指导教师：**　　提交时间：2015.3.27

The task of Digital Image Processing

数字图像处理

School of Computer Science

No: 第一次作业

姓名：张建鹏

学号：2012302582

班号：10011204

层叠哈希的快速、准确图像匹配的 3D 重建

Jian Cheng, Cong Leng, Jiaxiang Wu, Hainan Cui, Hanqing Lu

模式识别国家实验室，自动化协会，中国科学研究院

{jcheng,cong.leng,jiaxiang.wu,hncui,luhq}@nlpr.ia.ac.cn

摘要

在 3D 重建中，图像匹配是最具挑战性的阶段，它通常占据了一半的电脑消耗并且不精确的匹配可能导致重建失败。所以，快速、准确的图像匹配对于 3D 重建来说是非常重要的。在这片文章中，我们推荐用串级散列的方法来加速图像的匹配速度。为了加速图像的匹配速度，被推荐的串级散列方法被设计成三层结构：散列查找，散列重映射，散列排序。每一层都采取了不同的方法和被证明过对噪声具有不敏感性的过滤策略。大量的实验表明，通过上百次的图像匹配，和无理性强力匹配相比，我们实现了能够使其匹配速度更快。在保持相似的精确度的情况下，我们的速度甚至是基于 KD 树匹配的十倍以上。

1. 介绍

在计算机视觉界，3D 重建是一个经典的、具有挑战性的难题，并且发现它在不同的领域都有应用。最近几年，大规模社会照片集合的 3D 重建（例如，旅游照片[18]）已经变成了一个新兴的研究课题，这吸引了越来越多的来自各个学院和工厂的研究人员。然而，3D 重建的计算是极其昂贵的。举个例子，它可能在一台简单的机器上花费超过一天的时间来重建一个只有一千张照片的项目。在运动（SFM）模型[1,4]的结构中，3D 重建途径可以被分为几个步骤：特征提取，图象匹配，跟踪生成和集合估计，等等。在这当中，图像匹配占据了主要的计算成本，甚至在某些情况下超过半数。此外,不准确的匹配结果可能会导致重建失败。因此,快速、准确的图像匹配是三维重建的关键所在。

图象匹配技术大致可以分为三类:点匹配、线匹配和区域匹配。由于其鲁棒性变化的启发,仿射变换和观点的变化,点匹配已收到关注,并且在过去的几十年里[13,2]，许多有效的算法已经被提出。然而,点匹配通常是非常耗时的。两两图像匹配中，经过详细比较两个图象中的所有特征点，其计算复杂度为 O(N2),其中 N 是在每一个图象中特征点的平均数量。

作为一种替代方法，近似最近邻（ANN）搜索已经被研究用来在 3D 重建中取代详尽匹配的方法。一个经典范式是基于树的方法，它趋向于用高效的数据结构来存储数据，并且使得搜索操作比那的相当快，通常它的复杂性为 O(log(N))。基于树的代表性的算法有 Kd-树，M-树和 ball-树[15]，在这当中 Kd-树可能是在文学 3D 重建中使用最为广泛的策略[1,4]。然而,高维度将在一定程度上恶化树型方法的效率,甚至使他们执行得比幼稚的方法更糟糕（如，线性扫描)。

由于在存储和速度两方面的效率,在过去的几年中[9,7],基于哈希的 ANN 搜索方法已经吸引了很多人的关注。散列将图象所有的特征表现转换为二进制码,然后以极快的速度进行按位异或操作。哈希策略第一次在 3 d 重建 LDAHash[19]引入匹配过程。LDAHash 对二值化前的描述符执行线性判别分析(LDA)。然而，LDAHash 使用一个详尽的线性搜索来找到匹配点,这极大地降低了它的效率。此外,LDAhash 是一个受监督并且和数据相关的方法，它在训练阶段需要人为的额外标签。

受到 LDAHash 的启发,在这篇文章中,我们提出一个串级散列结构加快图像匹配的 3 d 重建,我们把这个方法叫做 CasHash。该方法的优点是双重的。一方面,我们提出的串级散列结构包含三层，它从粗到细将图象映射表示为二进制代码,导致图像匹配的大大加速。另一方面,每一层的串级散列采用不同的测量和过滤策略,这是证明对噪声不敏感的特征点。

与最相似的 LDAHash 相比,我们的方法有三个主要的优点。首先,我们的串级散列结构增大、更快的加速图像匹配。第二,我们的方法可以被认为是一个抵抗的噪声点对的多级过滤器。第三,我们利用一个非监督和数据独立的方法,即位置敏感哈希(激光冲徊化)[3],生成散列函数。因此,我们的方法是数据独立的和自由训练的。 大量实验表明,图像匹配可以使我们的方法加速为 Kd-tree 基础方法的十倍或以上,同时保持类似的准确性。

2. 本文提出的方法

在本文中,我们提出一个名叫 CasHash 的串级散列结构,用来加速 3D 重建图像匹配。在我们的方法中,一个简单的散列算法,定位敏感哈希(LSH),被用来生成二进制代码(参见图 1)。因此,我们将给出一个
LSH 算法的简要介绍。

2.1. 定位敏感哈希

令 $X=\{x_1, x_2, \ldots, x_n\}$ 为一组数据点，这里 $x_i \in R^d$. 给定一个查询向量 q, 我们所感兴趣的是在 X 当中寻找一个和 q 最相似的向量。LSH 也许是基于 ANN 搜索设计的最广为人知的哈希算法，它的存在依赖于定位敏感哈希函数。假设 H 是一个哈希家族函数，该函数从 R^d 映射到海明空间 B。对任意的两个点 x、y, 从 H 当中随机选择一个功能函数 h，该功能函数受限于概率 $h(x)=h(y)$。如果它家族函数 H 满足一下情况，则它具有敏感定位性：

定义 1：我们把从 R^d 映射到 B 的系列函数 H 记做(R, cR, P1, P2)，H 对 D(,)是敏感的，如果对任意的 $x, y \in R^d$ 都有：

- $P_{rh\in H}(h(x)=h(y)) \geq P_1$, if $D(x,y) \leq R$
- $P_{rh\in H}(h(x)=h(y)) \leq P_1$, if $D(x,y) \geq cR$

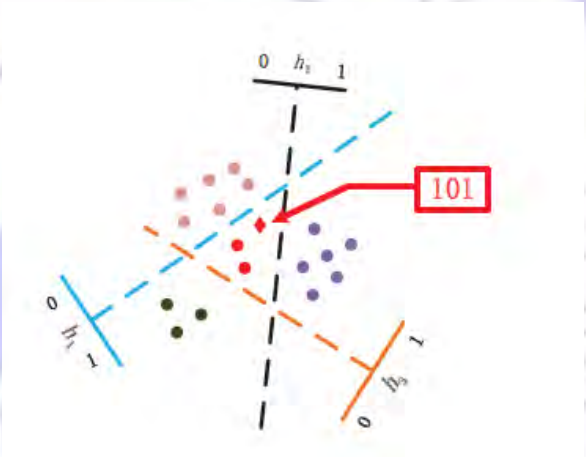

图 1 所示。位置敏感哈希

这里 D(.,.)是在原先空间 R^d 中的一个距离函数。显然，一个家族函数 H 只有当

$c>1$且$P_1>P_2$时才是有效的。给定一个有效的LSH函数，Gionis等人已经证实了检索(1+ε)到其邻域的访问时间是有界的，其海明距离[7]为$O(n^{\frac{1}{1+\varepsilon}})$。

Charikar 定义了一个哈希函数:h∈H广泛地使用内积相似[3]。更具体一点说，就是从一个d维高斯分布N(0,1)中，选择一个具有超平面的随机向量r，哈希函数定义如下：

$$h_r(q)=\begin{cases}1, if\ (r\cdot q>0)\\ 0, if\ (r\cdot q<0)\end{cases} \quad \cdots\cdots(1)$$

在[8]当中，Goemans等人证明了对任意的数据点x和y,

$$P_r(h_r(x)=h_r(y))=1-\frac{\theta(x,y)}{\pi} \quad \cdots\cdots(2)$$

这里$\theta(x,y)=\cos^{-1}(\frac{x^T y}{\|x\|\|y\|})$是在x,y间的角度。

定义$D(x,y)=\frac{\theta(x,y)}{\pi}$,Eq.(2),很容易发现：

如果$D(x,y)\le \mathrm{R}$，那么$P_r(h(x)=h(y))\ge 1\text{-}\mathrm{R}$；

如果$D(x,y)\ge \mathrm{cR}$，那么$P_r(h(x)=h(y))\le 1\text{-}\mathrm{cR}$。

所以说，如果P1=1-R并且P2=1-cR，只要近似系数C远大于1，我们就能得到P1>P2。这些满足定义1当中的性质。

因为概率P1和P2之间的差异很小，在实际情况下，一个“放大”的过程，需要连接多个不同的输出哈希函数。因为它的广泛使用，LSH算法也被扩展到P-基准距离[5]、Mahalaonbis距离[10]和内核相似[11]中。

2.2 层叠哈希

为了尽可能快地加快图象匹配，提出的层叠哈希结构被设计成包括三层：散列查找(2.2.1节),散列重新映射(2.2.2节),散列排名(2.2.3节)。我们的方法的流程图在图2中给出。我们设计的3 层散列映射功能从粗糙到细腻，将图像表示为二进制代码,以便更快的图像匹配。此外,每层的层叠哈希采用不同的测量和过滤策略,它可能会以交叉验证的方式过滤掉一些噪声的特征点。从这个意义上说,我们提出的层叠哈希方法在3D重建中对噪声更具抗敏感能力。

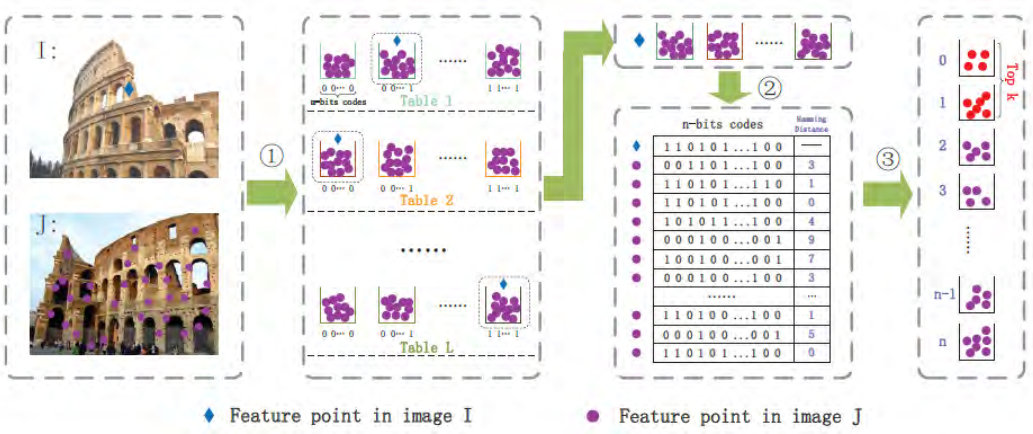

图 2。我们提出的层叠哈希方法的流程图。在图像 I 的特征点中,涉及三个步骤来找到它在图像 J 中的匹配点。第一,一个多表的短码哈希查找是用来进行粗搜索的。第二,返回的候选点将被重新映射到高维汉明空间，进行汉明距离查询计算。决赛,我们以汉明距离为关键来建立哈希表,以找到最准确的(或 K 最高的)候选集。

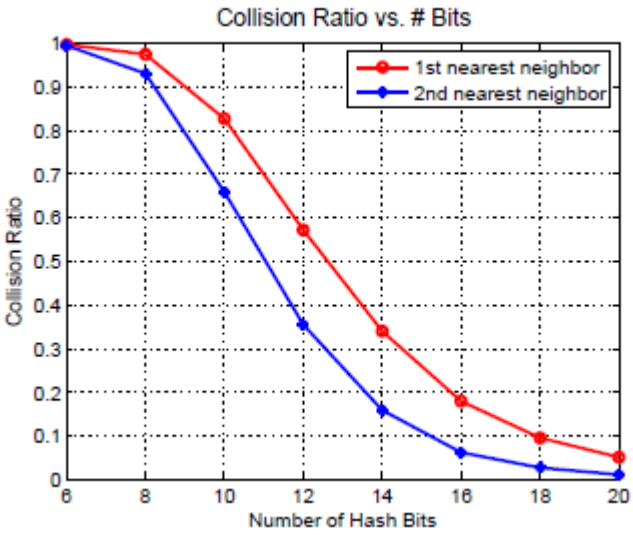

图 3。两个最近的邻点在同一个向量桶中发生冲突的比率和比特数量变化。

2.2.1 具有多表结构的哈希查找

在第一步中,我们运用短码的哈希查找进行粗搜索。特别是所有图象中的特征点用 LSH 技术嵌入 m 位二进制代码。对于图 I 中的特征点,为了其在图象 J 中的匹配点，使用 m 位二进制代码构造一个具有一组桶的查找表。图象 J 中所有点均落入相同的桶中的点 P 将被返回。以上过程称为哈希查找。

由于其复杂性是时间常数，哈希查找强调实际搜索速度。然而,使用长比特一个哈希表,哈希查找常常失败,因为汉明空间变得越来越稀疏并且只会有很少的样本落入同一个桶中。在本文中,我们采用多表策略来解决这个问题。特别的是,

我们生成 L 函数 $g_l(q)=(h_{1,l}(q),h_{2,l}(q),...,h_{m,l}(q)),l=1,2,...,L$，这里 $h_{s,l}(1\le s\le m,1\le l\le L)$ 是独立生成的，并且一致随机取自 H。总之，数据结构是以具有 m 比特的 L 哈希表来构建的，每个点 P 是被赋值给 $gl(P)$，l=1, . . . , L。

一方面, m 的值越大，越会导致在碰撞概率相似点(P1)和不同的点(P2)之间产生一个更大的差距。这种放大的好处是哈希函数更有选择性。另一方面, 如果 m 较大而 P1 较小, 这意味着 L 必须大, 以确保真正的邻居与查询点至少冲突一次。具体说，这个可能性是 $1-(1-P_1)^L$。在实践中, 我们可以在不同的应用里面选择不同的参数 m 和 L。

在三维重建的特征匹配中, 为了在图 I 中找到特征点 P 的匹配点, 我们通常需要从图像 J 中找两个最近的邻居点。一个在玩具数据上做的实验, 如图 3 所示。当比特 m 的数量增加时, 最近的(第二最近的)邻居点在查询中的同一个桶中产生冲突的比例在减少, 在 m > 20 时，它们将接近于零。与此同时, 随着 m 的增加, 汉明空间会更稀疏、将被归还候选点愈来愈少。这是非常有利于以下步骤的计算成本。为了找到一个匹配的准确性和计算成本之间的权衡, 在实验中, 我们设置 m = 8 或 10、 L = 6 来构造多个哈希表。

2.2.2 哈希重映射

在散列查找粗搜索后阶段, 通常可以进行精确搜索, 例如计算每个查询候选点之间的欧氏距离。

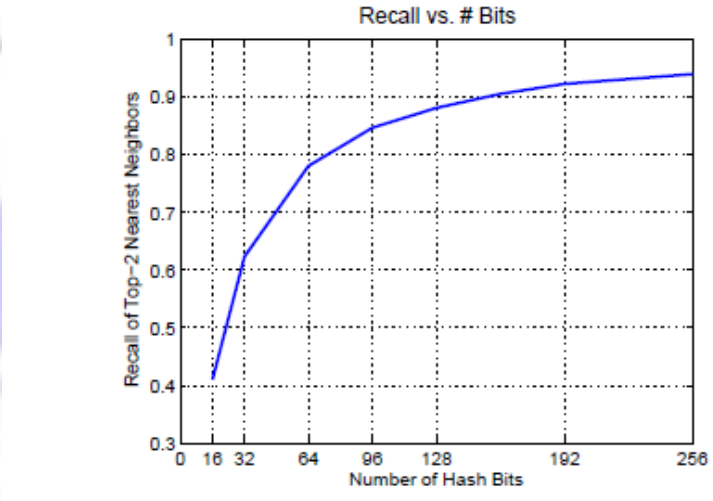

图 4。比特数变化时，前十位被返回后最近两个邻点的召回

然而,由于候选人的数量仍然很大，欧几里得距离的直接计算需要大量的计算量。为了解决这个问题,一个新的 n 位比特(n >m)哈希函数，用来重新映射到汉明空间哈希查找阶段所提供的候选点。之后,我们根据它们的汉明距离查询和顶部 k 项，对候选点进行排序，并将其选为最终的候选人(2.2.3 节中描述)。

当应用 LSH 到特征点中时,例如 SIFT 是排名性能密切相关的比特数。在 LSH 中,理论上可以保证长二进制代码导致更有识别力的哈希表达式,和更好的排名性能。我们在一个具有不同比特数的玩具数据上进行实验。当返回前十名(汉明距离)时，我们召回排名前 2 个点(欧氏距离)图 4 所示。在这篇文章中,我们将候选点重新映射到 128-d 的汉明空间。

2.2.3 哈希排列前 K 位

为了执行更精确的搜索,在第三个步骤中,我们希望在汉明空间寻找 k 个最近的邻点,然后在欧几里得空间找到它们之间的两个最近的邻点。作为最常用的方法,线性搜索找到前 k 项的时间复杂度是 O(kN)。我们提出一种基于树的数据结构称为 min-heap，来找到顶部 k 项，它的时间复杂度是 $O(\log(k)N)$ [12]。在本文中,我们提出一个基于哈希的计划来找到顶部 k 项,其时间复杂度是 O(N + k)。

详细地说，在数据库里的查询和所有点之间计算海明距离时,我们使用汉明距离作为密钥建立哈希桶。具有相同的汉明距离的点会落入相同的桶中。例如,所有汉明距离为 1 的查询点,都将落入 id 是# 1 的桶中。当我们使用 n 位汉明排名时,至少要建立 n + 1 个桶。值得注意的是，我们只需要扫描一次数据库来完成这个过程,因此时间复杂度是 O(N)。

这个过程完成后,我们访问的桶里的点的汉明查询距离是 0。如果点在这个桶的数量小于 k,然后进入下一个汉明距离查询是 1 的点。重复这个过程,直到获得点相加到 k。显然,获得的 k 个点是 k 个候选点,这个时间复杂度是 O(k)。因此,整个过程的时间复杂度是 O(N + k)。在大多数应用程序中,k 远低于 N。

前 k 个候选点被发现后,我们可以根据欧几里得距离在它们中找到两个最近的邻点,通过洛斯比测试将匹配的点保留下来[13]。

3.计算复杂度分析

我们和其他相关的图像匹配方法比较 CasHash 方法的计算复杂度。假设我们需

要执行两幅图像之间的图像匹配，图像包含 N 个 SIFT 要点。我们用 TE 和 TH 分别表示计算欧氏距离和汉明距离的耗费时间。现代的 CPU,它只需要一个 CPU 周期来计算在汉明空间中海明距离为 128 位的哈希码,因此 TH 比 TE 小得多。

对于蛮力匹配,线性搜索所需 N^2 倍欧氏距离的计算,因此,时间成本大约是 $O(T_E \cdot N^2)$。在基于 Kd-树的近似最近邻搜索,平均单个搜索复杂度是 O(logN),总时间成本是 $O(T_E \cdot N\log N)$。基于匹配的 LDAHash 也需要详尽的搜索,但它的汉明嵌入描述符可以减少时间成本 $O(T_E \cdot N^2)$。

基于匹配的 CasHash 方法有三层结构,平均将有 $LN/2^m$ 个点通过散列查找阶段。这些点的海明距离计算，每次查询需要的耗费为 $O(T_H \cdot LN/2^m)$。另一个主要的时间消耗就是对前 k 个候选点进行最后的搜索,这要耗费的代价为 $O(T_E \cdot k)$。一般来说,L 和 k 都不超过 10 。在查找阶段计算哈希的复杂度为 $O(dmLN)$，而在重新映射阶段则是 $O(ndLN/2^m)$。在实践中,查找和重新映射阶段可以实现离线。因此,整体计算复杂度是 $O(T_H \cdot LN^2/2^m + T_E \cdot Nk + dmLN + ndLN/2^m)$。理论上,m 越大匹配越快,但精度较低。在我们的实验中,在保证精度相当的情况下，m = 8 能够实现较快的速度。总的来说,很容易实现比 Kd-tree 基于图像匹配快 10 倍或更多，并且性能相当。广泛的实验提供了令人鼓舞的结果。

	boat	trees	ubc	wall
Brute	13.959s	28.003s	2.033s	18.326s
KDTree	1.278s	2.010s	0.566s	2.044s
LDAHash	0.924s	1.861s	0.144s	1.234s
CasHash	**0.128s**	**0.223s**	**0.029s**	**0.173s**

表 1。比较筛选匹配时间点的四个算法:蛮力算法,Kd-tree,LDAHash 和 CasHash 算法。

4.实验

在我们的实验中,SIFT 特征描述符被提取出来，用来进行两两图像匹配。为了评估我们的提议 CasHash 方法的速度和精度,我们把三个匹配策略相比,即强力匹配,Kd-tree 匹配[15],和 LDAHash 匹配[19]。我们使用打包机[18]来进行要点匹配及产生稀疏的点云。PMVS2 包[6]用于生成致密重建结果，使之可视化。三维重建结果的几个数据集,可以从专家收集的高质量的图像集到低质量图像集中进行下载。

所有的实验都在一个具有 Ubuntu 13.04 操作系统、i7 - 2600 CPU 和 8 gb 内存空间的 PC 机桌面运行。不管是并行计算还是 GPU 加速技术，都被用于我们的实验,以确保公平的比较。

4.1 牛津大学数据库中的结果

我们使用标准的牛津数据集[14]来评估我们的 SIFT 重点匹配性能，并与蛮力算法进行比较匹配,Kd-树匹配和 LDAHash 匹配。这个数据集包含具有不同几何和光度转换的图像,包括模糊、视角变化、缩放和旋转。图像之间的单应性矩阵对,因此在第一图像中对于每个 SIFT 关键点,其在第二图像中的预期位置是可以计算的。对于第二图象中的任何 SIFT 关键点,如果它的距离到预期位置低于某个阈值,然后我们假设这两个要点组成一对真实匹配。在我们的实验中,我们将距离阈值设置为 2.5。

具有不同方法的 SIFT 关键点匹配的时间被测量出来并且标记在表 1 中。由于有限的空间,只有四套结果记录在这里。在图 5 中,正如所预测的一样，我们观察到蛮力匹配达到最佳性能,因为其搜索策略是线性检查所有要点。然而,如表 1 所示，在所有方法中，它还需要最长的匹配时间。Kd-tree 匹配提供了大约 10 倍的匹配速度，并且仍然保持令人满意的性能。比起 Kd-tree 算法，LDAHash 匹配稍快,但会导致一个严重的回忆与 1-precision 下降曲线。我们提出的 CasHash 算法匹配达到 Kd-tree 的可比性能，并且只需要 1/10 Kd-tree 匹配时间。

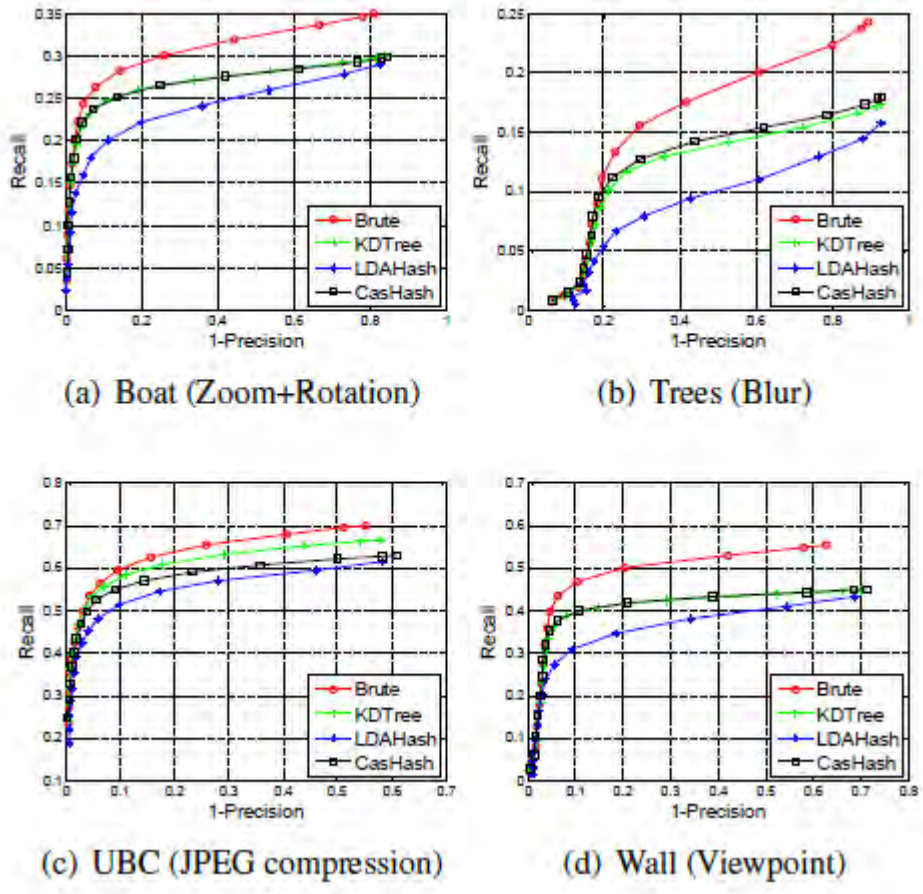

图 5。回忆与 1-precision 曲线在牛津数据库。

从上面的实验结果,我们可以看到,散列技术是加速关键点匹配一种有效的方法。然而,通过哈希将速度提高经常导致性能恶化,这一点已出现在许多其他应用程序里。LDAHash 匹配的性能在我们的实验中也证明了这一结论。然而,具有精心设计的级联结构,我们的 CasHash 匹配不仅极大地提高了匹配的速度,也保证了匹配性能堪比 Kd-tree 算法。这可能是受益于我们的级联结构，多个距离度量框架有助于过滤掉噪音对。

4.2 清华数据库中的结果

为了最后的 3 d 重建而评估不同的关键点匹配算法的有效性,有必要比较结果与真实的 3 d 模型的一致性。在我们的实验中,我们使用两个具有 3D 地面激光扫描仪获得的数据的数据集,分别包含 102 张和 193 张照片。

我们这些数据集上运行不同的匹配方法，使用打包机 PMVS2 一起生成重建结果。我们的匹配方法的三维重建结果如图 7 所示。匹配时间统计数据记录在表 2

进行比较。

我们使用相同的评价方法[17]。为简单起见,用 G 表示真实模型，R 表示候选模型 .评估精度就是 R 和 G 之间的亲密度，完整性即评估 R 对 G 建模的多少。

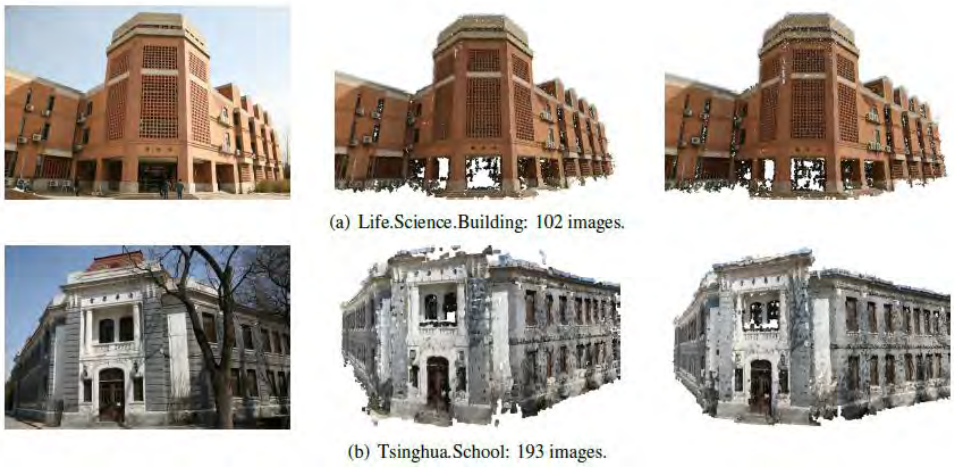

图 6。"Life.Science.Building"和"Tsinghua.School"的三维重建结果数据集。

左:真实的照片,中间:CasHash-8Bit;右:CasHash-10Bit。

Method	Dataset-Life.Science.Building		
	T_{Match}	Speed-Up	Points
Brute	76655s	1.00×	457907
KDTree	4689s	16.35×	435275
LDAHash	4782s	16.03×	490146
CasHash-8Bit	639s	119.96×	**497944**
CasHash-10Bit	**316s**	**242.58**×	407562
Method	Dataset-Tsinghua.School		
	T_{Match}	Speed-Up	Points
Brute	46488s	1.00×	680966
KDTree	4877s	9.53×	683461
LDAHash	2522s	18.43×	684524
CasHash-8Bit	402s	115.64×	676223
CasHash-10Bit	**243s**	**191.31**×	**697366**

表 2。清华数据库上重建性能。

R 中的点和它们在 G 中各自最近的点被计算出来，用来测量准确性。精度在 X % 的定义是最小距离值,所以在 R 中 X %点在 G 的这个距离之内。同样,完整性是通过 G 中的点和它们在 R 中最近的点之间的距离来衡量的。完整性在 X %上的定义

是最小距离点的值,G 中 X%的点包含在 R 的距离中。准确性和完整性评价结果显示在表 3 和表 4。

从表 3 我们可以看到，在数据集“Life.Science.Building”中，我们的 CasHash-8Bit 匹配策略明显比所有其他方法好。至于数据集“Tsinghua.School”,对 Kd-tree 匹配方法的精确统计是优于其他方法的。如表 4 所示的两个数据集所示，CasHash-8Bit 在完整的统计结果中始终优于其它。

根据上述实验结果,在大多数情况下，我们 CasHash 匹配策略比 Kd-tree 和 LDAHash 方法更能够实现类似的(甚至略好)的性能。

4.3 在 Flicker 数据库中的结果

随着在互联网上上传图片数量的日益增多,我们可以仅基于这些 web 图像，重建著名旅游景点的 3 D 场景。我们从 flickr 网站收获四个著名的地标的照片集:泰姬陵,自由女神像,巴黎圣母院和罗马圆形大剧场。

我们在图 7 中展示我们的重建结果,在表 5 中的是对所有匹配的方法的重建统计的比较。如表 5 所示,与常用 Kd-tree 匹配策略相比，我们的 CasHash 匹配策略达到对 SIFT 关键点匹配 10 倍以上的提升速度。同时,我们重建性能与其他关键点匹配方法保持在同等水平,考虑用最后密集重建阶段点集的总量作为评价指标。

5。结论

在本文中,我们提出了一种级联散列方法加快图像匹配。大量实验表明,我们的方法在成百上千的实验中对图象匹配时的速度都能够比蛮力匹配更快,在保持相当准确性的同时，甚至是比 Kd-tree 匹配快十倍或者更多。

Method	Life.Science.Building									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Brute	4.95e-3	9.88e-3	1.49e-2	2.01e-2	2.58e-2	3.25e-2	4.17e-2	5.86e-2	1.05e-1	2.93
KDTree	4.68e-3	9.23e-3	1.38e-2	1.86e-2	2.41e-2	3.10e-2	4.07e-2	5.80e-2	1.03e-1	2.95
LDAHash	4.84e-3	9.56e-3	1.42e-2	1.92e-2	2.46e-2	3.14e-2	4.11e-2	5.72e-2	1.01e-1	2.94
CasHash-8Bit	5.39e-3	1.06e-2	1.58e-2	2.11e-2	2.68e-2	3.38e-2	4.40e-2	6.12e-2	1.05e-1	2.92
CasHash-10Bit	4.75e-3	9.47e-3	1.43e-2	1.96e-2	2.56e-2	3.30e-2	4.31e-2	6.10e-2	1.07e-1	2.92
Method	Tsinghua.School									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Brute	2.98e-3	5.93e-3	8.92e-3	1.21e-2	1.61e-2	2.13e-2	2.90e-2	4.17e-2	7.11e-2	2.25
KDTree	4.37e-3	8.98e-3	1.39e-2	1.91e-2	2.50e-2	3.19e-2	4.04e-2	5.29e-2	8.00e-2	1.93
LDAHash	2.99e-3	6.13e-3	9.61e-3	1.36e-2	1.85e-2	2.44e-2	3.23e-2	4.45e-2	7.35e-2	1.53
CasHash-8Bit	2.85e-3	5.70e-3	8.67e-3	1.21e-2	1.63e-2	2.19e-2	2.94e-2	4.11e-2	6.78e-2	1.43
CasHash-10Bit	3.87e-3	7.98e-3	1.25e-2	1.76e-2	2.34e-2	3.05e-2	3.97e-2	5.32e-2	8.18e-1	1.78

表 3.清华数据库中 3D 重建的精确度

Method	Life.Science.Building									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Brute	1.52e-2	2.06e-2	2.59e-2	3.18e-2	3.91e-2	4.93e-2	6.44e-2	8.96e-2	1.52e-1	1.95
KDTree	1.53e-2	2.06e-2	2.58e-2	3.15e-2	3.87e-2	4.88e-2	6.39e-2	9.04e-2	1.61e-1	1.90
LDAHash	1.50e-2	2.02e-2	2.52e-2	3.06e-2	3.75e-2	4.68e-2	6.07e-2	8.47e-2	1.45e-1	1.75
CasHash-8Bit	1.55e-2	2.09e-2	2.59e-2	3.15e-2	3.85e-2	4.82e-2	6.27e-2	8.71e-2	1.48e-1	1.52
CasHash-10Bit	1.56e-2	2.12e-2	2.66e-2	3.26e-2	4.00e-2	5.02e-2	6.58e-2	9.17e-2	1.58e-1	1.68
Method	Tsinghua.School									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Brute	1.33e-2	1.72e-2	2.05e-2	2.39e-2	2.77e-2	3.22e-2	3.82e-2	4.73e-2	6.62e-2	8.26e-1
KDTree	1.50e-2	1.94e-2	2.33e-2	2.71e-2	3.13e-2	3.61e-2	4.22e-2	5.12e-2	6.85e-2	8.28e-1
LDAHash	1.35e-2	1.75e-2	2.10e-2	2.45e-2	2.83e-2	3.28e-2	3.86e-2	4.73e-2	6.63e-2	7.85e-1
CasHash-8Bit	1.34e-2	1.73e-2	2.07e-2	2.41e-2	2.79e-2	3.23e-2	3.79e-2	4.65e-2	6.48e-2	9.63e-1
CasHash-10Bit	1.47e-2	1.91e-2	2.31e-2	2.72e-2	3.17e-2	3.71e-2	4.41e-2	5.46e-2	7.52e-2	7.11e-1

表 4.清华数据库中 3D 重建的完整性

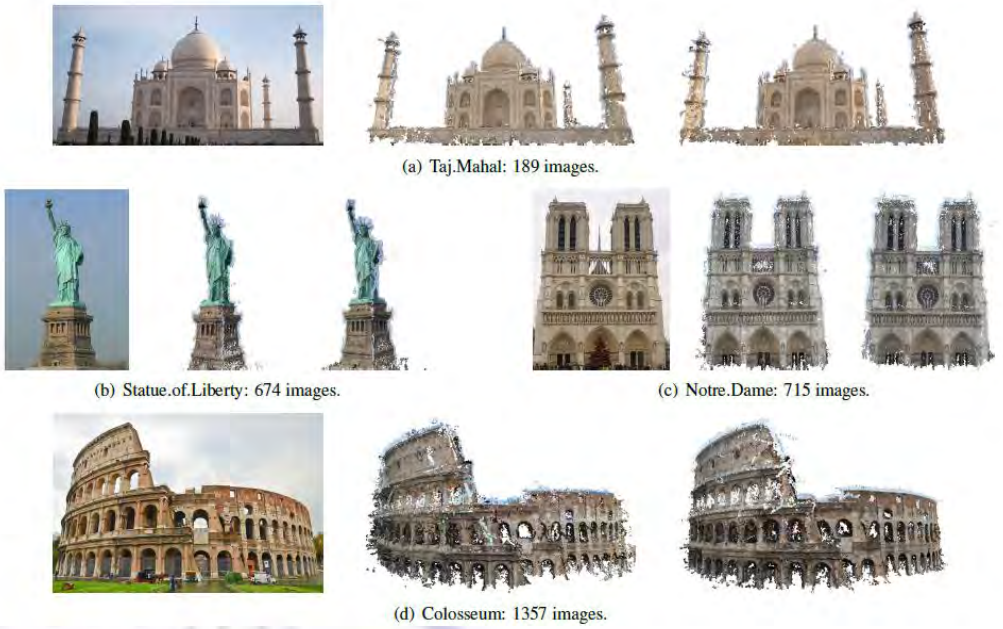

图 7。在 Flickr 上三维重建结果数据集"泰姬陵"、"自由女神像","巴黎圣母院"和"罗马圆形大剧场"。

左:真实的照片,中间:CasHash-8Bit;右:CasHash-10Bit。

Method	Taj.Mahal			Statue.of.Liberty		
	T_{Match}	Speed-Up	Points	T_{Match}	Speed-Up	Points
Brute	52575s	1.00×	**124038**	30608s	1.00×	88001
KDTree	5136s	10.24×	119165	9765s	3.13×	98674
LDAHash	1774s	29.64×	120353	874s	35.02×	123274
CasHash-8Bit	301s	174.67×	118331	358s	85.50×	203587
CasHash-10Bit	**183s**	**287.30**×	116224	**231s**	**132.50**×	**246206**
Method	Notre.Dame			Colosseum		
	T_{Match}	Speed-Up	Points	T_{Match}	Speed-Up	Points
Brute	396729s	1.00×	358121	12307s	1.00×	**540308**
KDTree	60663s	6.54×	347056	2430s	5.06×	445774
LDAHash	13136s	30.20×	413348	851s	14.46×	492040
CasHash-8Bit	2266s	175.08×	**484960**	222s	55.44×	393408
CasHash-10Bit	**1354s**	**293.01**×	400673	**196s**	**62.79**×	512508

表 5。重建统计四种匹配方法:蛮力,Kd-tree,LDAHash 和 CasHash。注意:词汇树技术已经被用于数据集"斗兽场"加速图像匹配[16]。

6. 感谢

作者要感谢詹益胡教授和魏高博士有益的建议。这个工作是由 973 计划项目部分支持的，批准号为 2010 cb327905 和国家自然科学基金，批准号为 No. 61332016。

引用

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In International Conference on Computer Vision (ICCV), 2009.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). Computer vision and image understanding,110(3):346–359, 2008.

[3] M. Charikar. Similarity estimation techniques from rounding algorithm.In ACM symposium on Theory of computing, pages 380–388,2002.

[4] D. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher. Discretecontinuous optimization for large-scale structure from motion. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2011.

[5] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni.Locality-sensitive hashing scheme based on p-stable distributions.In SCG,pages 253262,2004

[6]Y.Furukava and J.Ponce.Accurate,dense,and robust multiview stereopsis .Pattern Analysis and Machine Intelligence,IEEE Transactions on,32(8):1362,2010.

[7]A.Gionis,P.Indyk,R.Motwani,et al.Similartity search in high dimensions via hashing.In VLDB,wolume 99,pages 518-529.1999.

[8]M.X.Goemans and D.P.Williamson.Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming.Journal of the ACM(JACM),42(6):1115-1145,1995.

[9]P.Indyk and R.Motwani.Approximate nearest neighours : towards removing the curse of dimensionality,In ACM symposium on Theory of computing(STOC),pages 604-613,1998.

[10]P.Jain,B.Kulis,and K.Grauman.Fast image search for learned metrics.In IEEE Conference on Computer Vision and Pattern Recongnition(CVPR),2008.

[11]B.Kulis and K.Grauman.Kernelized locality-sensitive hashing for scalable image search.In IEEE International Conference on Computer Vision(ICCV),2009.

[12]C.E.Leiserson,R.L.Rivest,C.Stein,and T.H.Cormen.Introduction to algorithms.The MIT press,2001.

[13]D.Lowe.Distinctive image features from scale-invariant keypoints.International journal of computer vision,60(2):91-110,2004.

[14] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10):1615–1630, 2005.

[15] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In VISAPP (1), pages 331–340,2009.

[16] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2006.

[17] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski.A comparison and evaluation of multi-view stereo reconstruction algorithms.In Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on, volume 1, pages 519–528. IEEE,2006.

[18] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d.In ACM transactions on graphics(TOG),volume 25,pages 835-846.ACM,2006

[19]C.Strecha,A.M.Bronstein,M.M.Bronstein,and P.Fua.Ldahash:Improved matching with smaller descriptors.IEEE Transactions on Pattern Analysis and Machine Intelligence,34(1):66-78,2012.