基于 OpenCL 的 ICP 点云并行配准算法

刘 忠 建
(北方工业大学计算机学院 北京 100144)

摘 要 针对当前点云配准算法效率过低的问题,运用 OpenCL 实现了基于通用 GPU 的 kd-tree 并行搜索算法,进而实现了 ICP 点云并行配准算法。首先建立目标点云的三维空间 kd-tree,并使用 OpenCL 并行加速其搜索算法;然后将并行加速的 kd-tree 搜索算法运用于 ICP 算法,同时针对 ICP 算法的其他部分也使用 OpenCL 并行加速以确保配准过程尽可能高效。通过实验验证了所实现算法的高效性以及健壮性。

关键词 OpenCL GPU kd-tree 点云配准 ICP

中图分类号 TP391.41 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2016.11.043

PARALLEL REGISTRATION ALGORITHM OF ICP POINT CLOUD BASED ON OPENCL

Liu Zhongjian
(*College of Computer, North China University of Technology, Beijing 100144, China*)

Abstract In view of the problem of too low efficiency the current point cloud registration algorithm has, by using OpenCL we achieved the general GPU-based kd-tree parallel search algorithm, and then realised the parallel ICP point cloud registration algorithm. First we established the three-dimensional kd-tree of target point cloud, and used the OpenCL to parallelly accelerate its search algorithm. Then we applied the parallelly accelerated kd-tree search algorithm in ICP algorithm, at the same time the OpenCL parallel acceleration was also used aimed at the rest part of ICP algorithm to ensure as much as possible the efficient registration process. Experiments verified the efficiency and robustness of the implemented algorithm.

Keywords OpenCL GPU kd-tree Point cloud registration ICP

0 引 言

随着三维扫描技术的发展成熟与普及,三维应用技术得到了较好的发展。特别是在逆向工程、缺陷检测、文物保护、医学手术、3D 打印以及游戏娱乐等领域获得了广泛的关注与研究。由于部分遮挡等问题,所以为了获得完整的点云数据,需要进行多次扫描拼接配准。而三维点云配准一直以来都是三维扫描问题中的关键技术。

随着三维扫描器材设备的不断发展更新,扫描数据量不断增大,传统的配准算法随着数据量的激增效率逐渐降低,无法满足实时性的需求。图形处理单元 GPU、开放运算语言 OpenCL 以及计算机视觉与计算机图形学的发展为解决相关大数据量的问题提供了通用的、高性能的并行计算环境,使得实时解决点云配准问题成为可能。本文基于 OpenCL 对 kd-tree 搜索算法并行加速,实现了高效并行的 ICP[1] 配准算法,对于提高配准算法的效率具有十分显著的效果。

1 相关工作

点云配准算法通常是指 ICP 点云配准算法,该算法是通过迭代点云数据 $X = \{x_1, x_2, \cdots, x_{n_x}\}$ 和 $Y = \{y_1, y_2, \cdots, y_{n_y}\}$ 反复迭代计算旋转矩阵 R 与平移向量 t,使得对应点的距离和最小,即通过下面的目标函数最小化实现点云的配准工作。

$$E = \sum_{i=1}^{n_y} \| x_i - (Ry_i + t) \|_2^2 \quad (1)$$

已有围绕 ICP 配准算法做出的很多研究与改进,其中 Zhang[2] 提出引入 kd-tree 算法对搜寻对应点起到了很好的加速作用。Granger 等[3] 提出将最大期望算法 EM(Expectation Maximization algorithm)算法应用到 ICP 中,较好地解决了 ICP 配准算法初始配准的问题。Rusinkiewicz 等[4] 对有关 ICP 配准算法所做的改进作了一个全面的分析与总结。Gold 等人[5] 提出了一种名为 Softassign 的配准算法,具有较好的配准效果。但是 EM-ICP 算法与 Softassign 算法由于算法本身的原因需要计算两个点云数量相乘的矩阵,使得算法在点云数量集不断增大的情况下,可行性与实时性急速降低。Tamaki 等人[6] 使用了 CUDA (compute unified device architecture) 对 EM-ICP 配准算法以及 Softassign 配准算法进行了并行优化,但实验所使用的数量级也不能达不到数据量不断增加的需求,并且 CUDA 目前只支持 NVIDIA 系列显卡,不能做到对绝大多数 CPU、显卡等异构平台

收稿日期:2015-05-16。国家自然科学基金项目(61202229);北京市自然科学基金项目(4132018)。**刘忠建**,硕士生,主研领域:计算机图形学。

的通用性支持。国内针对点云配准的研究也是基于 CUDA 的 EM-ICP 与 Softassign 算法的研究,其中蔡静等人[7,8]采用的是下采样的方法来降低配准数据的数据量。下采样会对原始数据造成分辨率以及细节的缺失,本文认为这种做法是不太可取的。即使是下采样过后,采用这两种算法要求的计算数据量也是相当大的,这也是本文不采用这两个算法的一个重要原因。

2 ICP 配准算法

ICP 配准算法即迭代最近点配准算法,是一种基于自由形态曲面的配准算法。对于两个待匹配的点云数据集 X 和 Y,在数据点集 X 中找到一点 x_i 与 Y 点集中一点 y_i 对应,使目标函数式(1)最小。而 EM-ICP 配准算法[4]由于引入了最大期望算法,它的目标函数为:

$$E = \sum_{j=1}^{n_x}\sum_{i=1}^{n_y}\alpha_{ij}d_{ij}^2 \quad d_{ij} = \| x_j - (Ry_i + t) \| \tag{2}$$

$$\begin{cases}\alpha_{ij} = \dfrac{1}{C_i}\exp\left(\dfrac{-d_{ij}^2}{\sigma_p^2}\right)\\ C_i = \exp\left(\dfrac{-d_0^2}{\sigma_p^2}\right) + \sum_{k=1}^{nx+1}\exp\left(-\dfrac{-d_{ik}^2}{\sigma_p^2}\right)\end{cases} \tag{3}$$

其中 σ_p 是参数,d_0 是初始已知值,α_{ij} 为两点为对应点的概率。同样,Softassign 配准算法[5]中引入了双随机矩阵,再运用近似均值优化以及模拟退火算法,其目标函数为:

$$E = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}m_{ij}\| x_i - (R^{k-1}y'_j + t^{k-1}) \| - \alpha\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}m_{ij} - \frac{1}{\beta}\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}m_{ij}(\log m_{ij} - 1) \tag{4}$$

因此 EM-ICP 与 Softassign 算法所计算的数据量为 $n_x \times n_y$,而 ICP 所需要计算的数据量为 n_y。

ICP 配准算法如下:

输入 待配准点云数据 $X = \{x_1, x_2, \cdots, x_{n_x}\}$ 和 $Y = \{y_1, y_2, \cdots, y_{n_y}\}$、$R^0$、$t^0$;

输出 配准结果 R' 和 t'。

步骤 1 $R^0 \leftarrow I, t^0 \leftarrow 0$;

步骤 2 for $k = 1, \cdots,$ maxIterations do;

步骤 3 for each point y_i in Y do;

步骤 4 寻找到最近的点 x_{ik} in X:

$$ik = \underset{j=1,2,\cdots,n_x}{\operatorname{argmin}} \| x_j - (R^{k-1}y_i + t^{k-1}) \|$$

步骤 5 end for;

步骤 6 建立对应于 Y 的 $X^k = \{x_{1k}, x_{2k}, \cdots, x_{n_yk}\}$ 数据集;

步骤 7 利用四元数计算 R' 和 t';

步骤 8 $R^k \leftarrow R', t^k \leftarrow t'$;

步骤 9 end for。

有些研究学者认为 ICP 算法的关键问题不在于它的配准速度而在于其配准的精度,即 ICP 一直存在的初始位置的问题。当两个点云的初始位置距离很远或者重叠部分不是特别好时,ICP 配准算法的效果是相当差的,这也是很多学者倾向于认为配准点云实际上是一种粗配准(又叫预配准)的原因。另一个角度而言,构建一个模型或者一个空间的扫描建模时,是可以控制相邻点云数据的重叠以及距离的。因此,本文认为 ICP 算法在做扫描点云配准时还是优于其他配准算法,这也是本文用 ICP 算法做点云配准的主要原因。

3 kd-tree 算法

kd-tree 是一种分割 k 维数据空间的数据结构,主要应用于多维空间关键数据的搜索(如:范围搜索和最近邻搜索)。kd-tree 是二进制空间分割树的一种特殊情况。kd-tree 由 Zhang[2]首先引入到 ICP 算法中,用于最近邻搜索。kd-tree 算法分为两个部分,第一部分为创建 kd-tree,第二部分才是本文的目的:搜索最近邻的节点。

创建 kd-tree 分为递归算法与非递归的算法,为提高效率,本文采用的是非递归算法。而搜索算法也是采用非递归式的搜索,因此本文将搜索算法分为两个步骤,即先向下搜索直到叶子节点,找到第一个最近邻参照节点,随后利用父节点信息对满足条件的节点进行回溯搜索。

4 基于 OpenCL 的并行算法

开放计算语言 OpenCL 是一个为异构平台编写程序的框架,此异构平台可由 CPU、GPU 或其他类型的多核处理器组成。OpenCL 为开发人员提供了基于任务分割和数据分割的并行计算机制以及一套支持 C99 标准的内核 API 与控制 API,以帮助开发人员在 GPU 上运行一个并行的内核函数,充分提高算法运行效率。

4.1 GPU 下的 kd-tree 算法

kd-tree 的数据结构大大地方便了单个数据在 k 维空间中进行搜索,但是当需要搜索的数据量非常大时,kd-tree 的搜索还是会比较迟缓的。因此,为了进一步提高算法的效率,将 kd-tree 算法运用 OpenCL 移植到 GPU 上,通过并行对多个数据同时进行搜索来达到提高算法效率的目的。

为方便地将 kd-tree 的节点数据传输到 GPU 下,使用一个数组的形式来存储 kd-tree 的节点,将当前节点的子节点用一个整型数组来存储它们的序号,以方便查找。同样为了方便回溯查找可能的最近邻节点,设置一个整型数据位表示父节点的序号。采用数组结构存储 kd-tree 的节点主要是考虑到满足 GPU 下全局内存的管理。对于 GPU 下建立 kd-tree,本文方法会比直接使用 CPU 下的 kd-tree 繁琐一点。首先,利用输入的样本集在 CPU 下建立 kd-tree;然后将 CPU 下的 kd-tree 节点转换为 GPU 下所支持的数组存储形式;最后将 kd-tree 传输到 GPU 的全局内存中,用于搜索使用。下面主要介绍 GPU 下的最近邻搜索算法。

本文所使用的 kd-tree 搜索算法分为三个步骤:第一步,从根节点向下根据 kd-tree 特性采用二分搜索;第二步,逐个比较叶子节点中所包含的样本数据集,选取最近邻节点;第三步,如果分支节点与待查询节点距离小于第二步中得到的最近邻节点与待查询节点距离,采用回溯查找的策略以确定最近邻节点,否则返回第二步中的结果。

GPU 下的 kd-tree 最近邻搜索算法如下:

输入 将 kd-tree、点云数据集 X 和 Y 加载到 GPU 的全局内存;

输出 对应点点集与对应点点集所对应的欧氏距离。

步骤 1 分配 n 个线程,n 为查询的点集数量;

步骤 2　在 GPU 中为输出分配相应的存储空间;

步骤 3　对每一个线程并行执行 kd-tree 搜索程序:

(1) 如果 kd-tree 是空的,则设距离为无穷大并返回;

(2) 沿 kd-tree 树向下搜索直到叶子结点;

(3) 如果分支节点与待查询节点的距离小于步骤(2)中搜索到的最近邻节点与待查询节点的距离,则回溯搜索,以确定最近邻节点;否则返回步骤(2)中的查询结果。

步骤 4　将 GPU 内存中的结果拷贝到内存中。

在实验中,发现了两个问题:第一个是当数据量较大的时候,树的深度在一定程度上影响着搜索算法的效率。根据实验,本文将树的最大深度定在 13 层会取得较好的结果,然后对叶节点中所包含的样本数据集再采用逐个比较的方式选取最近邻节点。第二个问题则是回溯算法的必要性问题。首先 ICP 算法本身就是一种采用最近邻近似寻求对应节点以解决配准问题的一个算法。其次,kd-tree 按照二维划分所找到的近似最近邻节点与真实最近邻节点的差距十分小。回溯算法需要更新最近邻节点的可能性也不是特别高。按照以上分析,本文做了关于回溯算法必要性以及回溯栈大小的影响的实验。本文将在第 5 节给出实验结果。

4.2　GPU 下的 ICP 算法

原始的 ICP 算法主要包括三大步骤:第一步是利用最近邻搜索查找对应点;第二步是利用四元数估计变换矩阵;第三步是通过变换矩阵转换数据。当目标方程达到误差标准时,退出算法,否则转到第一步继续执行。这三个步骤构成了 ICP 算法的基本结构,除第一步的对应点搜索外,其他两步包含大量的矩阵操作。第二步包含两个点云的交叉协方差矩阵与重心的计算都可以使用 GPU 加速。此外,第三步数据的转换其实就是两个矩阵相乘,因此也可以运用 GPU 加速,以提高算法的效率。

Bouaziz 等人[9]提出了一种新的衡量 ICP 算法的目标方程,称作 l_p-ICP。即式(1)可改为:

$$E = \sum_{i=1}^{n_y} \| x_i - (Ry_i + t) \|_2^p \qquad (5)$$

该算法在文献[9]中得到证实,当 $p \in [0,1]$ 的区间内时,可以有效地减少错误的对应点以及噪声点对算法的影响,以提高 ICP 算法配准的精度。但由于该文献采用了一种优化算法,迭代优化次数比较多,算法的执行时间比较长。本文简单地将这个方法使用在原始的 ICP 算法中用来排除错误对应点以及噪声点的影响,提高配准的精度。

5　实验结果及分析

本文将通过三个不同实验设计来验证本文所实现的算法,分别验证本文算法的运行效率与配准的精度。实验的编译环境为:Ubuntu 12.04 64 位。实验的硬件环境为:CPU 为 i7-4770 处理器,主频 3.40 GHz;GPU 为 AMD HD8490 显卡,显存为 1 GB,位宽为 64 位,主频为 875 MHz,流处理器 160 个,该显卡属于低端显卡。NVIDIA 显卡只需一个流处理器就能发挥作用,而 AMD 的显卡对流处理器定位不一样,要 5 个流处理器单元一组才能工作。实验所用数据为 Kinect 拍摄的室内场景。实验数据以及效果如图 1 和图 2 所示。

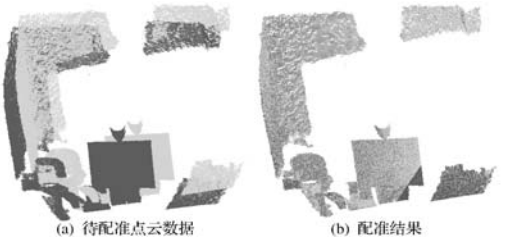

图 1　拍摄数据实验效果

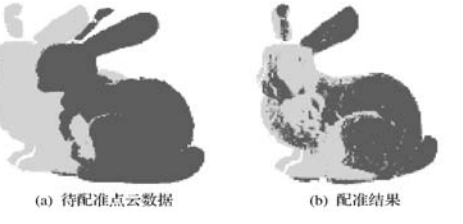

图 2　斯坦福兔子实验效果

5.1　基于 OpenCL 下的 ICP 与 CPU 下的 ICP 比较

将本文的算法与未使用 OpenCL 加速的情况下以及 Open PCL(Open Point Cloud Library)中所实现的 ICP 算法进行了运行效率对比,本文改进后算法具有较好的效率,特别是使用 OpenCL 并行加速之后的算法,效率有了较大的提升。实验结果如图 3 所示。

PCL-ICP 为 PCL 库中的 ICP 算法,KDtree-ICP 为本文优化之后使用 kd-tree 所实现的 ICP 算法,OpenCL-ICP 为本文最后实现的基于 OpenCL 的 ICP 算法。由于本文尽可能简化、优化 kd-tree算法,使得该算法更适用于 ICP 算法,从实验结果中可以明显发现改进之后的算法在相同的实验环境下比 PCL 库中所实现的算法有了较好的提高。而从实验中可以明显看到使用 OpenCL 之后的算法的执行效率有了较大的提升,但由于 GPU 计算核心的限制,在数据量较大时,算法的效率还是不够理想。如果采用性能更好的显卡,可以获得更好的结果。

5.2　回溯栈的大小对算法的影响

本文在实验中发现回溯栈的大小对算法具有较大的影响。首先从 ICP 算法本身考虑,ICP 算法本身就是通过最近邻节点近似表示对应点,通过 kd-tree 进行筛选之后的节点对于搜索点已经是一个十分近似的最近邻节点,理论上足以用来表示近似最近邻节点。再从 kd-tree 的构造上来看,本文并不是将 kd-tree 一直划分到只剩下一个节点,而是将树建到一定的深度就不再往下继续划分子树。因此每个叶子节点中包含有多个数据节点,在搜索过程中当搜索到叶子节点时是采用逐一比较的方法确定最近节点,这样就再一次从概率的角度缩小了需要再次回溯找到最近邻节点的概率。实验所用数据量为 298 227,实验结果如图 4 所示。

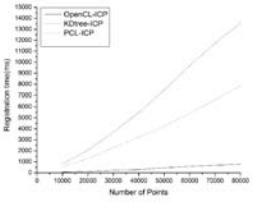

图 3　算法效率的实验

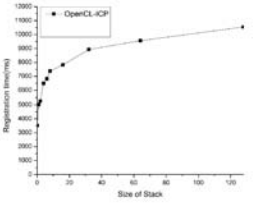

图 4　回溯栈大小对算法的影响

(下转第 192 页)

题,混沌烟花算法和烟花算法分别在第 9 代和第 230 代找到问题的最优解,其他两种算法未找到最优解;对于 Car6 问题,只有 PSO 未找到问题最优解,混沌烟花算法、烟花算法和萤火虫算法分别在 170 代、220 代和 280 代找到该问题的最优解。

4 结 语

在分析基本算法寻优机理的基础上,主要从编码、搜索半径、选择策略和精英混沌搜索四方面对原始算法加以改进。针对置换流水车间基准问题(Car 类和 Rec 类)求解,混沌烟花算法在寻优率、寻优稳定性、寻优速度等方面较之烟花算法、粒子群算法和萤火虫算法具有一定的优势。对于烟花算法的改进策略是有效的,改进后的烟花算法可以用来求解置换流水车间问题。将来可以将改进算法应用于大规模置换流水车间调度以及其他离散调度(作业车间、可重入调度等)中去。

参 考 文 献

[1] Tan Y, Zhu Y C. Fireworks algorithm for optimization[C]//Advances in Swarm Intelligence. Springer, 2010: 355 - 364.

[2] He W R, Mi G Y, Tan Y. Parameter optimization of local-concentration model for spam detection by using local-concentration model for spam detection by using fireworks algorithm[C]//Advances in swarm intelligence. Springer, 2013: 439 - 450.

[3] Imran A M, Kowsalya M. A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using Fireworks Algorithm[J]. International Journal of Electrical Power and Energy Systems, 2014, 62: 312 - 322.

[4] Gao H Y, Diao M. Cultural firework algorithm and its application for digital filters design[J]. International Journal of Modelling Identification and Control, 2011, 14(4): 324 - 331.

[5] Zheng S Q, Tan Y. A unified distance measure scheme for orientation coding in identification[C]//Information Science and Technology (ICIST), 2013 International Conference on. IEEE, 2013: 979 - 985.

[6] Zheng Y J, Song Q, Chen S Y. Multiobjective fireworks optimization for variable-rate fertilization in oil crop production[J]. Applied Soft Computing, 2013, 13(11): 4253 - 4263.

[7] 李兵, 蒋慰孙. 混沌优化方法及其应用[J]. 控制理论与应用, 1997, 14(4): 613 - 615.

[8] 李新杰, 胡铁松, 郭旭宁, 等. 0 - 1 测试方法的径流时间序列混沌特性应用[J]. 水科学进展, 2012, 23(6): 861 - 868.

[9] 娄素华, 吴耀武, 熊信艮. 电力系统无功优化的变尺度混沌优化算法[J]. 电网技术, 2005, 29(11): 20 - 24, 29.

[10] 张沫. 改进混合蛙跳算法的云计算资源调度[J]. 计算机应用与软件, 2015, 32(4): 330 - 333.

[11] Garey M R, Johnson D S, Sethi R. The complexity of flow shop and job shop scheduling[J]. Mathematics of Operations Research, 1976, 1(2): 117 - 129.

[12] 谭营, 郑少秋. 烟花算法研究进展[J]. 智能系统学报, 2014, 9(5): 515 - 528.

[13] Qian B, Wang L, Huang D X, et al. An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers[J]. Computers and Operations Research, 2009, 36(1): 209 - 233.

[14] Awad El-Gohary, A S Al-Ruzaiza. Chaos and Adaptive Control in Two Prey, One Predator System With Nonlinear Feedback[J]. Chaos, Solitons and Fractals, 2007, 34(2): 443 - 453.

[15] 刘长平, 叶春明. 基于逻辑自映射的变尺度混沌粒子群优化算法[J]. 计算机应用研究, 2011, 28(8): 2825 - 2827.

[16] 王凌. 车间调度及其遗传算法[M]. 北京: 清华大学出版社, 2003.

(上接第 187 页)

在实验中发现,随着回溯栈的增大,算法的效率会逐渐降低,但是会有一个最高点;而算法配准的精度当回溯栈大于 1 之后就没有太大的变化。这样就刚好验证了本文之前的假设:kd-tree 算法需要回溯更新最近邻节点的概率比较小,或者说大部分的回溯都是无用的,抑或回溯一次与第一次找到的最近邻节点已经足以表示近似最近邻节点。

5.3 l_p-ICP 目标函数的影响

根据 Bouaziz 等人[9]的方法,为了使算法简单,本文只使用了文献[9]中所使用的目标函数来替代本文中使用的目标函数进行实验。可能由于实验中并没有使用该文献所采用的优化算法,算法配准精度的提升并不明显,但是随着实验所取 p 值的减小,算法的配准精度整体是提高的。

6 结 语

点云配准一直都影响着逆向工程、扫描建模以及模式识别等重要领域。本文根据实验以及算法特性优化 kd-tree 算法,以使其更适用于 ICP 点云配准算法。并使用 OpenCL 平台,运用通用的 GPU 并行加速技术,实现并行 ICP 点云配准算法,在一定程度上提高了算法的执行效率,在实际应用中具有一定的价值。传统 ICP 算法由于采用最近邻查找对应点的算法特性,使得该算法很容易陷入局部最优而导致错误的配准结果。但本文并没有考虑这一影响,所以接下来的工作是如何能够较好地对任意输入的待配准数据进行初始配准,使算法在任意输入下都具有较好的健壮性。

参 考 文 献

[1] Besl P J, Mckay N D. A method for registration of 3-d shapes[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, 14(2): 239 - 256.

[2] Zhang Z Y. Iterative Point Matching for Registration of Free-From Curves and Surfaces[J]. International Journal of Computer Vision, 1994, 13(2): 119 - 152.

[3] Granger S, Pennec X. Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration[C]//Proceedings of the 7th European Conference on Computer Vision (ECCV2002), 2002: 418 - 432.

[4] Rusinkiewicz S, Levoy M. Efficient Variants of the ICP Algorithm[C]//Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling, 2001: 145 - 152.

[5] Gold S, Rangarajan A, Lu C P, et al. New algorithms for 2D and 3D point matching: pose estimation and correspondence[J]. Pattern Recognition, 1998, 31(8): 1019 - 1031.

[6] Tamaki T, Abe M, Raytchev B, et al. Softassign and EM-ICP on GPU[C]//Proceedings of the 2010 1st International Conference on Networking and Computing. Washington DC, USA: IEEE Computer Society, 2010: 179 - 183.

[7] 蔡静, 董琳, 孙晓鹏. 3D 人耳点云配准的并行 Softassign 算法[J]. 计算机工程与设计, 2013, 34(10): 3629 - 3634.

[8] 董琳, 何扬. 基于 EM-ICP 的三维人脸简化点云并行配准算法[J]. 微型机与应用, 2013, 32(16): 38 - 41.

[9] Bouaziz S, Tagliasacchi A, Pauly M. Sparse Iterative Closest Point[J]. Computer Graphics Forum, 2013, 32(5): 113 - 123.