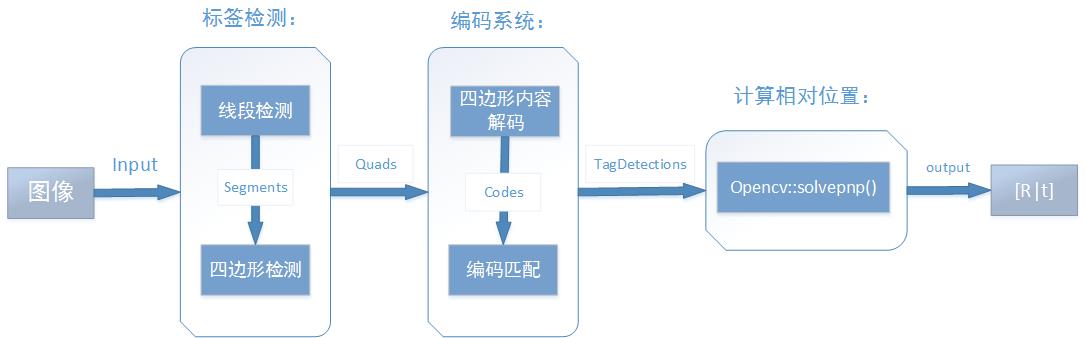
# Apriltags

识别图像中已知大小的二维码，获取每个二维码标签的ID，并估计已标定的相机与每个二维码的相对位置关系。

算法流程：



## 1. 标签检测模块

估计图像中可能的标签的位置，检测器尝试寻找内部比外部更暗的四边形.

#### （1）检测图像中的线段

* 线段检测方法首先进行计算每个像素点的梯度大小和方向。
* 应用基于梯度的聚类方法（gradient-based clustering method），拥有相似梯度方向和大小的像素被聚合成一个单元。
* 将像素单元进行合并：

对于每个边，我们判断相连的单元是否应该被合并，对于一个单元n，

D(n) : 梯度方向的范围(the range of gradient directions)；

M(n): 梯度大小的范围(the range of magnitudes)。

对于两个单元m、n，如果他们满足以下两个条件，将被合并：



D()和M()得知越小，意味着单元内部的变化越小。如果两个单元合并后和单 独每个单元一致，那么这两个单元就可以合并。参数KD、KM 可以适当调节 单元内部变化情况。

* 调整每个线段让其左侧置暗，右侧置亮。

这部分计算是整个检测阶段最慢的部分，可以通过降低分辨率来提升计算速度。

#### （2）提取图像中的四边形

这部分的任务是找到可以组成四边形的线段序列。

深度优先递归搜索算法：（递归4层）

在每一层搜索中添加一条边到四边形。在第一层，我们考虑所有线段，在第二层到第四层，我们考虑那些和之前线段结束的地方足够近的所有线段，并且要符合逆时针方向。

## 2. 编码模块

#### （1）解读检测区域内的编码

最后一步是从数据域内读取字节。

计算每个字节区域相关标签的坐标，通过单应矩阵将他们映射到图像坐标系内，然后对像素结果进行二值化处理。

#### （2）编码匹配

判断被检测出来的四边形内的编码是否是有效编码。

编码模块的目的是：

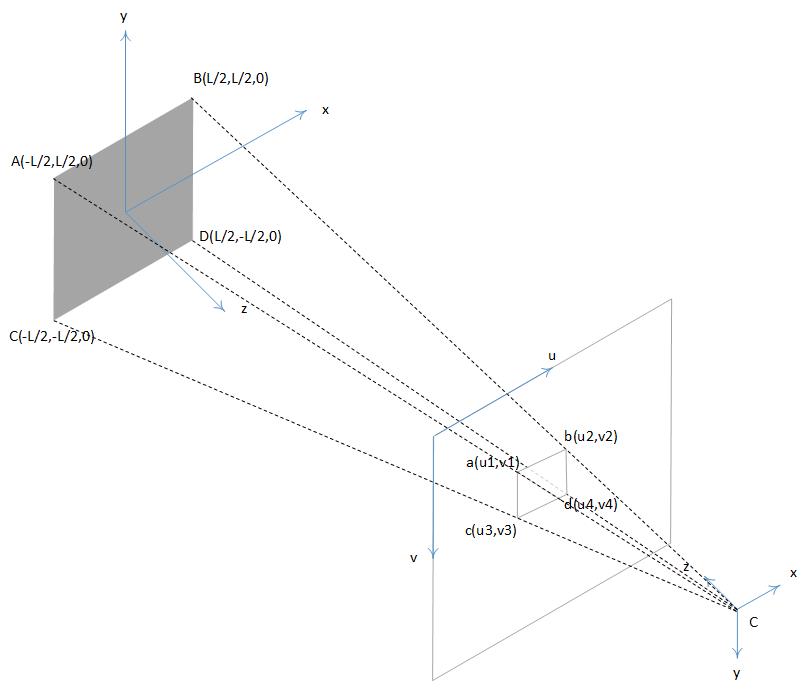
* 最大化可识别的编码数
* 最大化可检测或改正的字节数
* 最小化false positive/inter-tag confusion rate
* 最小化每个标签的字节总数

Apriltags编码系统基于lexicode。传统的lexicode有两个参数：编码的字节数和两个编码之间的海明距离（一个最小海明距离为10的36位编码我们称之为36h10 code）。

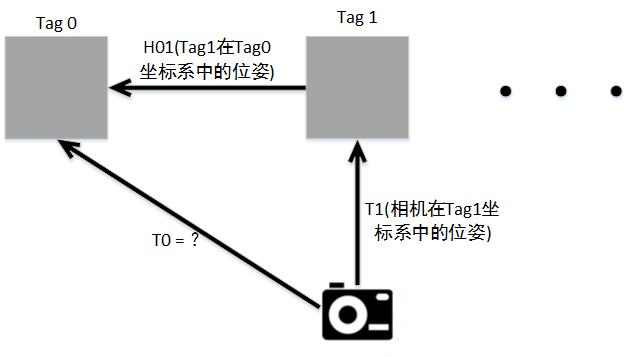
海明距离（Hamming distance）：两个码字的对应比特取值不同的比特数称为这两个码字的海明距离。在一个有效编码集中,任意两个码字的海明距离的最小值称为该编码集的海明距离。

例如:10101和00110从第一位开始依次有第一位、第四、第五位不同，则海明距离为3。

## 3. 相对位置的估计



我们已知标签(ABCD)的边长L，可以求出标签四个定点在标签坐标系中的坐标。标签投影到相机成像平面，可知标签四个定点在相机成像平面内的坐标，结合相机的内参矩阵，可以通过opencv::solvepnp()函数求得标签在相机坐标系内的位姿T( [R|t])。



当进行多标签定位的时候，我们可以求出相机相对于每一个标签的位姿，但我们希望得到相机相对于一个统一的坐标系，及世界坐标系的位姿，这里我们以第一个标签(Tag0)的坐标系为世界坐标系，我们需要先求出每一个标签相对于第一个坐标系的位姿(H01，H02，H0i...)，如果相机检测出Tagi，我们记相机在Tagi坐标系中的位姿为Ti，那么相机在Tag0中的位姿T0为：

T0 = Ti x H0i