基于 GPU 和 Kinect 的快速物体重建

刘 鑫 [1] 许华荣 [2] 胡占义 [1]

摘 要 便宜的物体快速三维建模技术是当前计算机视觉领域重要的研究课题. 给出了一种基于 Kinect 传感器的快速物体重建方法, 以及基于该方法的一种图形处理器 (Graphic processing unit, GPU) 原型系统实现. 本文方法主要分为两步: 1) 系统的初始标定; 2) 全自动的物体重建. 对于系统初始标定, 提出了一种简单易用的粗标定方法; 对于物体重建, 提出一种全自动的快速物体重建方法. 本文方法鲁棒性高, 在出现点云配准错误时仍然能够稳定地得到较理想的重建模型. 针对环闭合 (Loop-closure) 问题, 提出了一种全局的点云配准方法. 对几类物体的重建实验结果表明, 本文方法方便实用, 且能得到较好的重建效果. 此外, 本文还探索了有遮挡物体的重建问题. 将本文方法应用于有遮挡物体的重建, 也取得了较好的重建效果.

关键词 三维物体建模, 图形处理器, Kinect, 遮挡问题, 环闭合

引用格式 刘鑫, 许华荣, 胡占义. 基于 GPU 和 Kinect 的快速物体重建. 自动化学报, 2012, **38**(8): 1288−1297

DOI 10.3724/SP.J.1004.2012.01288

GPU Based Fast 3D-Object Modeling with Kinect

LIU Xin[1] XU Hua-Rong[2] HU Zhan-Yi[1]

Abstract The cost-effective 3D-object modeling is a topic in computer vision field that is full of significance. In this paper, we report a method for object modeling with Kinect, as well as its implementation on graphic processing unit (GPU). Our modeling solution is divided into two steps: one is the system calibration, and the other is the fast 3D-object modeling. We present a easy-to-use calibration method for the first step, and a fully automatic modeling method for the other. Besides, our method is robust to registration of point clouds. For the well-known loop closure problem, we present a global registration method. Our method is validated by the modeling of several difficult real-world objects. In addition, the modeling of the objects with occlusion is also investigated and satisfactory results are obtained.

Key words 3D object modeling, graphic processing unit (GPU), Kinect, occlusion, loop-closure

Citation Liu Xin, Xu Hua-Rong, Hu Zhan-Yi. GPU based fast 3D-object modeling with Kinect. *Acta Automatica Sinica*, 2012, **38**(8): 1288−1297

目前, 物体三维模型在设计仿真、虚拟现实、3D 电影和文化保护等诸多领域, 应用十分广泛. 然而, 现有的三维重建技术常常基于一些复杂和昂贵的传感器, 如结构光相机或激光测距仪等. 若三维重建能够更加便宜, 使得三维物体模型如照片和视频一样容易获得, 则物体模型能够被应用到更多的领域, 如商务网站平台和在线购物网站等. 因此, 如何快速便宜地对物体建模, 是计算机视觉的一个重要目标.

Kinect 传感器的出现, 可以说是一场革命性的变革. Kinect 本来是微软公司开发的 Xbox360 主机的周边外设, 主要用于人机实时交互, 但也有一些文献报道将其用在三维重建中[1−6]. Engelhard 等[1] 利用 Kinect 提供的 RGB-D 相机实现了一个实时的视觉 SLAM 系统, 该系统能够用于场景重建. 其主要方法是利用彩色摄像机进行 SURF 特征匹配[7], 先获得摄像机位置的初值, 然后用 ICP (Iterative closest point) 算法[8] 进行点云配准并对相机位置进行优化. Henry 等[2−3] 利用 Kinect 实现了一个交互式的三维重建系统, 该系统仅选取关键帧进行 ICP 配准. 这两种方法均需要进行图像特征提取与匹配, 而一般的物体图像则较难提取出可靠的匹配特征. Izadi 等[4−5] 给出了一种基于 GPU 并行计算的实时定位与重建系统, 并实现了动态场景的增强现实应用. 但该系统重建结果依赖于实时的 ICP 配准, 配准错误影响系统的稳定性, 而配准误差使得重建的三维模型存在一定的环闭合 (Loop-closure) 问题. Tong 等[6] 给出了一种基于 Kinect 的人体 (有轻微形变的非刚体) 重建方法. 该方法首先需要对

收稿日期 2012-01-10 录用日期 2012-05-22
Manuscript received January 10, 2012; accepted May 22, 2012
国家自然科学基金 (60905020), 福建省科技厅重点项目 (2011H0032) 资助
Supported by National Natural Science Foundation of China (60905020) and the Key Project of Fujian Province (2011H0032)
本文责任编委 贾云得
Recommended by Associate Editor JIA Yun-De
1. 中国科学院自动化研究所模式识别国家重点实验室 北京 100190 2. 厦门理工学院计算机科学与技术系 厦门 361024
1. National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190 2. Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361024

人体进行建模. 然后利用图像特征点实现相邻帧的局部配准, 并进行全局优化. 该系统通过局部配准与全局优化的反复迭代来获得人体模型. 该系统需要对人体进行建模, 不适用于一般的物体重建, 因为对一般的物体图像, 提取可靠的匹配特征是很困难的. 此外, 反复迭代的配准策略会一定程度地影响算法的时间性能. 对于物体重建, 基于 Kinect 的重建技术有如下的优势: Kinect 能够快速获取场景的深度信息; Kinect 是一种主动传感器, 它不受环境可见光谱的干扰[9−10]; Kinect 的核心设备是彩色摄像机、红外线发射器和红外线 CMOS 摄影机, 这些设备都比较廉价, 因而 Kinect 的售价也较为低廉; 此外, Kinect 的操作与普通摄像机类似, 易于使用.

本文旨在报道一种基于 GPU 和 Kinect 的物体三维重建方法以及该方法的 GPU 实现. 其中, 利用 Kinect 来获取物体的三维点云, 而利用基于 GPU 的快速算法来对获得的点云进行配准和优化. 本文方法的主要特点有: 1) 该方法是一种全自动的物体重建方法. 与文献 [1−5] 的方法相比, 利用本文方法进行重建时无需手动旋转物体或移动 Kinect 传感器. 2) 该方法鲁棒性高, 在出现点云配准错误时仍然能够得到理想的重建模型. 与文献 [1−3, 6] 的方法相比, 本文方法不需要提取图像特征点进行局部配准. 这使得本文方法适用于纹理并不丰富的物体重建. 由于利用了转台的匀速旋转特性, 本文的局部配准方法也不同于文献 [4−5] 中的配准方法. 3) 系统不需要精确标定, 只需要粗略标定即可. 4) 能比较有效地处理 Loop-closure 问题. 与文献 [6] 的方法相比, 本文方法的局部配准和全局配准仅需进行一次. 此外, 为了探索有遮挡物体的重建问题, 我们将本文方法应用于有遮挡物体的重建, 也取得了比较满意的重建效果.

1　ICP 算法及 EM-ICP 算法

ICP 算法和 EM-ICP 算法是本文方法用到的核心算法. ICP 算法是经典的点云配准算法. 设有两个三维点集 $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$ 和 $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m\}$, ICP 算法的目的就是要找到旋转矩阵 R 和位移向量 $\boldsymbol{t}$, 使得 X (固定) 和 Y' (Y' 是 Y 经过 R 和 $\boldsymbol{t}$ 变换后的集合) 的残留误差最小. 对 ICP 算法的详细介绍可参阅文献 [7, 11−12], 这里不再赘述.

EM-ICP 算法是 ICP 算法的变种算法, 它们的主要区别是: 对于 ICP 算法, 点集 X 中的每一个点 $\boldsymbol{x}_j$ ($j = 1, 2, \cdots, n$), 在点集 Y 中仅有一个独一无二的对应点; 相反, 对于 EM-ICP 算法, 点集 Y 中的每个点都与 $\boldsymbol{x}_j$ 有一个对应关系, 以权重来表示可信程度. 因此, EM-ICP 算法通过最小化如下的误差函数来找到旋转矩阵 R 和位移向量 $\boldsymbol{t}$, 使得 X 和 Y' 的残留误差最小.

$$E = \sum_{j=1}^{n}\sum_{i=1}^{m}\alpha_{ij}d_{ij}^2,\ \ d_{ij} = \|\boldsymbol{x}_j - (R\boldsymbol{y}_i + \boldsymbol{t})\| \quad (1)$$

其中, α_{ij} 表示点 $\boldsymbol{x}_j$ 与点 $\boldsymbol{y}_i$ 匹配的概率, 由下式得到:

$$\alpha_{ij} = \frac{1}{C_i}\exp\left(\frac{-d_{ij}^2}{\sigma_p^2}\right) \quad (2)$$

$$C_i = \exp\left(\frac{-d_0^2}{\sigma_p^2}\right) + \sum_{k=1}^{n+1}\exp\left(\frac{-d_{ik}^2}{\sigma_p^2}\right) \quad (3)$$

其中, σ_p 和 d_0 是常数. 式 (1) 可重写为[11]

$$E = \sum_{i=1}^{m}\lambda_i^2\|\boldsymbol{x}_i' - (R\boldsymbol{y}_i + t)\| \quad (4)$$

其中

$$\lambda_i = \sum_{j=1}^{n}\sqrt{\alpha_{ij}},\ \ \boldsymbol{x}_i' = \frac{1}{\lambda_i}\sum_{j=1}^{n}\sqrt{\alpha_{ij}}\boldsymbol{x}_j \quad (5)$$

EM-ICP 算法如算法 1 所示.

算法 1. EM-ICP 算法

1. $R^0 \leftarrow I$, $\boldsymbol{t}^0 \leftarrow \boldsymbol{0}$, $k \leftarrow 0$, $\sigma_p \leftarrow sigma_p$, $\sigma_{\text{recon}} \leftarrow sigma\_\text{inf}$, $d_0^2 \leftarrow d_{02}$.
2. **while** $\sigma_p > \sigma_{\text{recon}}$ **do**
3. 对所有的 i, j ($i = 1, 2, \cdots, n$; $j = 1, 2, \cdots, m$), 由式 (2) 计算 α_{ij}.
4. 找到使得式 (4) 最小的旋转矩阵 R^* 和位移向量 $\boldsymbol{t}^*$[11−12].
5. $R^k \leftarrow R^*$, $\boldsymbol{t}^k \leftarrow \boldsymbol{t}^*$, $k \leftarrow k + 1$.
6. $\sigma_p \leftarrow \sigma_p \times sigma_{\text{factor}}$.
7. **end while**.

相较于 ICP 算法, EM-ICP 算法对初值不敏感, 而且不容易陷入局部最优解[13−14]. 对 EM-ICP 算法的详细介绍可参阅文献 [13−16].

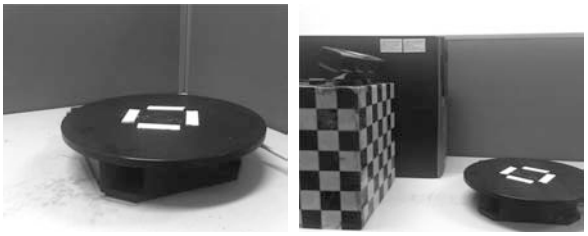
2　系统构架及主要原理和步骤

2.1　系统构架

本文的物体重建系统主要包括三个设备: 一个 Kinect 传感器、一个旋转平台和一台配置有 GPU 的计算机, 如图 1 所示. 旋转平台通过一个步进电机与计算机连接. 步进电机控制旋转平台的旋转速度, 实现平台上物体的旋转运动. 物体旋转的角度 ω 可由下式得到:

$$\omega = \frac{v_s\omega_0 tm}{360} \quad (6)$$

其中, v_s 为步进电机的旋转速度 (step/s), ω_0 为一个常数, tm 为转台运行时间.



(a) 转台 (b) 系统配置

(a) Turntable (b) System configuration

图 1 系统实物图

Fig. 1 System platform

2.2 主要原理和步骤

将欲重建的物体置于转台中央, 便可启动系统, 进行物体重建. 系统首先驱动转台旋转, 然后利用 Kinect 传感器每隔时间 tm_0 采集一次视差图. 由采集的视差图, 系统在线计算得到 RGB 相机坐标系下的三维点云, 并实时与邻近点云进行局部配准. 在旋转与采集结束后, 由获得的三维点云序列 $(X_0, X_1, \cdots, X_{n-1})$ 和局部配准关系, 通过一个全局配准步骤, 计算所有点云与点云 X_0 的坐标变换关系. 最后, 合并点云并过滤, 得到物体的三维模型. 由以上的重建原理, 我们的物体重建方法主要有如下几个步骤.

2.2.1 标定 Kinect 传感器

标定内容包括: 红外摄像机相关参数、彩色摄像机的内参数矩阵和红外摄像机与彩色摄像机的刚体变换. 对于空间中一点 P, 由 Kinect 采集得到了其在红外摄像机坐标系下的视差 d, 可由标定的红外摄像机相关参数计算点 P 在红外摄像机坐标系下的三维坐标. 然后, 根据彩色摄像机与红外摄像机的旋转平移关系和彩色摄像机的内参数矩阵, 计算点 P 在彩色摄像机坐标系下的三维坐标以及其对应的图像坐标. 对 Kinect 的标定仅需进行一次.

2.2.2 标定 Kinect 与转台关系

在固定了 Kinect 传感器和转台的相对位置之后, 要对 Kinect 与转台的关系进行粗标定. 由于三维模型的生成依赖于点云配准的结果, 而点云配准错误将导致重建失败. 因此, 我们用粗标定值来保证配准结果不出现大的误差, 以增强系统的鲁棒性. 粗标定的方法如下:

1) 每隔时间 tm_0 采集一次视差图. 由式 (6) 可知, $\omega_{01} = \omega_{12} = \cdots = \omega_{n-2,n-1}$ ($\omega_{i,i+1}$ 表示相邻两次采集转台旋转的角度). 设相邻两点云的真实变换关系为 $(\bar{R}_{i,i+1}\ \bar{\boldsymbol{t}}_{i,i+1})$ $(i = 0, 1, \cdots, n-1)$, 则有:

$$(\bar{R}_{01}\ \bar{\boldsymbol{t}}_{01}) = (\bar{R}_{12}\ \bar{\boldsymbol{t}}_{12}) = \cdots = (\bar{R}_{n-2,n-1}\ \bar{\boldsymbol{t}}_{n-2,n-1}) \tag{7}$$

但是对于真实物体的重建, 由于转台旋转误差、物体不可能严格放在正中、Kinect 传感器采集误差和 Kinect 参数标定误差等因素, 各变换关系并非严格相等的, 于是有:

$$(\bar{R}_{01}\ \bar{\boldsymbol{t}}_{01}) \approx (\bar{R}_{12}\ \bar{\boldsymbol{t}}_{12}) \approx \cdots \approx (\bar{R}_{n-2,n-1}\ \bar{\boldsymbol{t}}_{n-2,n-1}) \tag{8}$$

2) 由式 (8) 可知, 相邻点云有相近的坐标变换关系. 于是, 我们为相邻点云的坐标变换标定一个初值, 记为 $(R_0\ \boldsymbol{t}_0)$. 该值的用处有两个: 1) 作为点云配准的初始值, 从而提高配准的精度, 加快配准速度; 2) 以该值为标准来检验点云配准的结果, 若配准结果与初始值相差太大, 则表明配准失败. 可将初始值作为相邻点云变换关系的近似值, 保持系统的稳定.

$(R_0\ \boldsymbol{t}_0)$ 的标定结果不需要特别精确, 仅仅得到一个合理的初值就可以了, 精确的解可由后续的配准和优化步骤得到. 因此, 本文粗标定没有采用复杂的标定方法或标定工具. 此外, 在 Kinect 和转台位置关系固定后, 粗标定仅需进行一次.

2.2.3 点云获取、配准及优化

这一步对置于转台中央的物体进行重建, 其主要流程分为在线计算和离线计算两个部分. 在线计算获得物体三维点云并进行局部配准, 离线计算对获得的点云序列进行全局配准并优化. 该步骤的流程图如图 2 所示.

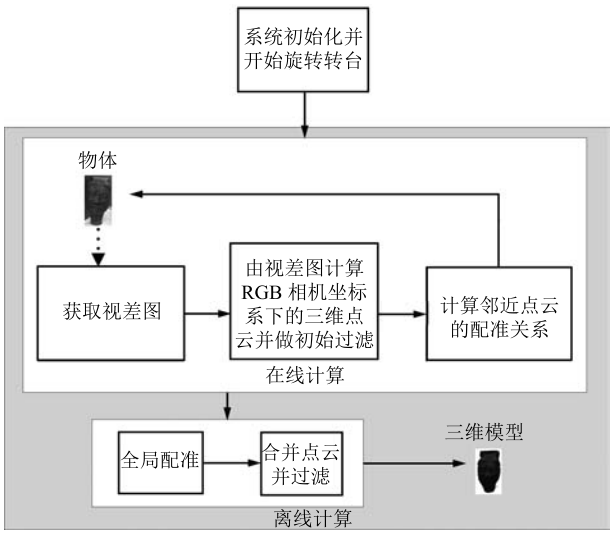


图 2 重建系统流程图

Fig. 2 Flow diagram of the system

我们将在后续各节详细介绍本文提出的物体重建方法: 第 4 节给出标定 Kinect 和标定 Kinect 与

转台的关系 $(R_0\ \boldsymbol{t}_0)$ 的方法; 第 5 节介绍进行点云获取、配准及优化的方法; 第 6 节给出本文方法基于 GPU 的系统实现.

3 标定 Kinect 以及 Kinect 与转台的关系

3.1 标定 Kinect

Kinect 传感器的标定包括三个部分[10, 17]:

1) 标定红外摄像机相关参数, 包括标定固定常数 α, β, b 和红外摄像机的内参数矩阵 K_{IR}.

$$K_{IR}=\begin{bmatrix} f_{uIR} & 0 & u_{0IR} \\ 0 & f_{vIR} & v_{0IR} \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

其中, $(f_{uIR}\ f_{vIR})$ 是红外摄像机的焦距, $(u_{0IR}\ v_{0IR})$ 是红外摄像机的主点坐标. 标定这些常数后, 可由空间点 P 在红外摄像机的原始视差 d (取值范围为 0 ∼2 047), 计算点 P 在红外摄像机坐标系下的坐标 $(x_{IR}\quad y_{IR}\quad z_{IR})^{\mathrm{T}}$.

$$x_{IR}=\frac{b(u_{IR}-u_{0IR})}{d} \tag{10}$$

$$y_{IR}=\frac{bf_{uIR}(v_{IR}-v_{0IR})}{f_{vIR}d} \tag{11}$$

$$z_{IR}=\frac{1}{\alpha(d-\beta)} \tag{12}$$

2) 标定彩色摄像机的内参数矩阵

$$K_{RGB}=\begin{bmatrix} f_{uRGB} & 0 & u_{0RGB} \\ 0 & f_{vRGB} & v_{0RGB} \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

其中, $(f_{uRGB}\ f_{vRGB})$ 是彩色摄像机的焦距, $(u_{0RGB}\ v_{0RGB})$ 是彩色摄像机的主点坐标.

3) 标定红外摄像机与彩色摄像机的刚体变换 $T_{RGB-IR}=(R_{RGB-IR}\quad \boldsymbol{t}_{RGB-IR})$. 本文系统使用的 Kinect 传感器各参数均利用文献 [17] 中提供的标定包进行标定.

3.2 标定 Kinect 与转台的关系

我们采用一种基于 EM-ICP 算法的粗标定方法: 标定 $(R_0\ \boldsymbol{t}_0)$. 首先, 将某一物体置于转台中央, 旋转转台一周, 每隔 tm_0 采集一次物体的点云数据. 采集到的 n 组点云记为 $(X_0, X_1, \cdots, X_{n-1})$. 然后, 以 $(I\ 0)$ 为初值, 由 EM-ICP 算法计算对所有相邻点云变换关系的估计 $(R_{i,i+1}\ \boldsymbol{t}_{i,i+1})$ $(i=0,\ 1,\ \cdots,\ n-2)$. 由 $(R_{i,i+1}\ \boldsymbol{t}_{i,i+1})$, 计算对点云 X_0 和 X_{n-1} 变换关系的估计 $(R^i_{0,n-1}\ \boldsymbol{t}^i_{0,n-1})$:

$$\begin{pmatrix} R^i_{0,n-1} & \boldsymbol{t}^i_{0,n-1} \\ 0 & 1 \end{pmatrix}=\begin{pmatrix} R_{i,i+1} & \boldsymbol{t}_{i,i+1} \\ 0 & 1 \end{pmatrix}^{n-1} \tag{14}$$

由此得到 X_0 和 X'_{n-1} (X'_{n-1} 是 X_{n-1} 经过 $R_{0,n-1}$ 和 $\boldsymbol{t}_{0,n-1}$ 变换后的点云) 的残留误差 ε^i. 使得该残留误差最小的 $(R^i_{0,n-1}\ \boldsymbol{t}^i_{0,n-1})$ 即为标定结果 $(R_0\ \boldsymbol{t}_0)$. 即:

$$(R_0\ \boldsymbol{t}_0)=\arg\min_i \varepsilon\left(R^i_{0,n-1}\ \boldsymbol{t}^i_{0,n-1}\right) \tag{15}$$

整个标定无需特别的标定设备, 我们用一本英语字典进行了粗标定.

4 点云获取、配准及优化

4.1 点云采集与在线局部配准

重建开始后, 系统首先实时获取点云, 并计算相邻 c 组点云的相对关系. 对于空间中一点 P, 由 Kinect 采集得到了其在红外摄像机坐标系下的视差 d, 可由式 (10) ∼ (12) 计算点 P 在红外摄像机坐标系下的三维坐标. 然后, 根据式 (16) 和式 (17), 计算点 P 在彩色摄像机坐标系下的三维坐标 $(x_{RGB}\quad y_{RGB}\quad z_{RGB})^{\mathrm{T}}$ 以及其对应的图像坐标 $(u_{RGB}\quad v_{RGB})^{\mathrm{T}}$.

$$\begin{pmatrix} x_{RGB} \\ y_{RGB} \\ z_{RGB} \end{pmatrix}=R_{RGB-IR}\begin{pmatrix} x_{IR} \\ y_{IR} \\ z_{IR} \end{pmatrix}+t_{RGB-IR} \tag{16}$$

$$\frac{1}{z_{RGB}}\begin{pmatrix} u_{RGB} \\ v_{RGB} \\ 1 \end{pmatrix}=K_{RGB}\begin{pmatrix} x_{RGB} \\ y_{RGB} \\ z_{RGB} \end{pmatrix} \tag{17}$$

本文以 $(R_0\ \boldsymbol{t}_0)$ 为初值, 利用 ICP 算法对相邻点云进行了配准. 对于配准后结果, 用 $(R_0\ \boldsymbol{t}_0)$ 为参考依据, 进一步去除了错误的结果. 具体方法见算法 2.

算法 2. 点云采集与在线局部配准

1. **while** 转台旋转 **do**
2. 由 Kinect 传感器获取当前视差图.
3. 计算 RGB 相机坐标系下的三维点云 X_i.
4. **for** 点云 X_i 的前 c 组点云 X_j ($j=i-c$, $i-c+1, \cdots, i-1$; $j\geq 0$) **do**
5. 计算点云之间变换关系的初值 $\hat{R}_{ji}$ 和 $\hat{t}_{ji}$:
$$\begin{pmatrix} \hat{R}_{ji} & \hat{\boldsymbol{t}}_{ji} \\ 0 & 1 \end{pmatrix}=\begin{pmatrix} R_0 & \boldsymbol{t}_0 \\ 0 & 1 \end{pmatrix}^{i-j}$$
6. 以 $\hat{R}_{ji}$ 和 $\hat{\boldsymbol{t}}_{ji}$ 为初值, 由 ICP 算法得到 X_j 与 X_i 的相对变换关系 R'_{ji} 和 $\boldsymbol{t}'_{ji}$.
7. 计算 $(\hat{R}_{ji}\ \hat{\boldsymbol{t}_{ji}})$ 与 $(R'_{ji}\ \boldsymbol{t}'_{ji})$ 的均方误差 ε.

8. **if** $\varepsilon < \varepsilon_{\inf}$ **do**
9. $R_{ji} \leftarrow R'_{ji}, \boldsymbol{t}_{ji} \leftarrow \boldsymbol{t}'_{ji}$
10. **else**
11. $R_{ji} \leftarrow \hat{R}_{ji}, \boldsymbol{t}_{ji} \leftarrow \hat{\boldsymbol{t}}_{ji}$
12. **end if**.
13. **end for**.
14. $i \leftarrow i+1$.
15. **end while**.

4.2 计算初始点云与最后点云的变换关系

通过点云采集与在线计算, 得到了点云序列 $(X_0, X_1, \cdots, X_{n-1})$ 和邻近点云的配准关系. 此外, 由于点云序列是将转台旋转一周采集得到的, 则初始采集的 c 个点云 X_i $(i = 0, 1, \cdots, c-1)$ 与最后采集的 c 个点云 X_j $(j = n-c, n-c+1, \cdots, n-1)$ 也可视为邻近的点云, 需要计算它们的局部配准关系作为全局配准的输入. 由于局部配准的累积误差, 算法 2 第 5 步的计算结果已经不再适合作为本步局部配准的初值, 因此, 我们首先以 $(I\ 0)$ 为初值, 由 EM-ICP 算法得到对点云 X_0 和 X_{n-1} 相对变换关系的估计 $(R_{0,n-1}\ \boldsymbol{t}_{0,n-1})$. 然后, 计算对点云 X_j $(j = n-c, n-c+1, \cdots, n-1)$ 与 X_i $(i = 0, 1, \cdots, c-1)$ 变换关系的估计 $(\hat{R}_{ji}\ \hat{\boldsymbol{t}}_{ji})$:

$$\begin{pmatrix} \hat{R}_{ji} & \hat{\boldsymbol{t}}_{ji} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_0^{\mathrm{T}} & -R_0^{\mathrm{T}}\boldsymbol{t}_0 \\ 0 & 1 \end{pmatrix}^{j} \times \begin{pmatrix} R_{0,n-1}^{\mathrm{T}} & -R_{0,n-1}^{\mathrm{T}}\boldsymbol{t}_{0,n-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_0^{\mathrm{T}} & -R_0^{\mathrm{T}}\boldsymbol{t}_0 \\ 0 & 1 \end{pmatrix}^{n-1-i} \tag{18}$$

最后, 以 $(\hat{R}_{ji}\ \hat{\boldsymbol{t}}_{ji})$ 为初值, 由 ICP 算法获得邻近点云之间的局部配准结果.

4.3 全局配准

由上两步获得邻近点云的配准关系后, 需要进一步进行全局配准, 来得到点云 X_i 与点云 X_0 的相对变换关系 $\tilde{R}_{0i}$ 和 $\tilde{\boldsymbol{t}}_{0i}$ $(i = 1, 2, \cdots, n-1)$. 首先, 通过 SVD 分解法[18] 计算旋转矩阵 $\tilde{R}_{0i}$. 设点云 X_i 和 X_j 的真实变换关系为 $(\bar{R}_{ij}\ \bar{\boldsymbol{t}}_{ij})$ $(i, j = 0, 1, \cdots, n-1;\ i \neq j)$. 构造矩阵 $\bar{A}$ 和矩阵 $\bar{R}$.

$$\bar{A} = \begin{bmatrix} \bar{R}_{01} & -I & 0 & \cdots & 0 & 0 \\ \bar{R}_{02} & 0 & -I & \cdots & 0 & 0 \\ \bar{R}_{0,n-1} & 0 & 0 & \cdots & 0 & -I \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \bar{R}_{n-2,n-1} & -I \end{bmatrix} \tag{19}$$

$$\bar{R} = \begin{bmatrix} I \\ \bar{R}_{01} \\ \bar{R}_{02} \\ \vdots \\ \bar{R}_{0,n-1} \end{bmatrix} \tag{20}$$

其中, 矩阵 $\bar{A}$ 是一个 $rel \times n$ 矩阵, rel 表示局部配准获得的临近点云变换关系的个数, 而 n 表示采集到的三维点云的个数. 而矩阵 $\bar{R}$ 是一个 $n \times 3$ 矩阵. 则有:

$$\bar{A}\bar{R} = 0 \tag{21}$$

由前两步的局部配准结果可得到对矩阵 $\bar{A}$ 的估计 A.

$$A = \begin{bmatrix} R_{01} & -I & 0 & \cdots & 0 & 0 \\ R_{02} & 0 & -I & \cdots & 0 & 0 \\ R_{0,n-1} & 0 & 0 & \cdots & 0 & -I \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & R_{n-2,n-1} & -I \end{bmatrix} \tag{22}$$

使得 $\|AR\|$ ($\|\cdot\|$ 表示矩阵范数) 最小的矩阵 R^* 即为对 $\bar{R}$ 的估计, 即:

$$R^* = \arg\min_{R} \|AR\| \tag{23}$$

对矩阵 A 进行 SVD 分解, 有:

$$A = U\Sigma V^{\mathrm{T}} \tag{24}$$

其中, 矩阵 V 的最后三列可表示为矩阵 R^*:

$$R^* = \begin{pmatrix} R_{00}^* \\ R_{01}^* \\ \vdots \\ R_{0,n-1}^* \end{pmatrix} \tag{25}$$

则 $\tilde{R}_{0i}$ $(i = 1, 2, \cdots, n-1)$ 可由下式得到:

$$\tilde{R}_{0i} = R_{0i}^* R_{00}^{*\mathrm{T}} \tag{26}$$

进而, 由得到的旋转矩阵 $\tilde{R}_{0i}$ 来计算位移量 $\tilde{\boldsymbol{t}}_{0i}$ $(i = 1, 2, \cdots, n-1)$. 构造矩阵 $\bar{B}$ 和向量 $\bar{\boldsymbol{a}}$.

$$\bar{B} = \begin{bmatrix} -\bar{R}_{01} & I & 0 & \cdots & 0 & 0 \\ -\bar{R}_{02} & 0 & I & \cdots & 0 & 0 \\ -\bar{R}_{0,n-1} & 0 & 0 & \cdots & 0 & I \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\bar{R}_{n-2,n-1} & I \end{bmatrix} \tag{27}$$

$$\bar{\boldsymbol{a}} = \begin{pmatrix} 0 \\ \bar{\boldsymbol{t}}_{01} \\ \bar{\boldsymbol{t}}_{02} \\ \vdots \\ \bar{\boldsymbol{t}}_{0,n-1} \end{pmatrix} \tag{28}$$

同矩阵 $\bar{A}$ 一样, $\bar{B}$ 也是一个 $rel \times n$ 矩阵. 对于点云 X_0 坐标系下的任一点 $\bar{\boldsymbol{x}}_0$, 其在点云 $X_1, X_2, \cdots, X_{n-1}$ 的坐标系下对应的坐标向量分别为 $\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \cdots, \bar{\boldsymbol{x}}_{n-1}$. 构造向量 $\bar{\boldsymbol{b}}$.

$$\bar{\boldsymbol{b}} = \begin{pmatrix} \bar{\boldsymbol{x}}_0 \\ \bar{\boldsymbol{x}}_1 \\ \vdots \\ \bar{\boldsymbol{x}}_{n-1} \end{pmatrix} \tag{29}$$

则有:

$$\bar{B}\bar{\boldsymbol{b}} - \bar{\boldsymbol{a}} = 0 \tag{30}$$

我们首先构造矩阵 $\bar{B}$ 的估计 B. $\tilde{R}_{0i}$ 可得到任意两点云的相对旋转关系 $\tilde{R}_{ij}$:

$$\tilde{R}_{ij} = \tilde{R}_{0j}\tilde{R}_{0i}^{\mathrm{T}} \tag{31}$$

对于局部配准得到的每一个矩阵 R_{ij}, 由式 (31) 计算其相对应的矩阵 $\tilde{R}_{ij}$, 由此得到矩阵 B.

$$B = \begin{bmatrix} -\tilde{R}_{01} & I & 0 & \cdots & 0 & 0 \\ -\tilde{R}_{02} & 0 & I & \cdots & 0 & 0 \\ -\tilde{R}_{0,n-1} & 0 & 0 & \cdots & 0 & I \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\tilde{R}_{n-2,n-1} & I \end{bmatrix} \tag{32}$$

向量 $\bar{\boldsymbol{a}}$ 的估计 $\boldsymbol{a}$ 为

$$\boldsymbol{a} = \begin{pmatrix} 0 \\ \boldsymbol{t}_{01} \\ \boldsymbol{t}_{02} \\ \vdots \\ \boldsymbol{t}_{0,n-1} \end{pmatrix} \tag{33}$$

利用广义逆矩阵[19] 可求得向量 $\bar{\boldsymbol{b}}$ 的估计 $\boldsymbol{b}$ 为

$$\boldsymbol{b} = \left(B^{\mathrm{T}}B\right)^{-1}B^{\mathrm{T}}a = \begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{n-1} \end{pmatrix} \tag{34}$$

则点云 X_0 和 X_i $(i = 1, 2, \cdots, n-1)$ 相对位移可表示为

$$\tilde{\boldsymbol{t}}_{0i} = \boldsymbol{x}_i - \tilde{R}_{0i}\boldsymbol{x}_0 \tag{35}$$

4.4 点云合并与过滤

利用上一步得到的点云相对变换关系 $\tilde{R}_{0i}$ 和 $\tilde{\boldsymbol{t}}_{0i}$ $(i = 1, 2, \cdots, n-1)$, 将点云 X_i $(i = 1, 2, \cdots, n-1)$ 都转换到点云 X_0 的坐标系下, 并合并所有点云. 此外, 由于 Kinect 传感器采集误差和 Kinect 参数标定误差等因素, 每组点云的边缘会有少量颜色为实验场景背景色的错误点, 需要进行剔除. 本文对合并点云的所有三维点进行了遍历. 若某三维点的 RGB 颜色与背景颜色相近, 则认定该点为错误点. 系统最后在重建模型中过滤了所有此类错误点.

5 GPU 实现

本文方法主要有如下几个步骤: 标定 Kinect 与转台的关系; 由视差图获得三维点云; 计算邻近点云坐标系的相对变换关系; 全局配准; 点云合并与过滤. 下面分别描述各步的 GPU 并行实现. 实现使用 CUDA[20] 作为 GPU 编程模型.

5.1 标定 Kinect 与转台的关系

标定 Kinect 与转台的关系主要有两个子步骤在 GPU 上实现: 1) EM-ICP 算法; 2) 计算 X_0 和 X_{n-1} 之间的最小残留误差. EM-ICP 算法的 CUDA 实现主要分成 5 步: 1) 计算 d_{ij}; 2) 标准化矩阵 $C = (\alpha_{ij})_{n\times m}$; 3) 计算 $\boldsymbol{x}'_i$; 4) 计算点云中心; 5) 得到旋转矩阵 R^* 和位移向量 $\boldsymbol{t}^*$. 实现包括 5 个 Kernel 函数和 4 个 CUBLAS 库函数[21]: Sgemv, Saxpy, Sgemm 和 Sscal. 具体实现细节可参考文献 [14].

计算 X_0 和 X_{n-1} 之间的最小残留误差如算法 3 所示. 算法 3 的第 4 步、第 5 步和第 6 步分别由一个 Kernel 函数实现. 对于计算点云 X'_{n-1}, 每个 CUDA-thread 并行地计算一个三维点的坐标变换. 对于第 5 步, 每个 CUDA-thread 分别搜索 GPU-KD 树, 获得 x'_i 的最近邻点及与最近邻点的距离. 最后一个 Kernel 函数计算所有距离的平均值.

算法 3. 计算 X_0 和 X_{n-1} 的最小残留误差

1. 由当前点云 X_0 建立 GPU-KD 树, 并将其下载到显存.
2. 将点云 X_0 和点云 X_{n-1} 下载到显存.
3. **for** 每个 $R_{i,i+1}$ 和 $\boldsymbol{t}_{i,i+1}$ **do**
4. 计算点云 X'_{n-1} (X_{n-1} 经 $R_{i,i+1}$ 和 $\boldsymbol{t}_{i,i+1}$ 变换后的点云).
5. 搜索 GPU-KD 树, 找到集合 X'_{n-1} 中的每个点的最近邻点.

6. 计算残留误差 ε.
7. **if** $\varepsilon < \varepsilon_{\min}$ **do**
8. $R_0 \leftarrow R_{i,i+1}$, $\boldsymbol{t}_0 \leftarrow \boldsymbol{t}_{i,i+1}$, $\varepsilon_{\min} \leftarrow \varepsilon$
9. **end if**.
10. **end for**.

5.2 由视差图获得三维点云

由 Kinect 传感器获得第 i 个视差图 D_i 后, 在 GPU 端计算各个点在 RGB 相机坐标系下的三维点坐标及其对应的图像坐标. CUDA 实现包括一个 Kernel 函数. 每个 CUDA-thread 并行地计算一个点的坐标变换.

5.3 计算邻近点云的坐标系相对变换关系

在线计算邻近点云的相对变换关系实现方法如算法 4 所示. CUDA 实现包括两个 Kernel 函数和一个 CUBLAS 库函数: Sgemm. 第一个 Kernel 函数的实现与第 5.1 节相同. 计算 S 使用函数 Sgemm, 并由另一个 Kernel 函数计算 X_i^* 的中心. 最后, 将结果拷贝回 CPU 端, 计算得到 R^k 和 t^k. 对于每组点云 X_i, GPU-KD 树仅需建立一次.

算法 4. 计算邻近 c 组点云的相对变换关系

1. 由当前点云 X_i 建立 GPU-KD 树, 并将其下载到显存.
2. **for** 每组点云 X_j $(i-c \le j < i)$ **do**
3. 将点云 X_j 下载到显存.
4. 计算点云之间变换关系的初值 $\hat{R}_{ji}$ 和 $\hat{\boldsymbol{t}}_{ji}$
$$\begin{pmatrix} \hat{R}_{ji} & \hat{\boldsymbol{t}}_{ji} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_0 & \boldsymbol{t}_0 \\ 0 & 1 \end{pmatrix}_{i-j}$$
5. $R^0 \leftarrow \hat{R}_{ji}$, $\boldsymbol{t}^0 \leftarrow \hat{\boldsymbol{t}}_{ji}$.
6. **for** $k = 1, \cdots, maxIterations$ **do**
7. 搜索 GPU-KD 树, 找到集合 X_j 中的每个点的最近邻点.
8. 构造相关集合 $X_i^* = \{\boldsymbol{x}_{i1*}, \boldsymbol{x}_{i2*}, \cdots, \boldsymbol{x}_{im*}\}$.
9. 找到使得 X_j 和 X_i^* 残留误差最小的 R^* 和 $\boldsymbol{t}^*$.
10. $R^k \leftarrow R^*$, $\boldsymbol{t}^k \leftarrow \boldsymbol{t}^*$.
11. **end for**.
12. **end for**.

此外, 计算初始的 c 组点云与最后 c 组点云的变换关系主要使用了 EM-ICP 算法和 ICP 算法, 实现方法与以上的实现方法类似, 不再赘述.

5.4 全局配准

计算矩阵 A 的 SVD 分解使用了 CULA 库函数[22]: Sgesvd. 其他的矩阵乘法与矩阵向量乘法运算, 均使用 CUBLAS 库的相关函数实现.

5.5 点云合并与过滤

点云的坐标变换及过滤由一个 Kernel 函数实现. 每个 CUDA-thread 并行地计算一个三维点的坐标变换, 并根据颜色的 RGB 值判断其是否为错误点, 若是则舍弃.

6 实验结果及分析

为了验证本章方法的效率, 我们在 7 个物体上进行了实验, 包括: 维纳斯头像、兵马俑头像、青叶碧玉 (常见室内植物)、亮丝草 (常见室内植物)、水壶、茶罐和蛐蛐角雕 (工艺品). 图 3 依次列出了这些物体的示例图片 (最左列) 和重建结果, 物体的原数据见表 1. 其中, 维纳斯头像、兵马俑头像和蛐蛐角雕的尺寸分别表示其长 × 宽 × 高, 而青叶碧玉、亮丝草、水壶和茶罐的尺寸分别表示其底座直径 × 高. 此外, 实验中, 系统设定为每 1 秒钟采集一次视差图, 对邻近的 3 组点云进行匹配, 即取 $tm_0 = 1\,\mathrm{s}$; $c = 3$.

图 3 物体图片及重建结果

Fig. 3 Object images and reconstruction results

实验用物体可分为 4 类: 维纳斯头像和兵马俑头像几乎没有遮挡, 且在各个视角下, 其表面形状变化较大; 水壶和茶罐几乎没有遮挡, 但各个视角下的表面形状几乎相同; 蛐蛐角雕有轻微的遮挡; 而青叶

碧玉和亮丝草有大量遮挡. 各个物体的重建结果如图 3 所示. 可见, 即使对于有遮挡物体和各个视角下表面形状非常相似的物体, 我们的方法仍然能够重建出视觉效果良好的三维模型. 此外, 我们将物体直立或平躺地置于转台中央, 分别重建出物体的三维模型, 并将得到的模型进行融合. 结果如图 4 所示. 其中, 各行的重建结果依次为维纳斯头像、水壶、茶罐和青叶碧玉; 左边两列分别是将物体直立和平躺所得的重建模型示例图片, 右边三列是融合后模型的各个侧面. 可见, 将物体以各种姿态置于转台中央, 重建出的三维模型均不完整. 然而, 将这些模型进行融合, 便能得到较为完整的物体三维模型.

表 1　物体元数据及重建结果

Table 1　Some meta-information and reconstruction results

物体	尺寸 (mm)	重建三维点个数
维纳斯头像	195 × 164 × 332	1 169 981
兵马俑头像	82 × 121 × 213	367 036
青叶碧玉	81 × 201	220 305
亮丝草	162 × 310	1 114 669
水壶	163 × 280	707 175
茶罐	98 × 187	319 312
蚰蜒角雕	90 × 120 × 170	49 308

图 4　模型融合结果

Fig. 4　Fusion models

为验证三维重建效果, 我们手工选取了若干点, 并测量了点与点之间的距离. 选取的测量值如图 5 所示, 包括: 1) 兵马俑头像发髻宽度; 2) 兵马俑头像发髻长度; 3) 亮丝草花盆上沿直径; 4) 青叶碧玉花盆下沿直径; 5) 水壶把手长度; 6) 水壶把手宽度; 7) 蚰蜒角雕下边沿长度; 8) 维纳斯头像左眼角到左鼻子角的距离; 9) 茶罐顶到底面花纹的距离. 我们将系统重建结果与手工测量值进行了比较, 结果见表 2. 可见, 测量数据的所有相对误差都小于 2%. 这表明该系统具有较好的三维重建效果, 基本能满足一般三维模型应用的需要. 此外, 为了验证重建结果是否闭合, 我们还统计了点云 X_0 和 X'_{n-1} (X_{n-1} 经过 $\tilde{R}_{0,n-1}$ 和 $\tilde{\boldsymbol{t}}_{0,n-1}$ 旋转变换) 的平均残留误差, 如表 3 所示. 由图 3 和表 3 可知, 重建结果几乎没有漂移, 且有较好的视觉效果.

图 5　测量结果

Fig. 5　Measurements

表 2　测量结果

Table 2　Results of measurements

编号	真实值 (mm)	测量值 (mm)	绝对误差 (mm)	相对误差 (%)
1	87.3	86.1	1.2	1.37
2	33.2	32.8	0.4	1.20
3	174.1	171.9	2.2	1.26
4	78.7	77.9	0.8	1.01
5	33.1	32.5	0.6	1.81
6	24.7	24.3	0.4	1.62
7	84.2	82.9	1.3	1.54
8	50.1	49.4	0.7	1.49
9	163.1	161.6	1.5	0.92

我们进行了多次实验, 由于进行了初始标定, 物体重建没有出现失败的情况. 一次初始标定的时间大约需要 2 分钟, 而转台与 Kinect 固定之后, 初始标定仅需进行一次. 我们统计了各次物体重建的系统运行时间, 结果如表 4 所示. 受使用的转台最大旋转速度的限制 (旋转一周需要 46 秒), 重建一个物体

的时间大约是 50 秒左右.

表 3 平均残留误差

Table 3 Residual-errors

物体	残留误差 (mm)	X_0 三维点个数
维纳斯头像	0.217	20 166
兵马俑头像	0.187	6 288
青叶碧玉	0.494	4 252
亮丝草	0.378	21 064
水壶	0.345	11 662
茶罐	0.221	7 145
蚰蜒角雕	0.198	578

表 4 物体重建时间 (s)

Table 4 Time consuming of objects reconstruction (s)

物体	初始化 Kinect	获取点云和在线计算	离线计算	总体时间
维纳斯头像	1.5	46.0	3.3	50.8
兵马俑头像	1.5	46.0	2.5	50.0
青叶碧玉	1.5	46.0	2.2	49.7
亮丝草	1.5	46.0	1.9	49.4
水壶	1.5	46.0	3.1	50.6
茶罐	1.5	46.0	2.3	49.8
蚰蜒角雕	1.5	46.0	1.5	49.0

7 总结

本文报道了一种基于 Kinect 和转台的快速物体重建方法. 该方法可对物体全自动实时重建, 在出现点云配准错误时仍然能够得到较理想的重建模型. 我们还给出了该方法基于 GPU 的原型系统, 该系统能够在 50 s 左右重建出物体的稠密三维点云. 几类物体的重建实验结果表明, 本文方法方便实用. 重建的三维模型几乎没有漂移, 基本能满足一般应用的需要. 对于有遮挡物体的重建, 本文方法也取得了较好的重建效果.

本文方法的使用也存在一些局限, 主要有: 1) 转台的转速限制了重建的效率. 若有旋转速度更快的转台, 本文方法的重建速度将会进一步的提高. 2) Kinect 传感器的最小有效距离限制了重建物体的大小. Kinect 获取视差图的最小有效距离大约是 0.5 m, 对于体积较小的物体, 将会无法获取到稠密的点云. 下一步工作包括将本文方法的结果与激光测距仪的重建结果进行比较, 以及探索如何对物体各姿态下的重建模型进行更准确的融合. 另外需要指出的一个问题是, 对于诸如圆柱、圆台等物体, 由于在物体的各个视角下的表面形状非常相似, 本文局部配准部分的失败率相对其他物体要高. 但由于采用了对转台和 Kinect 的粗标定信息以及最后的整体优化, 最终的重建对此类物体也基本能得到比较好的结果.

References

1 Engelhard N, Endres F, Hess J, Sturm J, Burgard W. Real-time 3D visual SLAM with a hand-held RGB-D camera. In: Proceedings of the 2011 RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum. Västeras, Sweden: Robotdalen, 2011

2 Henry P, Krainin M, Herbst E, Ren X, Fox D. RGB-D mapping: using depth cameras for dense 3D modeling of indoor environments. In: Proceedings of the 12th International Symposium on Experimental Robotics. Delhi, India: IEEE, 2010

3 Du H, Henry P, Ren X F, Cheng M, Goldman D B, Seitz S M, Fox D. Interactive 3D modeling of indoor environments with a consumer depth camera. In: Proceedings of the 13th International Conference on Ubiquitous Computing. Beijing, China: IEEE, 2011. 75−84

4 Izadi S, Newcombe R A, Kim D, Hilliges O, Molyneaux D, Hodges S, Kohli P, Davison A, Fitzgibbon A. KinectFusion: real-time dynamic 3D surface reconstruction and interaction. In: Proceedings of the 2011 International Conference on Computer Graphics and Interactive Techniques. Vancouver, Canada: ACM, 2011

5 Izadi S, Kim D, Hilliges O, Molyneaux D, Newcombe R, Kohli P, Shotton J, Hodges S, Freeman D, Davison A, Fitzgibbon A. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: Proceedings of the 2011 Annual ACM Symposium on User Interface Software and Technology. Santa Barbara, CA: ACM, 2011. 559−568

6 Tong J, Zhou J, Liu L G, Pan Z G, Yan H. Scanning 3D full human bodies using Kinects. *IEEE Transactions on Visualization and Computer Graphics*, 2012, **18**(4): 643−650

7 Bay H, Ess A, Tuytelaars T, van Gool L. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 2008, **110**(3): 346−359

8 Besl P J, McKay H D. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992, **14**(2): 239−256

9 Konolige K, Mihelich P. Technical description of Kinect calibration [Online], available: http://www.ros.org/wiki/kinect_calibration/technical, November 3, 2011

10 Gu Zhao-Peng. A Study on Monocular Simultaneous Localization and Mapping [Ph. D. dissertation]. Institute of Automation, Chinese Academy of Sciences, China, 2011
(顾照鹏. 单目视觉同步定位与地图创建技术研究 [博士学位论文], 中国科学院自动化研究所, 中国, 2011)

11 Liu Y H. Automatic registration of overlapping 3D point clouds using closest points. *Image and Vision Computing*, 2006, **24**(7): 762−781

12 Horn B K P. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 1987, **4**(4): 629−642

13 Granger S, Pennec X. Multi-scale EM-ICP: a fast and robust approach for surface registration. In: Proceedings of the 7th European Conference on Computer Vision. Copenhagen, Denmark: Springer-Verlag, 2002. 418−432

14 Tamaki T, Abe M, Raytchev B, Kaneda K. Softassign and EM-ICP on GPU. In: Proceedings of the 2010 1st International Conference on Networking and Computing. Washington DC, USA: IEEE, 2010. 179−183

15 Dewaele G, Devernay F, Horaud R. Hand motion from 3D point trajectories and a smooth surface model. In: Proceedings of European Conference on Computer Vision. Prague, Czech: Springer, 2004. 495−507

16 Tamaki T. Pose estimation and rotation matrices. IEICE Technical Report SIS2009-23, 2009, **109**(203): 59−64

17 Herrera C D, Kannala J, HeikkilAa J. Accurate and practical calibration of a depth and color camera pair. In: Proceedings of the 14th international conference on Computer analysis of images and patterns. Seville, Spain: Springer, 2011. 437−445

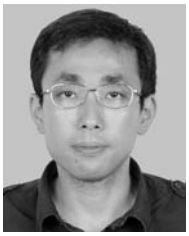
18 Martinec D, Pajdla T. Robust rotation and translation estimation in Multiview reconstruction. In: Proceedings of 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Minneapolis, Minnesota, USA: IEEE, 2007. 1−8

19 Hartley R, Zisserman A. *Multiple View Geometry in Computer Vision* (Second edition). London: Cambridge University Press, 2004

20 CUDA 3.0. CUDA Programming Guide 3.0 [Online], available: http://developer.download.nvidia.com/compute/cuda/, December 5, 2011

21 CUBLAS library 2.0. CUBLAS Library User′s Guide [Online], available: http://developer.download.nvidia.com/compute/cuda/, December 5, 2011

22 CULA library. CULA Library User′s Guide [Online], available: http://www.culatools.com/, December 5, 2011



刘 鑫 中国科学院自动化研究所博士研究生. 2004 年获北京师范大学信息学院计算机系学士学位. 主要研究方向为三维重建和 GPU 通用计算. 本文通信作者. E-mail: liux_skylark@tom.com
(**LIU Xin** Ph. D. candidate at the Institute of Automation, Chinese Academy of Sciences. He received his bachelor degree from Beijing Normal University in 2004. His research interest covers 3D reconstruction and GPU computing. Corresponding author of this paper.)

许华荣 厦门理工学院副教授. 主要研究方向为计算机视觉和数字图像处理.
E-mail: huarong_xu@sina.com
(**XU Hua-Rong** Associate professor in the Department of Computer Science and Technology, Xiamen University of Technology. His research interest covers computer vision and image processing.)

胡占义 中国科学院自动化研究所研究员. 主要研究方向为摄像机标定与三位重建, 视觉机器人导航, 基于图像的建模与绘制.
E-mail: huzy@nlpr.ia.ac.cn
(**HU Zhan-Yi** Professor at the Institute of Automation, Chinese Academy of Sciences. His research interest covers robot vision, which include camera calibration and 3D reconstruction, vision guided robot navigation, image based modeling and rendering.)