原 ORBSLAM2学习（三）：DBoW2理论知识学习

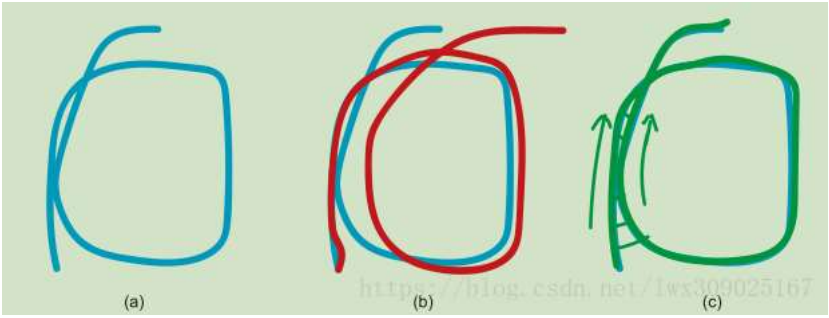
2018年05月31日 17:23:00 Mega_Li 阅读数 1462 更多

0 前言

开始ORBSLAM2中DBoW2的学习，首先说一下DBoW2引入的背景和原因。

VSLAM中由于是通过计算帧间运动得到全局运动信息，不可避免的会产生累计误差，长期估计的结果将不可靠。那如何应对这个问题？如果机器人经过的地方，且能够识别出这是“曾经的偶遇”，那么即使此时通过SLAM计算得到的位置有大的偏差，也能够有信心地把它修改为上一次经过这里时

如下图所示，（a）为真实运动路径，观察机器人在后期有回到之前经过地方的表现；（b）中红色路径是单纯通过帧间运动估计得到的全局路径影响十分明显；（c）则是利用上面说的思想对全局路径“纠偏”后的结果。



基本原理是这样，可实现起来并不简单，VSLAM中把该部分内容称为回环检测。由于输入信息主要是图像，回环检测的基础是首先机器人要能够将图像和曾经某处图像的“相似性”。一般来说，环境变化不大时，图像越为相似，是同一场景的可能性越高。

1 DBoW2简介

DBoW2是一位学术大神的开源项目，github地址为https://github.com/dorian3d/DBoW2，它能够快速稳定地计算两幅图像之间的相似程度。的学术论文是《Bags of Binary Words for Fast Place Recognition in Image Sequences》，该篇文章的内容包含两部分：vocabulary tree（字典）像相似度的计算+基于此进行回环检测的策略，DBoW2主要对应前部分。DBoW2是一个开源C++库，可以将图像转换为bag-of-word的表示（词袋模用特征提取算法（可选取）提取图像的特征描述子，然后根据一定的策略将这些特征描述子聚类为“单词”，并使用树的形式组织，得到一个vocabul后将利用vocabulary tree将图像转换为{单词，权值}的向量表示，通过计算向量间距离的方式计算图像之间的相似性。下面就DBoW2中的相关内容做

2 DboW2建立vocabulary tree

2.1 Kmeans++算法

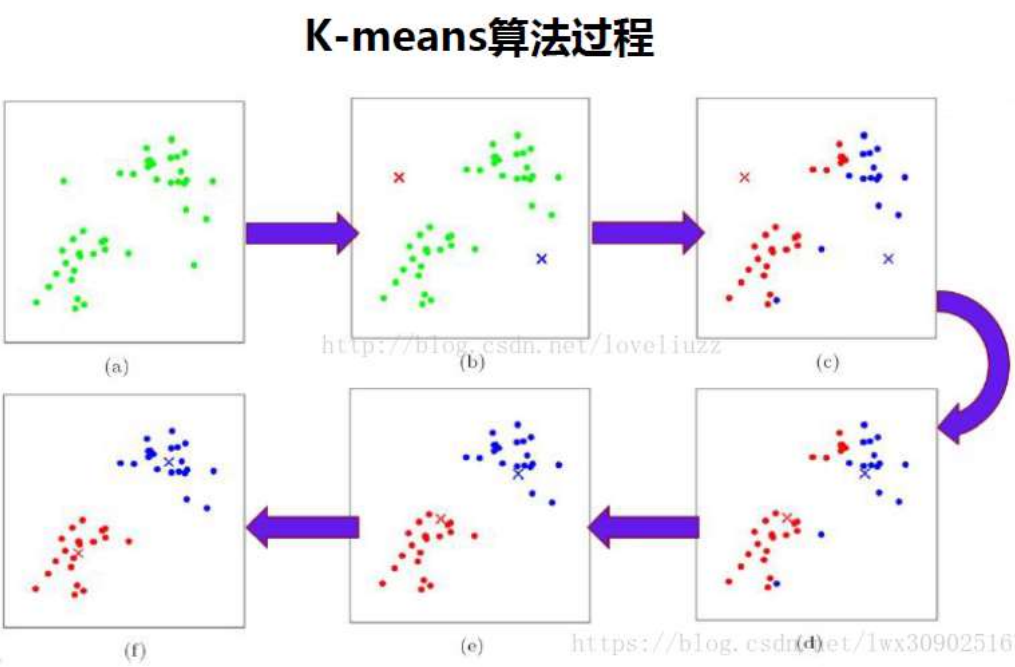
该过程涉及到聚类操作，使用了Kmeans++算法，该算法基于Kmeans算法改进而来。Kmeans算法也称为K-平均或者K-均值，是一种广泛使用的类算法，算法基本步骤如下：

step1.在样本集中指定k个初始化中心a1,a2,...,ak

step2.对于每个样本Xi，使用某种距离定义计算其与各个初始化中心的距离，并且选择距离最小的作为其归属子集

step3.对于k个子集，计算其样本平均值作为该子集新的中心ak

step4.重复step2-3，直至满足终止条件（迭代次数/中心变化大小等）



算法优点：流程简单；处理大数据集时算法具有较好的伸缩性和高效率；子集近似于高斯分布时效果不错

算法缺点：种类k需要人为指定；对于初始中心点的选取比较敏感；噪声（离群值）对结果影响较大

Kmeans++算法是基于Kmeans改进而来，主要改进点在于中心点的初始化上，不像原始版本算法的随机生成，它通过一些策略使得k个初始中心尽量地远，以期获得这些中心点具有更好的代表性，有利于后面的分类操作的效果。

Kmeans++算法中中心点初始化的流程如下：

step1.从n个样本中随机选取一个点作为第一个中心点；

step2.计算样本中每个点和距离它最近的中心点之间的距离Di，根据策略选择新的中心点

step3.重复step2直至得到k个中心点

step2中策略具体如下：n个Di组织为序列D，为避免噪声影响不直接选取其中最大值对应的点，而应该选取较大值对应的点。那么如何选取较大值Di对应的线段依次连接起来，然后随机的在这整条线段上选择一个点，从概率上来说该点落到最长线段（对应最大的Di）的概率最大。数学实现上就是Sum(D) = D1+...+Dn，然后生成位于[0,1]之间的随机数random，用Sum(D)*random得到阈值f，之后从D1开始累加保存结果到CurSum直至CurSu个加入的Di对应的点就是新的中心点。

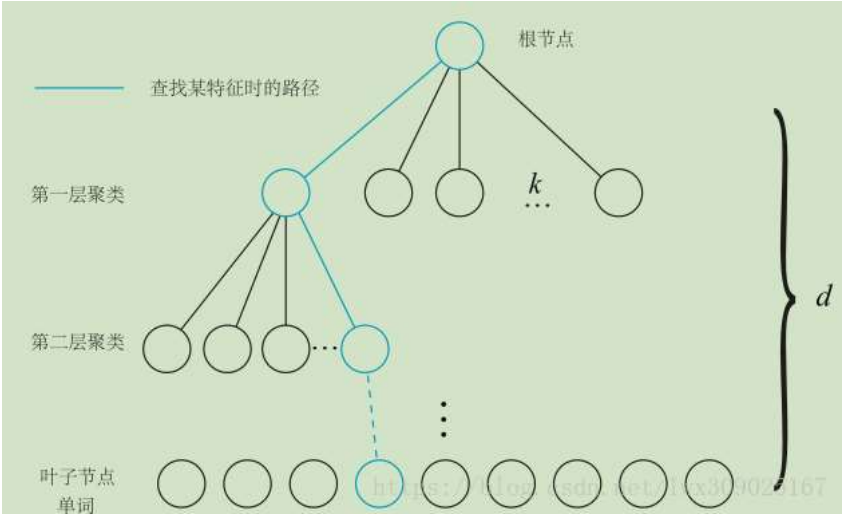
2.2 建立vocabulary tree

给定了样本图像集合后，DBoW2首先根据指定了的特征提取方法得到N（非常大）个特征和对应的特征描述子，使用Kmeans++聚类方法做如下

step1.在根节点将样本聚为k类

step2.对第一层的每个节点，把属于该节点的样本再聚为k类

step3.重复step2，最后得到叶子层，每个叶子对应一个word，最终得到的树就是vocabulary tree，如下图所示。



图中的树能够容纳k^d个单词。为什么要组织为树的形式？一个原因在于它建立了节点之间的“父子”关系，父节点下面的子节点相似度更高。假个节点，想要确定它和已知的节点中哪个最为相似，使用穷举法需要比较K^d次；利用vocabulary tree，我们只需要从根节点开始逐层地和各个节点对似度最高的那个节点下方的分支作为新的分支继续做对比，则只需比较k*d次，计算速度大大提升。

3 利用vocabulary tree计算图像相似度

3.1 TF-IDF

vocabulary tree建立了，对于两幅图像如何利用它计算其相似度？

首先我们需要将图像转换为单词表示的形式，基本操作为提取得到图像的特征，然后对每个特征计算得到它的单词归属。这样图像就变成了一个类w2,....wn)的序列，其中wn为单词的索引。

不过到这里又引出新的问题，单词那么多，难道它们的重要性都是一致的吗？一方面来说，如果图像中属于某一类单词的特征很多，那我们一般认的重要性会更大。另一方面来说，如果某个单词在我们构造的vocabulary tree中出现频率很低（生僻字），那么如果图像中出现了该单词，则这个单词应该大一些（少见）。也就是涉及到了单词的权值计算问题，DboW2使用了TF-IDF（Term Frequency-Inverse Document Frequency）来计算单词

单词的IDF部分权值在建立vocabulary tree时已经确定，假设字典中所有特征数量为n，单词wi中具有的特征数量为ni，则该单词的IDF权值IDFi=词TF部分权值由参与比较的图像确定，图像A中所有特征数量为m，单词wi包含的特征数量为mi，则该部分权值TFi=mi/m。单词wi总的权值等于TFi *

3.2 计算图像间相似度

增加了权值后，一幅图像A利用字典转换后得到的序列为

$$A=\{(w1,weight1),(w2,weight2),...,(wn,weightn)\}$$

我们把它看做向量Va，则图像B可转换得到向量Vb，计算图像间相似性问题转换为计算两个向量之间差异的问题。类似于定义范数，计算差异的标很多，DBoW2中使用了如下的计算形式：

$$s\left(\boldsymbol{v}_A-\boldsymbol{v}_B\right)=2\sum_{i=1}^{N}\left|\boldsymbol{v}_{Ai}\right|+\left|\boldsymbol{v}_{Bi}\right|-\left|\boldsymbol{v}_{Ai}-\boldsymbol{v}_{Bi}\right|.$$

有关原理部分基本上记录完毕，不过实际代码实现中有很多的细节内容需要考虑，DBoW2的代码分析放到下一篇博客中吧。

4 参考博客

[1] https://www.cnblogs.com/jian-li/p/5664559.html

[2] https://blog.csdn.net/loveliuzz/article/details/78783773

[3] https://blog.csdn.net/u012102306/article/details/52212870

[4]http://www.cnblogs.com/jian-li/p/5666556.html