# 回环检测的意义

前端提供特征点的提取和轨迹、地图的初值，而后端负责对所有这些数据进行优化。回环检测解决整个SLAM过程中的累计误差，形成全局一致的轨迹和地图。

回环检测的方法：基于外观(Appearance based)的几何关系。

需要理解的概念：感知偏差(Perceptual Aliasing)又称假阳性(False Position FP)：不一样的地点，看起来很像的两张图像；感知变异(Perceptual Variability)又称假阴性(False Negative FN)：一样的地点，由于其他原因导致看起来不一样的图像。

| 算法 / 事实 | 是回环 | 不是回环 |
| --- | --- | --- |
| 是回环 | 真阳性(TP) | 假阳性(FP) |
| 不是回环 | 真阴性(TN) | 假阴性(FN) |

准确率(Precision)：

$$Precision = TP/(TP + FP)$$

召回率(Recall)：

$$Recall = TP/(TP + FN)$$

注：在SLAM系统中，对准确率要求更高，对召回率相对宽松。

# 词袋模型(Bag-of-Words)BoW

需要理解：单词(Word)与字典(Dictionary)(排过序的)

单词可以理解为图像中具有的一些属性。字典可以理解为很多图像的单词的集合。

词袋模型可以理解为一幅图像中是否具有字典中的单词，有则记为1或者有几个这样的单词记为几，无则记为0，进而构成一个描述图像的向量。

# 字典

一个单词与一个单独的特征点不同，它不是从单幅图像上提取出来的，而是某一类特征的组合。字典的生成问题类似于一个**聚类**问题。

K-means(K均值)生成字典，k叉树用于加速查找。

对于一个k分支、深度为d的树，可以容纳$k^d$个单词。(DBoW库生成字典的方法)

```
DBoW3::Vocabulary vocab;
vocab.create(descriptors);        // 默认构造函数，k=10,d=5
vocab.save("vocabulary.yml.gz");
```

# 相似度计算

---

给每个单词赋予权值，以表示其对区分图像做出的贡献。

TF-IDF(Term Frequency - Inverse Document Frequency)：文本检索中常用的一种加权方式。TF的思想是，某单词在一幅图像中经常出现，它的区分度就高。另外，IDF的思想是，某单词在字典中出现的频率越低，分类图像时区分度越高。

建立字典时计算IDF：统计某个叶子节点$w_i$中的特征数量$n_i$相对于所有特征数量$n$的比例。$IDF_i = log\frac{n}{n_i}$

计算TF：某个叶子节点$w_i$在单幅图像中出现的频率。假设图像A中单词$w_i$出现了$n_i$次，而一共出现的单词次数为$n$。 $TF_i = \frac{n_i}{n}$

单词$w_i$的权重等于TF乘IDF之积：

$$\eta_i = TF_i \times IDF_i$$

引入权重后，对于某图像A，其BoW表示：

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

向量$v_A$是稀疏向量，其非零部分表示图像A含有那些单词，这些部分的值为TF-IDF的值。

给定$v_A$和$v_B$，可使用$L_1$范数形式计算两幅图像的差异：

$$s(v_A - v_B) = 2\sum |v_{Ai}| + |v_{Bi}| - |v_{Ai} - v_{Bi}|$$

利用词袋模型进行相似度计算时，可分为直接图像之间比较或图像与数据库进行比较；

图像之间直接比较：

```
// vocab为提前训练好的字典
for{int i = 0; i < image.size(); i++}
{
    DBoW3::BowVector v1;
    vocab.transform(descriptors[i], v1);
    for{int j = i; j < image.size(); j++}
    {
        DBoW3::BowVector v2;
        vocab.transform(descriptors[j], v2);
        double score = vocab.score(v1, v2);
        cout << "image " << i << " vs image " << j << " :
" << score << endl;
```

```
    }
    cout << endl;
}
```

图像与数据库之间的比较：

```
DBoW3::Database db(vocab, false, 0);
for(int i = 0; i < image.size(); i++)
    db.add(descriptors[i]);
for(int i = 0; i < image.size(); i++)
{
    DBoW3::QueryResults ret;
    db.query(descriptor[i], ret, 4);   // max result = 4,
输出匹配度最大的四组
    cout << "searching for image " << i << " returns " <<
ret << endl << endl;
}
```

# 试验分析

在实际应用中，通常使用图像与数据库进行比较的形式进行回环检测，同时增大字典规模，能够较好的提高相似度计算的精度。

## 相似性评分的处理

利用先验相似度$s(v_t, v_{t-\Delta t})$，表示某时刻关键帧图像与上一时刻的关键帧的相似性。然后其他的分值都参照这个值进行归一化：

$$s(v_t, v_{t_j})' = s(v_t, v_{t_j}) / s(v_t, v_{t-\Delta t})$$

通常如果当前帧与之前某个关键帧的相似度超过当前帧与上一时刻关键帧相似度的3倍，就认为可能存在回环。

## 关键帧的处理

用于回环检测的帧最好稀疏一些

## 检测之后的验证

1. 设立回环的缓存机制，时间上的一致性
2. 空间上的一致性，即对回环检测到的两个帧进行特征匹配，估计相机运动。然后，把运动放到之前的位姿图中，检查与之前的估计是否有很大的出入。

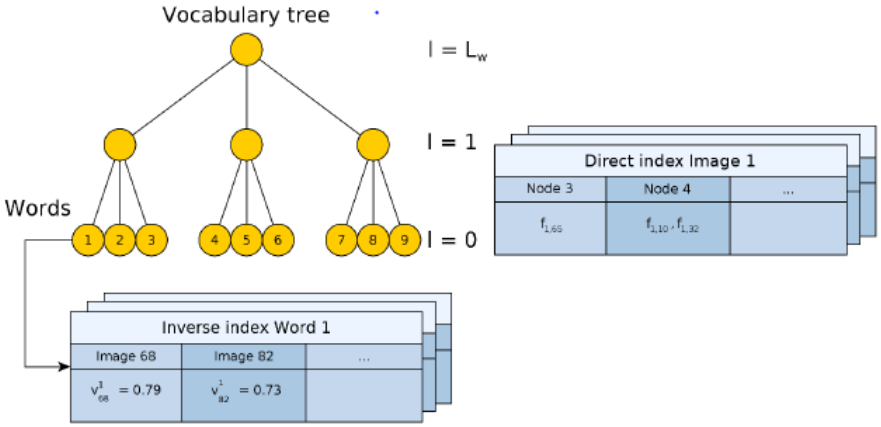


Fig. 1. Example of vocabulary tree and direct and inverse indexes that compose the image database. The vocabulary words are the leaf nodes of the tree. The inverse index stores the weight of the words in the images in which they appear. The direct index stores the features of the images and their associated nodes at a certain level of the vocabulary tree.

至此，才明白这个图的含义。

参考链接