文章编号: 1001-0920 (2013) 09-1403-06

# 大场景三维重建中多核并行捆集调整算法

佟国峰, 蒋昭炎, 叶 柠, 徐心和

(东北大学 信息科学与工程学院，沈阳 110819)

**摘 要:** 将三维重建中捆集调整算法用于优化重建结果, 是非常关键的步骤, 然而传统单核串行算法耗时量大不太适合大场景重建. 对此, 首先对捆集调整算法本身进行了改进; 然后在此基础上提出了多核并行捆集调整算法并采用图像处理器 (GPU) 实现该算法. 实验表明, 所提出的多核并行捆集调整算法提高了算法优化参数的精度和处理速度.

**关键词:** 三维重建；捆集调整；并行运算；图像处理器

**中图分类号:** TP301.6 **文献标志码:** A

# Multi-core bundle adjustment algorithm using parallel processing in large-scale 3D scene reconstruction

*TONG Guo-feng*, *JIANG Zhao-yan*, *YE Ning*, *XU Xin-he*

(College of Information Science and Engineering，Northeastern University，Shenyang 110819，China．Correspondent: TONG Guo-feng，E-mail：tongguofeng@ise.neu.edu.cn)

**Abstract:** The bundle adjustment algorithm is used to optimize the final reconstruction results of 3D scene reconstruction, which is a very critical step in 3D scene reconstruction. However, the serial bundle adjustment algorithm is large time-consuming and not suitable for large-scale scene reconstruction. Therefore, the bundle adjustment algorithm is improved. Then the multi-core parallel bundle adjustment algorithm is proposed on the basis of the improved algorithm, and the graphic processing unit(GPU) is used to realize the algorithm. The experiment shows that the accuracy and the processing speed of the algorithm are greatly improved by using the proposed method.

**Key words:** 3D scene reconstruction；bundle adjustment；parallel computation；graphic processing unit(GPU)

## 0 引 言

基于图像的三维重建, 一直是计算机视觉领域研究的热点问题, 而针对大场景的三维重建, 其较大的科研和实用价值日益得以体现, 重建出来的三维模型有着广泛的应用, 例如可用于文物保护, 数字城市的建设等. 大场景的三维重建具有如下特点: 1) 至少对单个建筑物级别的场景进行重建; 2) 重建使用图像多, 至少需上百张图像, 有时甚至达到上万张; 3) 对算法处理速度要求高. 国内外学者对此课题已有了深入的研究, 1996 年 Debevec 等[1]实现了一个半自动的 Facade 方法, 指定几张图像中的关键边缘, 然后自动重建三维结构. 2009 年, Sinha 等[2]实现了一个相似的系统, 该系统利用消影点来帮助用户建立 3D 模型. 1998 年, 麻省理工学院 Teller[3-4]的 city scanning project 是最早的基于地面的全自动城市重建工作之一, 相机的参数是使用 SfM 和 GPS 数据来自动估计的. 2004 年, Shindler等[5-7]在 4D cities 项目中同样恢复了相机的位姿和稀疏的边缘结构, 提出了边缘匹配和基于线的 SfM 方法. 2009 年, 华盛顿大学的 GRAIL (graphics and imaging laboratory) 实验室成功完成了著名的 “Building rome in a day”工程[8], 实现了对罗马城的稀疏点云重建. 但存在两个方面的问题: 1) 半自动三维重建系统虽然鲁棒性较好, 但并不适合大场景重建; 2) 全自动重建方法的过程复杂, 空间和时间复杂度都很高, 也不太适合大场景重建.

在全自动三维重建算法中, 捆集调整算法主要用于优化重建最终结果, 是不可缺少的步骤, 但单核串行捆集调整算法至少占据了重建全过程 75% 的时间, 且随着图像数量的增加, 其耗时会呈指数增加. 为提高三维重建算法的处理速度, 使其适合大场景三

**收稿日期:** 2012-08-02；**修回日期:** 2013-02-26.

**基金项目:** 国家自然科学基金项目(61175031)；教育部基础科研基金项目(N110204004).

**作者简介:** 佟国峰(1972−), 男, 副教授, 博士, 从事计算机视觉等研究；蒋昭炎(1988−), 男, 硕士, 从事并行算法的研究.

维重建, 本文对三维重建中捆集调整算法进行了改进. 首先对捆集调整算法本身作了两方面改进, 一方面针对输入优化参数初值具有不同的不确定度, 提出带权值的捆集调整算法, 以提高捆集调整优化参数的精度, 另一方面采用预先共轭梯度法解决捆集调整算法中的求解方程问题, 以简化算法复杂度; 其次, 基于改进后的捆集调整算法, 提出多核并行捆集调整算法, 提高了算法处理速度; 最后, 鉴于采用SIMT架构的GPU的超强并行计算能力, 利用GPU去实现本文提出的多核并行捆集调整算法.

## 1 一种改进的捆集调整算法

### 1.1 捆集调整原理

捆集调整是同时对所有参数进行优化的方法[9]. 在计算机视觉领域, 捆集调整主要用于优化重建结果以提供整体最优的三维结构和摄像机参数. 在三维重建中整体最优是指所有3D点出现的每幅视图中重投影点和被测量的图像点之间的重投影误差和最小[10]. 设$x$是所有摄像机的参数和重构3D点坐标组成的向量, $f(x)=[f_1(x),f_2(x),\cdots,f_k(x)]$是3D重投影误差向量, 则捆集调整优化问题可以简化为求解如下所示的非线性最小二乘问题:

$$x^*=\arg\min_x\sum_{i=1}^{k}\|f_i(x)\|^2. \tag{1}$$

本文利用Levenberg-Marquart (L-M)算法[11]来解决上述非线性最小二乘问题. 具体算法描述如下: 设$J(x)$是$f(x)$的雅可比矩阵, 每次LM迭代解决如下线性最小二乘问题:

$$\delta_k^*=\arg\min_{\delta_k}\|J(x_k)\delta_k+f(x_k)\|^2+\lambda_k\|D(x_k)\delta_k\|^2. \tag{2}$$

其中: $x_k$是第$k$次迭代时所取参数向量, $D(x_k)$是非负对角矩阵, 这里取矩阵$J(x_k)^IJ(x_k)$对角线元素的平方根, $\lambda_k(\lambda_k>0)$是控制每次迭代步长的参数, 其值由雅可比矩阵$J(x_k)$接近$f(x_k)$的程度决定, 接近程度越高其值也越大. 如果第$k$次迭代有$\|f(x_k+\delta_k^*)\|<\|f(x_k)\|$, 则$x_{k+1}=x_k+\delta_k^*$, 进入下次迭代过程, 一直迭代到$\|\delta^*\|<\varepsilon$为止.

综上所述, LM算法迭代过程关键的问题是求解$\delta$使下式取得最小值:

$$\|J(x)\delta+f(x)\|^2+\lambda\|D(x)\delta\|^2, \tag{3}$$

所以求解式(2)等价于求如下关于$\delta$方程的解:

$$(J^{\mathrm{T}}J+\lambda D^{\mathrm{T}}D)\delta=-J^{\mathrm{T}}f, \tag{4}$$

其中$H_\lambda=J^{\mathrm{T}}J+\lambda D^{\mathrm{T}}D$是广义海赛矩阵.

在三维重建捆集调整中, 参数矩阵$x=[x_c,x_p]$由摄像机参数$x_c$向量和3D点参数$x_p$向量组成, 类似对于$D,\delta$和$J$, 用下标$c$和$p$分别代表摄像机部分和3D点部分.

设$U=J_c^{\mathrm{T}}J_c$, $V=J_p^{\mathrm{T}}J_p$, $U_\lambda=U+\lambda D_c^{\mathrm{T}}D_c$, $V_\lambda=V+\lambda D_p^{\mathrm{T}}D_p$和$W=J_c^{\mathrm{T}}J_p$, 可以将式(4)写成线性矩阵块结构, 即

$$\begin{bmatrix}U_\lambda & W\\ W & V_r\end{bmatrix}\begin{bmatrix}\delta_c\\ \delta_p\end{bmatrix}=-\begin{bmatrix}J_c^{\mathrm{T}}f\\ J_p^{\mathrm{T}}f\end{bmatrix}. \tag{5}$$

### 1.2 改进的捆集调整算法

本文对捆集调整算法作了两方面的改进, 一方面针对优化参数初值有不同的不确定度, 引入了带权值的雅可比矩阵, 从而使原算法改进成一种带权值的捆集调整算法; 另一方面针对三维重建捆集调整所得式(4)中广义海赛矩阵$H_\lambda$为高维稀疏矩阵, 直接求解会增大算法时间复杂度和空间复杂度, 因此采用共轭梯度法(CG)求解方程(4).

优化参数时, 输入的参数初值具有不同的不确定度, 例如摄像机外参比摄像机内参初值不确定度一般高1 000倍左右, 如果将其作为同等不确定度进行优化, 必定会降低优化结果精度, 而且还会减慢优化算法收敛速度. 因此, 本文提出带权值的捆集调整算法, 将参数不确定度引入到雅可比矩阵$J$中, 提高了算法优化精度, 其雅可比矩阵的改进形式如下:

$$J=\begin{bmatrix}c_1\dfrac{\partial f_1}{\partial x_1} & c_2\dfrac{\partial f_1}{\partial x_2} & \cdots & c_m\dfrac{\partial f_1}{\partial x_m}\\ c_1\dfrac{\partial f_2}{\partial x_1} & c_2\dfrac{\partial f_2}{\partial x_2} & \cdots & c_m\dfrac{\partial f_2}{\partial x_m}\\ \vdots & \vdots & \ddots & \vdots\\ c_1\dfrac{\partial f_n}{\partial x_1} & c_2\dfrac{\partial f_n}{\partial x_2} & \cdots & c_m\dfrac{\partial f_n}{\partial x_m}\end{bmatrix}, \tag{6}$$

其中$c_i$为第$i$个参数的不确定度.

方程(4)的系数矩阵为高维稀疏矩阵, 求解其精确形式较为繁琐, 为了简化算法复杂度, 本文提出用预先共轭梯度法求解方程. 共轭梯度法求解方程$Ax=b$有以下优点: 共轭梯度算法中, 系数矩阵$A$的作用仅仅是由已知向量$P$产生向量$w=AP$, 对提供$A$比较困难时, 可以无需$A$的精确形式求解方程$Ax=b$. 对于求解方程(4), 向量

$$w=H_\lambda P=J^{\mathrm{T}}(JP)+\lambda D^{\mathrm{T}}DP. \tag{7}$$

已知雅克比矩阵$J$和向量$P$, 可简单地求解$JP$, $D^{\mathrm{T}}D$又是对角阵, 可以容易地得到向量$w$, 这比直接求解系数矩阵$A$简化得多. 为了提高共轭梯度法的收敛速度, 本文采用预先共轭梯度法, 根据方程(5)的线性矩阵块结构, 取预先矩阵

$$M_\lambda=\begin{bmatrix}U_\lambda & 0\\ 0 & V_\lambda\end{bmatrix}. \tag{8}$$

根据上述分析, 改进的捆集调整算法伪代码如

下:

迭代优 = 初值, $k = 0$;

while $\|f(x_k) > \varepsilon\|$ and $k < k_{\max}$.

利用共轭梯度法求方程

$$(J^{\mathrm{T}}J + \lambda D^{\mathrm{T}}D)\delta_k = -J^{\mathrm{T}}f.$$

将求得的 $\delta_k$ 代入方程

$\delta_k^* = \|J(x_k)\delta_k + f(x_k)\|^2 + \lambda\|D(x_k)\delta_k\|^2$,

if $\|f(x_k + \delta_k^*)\| < \|f(x_k)\|$,

$x_{k+1} = x_k + \delta_k^*$, $k = k + 1$.

else

跳出程序

end

end

## 2　多核并行捆集调整算法

### 2.1　多核并行捆集调整算法原理

首先, 分析捆集调整算法中耗时量大的主要计算步骤, 由捆集调整算法伪代码可得, 捆集调整算法主要耗时的步骤如下:

1) 计算 3D 点重投影误差 $f$;

2) 计算雅可比矩阵 $J = [J_c, J_p]$;

3) 计算矩阵与向量的乘积 $Jx = J_c x_c + J_p x_p$;

4) 计算矩阵与向量的乘积 $J^{\mathrm{T}}y = [J_c^{\mathrm{T}}y, J_p^{\mathrm{T}}y]$;

5) 计算预先矩阵 $M_\lambda$.

为了估计它们在整个捆集调整算法的耗时量, 利用单核 CPU 运行本文改进后的捆集调整算法, 统计上述主要耗时计算步骤在整个捆集调整算法的耗时情况. 分别用北京天坛祈年殿和北京故宫保和殿场景做测试, 统计算法迭代 60 次的数据如表 1 所示.

**表 1　捆集调整算法主要步骤耗时情况**　　%

| 目标 | $f$ | $J$ | $M_1^{-1}$ | $Jx$ | $J^{\mathrm{T}}y$ |
|---|---|---|---|---|---|
| 祈年殿 | 13 | 5.5 | 3.4 | 27 | 56 |
| 保和殿 | 0.8 | 3.4 | 1.8 | 29 | 59 |

由表 1 可得计算 3D 点重投影误差 $f$, 雅可比矩阵 $J$ 以及预先矩阵 $M_\lambda$ 耗时量不多, 这是因为算法迭代一次的过程中只需计算一次, 而计算矩阵与向量的乘积 $Jx$ 和 $J^{\mathrm{T}}y$ 耗时大, 这缘于利用预先梯度法求解方程需要多次计算 $Jx$ 和 $J^{\mathrm{T}}y$, 这要求它们的并行计算程度尽量大. 由表 1 还可知, 上述 5 个计算步骤占据整个捆集调整算法绝大部分时间, 只要分别实现它们的并行化计算, 就相当于实现了整个捆集调整算法的并行化计算.

其次, 分析上述 5 个耗时量大的主要计算步骤并行化计算的可能性以及如何并行化. 为实现多核多线程并行化算法, 要求每个线程相对独立以及数据之间没有依赖性. 对于计算 3D 点重投影误差 $f$, 每个图像 3D 点重投影误差 $f$ 的计算没有关联性, 这就可以同时并行计算每个图像 3D 点重投影误差 $f$. 对于计算雅可比矩阵 $J$, 同样每个图像点的计算保持着独立性, 每个图像点可以对应一个线程, 可以多核并行计算雅可比矩阵 $J$. 对于计算矩阵与向量的乘积 $Jx$ 和 $J^{\mathrm{T}}y$, 每张图像所对应的摄像机的所有摄像机参数的计算保持着独立性以及每个 3D 点的计算保持着独立性, 可以并行计算每个摄像机参数所对应的矩阵元素与向量元素相乘以及每个 3D 点所对应的矩阵元素与向量元素相乘. 最后, 并行计算预先矩阵 $M_\lambda$ 的计算原理跟计算矩阵与向量相乘原理相同.

### 2.2　多核并行捆集调整算法在 GPU 上的实现

算法并行实现可利用多核 CPU 和 GPU 两种方式, 但一般 CPU 处理器内核数量是有限的, 并行计算能力不是很理想. 而采用的 SIMT 架构 GPU 拥有几百个甚至上千个流处理器, 非常适合计算密集型和高度数据并行的应用, 从而本文选择由 GPU 实现本文提出的多核并行捆集调整算法. 在 SIMT 结构中, 线程是指令执行的基本单元, 多个线程可形成一个线程块, 线程块内每 32 个线程组织成一个 warp. 同一个 warp 的线程将在多个处理单元上按照同一个指令流并行执行. 在 GPU 多线程系统实现中, 需要注意两大主要问题: 1) 最大化处理器的占有率; 2) 优化数据的存取结构. 这牵涉到线程的组织结构和数据存储结构的设计, 下面将主要从这两方面介绍如何在 GPU 上实现捆集调整.

1) 利用 GPU 并行计算 3D 点重投影误差 $f$.

为最大化 GPU 处理器占有率, 设计 3D 点重投影误差 $f$ 并行运算时, 每个图像点对应一个线程, 总共有 64 个线程块, 每个图像点的计算都是并行的. 为提高 3D 点重投影误差 $f$ 的存取效率, 可将数据存储在纹理储存器中.

2) 利用 GPU 并行计算雅可比矩阵 $J = [J_c, J_p]$.

为实现 GPU 的高度数据并行性, 雅可比矩阵 $J = [J_c, J_p]$ 的求解采取与计算 3D 点相同的重投影误差的线程组织结构, 即每个图像点对应一个线程. 为了充分利用 GPU 的有限 RAM 储存器, 雅可比矩阵采用块压缩稀疏行格式存储.

3) 利用 GPU 并行计算矩阵与向量的乘积 $Jx = J_c x_c + J_p x_p$ 和 $J^{\mathrm{T}}y = [J_c^{\mathrm{T}}y, J_p^{\mathrm{T}}y]$.

雅可比矩阵与向量相乘 $Jx = J_c x_c + J_p x_p$ 与雅可比逆矩阵与向量相乘 $J^{\mathrm{T}}y = [J_c^{\mathrm{T}}y, J_p^{\mathrm{T}}y]$ 的并行计算原理相同, 这里将分析前者的并行计算. 分两个方面分析在 GPU 上实现雅可比矩阵与向量的乘积 $Jx =$

$J_c x_c + J_p x_p$, 一方面是向量 $x$ 的存储结构: 先存储向量 $x$ 中有关摄像机的参数, 再存储向量 $x$ 中有关3D点的参数, 并且分别按摄像机和3D点ID号的大小顺序存储; 另一方面是线程内核的设计, 同样为了最大化GPU处理器的占有率, 每个图像点对应一个线程, 线程块的宽度是32的整数倍, 每个线程块有64个线程, 即两个warp.

4) 利用GPU并行计算预先矩阵 $M_\lambda$.

求 $M_\lambda$ 相当于求如下两个矩阵:

$$U_\lambda = U + \lambda D_c^{\mathrm{T}} D_c = J_c^{\mathrm{T}} J_c + \lambda \mathrm{diag}(J_c^{\mathrm{T}} J_c), \quad (9)$$

$$V_\lambda = V + \lambda D_p^{\mathrm{T}} D_p = J_p^{\mathrm{T}} J_p + \lambda \mathrm{diag}(J_p^{\mathrm{T}} J_p). \quad (10)$$

进而可以简化求 $J_c^{\mathrm{T}} J_c$ 和 $J_p^{\mathrm{T}} J_p$, 矩阵与矩阵相乘可以等价为矩阵与多个向量相乘, 因而有

$$J_c J = J_c(x_1, \cdots, x_n) = (J_c x_1, \cdots, J_c x_n), \quad (11)$$

$$J_p J = J_p(x_1, \cdots, x_m) = (J_p x_1, \cdots, J_p x_m). \quad (12)$$

从式(10)和(11)很容易看出, 在GPU上计算预先矩阵 $M_\lambda$ 与计算雅可比矩阵与向量的乘积的方法类似. 由于 $M_\lambda$ 是稀疏矩阵(很多矩阵元素上的值为零), 同样为了节省GPU中非常有限的RAM资源[7], 仍采用块压缩稀疏行(BCSR)格式存储.

## 3 实验结果及分析

本文采集了北京天坛祈年殿356张图像, 北京故宫交泰宫135张图像, 北京故宫保和殿256张图像和北京乾清宫302张图像, 分别用未改进的捆集调整的三维重建算法和本文提出的多核并行捆集调整的三维算法进行重建, 算法除捆集调整部分不相同外, 其他部分都相同, 比较了它们的重建精度以及算法的处理速度.

1) 重建精度比较. 首先展示上述两种三维重建算法重建得到的三维模型, 图1为重建场景所用的部分图像, 图2和图3是三维重建得到的模型.

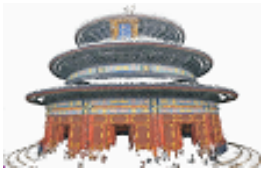
(a) 祈年殿图像

(b) 交泰宫图像

(c) 保和殿图像

(d) 乾清宫图像

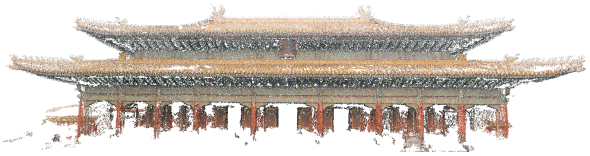
**图 1 三维重建所用的部分图像**



(a) 祈年殿模型

(b) 交泰宫模型

(c) 保和殿模型



(d) 乾清宫模型

**图 2 未改进的捆集调整的算法重建得到的三维模型**

(a) 祈年殿模型

(b) 交泰宫模型

(c) 保和殿模型

(d) 乾清宫模型

**图 3 多核并行捆集调整的算法重建得到的三维模型**

三维重建中各参数具有很强的关联性, 其中某参数的精度能代表整体三维重建精度, 因此比较三维重建精度可以通过比较重建得到的摄像机位置精度. 在采集北京天坛祈年殿数据时, 用徕卡TCR402型号全站仪测量摄像机位置, 并与用上述两种算法重建得到的摄像机位置(其数据已经引入尺度)相比较, 得到的数据如表2所示.

表2中, 真实值 $(x, y, z)$ 是利用全站仪测量得到的摄像机位置坐标, CPU $(x, y, z)$ 和GPU $(x, y, z)$ 分别是由未改进捆集调整和本文提出的多核并行捆集调整的三维重建算法重建得到的摄像机位置的坐标, 其值引入了尺度并将第1张图的摄像机位置作为坐标

**表 2　重建精度比较**　　m

| 图像号 | 真实值$(x, y, z)$ | CPU $(x, y, z)$ | CPU 误差/% | GPU $(x, y, z)$ | GPU 误差/% |
|---|---|---|---|---|---|
| 1 | 0.00, 0.00, 0.00 | 0.00, 0.00, 0.00 | 0.0 | 0.00, 0.00, 0.00 | 0.0 |
| 2 | −0.70, 0.80 , 0.07 | −0.65, 0.75, 0.069 | 2.80 | −0.68, 0.79, 0.07 | 1.37 |
| 3 | 1.65, 1.75, 0.069 | 1.55, 1.63, 0.073 | 4.78 | 1.68, 1.78, 0.06 | 2.63 |
| 4 | 2.05, 2.05, 0.065 | 1.93, 2.34, 0.074 | 6.57 | 2.08, 1.95, 0.07 | 4.17 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 356 | 36.89, 31.75, 0.19 | 38.09, 30.15, 0.17 | 6.27 | 37.49, 30.45, 0.18 | 3.62 |

原点. 表 2 中误差的计算公式如下 (CPU 误差的平均值是 4.95%, GPU 误差的平均值是 3.12%):

$$\mathrm{ess} = \left(\frac{|x - x_0|}{x_0} + \frac{|y - y_0|}{y_0} + \frac{|z - z_0|}{z_0}\right) \times \frac{1}{3} \times 100\%. \tag{13}$$

从直观视觉上, 上述两种三维重建算法重建出图 2 和图 3 的模型只有极小的差距, 但由表 2 数据计算得到的平均误差可知, 本文提出的多核并行捆集调整的三维重建算法重建结果平均误差小, 重建精度高.

2) 算法处理速度比较. 实验测试平台所有计算机显示卡型号是华硕 GTX560, CPU型号是 Intel i5-2500K, 记录了两种算法的耗时情况, 如表 3 所示.

**表 3　重建算法耗时情况**

| 场景名称 | 图像数/张 | CPU 用时/h | GPU 用时/min |
|---|---|---|---|
| 祈年殿 | 356 | 37 | 97 |
| 乾清宫 | 302 | 29 | 79 |
| 保和殿 | 256 | 12 | 36 |
| 交泰宫 | 135 | 2.35 | 7 |

表 3 中 CPU 用时和 GPU 用时分别代表未改进捆集调整与本文提出的多核并行捆集调整的三维重建算法运行时间情况. 由表 3 可得, 后者的用时明显比前者少, 本文提出的多核并行捆集调整的三维重建算法极大地提高了处理速度.

统计了随着图像张数的增加未改进的捆集调整与本文提出的多核并行捆集调整的三维重建算法运行时间情况, 其统计结果绘制的曲线如图 4 所示.

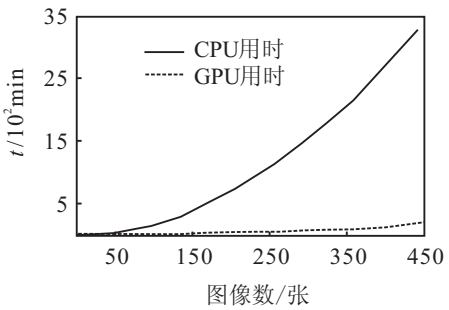



**图 4　CPU 算法和 GPU 算法用时统计图**

由图 4 中的曲线可知, CPU 用时曲线随着图像张数的增加其曲线斜率的增加速度要比 GPU 用时曲线的快, 说明随着图像张数的增加, 本文提出的多核并行捆集调整比未改进的捆集调整的三维重建算法运行时间增加速度要平缓, 由此可得出: 随着图像的增多, 本文提出的多核并行捆集调整的三维重建算法处理速度优势更加明显.

## 4　结　论

本文提出的多核并行捆集调整算法主要有 3 个方面的贡献: 1) 改进了捆集调整算法本身; 2) 使捆集调整算法最大程度并行化; 3) 在 GPU 上实现本文提出的多核并行捆集调整算法. 经过实验验证, 本文提出的多核捆集调整算法有如下 3 大优点:

1) 硬件上容易实现. 不象利用多核 CPU 并行计算加速那样, 需要昂贵的专业多核 CPU 计算机. 随着 GPU 迅速发展, 其已具有超强的浮点计算能力, 而计算开发平台 (CUDA)的出现使得在 GPU 上进行通用计算变得简单可行, 且 GPU 的价格也相对便宜.

2) 极大地提高了三维重建算法整体处理速度. 本文提出的多核并行捆集调整算法能使三维重建算法整体运行速度至少提高 20 倍.

3) 重建精度高. 改进了捆集调整算法本身, 提出了带权值的捆集调整算法; 同时提出了多核并行捆集调整算法, 且采用拥有超强并行计算能力的 GPU 实现该并行算法, 极大地提高了捆集调整算法处理速度. 这就允许增大捆集调整迭代次数, 提高捆集调整算法优化精度, 从而可得到更精确的重建结果.

### 参考文献(References)

[1] Debevec P, Taylor C, Malik J. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach[C]. SIGGRAPH 96 Conf on Proceedings. 1996: 11-20.

[2] Sinha, Steedly, Szeliski. Piecewise planar stereo for image-based rendering[C]. Int Conf on Computer Vision. Kyoto: IEEE, 2009: 23-40.

[3] Seth Teller. Automated urban model acquisition: Project rationale and status[C]. Image Understanding Workshop. Washington: IEEE, 1998: 455-462.

[4] Seth Teller, Matthew Antone, Zachary Bodnar. Calibrated, registered images of an extended urban area[J]. J of Computer Vision, 2003, 53(1): 3-107.

[5] Schindler, Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex manmade environments[C]. Computer Vision and Pattern Recognition. Washington: IEEE, 2004: 203-209.

[6] Schindler, Dellaert. Probabilistic temporal inference on reconstructed 3d scenes[C]. Computer Vision and Pattern Recognition. San Francisco: IEEE, 2010: 456-475.

[7] Schindler, Krishnamurthy, Dellaert. Line-based structure from motion for urban environments[J]. 3D Data, Processing, Visualization and Transmission, 2006, 32(1): 16-32.

[8] Sameer Agarwal, Noah Snavely, Ian Simon. Building Rome in a day[C]. Int Conf on Computer Vision. Kyoto: IEEE, 2009: 1223-1421.

[9] Sameer Agarwal, Noah Snavely, Richard Szeliski. Bundle adjustment in the large[C]. European Conf on Computer Vision. Greece: IEEE, 2010: 3057-3063.

[10] Richard Hartley, Andrew Zisserman. Multiple view geometry in computer vision second edition[M]. UK: Cambridge University Press, 2004.

[11] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares[J]. The Quarterly of Applied Mathematics, 1944, 2(17): 164-168.

(上接第1402页)

[8] Huang L, Lai Y C, Chen G R. Understanding and preventing cascading breakdown in complex clustered networks[J]. Physical Review E, 2008, 78(3): 036116.

[9] Schafer M, Scholz J, Greiner M. Proactive Robustness control of heterogeneously loaded networks[J]. Physical Review Letters, 2006, 96(10): 108701.

[10] Yang R, Wang W X, Lai Y C, et al. Optimal weighting scheme for suppressing cascades and traffic congestion in complex networks[J]. Physical Review E, 2009, 79(2): 026112.

[11] Li P, Wang B H, Sun H, et al. A limited resource model of fault-tolerant capability against cascading failure of complex network[J]. The European Physical J B, 2008, 62(1): 101-104.

[12] Wu Z X, Peng G, Wang W X, et al. Cascading failure spreading on weighted heterogeneous networks[J]. J of Statistical Mechanics: Theory and Experiment, 2008(5): 05013.

[13] Wang J W, Rong L L, Zhang L, et al. Attack vulnerability of scale-free networks due to cascading failures[J]. Physica A, 2008, 387(26): 6671-6678.

[14] Wang W X, Chen G R. Universal robustness characteristic of weighted networks against cascading failure[J]. Physical Review E, 2008, 77(2): 026101.

[15] Mirzasoleiman B, Babaei M, Jalili M, et al. Cascaded failures in weighted networks[J]. Physical Review E, 2011, 84(4): 046114.

[16] NLANR. AS graphs[EB/OL]. (2001-3)[2010-5-25]. http://www. guidocaldarelli.com.

[17] Kurant M, Thiran P. Layered complex networks[J]. Physical Review Letters, 2006, 96(13): 138701.

[18] Ercsey-Ravasz M, Toroczkai Z. Centrality scaling in large networks[J]. Physical Review Letters, 2010, 105(3): 038701.

# 下 期 要 目

压缩感知综述 · · · · · 尹宏鹏, 等
基于干扰观测器的高超音速飞行器鲁棒反步控制 · · · · · 王首斌, 等
一种基于最大边界投影和$l_{2,1}$范数正则化的属性选择算法 · · · · · 夏建明, 杨俊安
“报童问题”中风险偏好下的条件风险值及其优化 · · · · · 简惠云, 许民利
供应商损失厌恶情形下组装供应链协调 · · · · · 付 红, 等
一种基于相角映射的改进多目标粒子群优化算法 · · · · · 李 婷, 等
基于多块KPCA和SDG的故障诊断方法 · · · · · 王雅琳, 等
振荡型GM(1,1)幂模型及其应用 · · · · · 王正新
基于簇内不平衡度量的粗糙$K$-means聚类算法 · · · · · 张腾飞, 等
具有动态不确定性的自适应动态面控制 · · · · · 张天平, 高志远