ 并转化后的齐次形式.

$\boldsymbol{N}_4$ 中使用的顶点和法向量在 $l=1$ 中存在重复信息,由这些信息可以计算出一个相应的协方差矩阵,表示为 $\boldsymbol{N}_{41}$,执行与 $\boldsymbol{N}_{31}$ 相同的操作

$$\boldsymbol{N}_{41}=\begin{bmatrix}\boldsymbol{R}_2' & \boldsymbol{t}_2' & 0_{33} & 0\\ 0_{33} & 0 & \boldsymbol{R}_2' & \boldsymbol{t}_2'\end{bmatrix}\overline{\boldsymbol{N}}_{42}\begin{bmatrix}(\boldsymbol{R}_2')^{\mathrm{T}} & 0_{33}\\ (\boldsymbol{t}_2')^{\mathrm{T}} & 0^{\mathrm{T}}\\ 0_{33} & (\boldsymbol{R}_2')^{\mathrm{T}}\\ 0^{\mathrm{T}} & (\boldsymbol{t}_2')^{\mathrm{T}}\end{bmatrix}. \tag{11}$$

$l=1$ 层的协方差 $\boldsymbol{N}_1$ 由式(5)计算,最终相加每一层计算的协方差矩阵,并减去重复信息对应的协方差矩阵,得到最终的多层ICP的协方差矩阵为

$$\boldsymbol{N}=\boldsymbol{N}_1+\boldsymbol{N}_2+\boldsymbol{N}_3+\boldsymbol{N}_4-\boldsymbol{N}_{41}-\boldsymbol{N}_{31}-\boldsymbol{N}_{21}-\boldsymbol{N}_{42}-\boldsymbol{N}_{32}-\boldsymbol{N}_{43}. \tag{12}$$

2.3 GPU并行实现

在实际操作中,惯性传感器采集数据的频率大于深度相机获取图像的频率,为了融合两者的数据需要进行数据对齐,通过计算 $\boldsymbol{X}$ 的预测值 $\boldsymbol{X}^-$ 实现,定义为

$$\boldsymbol{X}^-=\begin{cases}\boldsymbol{t}_k^-=\boldsymbol{t}_{k-1}^+,\\ \boldsymbol{R}_k^-=\boldsymbol{R}_{k-1}^+\exp((\omega^\wedge)\delta t).\end{cases} \tag{13}$$

其中:脚标 $k-l$ 为上一帧图像索引,脚标 k 为当前帧图像索引,$\boldsymbol{t}_{k-1}^+$ 和 $\boldsymbol{R}_{k-1}^+$ 为上一帧图像估算出的刚体变换,$\omega^\wedge$ 为由陀螺仪中读取的角速度信息并转化为李代数形式,δt 为陀螺仪中两个连续数据的时间间隔. 式(13)表示对陀螺仪中读取的旋转分量进行积分,直到获得一帧新的图像数据,加速度计中读取的平移分量比较容易受到噪声和偏置的干扰,因此直接使用上一帧图像估计出的平移分量代替.

将 $\boldsymbol{X}^-$ 输入到多层ICP中估算出一个 $\hat{\boldsymbol{X}}$,并计算多层ICP的协方差 $\boldsymbol{N}$,如图2所示,包含如下步骤.

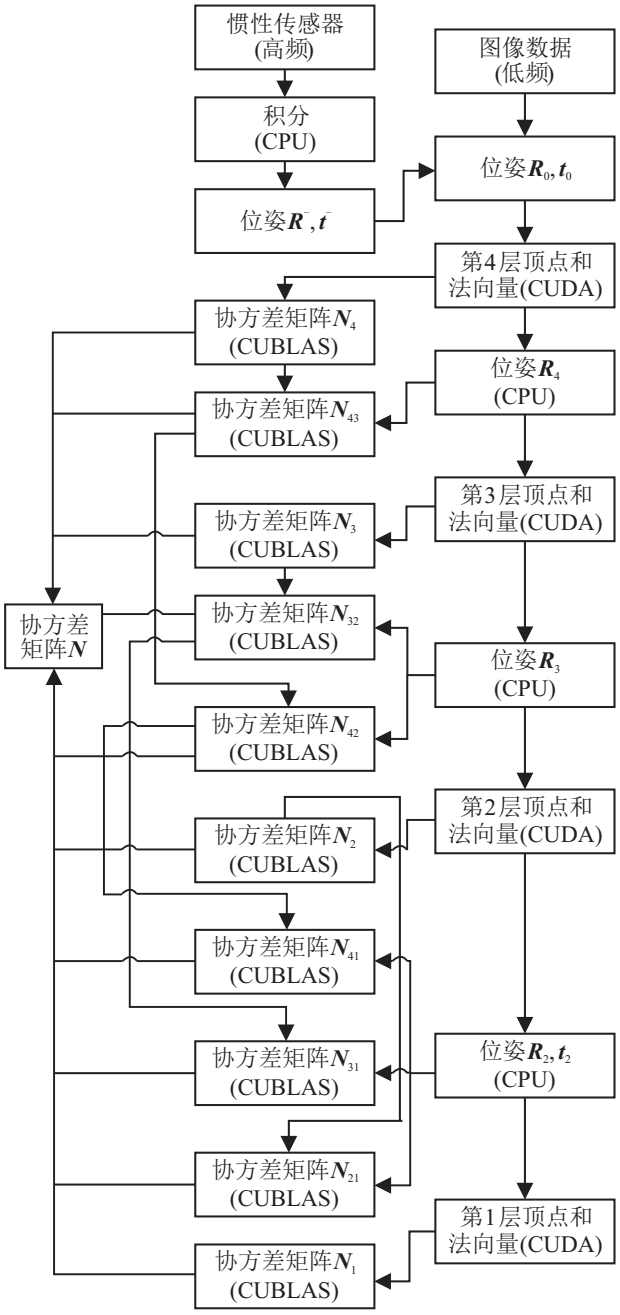


图 2　协方差矩阵计算框图

Step 1: $\boldsymbol{R}_k^-, \boldsymbol{t}_k^-$ 设置为多层 ICP 的初值 $\boldsymbol{R}_0 = \boldsymbol{R}_k^-, \boldsymbol{t}_0 = \boldsymbol{t}_k^-$;

Step 2: $\boldsymbol{R}_0, \boldsymbol{t}_0$ 交换到 CUDA(GPU 并行运算库)中;

Step 3: 由式(5)计算 $\boldsymbol{N}_4$, 顶点和法向量的求和结果同时被用于计算ICP的雅可比矩阵;

Step 4: 雅可比矩阵交换到CPU, 并使用乔列斯基分解求出 $\boldsymbol{R}_4$, 更新 $\boldsymbol{R}_0$ 并交换回CUDA中, 为计算下一层位姿做准备;

Step 5: 由式(6)在CUBLAS(GPU矩阵运算库)中进行计算;

Step 6: 由式(5)进行计算, 顶点和法向量求和结果被用于计算ICP的雅可比矩阵;

Step 7: 雅可比矩阵交换到CPU中, 并使用乔列斯基分解求出 $\boldsymbol{R}_3$, 更新 $\boldsymbol{R}_0$ 并交换回CUDA中, 为计算下一层位姿做准备;

Step 8: $\boldsymbol{N}_{32}$、$\boldsymbol{N}_{42}$ 由式(7)和(8)在CUBLAS(GPU矩阵运算库)中进行计算;

Step 9: 由式(5)计算 $\boldsymbol{N}_2$, 顶点和法向量求和结果被用于计算ICP的雅可比矩阵;

Step 10: $\boldsymbol{R}_0, \boldsymbol{t}_0$ 交换回CUDA中, 为计算下一层位姿做准备;

Step 11: 根据式(9)~(11), 在CUBLAS(GPU矩阵运算库)中计算 $\boldsymbol{N}_{21}, \boldsymbol{N}_{31}, \boldsymbol{N}_{41}$;

Step 12: 由式(5)计算 $\boldsymbol{N}_1$;

Step 13: 由式(12)计算出最后的协方差矩阵 $\boldsymbol{N}$.

上述步骤使用了 GPU 并行加速方法, 如 $\boldsymbol{N}_1$ 的计算是在分辨率为 640×480 的深度图像上进行的, 参与计算的顶点数量级达到500 000. 在实现中, 通过GPU并行计算来提升实时性能, 计算时, 需要将GPU与CPU之间的数据进行交换, 进行交换的矩阵最大维度是 6×6, 所以数据交换消耗的带宽也非常少. 将计算得到的协方差代入式(3)计算卡尔曼增益 $\boldsymbol{K}$, 进而由式(2)计算不变卡尔曼滤波器对应的不变输出项 $\boldsymbol{E}$, 并得到最终的位姿如下:

$$\boldsymbol{X}^+ = \begin{cases} \boldsymbol{t}_k^+ = \hat{\boldsymbol{t}} + (\boldsymbol{e}_\omega^\wedge \hat{\boldsymbol{t}} + \boldsymbol{e}_v), \\ \boldsymbol{R}_k^+ = \exp((\boldsymbol{e}_\omega^\wedge)\Delta\boldsymbol{t})\hat{\boldsymbol{R}}. \end{cases} \tag{14}$$

其中: $\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}$ 为多层ICP输出的旋转量和平移量, $\Delta\boldsymbol{t}$ 为连续两帧深度图像的时间间隔.

3　基于随机蕨和惯性传感器的重定位

文献[11]提出了随机蕨定位算法, 在视觉重定位上取得了较好的结果. 本节结合随机蕨定位算法与惯性传感器数据, 建立相机运动的速度模型, 可以在相机运行较快的场景实现视觉惯性里程计的重定位.

3.1　图像编码与蕨的建立

将采集到的彩色图像分成3个通道, 并且使用降采样得到图像金字塔, 将3个通道和深度图像中的最顶层取出(分辨率为 80×60), 随机取500个位置, 形成500个包含 $\theta_1, \theta_2, \theta_3, \theta_4$ 的蕨, 每个节点作如下处理:

$$f(\boldsymbol{I}, \theta, \tau) = \begin{cases} 1, & \boldsymbol{I}(\theta) \geqslant \tau; \\ 0, & \boldsymbol{I}(\theta) < \tau. \end{cases} \tag{15}$$

其中: $\boldsymbol{I}$ 为像素值, 彩色三通道加上深度共4个像素对应 $\theta_1, \theta_2, \theta_3, \theta_4$; τ 为阈值, 4个像素值(代表节点)对应4个阈值, 彩色三通道在0~255随机选取, 深度值在800~4 000随机选取. 至此每个节点被编码成 $\boldsymbol{b}_F = f_1, f_2, f_3, f_4$, 500个离散的蕨可以编码一幅图像 $\boldsymbol{b}_C =$

$\boldsymbol{b}_{F1}, \boldsymbol{b}_{F2}, \cdots, \boldsymbol{b}_{F500}$.

3.2 汉明距离与关键帧的选择

给定两个时刻的图像$\boldsymbol{I}_1, \boldsymbol{I}_2$,它们的随机蕨编码为$\boldsymbol{b}_C^{I_1}, \boldsymbol{b}_C^{I_2}$,帧间不相似性可以使用汉明距离度量,有

$$\kappa = \mathrm{BLOCKHD}(\boldsymbol{b}_C^{I_1}, \boldsymbol{b}_C^{I_2}) = \frac{1}{m}\sum_{k=1}^{m} \boldsymbol{b}_F^{I_1} \equiv \boldsymbol{b}_F^{I_2}. \quad (16)$$

其中:m为离散蕨的个数;符号$\equiv$定义为,如果$\boldsymbol{b}_F^{I_1}$和$\boldsymbol{b}_F^{I_2}$所有位都相同则返回0,如果有一位不同则返回1. 求得最终结果κ归一化到$[0, 1]$. 在实际实现的过程中定义一个全局表B,由于有4个节点,全局表共有2^4个编码,每个编码对应帧的id,从2^4个编码中统计具有两帧id编码的个数为q,式(16)可以改写为

$$\kappa = \mathrm{BLOCKHD}(\boldsymbol{b}_C^{I_1}, \boldsymbol{b}_C^{I_2}) = \frac{m-q}{m}. \quad (17)$$

当前帧图像与关键帧集合中的每一帧进行比较,如果所有κ大于一个预先设置的阈值τ_k(实验中为0.7),则将此帧加入到关键帧集合$\boldsymbol{H}$中.

3.3 关键帧集合中数据格式

一个关键帧包含如下数据:1)关键帧的帧号id;2)关键帧的位姿$\boldsymbol{X}$以及这个位姿转化成的六自由度向量$\boldsymbol{\xi}$;3)一个对角协方差矩阵$\boldsymbol{\Sigma}$,该矩阵在下一节中描述.

3.4 位姿恢复

当相机出现定位失败的情况时,需要对相机进行重定位,将待定位的帧与关键帧集合中的所有帧进行不相似度匹配,取其中n个κ值最低的关键帧进行一次多层ICP配准,在配准过程中,首先根据选中的关键帧进行速度建模. 认为捕获当前帧的相机运动是一个恒速模型,符合高斯分布,即$p(\boldsymbol{\xi}) \sim N(\boldsymbol{\xi}_i, \boldsymbol{\Sigma}_i)$. 其中:$\boldsymbol{\xi}$为当前帧的六自由度向量;$\boldsymbol{\xi}_i$为匹配好的关键帧的六自由度向量;$\boldsymbol{\Sigma}_i$为关键帧的对角协方差矩阵,表示6个自由度变化的快慢,可以通过惯性传感器获得. 建立该模型后,在进行ICP求$\Delta\boldsymbol{\xi}$时,能量函数的求解变为如下形式:

$$(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \boldsymbol{\Sigma}^{-1})\Delta\boldsymbol{\xi} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{r} + \boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi}_i - \boldsymbol{\xi}^{(k)}), \quad (18)$$

其中$\boldsymbol{\xi}^{(k)}$为$\boldsymbol{\xi}$第k次迭代的位姿更新结果. 对n个匹配的关键帧作上述形式的重定位,直到找到准确位姿为止.

4 实验结果与分析

4.1 实验硬件、软件、数据集

系统采用C++编写软件,运行在CPU为i7-4790,GPU为GTX770的计算机上,使用深度相机为primesense 1.09,惯性传感器为LPMS-CU. 此款惯性传感器包含三轴陀螺仪测角速度,三轴线加速度计测线速度,三轴地磁仪测地磁场方向,九轴数据使偏置对惯性传感器的影响非常小,只有1.8°/h. 深度相机与惯性传感器固定在一起如图3所示. 实验中,测试的数据集见文献[14],该数据集包含许多使用深度相机拍摄的室内视频序列,但是没有提供惯性传感器数据,惯性传感器数据可通过文献[15]进行模拟,惯性传感器真值可以根据地标通过不同的C^2连续B样条拟合,将这些真值加上相应的高斯白噪声即可模拟出惯性传感器的实际测量值. 高斯白噪声的相关参数由LPMS-CU中获得,如表1所示.

图3 深度相机和惯性传感器

表1 惯性传感器相关参数

陀螺仪			加速度计		
σ_{v_1}	σ_{v_2}	σ_{v_3}	σ_{ω_1}	σ_{ω_2}	σ_{ω_3}
0.0166	0.0392	0.0416	0.0069	0.0082	0.0085

4.2 数据集验证本文实验

使用第4.1节拟合出的惯性传感器数据以及数据集中的视频序列将本文方法与其他方法[8-9,16-17]进行比较以证明所提出的方法在相机定位准确性上的提升. 实验运行在fr2-360-hemisphere和fr1-desk2两个序列上,保存产生的相机轨迹与数据集中提供的轨迹真值作比较,并利用文献[14]提供的公式计算相关误差,结果如表2和表3所示. 由表2和表3可见,通过文献[17]得到的结果误差较大,这是因为文献[17]采用视觉里程计的方法对初值依赖较大. 文献[8-9,16]及本文方法通过加入惯性设备改善初值,并通过卡尔曼滤波器融合相机位姿的估计值与惯性传感器测量值减小误差. 本文采用由粗到细的多层结构迭代定位相机,所得的结果误差最小.

表2 fr2-360-hemisphere相关误差

	文献[8]	文献[16]	文献[17]	文献[9]	本文
均方根误差	0.0716	0.0723	0.0846	0.0645	0.0569
平均误差	0.0693	0.0702	0.0813	0.0613	0.0527
中值误差	0.0674	0.0693	0.0799	0.0606	0.0508
标准误差	0.0659	0.0665	0.0783	0.0599	0.0501
最小误差	0.0326	0.0344	0.0446	0.0289	0.0229
最大误差	0.1575	0.1611	0.1875	0.1164	0.1033

表 3　fr1-desk2相关误差

	文献[8]	文献[16]	文献[17]	文献[9]	本文
均方根误差	0.063 7	0.072 5	0.078 1	0.058 5	0.052 9
平均误差	0.061 3	0.071 0	0.076 4	0.057 3	0.050 1
中值误差	0.060 4	0.070 3	0.075 2	0.056 9	0.048 8
标准误差	0.059 1	0.069 6	0.074 3	0.054 5	0.047 9
最小误差	0.022 6	0.024 4	0.025 5	0.019 7	0.015 9
最大误差	0.113 3	0.139 6	0.141 4	0.097 5	0.084 3

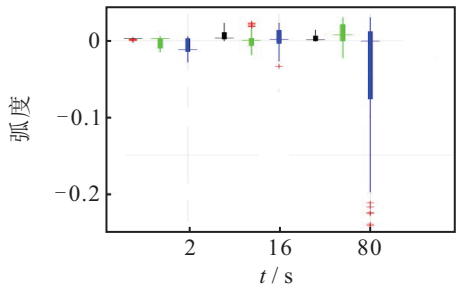
使用第4.1节拟合出的惯性传感器数据以及数据集中的视频序列验证层数设置对本文方法的影响,实验运行在视频序列fr2-large-no-loop上,使用文献[14]的公式分析相关误差,结果如表4所示. 比较这些数据可以发现,层数越深精度越高. 本文设置的图像金字塔同时兼顾多层ICP估计和重定位中图像编码的设置,选择了4层. 如果选择5层,则图像分辨率为40 × 30,选取500个位置过于稠密,不利于重定位检测;选择3层定位精度不如4层.

表 4　fr2-large-no-loop相关误差

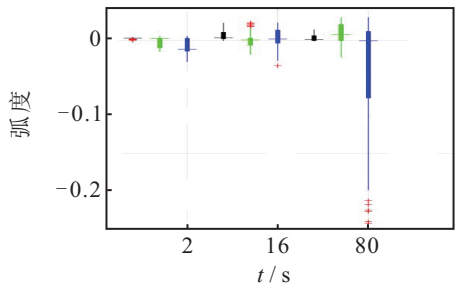
	2层	3层	4层	5层
均方根误差	0.063 7	0.072 5	0.078 1	0.058 5
平均误差	0.061 3	0.071 0	0.076 4	0.057 3
中值误差	0.060 4	0.070 3	0.075 2	0.056 9
标准误差	0.059 1	0.069 6	0.074 3	0.054 5
最小误差	0.022 6	0.024 4	0.025 5	0.019 7
最大误差	0.113 3	0.139 6	0.141 4	0.097 5

4.3　实际场景中验证本文实验

本节在实际场景中验证所提出的算法可以提高相机定位的精度,一个有效途径是通过观测漂移的减少验证算法的精度. 为了观测漂移,设计实验将固定好的深度相机和惯性传感器放置在一个水平放置的转盘上,当转盘转动时,两个设备绕着一个垂直轴进行旋转,记录俯仰角和翻滚角的测试值(位姿矩阵转换为欧拉角形式),理想状态下由于绕着垂直轴进行旋转,俯仰角和翻滚角为零,观测俯仰角和翻滚角在零值附近的波动误差即可验证漂移减少的效果[8-9]. 利用文献[9,17]和本文方法进行测试,相关观测数据如图4所示. 图中记录了3种方法在2 s、18 s、80 s三个时间点的统计结果,为了方便比较,3种方法被分开进行显示,在每个时间点的3列数据中,左边一列对应本文方法的结果,中间一列对应文献[9]的结果,右边一列对应文献[17]的结果. 由比较实验结果可以看出,本文所提出的方法可以更大程度地减少漂移.



(a)　俯仰角观测结果



(b)　翻滚角观测结果

图 4　漂移观测图

4.4　算法实时性能验证

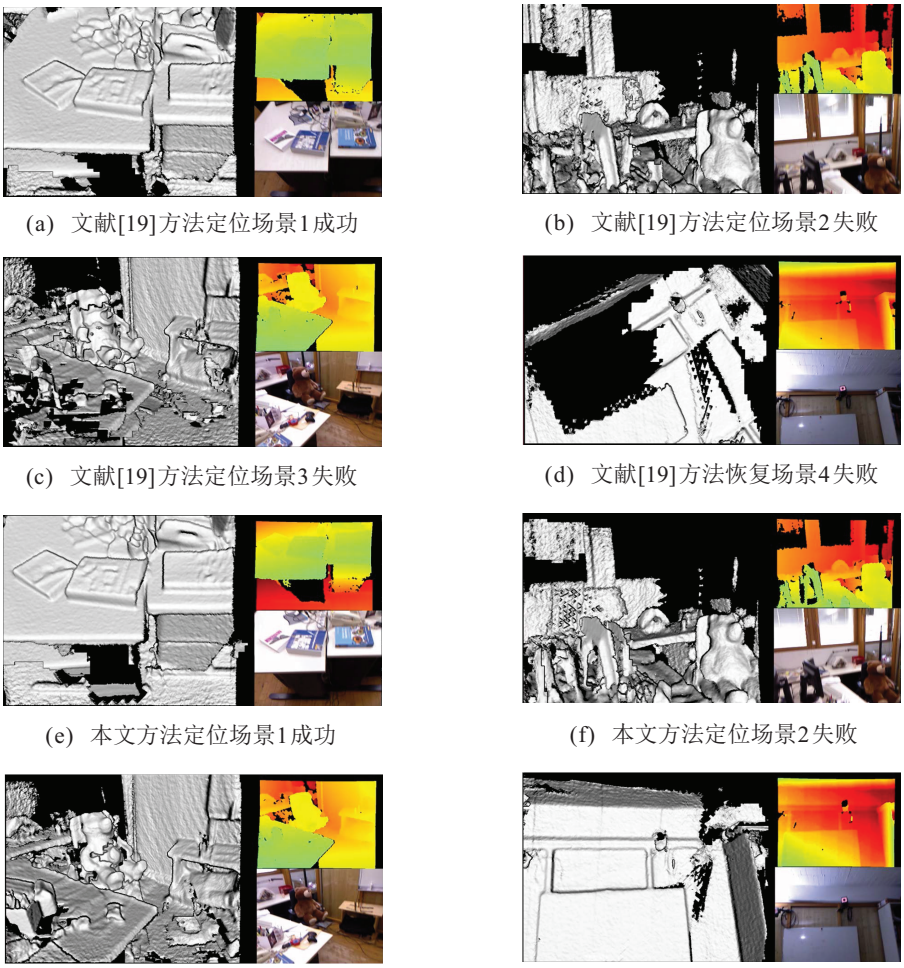
本节验证GPU加速可以有效提高系统实时性能,结果如表5所示. 文献[9]操作约100 000个点且运行在CPU上,所花费的时间大于本文提出的方法. 文献[16,18]和本文均操作约500 000个点,且运行在GPU上,由于本文提出的方法使用多层ICP方法,需要在多层上估计位姿且每层都要实现CPU与GPU交换数据,时间上要比使用单层ICP[16,18]的时间略多.

表 5　实时性能验证

方法	设备	点的数量	时间/ms
文献[9]	CPU	100 000	38.64
文献[18]	CPU和GPU	500 000	14.68
文献[16]	CPU和GPU	500 000	13.68
本文	CPU和GPU	500 000	18.26

4.5　重定位效果验证

本节通过重建视频序列中的三维场景来直观显示重定位方法的效果. 实验运行在视频序列上,这是一个有挑战性的视频序列,非常容易追踪相机失败,但是场景中提供了丰富的可以重定位的信息. 图5通过4组图像显示了文献[19]和本文方法进行定位的结果,在图像中,左边部分是重建产生的光线图,右上部分是使用重定位ICP进行配准的效果图像,右下部分是当前帧对应的场景图像. 图5(a)～图5(d)为文献[19]的追踪过程,图5(e)～图5(h)为本文追踪相似场景的过程. 图5(a)和图5(e)是追踪成功的场景,产生了正常的光线图;图5(b)、图5(c)、图5(f)、图5(g)是追踪失败的场景,产生的光线图中场景较乱;图5(d)由



(a) 文献[19]方法定位场景1成功　(b) 文献[19]方法定位场景2失败

(c) 文献[19]方法定位场景3失败　(d) 文献[19]方法恢复场景4失败

(e) 本文方法定位场景1成功　(f) 本文方法定位场景2失败

(g) 本文方法定位场景3失败　(h) 本文方法恢复场景4成功

图5 重定位示意图

于相机运动较快导致重定位没有成功,产生的光线图仍然很乱;图5(h)使用重定位成功,重新恢复了正常的光线图. 实验结果表明,本文所提出的方法可以有效实现重定位.

为了定量显示所提出的重定位算法,表6显示了在实际场景中的实现重定位的百分比,将摄像机在室内旋转100圈,第4.4节中相似候选关键帧数选择为5,基线短表示相机运行较慢,基线长表示相机运行较快,记录实现重定位的次数,从而计算百分比. 结果显示,在相机运行较慢时,本文方法的效果与文献[11]相似,在相机运行较快时,本文方法的效果明显好于文献[11].

表6 重定位性能验证

方法	相似候选关键帧数n	基线长度	重定位占比/%
本文	5	短	81.4
文献[11]	5	短	80.7
本文	5	长	74.8
文献[11]	5	长	60.5

5 结 论

本文设计了一种定位准确且能重定位的视觉惯性里程计. 定位部分采用多层ICP估计值和惯性传感器数据融合的方法增加定位的精度，多层ICP的协方差矩阵被估计并用于量化估计位姿的误差,使用不变卡尔曼滤波可以减少该误差. 定位的关键步骤使用GPU并行加速,有效地提高了系统的实时性能. 当出现相机定位失败时,视觉惯性里程计的重定位部分利用惯性传感器数据建立一个恒速模型,结合随机蕨方法可以得到有效的重定位信息. 通过实验验证并与已有结果比较,表明了所提出方法有较高的定位精度并且可以实现相机较快运行下的重定位.

参考文献(References)

[1] 罗杨宇, 刘宏林. 基于光束平差法的双目视觉里程计研究[J]. 控制与决策, 2016, 31(11): 1936-1944.
(Luo Y Y, Liu H L. Research on binocular vision odometer based on bundle adjustment method[J]. Control and Decision, 2016, 31(11): 1936-1944.)

[2] 李永锋,张国良,王蜂, 等. 基于快速视觉里程计和大

回环局部优化模型的改进VSLAM算法[J]. 机器人, 2015, 37(5): 557-565.
(Li Y F, Zhang G L, Wang F, et al. Improved VSLAM algorithm based on fast visual odometry and large loop local optimization model[J]. Robot, 2015, 37(5): 557-565.)

[3] 蔡迎波. 基于多状态约束的视觉/惯性组合导航算法研究[J]. 光学与光电技术, 2015, 13(6): 58-62.
(Cai Y B. Multi-State constraint algorithm for vision-aided inertial navigation system[J]. Optics & Optoelectronic Technology, 2015, 13(6): 58-62.)

[4] Cheviron T, Hamel T, Mahony R, et al. Robust nonlinear fusion of inertial and visual data for position, velocity and attitude estimation of UAV[C]. IEEE Int Conf on Robotics and Automation. New York: IEEE, 2007: 2010-2016.

[5] Huang G P, Mourikis A I, Roumeliotis S I. A first-estimates jacobian EKF for improving SLAM consistency[C]. Experimental Robotics. Berlin: Springer, 2009: 373-382.

[6] Martin P, Salaün E. Generalized multiplicative extended kalman filter for aided attitude and heading reference system[C]. AIAA Guidance, Navigation, and Control Conf. New York: AIAA, 2011: 1-13.

[7] Bonnabel S, Martin P, Salaun E. Invariant extended kalman filter: Theory and application to a velocity-aided attitude estimation problem[C]. IEEE Conf on Decision and Control. New York: IEEE, 2009: 1297-1304.

[8] Hervier T, Bonnabel S, Goulette F. Accurate 3D maps from depth images and motion sensors via nonlinear kalman filtering[C]. IEEE Int Conf on Intelligent Robots and Systems. New York: IEEE, 2012: 5291-5297.

[9] Barczyk M, Bonnabel S, Deschaud J E, et al. Invariant EKF design for scan matching-aided localization[J]. IEEE Trans on Control Systems Technology, 2015, 23(6): 2440-2448.

[10] Zhang T, Wu K, Song J, et al. Convergence and consistency analysis for A 3D invariant-EKF SLAM[J]. IEEE Robotics & Automation Letters, 2017, 2(2): 733-740.

[11] Glocker B, Shotton J, Criminisi A, et al. Real-time RGB-D camera relocalization via randomized ferns for keyframe encoding[J]. IEEE Trans on Visualization & Computer Graphics, 2015, 21(5): 571-583.

[12] Zhang Z. Microsoft kinect sensor and its effect[J]. IEEE Multimedia, 2012, 19(2): 4-10.

[13] Newcombe R A, Izadi S, Hilliges O, et al. Kinect fusion: Real-time dense surface mapping and tracking[C]. IEEE Int Symposium on Mixed and Augmented Reality. New York: IEEE, 2011: 127-136.

[14] Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems[C]. IEEE Int Conf on Intelligent Robots and Systems. New York: IEEE, 2012: 573-580.

[15] Huai J, Toth C K, Grejner-Brzezinska D A. Stereo-inertial odometry using nonlinear optimization[C]. Int Technical Meeting of the Satellite Division of the Institute of Navigation. Salt Lake Lity: IEEE, 2015: 2087-2097.

[16] Zhang G S, Huang W J, Shen Y, et al. Fast and accurate 3d reconstruction using a moving depth camera and inertial sensor[C]. Chinese Automation Congress. New York: IEEE, 2017: 1255-1260.

[17] Ren C Y, Reid I. A unified energy minimization framework for model fitting in depth[M]. Berlin: Springer Heidelberg, 2012: 72-82.

[18] 张国山, 尚红霞, 王欣博. 陀螺仪在基于Hash结构的三维重建中的应用[J]. 天津大学学报: 自然科学版, 2016, 49(11): 1132-1137.
(Zhang G S, Shang H X, Wang X B. Application of gyroscope in 3D reconstruction based on Hash structure[J]. J of Tianjin University: Science and Technology, 2016, 49(11): 1132-1137.)

[19] Kähler O, Prisacariu U A, Murray D W. Real-time large-scale dense 3D reconstruction with loop closure[C]. The 14th European Conf on Cmputer. Berlin: Springer, 2016: 500-516.

作者简介

黄伟杰 (1987–), 男, 博士生, 从事三维重建及其应用的研究, E-mail: huang1987515@126.com;

张国山 (1961–), 男, 教授, 博士生导师, 从事非线性系统控制、智能控制与工程应用、三维重建等研究, E-mail: zhanggs@tju.edu.cn.

(责任编辑：郑晓蕾)