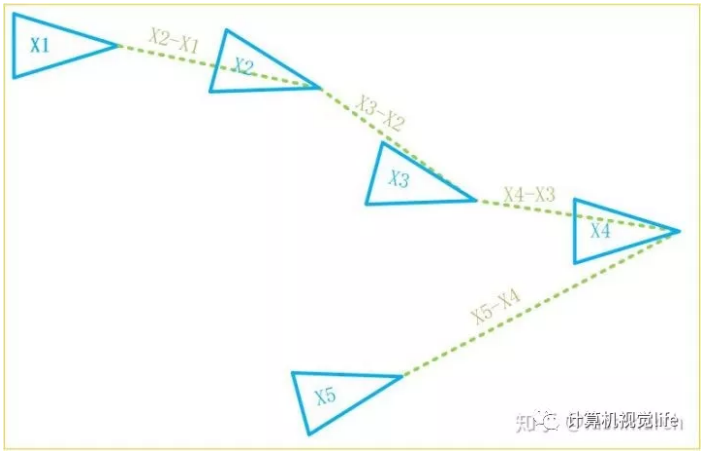
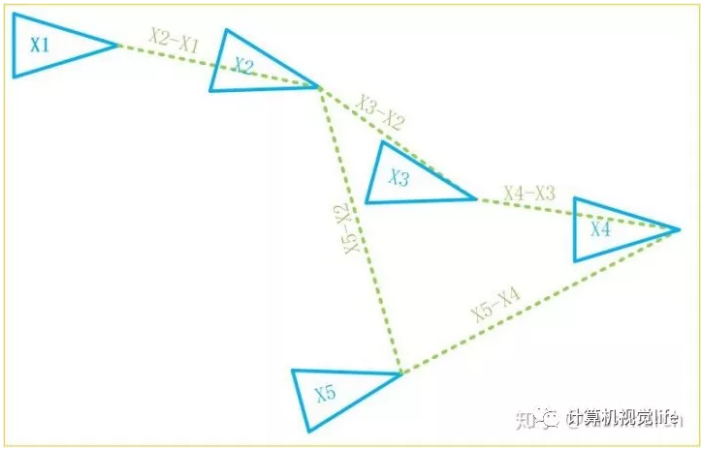
详解｜SLAM回环检测问题

在视觉SLAM问题中，位姿的估计往往是一个递推的过程，即由上一帧位姿解算当前帧位姿，因此其中的误差便这样一帧一帧的传递下去，也就是我们所说的**累积误差。**

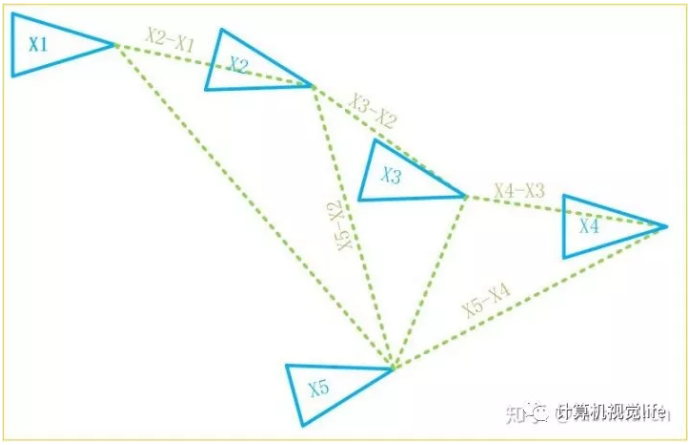
如下图所示，我们的位姿约束都是与上一帧建立的，第五帧的位姿误差中便已经积累了前面四个约束中的误差。



但此时，如果我们发现第五帧位姿不一定要由第四帧推出来，还可以由第二帧推算出来，那显然这样计算误差更小呀，因为只有两个约束的误差了嘛。像这样与之前的某一帧建立位姿约束关系就叫做**回环。**回环通过减少了约束数，起到了减小累积误差的作用。



那现在又有新的问题了，我们怎么知道可以由第二帧推算第五帧位姿呢？就像下图，可能第一帧、第三帧也可以呀。确实，我们之所以用前一帧递推下一帧位姿，因为这两帧足够近，肯定可以建立两帧的约束，但是距离较远的两帧就不一定可以建立这样的约束关系了。找出可以建立这种位姿约束的历史帧，就是**回环检测**。



那我们现在的重点就是回环检测了。其实我们完全可以把以前的所有帧都拿过来和当前帧做匹配，匹配足够好的就是回环嘛，但问题是计算量太大了，两帧匹配本来就慢，这样做的话还没有比较好的初值，需要匹配的数目又如此巨大，CPU和我们都会疯的。

但其实，任意两帧是否构成回环可以由更简单的方法做一个初步的筛选，就像一帧中有一个房子，另一帧中是一棵树，那这两帧明显关系不大嘛。通过这种方式，我们便可以对回环做出初步筛选。而这里说的房子、树就是词袋模型中的**单词。**也就是描述子的进一步抽象集合。

词袋模型

单词：差距较小的描述子的集合

字典：所有的单词

因此每一帧都可以用单词来描述，也就是这一帧中有哪些单词，这里只关心了有没有，而不必关心具体在哪里。只有两帧中单词种类相近才可能构成回环。

因此，现在利用词袋模型我们将回环检测大致分为了以下三个步骤：

1. 构建字典（所有单词的集合）$D = (\omega_1, \omega_2, \omega_3 \dots \omega_{n-1}, \omega_n)$
2. 确定一帧中具有哪些单词，用向量表示（1表示具有该单词，0表示没有）
$$F = 1 \cdot \omega_1 0 \cdot \omega_2 0 \cdot \omega_3 \dots 1 \cdot \omega_{n-1} 0 \cdot \omega_n$$
3. 比较两帧描述向量的差异

字典结构

字典由单词组成，而单词来自于描述子。并不是说一个描述子就是一个单词，而是一个单词表示了一组多个描述子，同组内的描述子差异较小。例如，描述子由256位组成，则描述子的种类便有 2^{256} 种，这个数相当大了，我们确定单词，构建字典的过程就类似于将这 2^{256} 种描述子进行分类（聚类）的过程，我们可以指定具体分成多少类。那就很像我们现实中的字典了，有厚有薄，也就是看我们分类的多少了。

那现在字典的构建也就是一个描述子聚类的过程。聚类算法也有很多了，这里采用了K-means算法，其过程也很简单：

1. 随机选取 k 个中心点：$c_1, \ldots, c_k$；

2. 对每一个样本，计算与每个中心点之间的距离，取最小的作为它的归类；

3. 重新计算每个类的中心点。

4. 如果每个中心点都变化很小，则算法收敛，退出；否则返回 1。

摘自《视觉SLAM14讲》

也可以用下图表示：

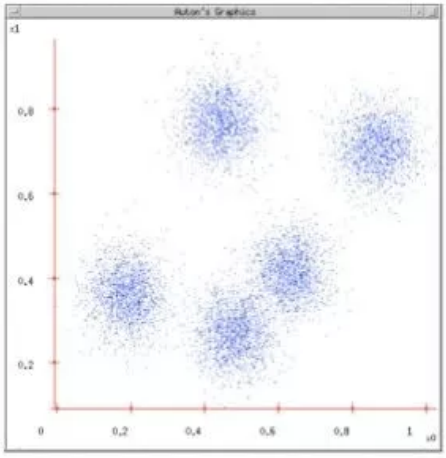


图1

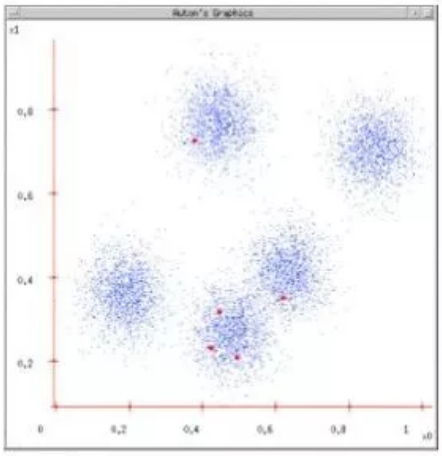


图2

摘自https://www.cnblo

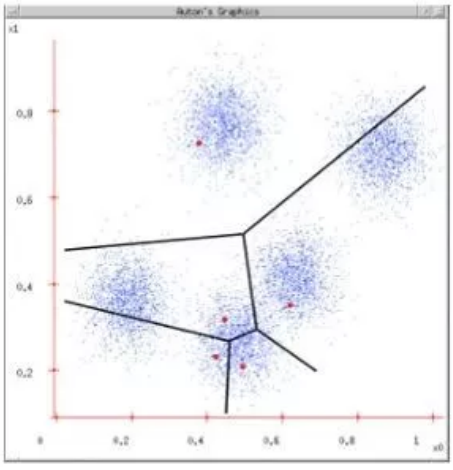


图3

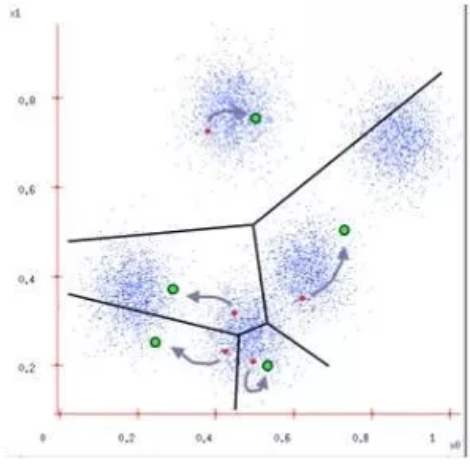
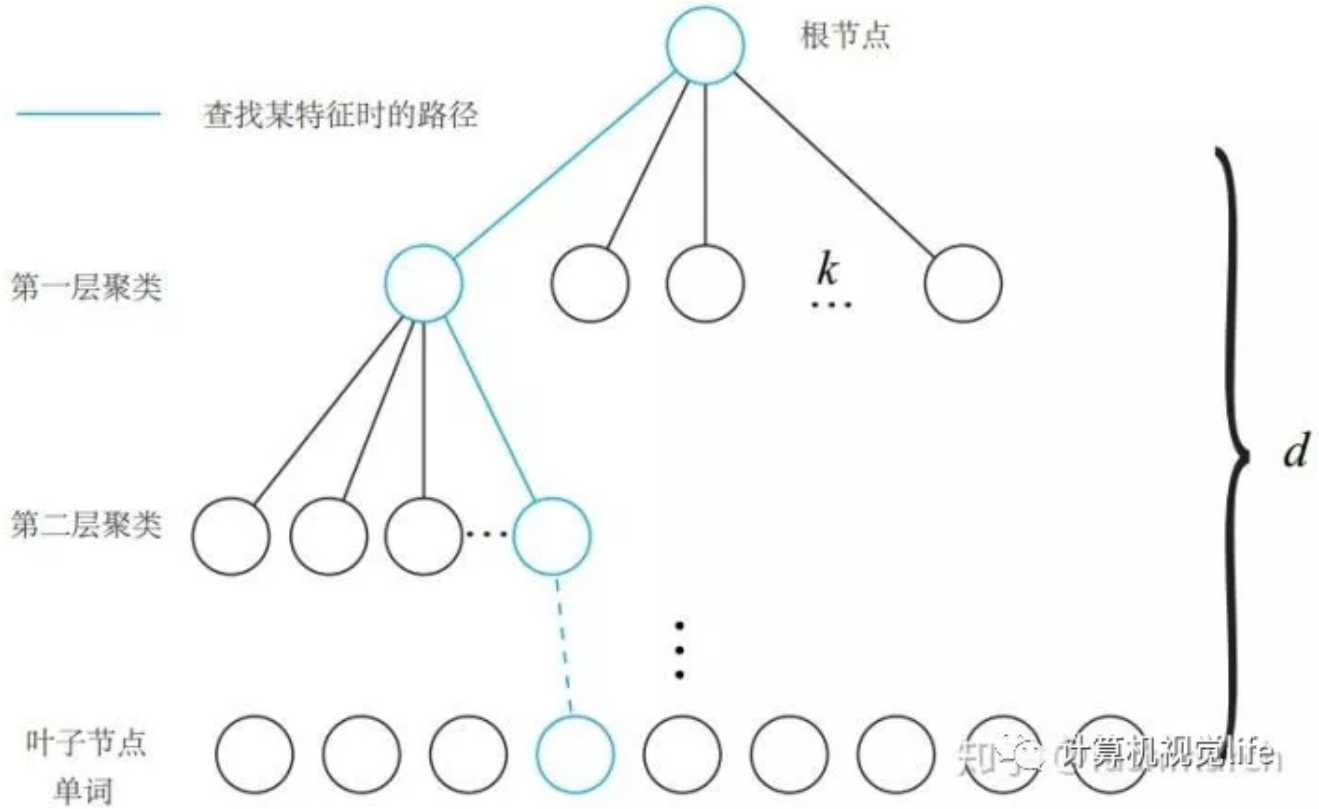


图4

gs.com/ybjourney/p/4714870.html

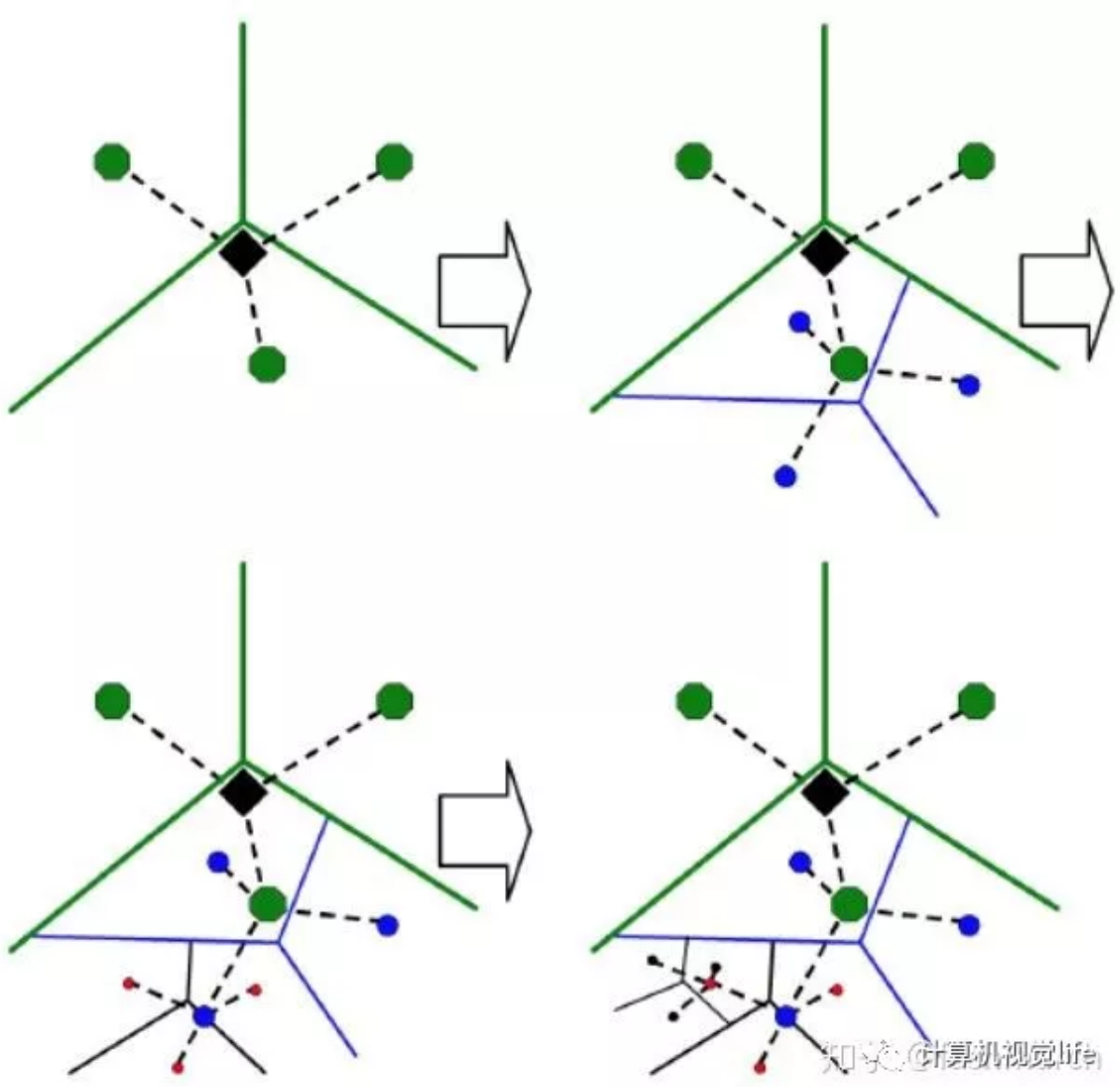
现在通过聚类我们获得了字典，但这里又有一个问题。回想我们平时查字典的过程，不可能直接打开一个一个去找吧，我们会利用目录，首字母等方法方便我们查找。这里也一样，一个个去查找速度太慢了，因此我们将字典构建成一个**K叉树**的结构来加速查找。如下图所示：



摘自《视觉SLAM14讲》

我们在每一层中都用K-means算法进行了聚类，分成了k个类，共有这样的d层（不包括根节点）因此在查找过程中，我们逐层向下，最终找到的叶节点也就是最终的单词。

还可以用下图理解：



摘自

[2]

相似度计算

现在我们有了字典，但还有一点需要注意。我们利用单词表示图像，目的是发现回环，也就是两帧图像具有较高的相似性。那其实不同的单词对这一目的的贡献性是不同的。例如，我这篇文章中有很多“我们”这个词，但这个词并不能告诉我们太多信息。而有些单词例如“回环”、“K-means”就比较具有代表性，能大概告诉我们这句话讲了什么。因此不同的单词应该具有不同的权重。

我们用两个量来描述这种权重：

- **IDF（Inverse Document Frequency）：描述单词在字典中出现的频率（构建字典时），越低越具有代表性**

$$IDF_i = ln(\frac{n}{n_i})$$

n 为所有描述子数， n_i 为该单词出现次数。ln 的作用大概是降低量级，毕竟 n 很大。

- **TF（Term Frequency）：单词在单帧图像中出现的频率，越高越具有代表性**

$$TF_i = \frac{n_i}{n}$$

n 为一帧图像中所有单词数， n_i 为一帧图像中该单词出现次数。

因此将一帧图像转化为单词表示时，我们要计算其单词的权重：

$$\eta_i = TF_i \times IDF_i$$

因此一帧图像 A 由单词 w 、及对应的权重 η 表示：

$$A = \{(w_1, \eta_1), (w_2, \eta_2), \ldots, (w_N, \eta_N)\} \triangleq v_A$$

同时，这里我们要将一帧图像中的所有的权重归一化：

$$\sum_{i=1}^{N} \eta_i = 1$$

我们在计算两帧图像的差异时，就比较对应的权重即可。

计算q和d两帧的差异（p表示范数，常取L1范数）

$$\begin{aligned}
\|\mathbf{q} - \mathbf{d}\|_p^p &= \sum_i |q_i - d_i|^p \\
&= \sum_{i|d_i=0} |q_i|^p + \sum_{i|q_i=0} |d_i|^p + \sum_{i|q_i\neq 0, d_i\neq 0} |q_i - d_i|^p \\
&= \|\mathbf{q}\|_p^p + \|\mathbf{d}\|_p^p + \sum_{i|q_i\neq 0, d_i\neq 0} \left(|q_i - d_i|^p - |q_i|^p - |d_i|^p\right) \\
&= 2 + \sum_{i|q_i\neq 0, d_i\neq 0} \left(|q_i - d_i|^p - |q_i|^p - |d_i|^p\right)
\end{aligned}$$

第二个等号进行了一个分类：只在q中有的单词、只在d中有的单词、在q、d中都有的单词。

第四个等号：因为进行了归一化，前两项都等于1

而两帧相似度的评分定义如下（越大表示越相似）：

$$\begin{aligned}
s(v_q, v_d) &= 1 - \frac{1}{2}\left|\frac{v_q}{|v_q|} - \frac{v_d}{|v_d|}\right| \\
&= \frac{1}{2} \sum_{i|q_i\neq 0, d_i\neq 0} \left(|v_{qi}| + |v_{di}| - |v_{qi} - v_{di}|\right)
\end{aligned}$$

当评分s足够大时即可判断两帧可能为回环。

回环处理

在这里简单说一下，回环出现后的处理，看得不多，如有错误还请指正。

当然回环的判断也并没有这么简单，含有很多的筛选环节，毕竟错误的回环将带来巨大灾难，宁可不要。例如某一位姿附近连续多次（ORB-SLAM中为3次）与历史中某一位姿附近出现回环才判断为回环；回环候选帧仍然要匹配，匹配点足够才为回环。

在判断出现回环后，两帧计算Sim3变换（因为有尺度漂移），也就是从历史帧直接推算当前位姿。

当我们用第m帧推算了回环帧n的位姿时，使得n的位姿漂移误差较小，但其实同时可以用第n帧来计算n-1帧的位姿，使n-1帧的位姿漂移误差也减小。因此，这里还要有一个**位姿传播。**

另外我们可以优化所有的位姿，也就是进行一个**位姿图优化**（由位姿变换构建位姿约束）。

最后，我们还可以进行一起全局所有变量的BA优化。

总结

总结来说，词袋模型通过描述一帧图像中有哪些单词，来加速寻找可能闭环帧的过程。

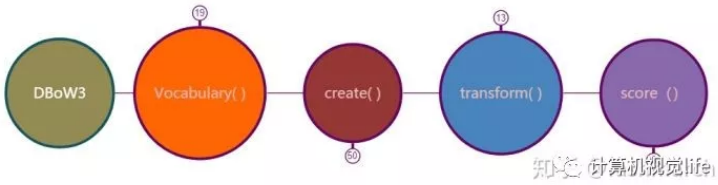
另外我们也可以利用单词加速特征点的匹配，例如帧A中的特征点a属于单词 w_1 ,那在帧B中寻找匹配时，也要去找属于单词 w_1 的特征点。

其实感觉词袋模型更近一步就是**语义**了，也就是将单词赋予真实的语义信息，同样可以起到闭环帧筛选，两帧特征点辅助匹配的作用。

代码

最后简单看一下DBoW3中的代码

其中比较重要的大概就是以下几个函数：



Vocabulary（）

构造函数，可以从文件中读取字典，由图像序列生成字典、复制字典等

其默认的树状结构为分支数k=10，深度L=5（不包括根节点）；

权重为TF_IDF;

评分类型为L1范数

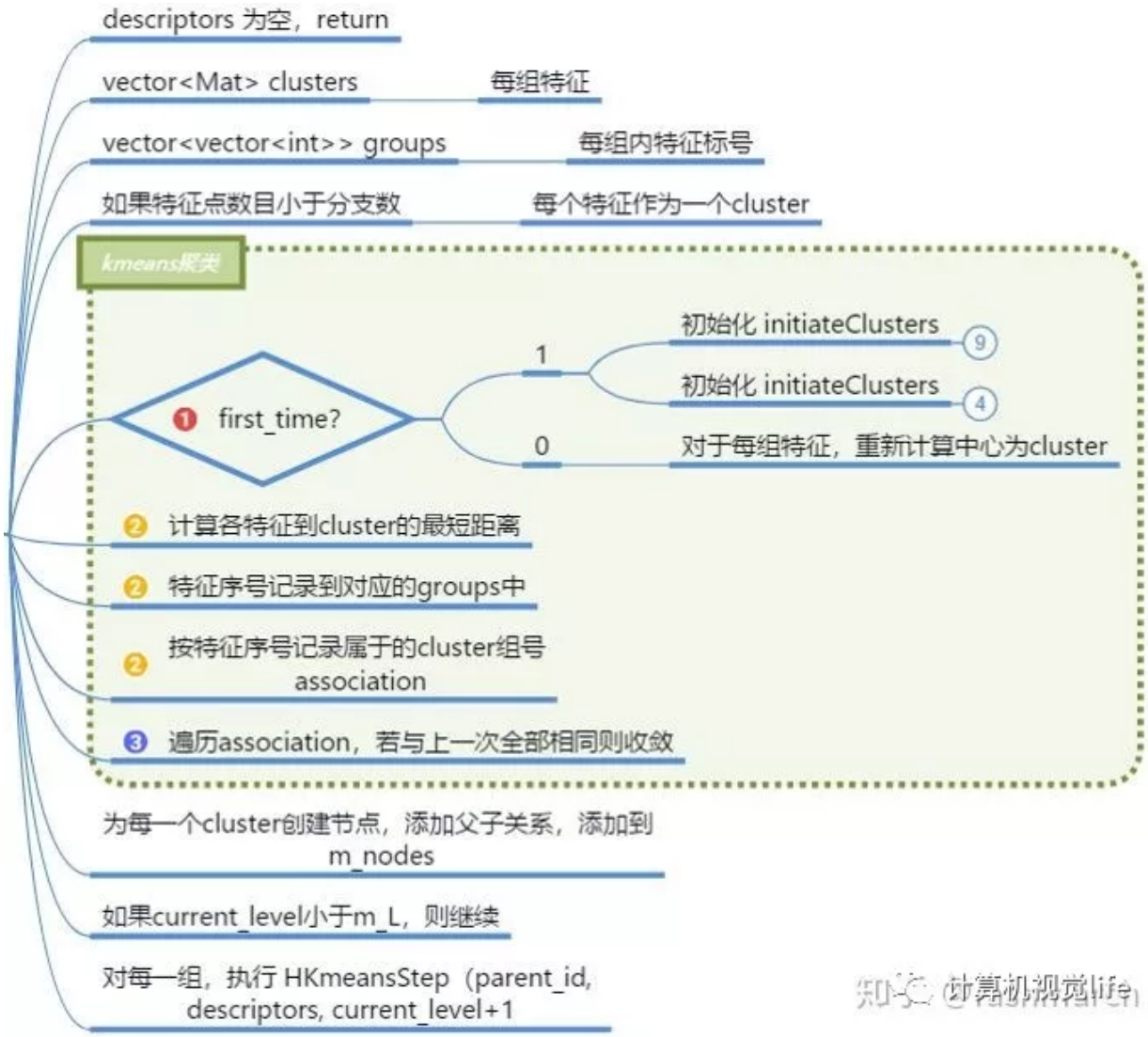
create（）

```
void Vocabulary::create (
    const std::vector<std::vector<cv::Mat> > &training_features )
{
    m_nodes.clear();
    m_words.clear();

    // expected_nodes = Sum_{i=0..L} ( k^i )
    int expected_nodes =
        ( int ) ( ( pow ( ( double ) m_k, ( double ) m_L + 1 ) - 1 ) / ( m_k - 1 ) );

    m_nodes.reserve ( expected_nodes ); // avoid allocations when creating the tree


    std::vector<cv::Mat> features;
    getFeatures ( training_features, features );


    // create root
    m_nodes.push_back ( Node ( 0 ) ); // root

    // create the tree
    HKmeansStep ( 0, features, 1 );

    // create the words
    createWords();

    // and set the weight of each node of the tree
    setNodeWeights ( training_features );

}
```


其主要步骤为：构建树、构建单词、计算权重（IDF）

构建树：

主要就是一个k_means过程，不过每层都要聚类一次，所以有一个递归。

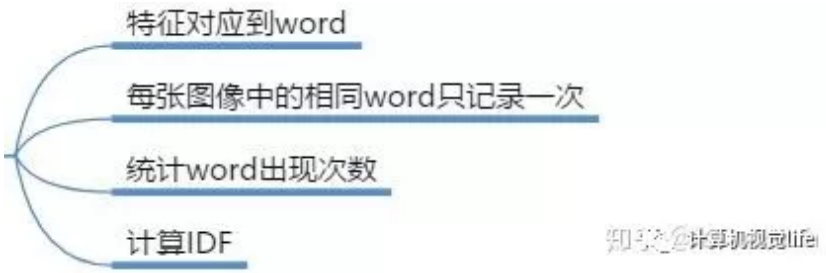


构建单词

寻找叶节点，添加为单词

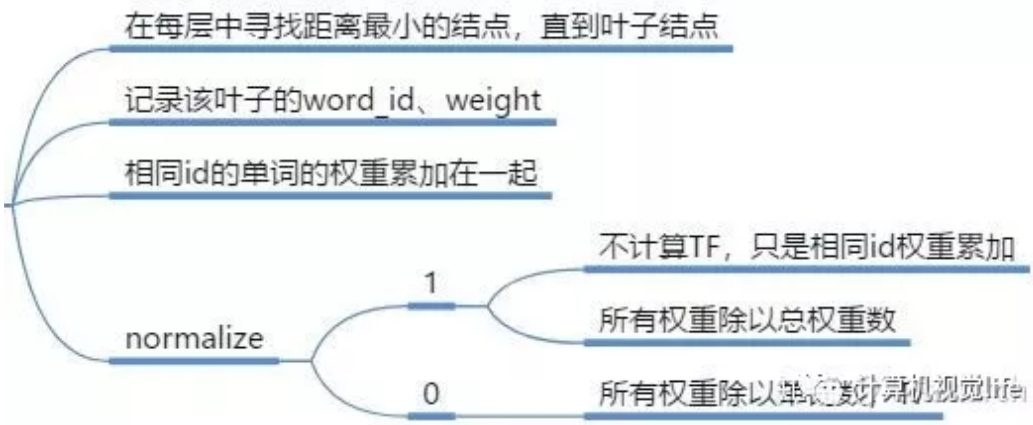
计算权重

每个字典的IDF是不变的，所以在这里计算IDF部分的权重



transform（）

该函数将一帧图像转换为单词表示



当然这里默认是要normalize的

我们再来看一下权重计算公式

$$\eta_i = TF_i \times IDF_i = \frac{n_i}{n} IDF_i$$

normalize之前我们相当于计算的是

$$n_i IDF_i$$

我们将它归一化，而没必要计算准确的TF，但也相当于综合考虑了TF和IDF。

score（）

计算两帧相似度的评分

寻找具有相同单词的权重按公式计算即可。

参考文献：

[1] 高翔 《视觉SLAM十四讲》

[2] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,”in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), vol. 2, pp. 2161–2168, IEEE, 2006.