**目标：**

* 我们将看到BRIEF算法的基础知识

## api：

### star = cv2.xfeatures2d.StarDetector\_create()

### brief = cv2.xfeatures2d.BriefDescriptorExtractor\_create()

### kp = star.detect(img,None)

### # compute the descriptors with BRIEF

### kp, des = brief.compute(img, kp)

**理论**

我们知道 SIFT 算法使用的是 128 维向量作为描述符。由于它是使用的浮点数，所以要使用 512 个字节。同样 SURF 算法最少使用 256 个字节（64 维描述符）。创建一个包含上千个特征的向量需要消耗大量的内存，在嵌入式等资源有限的设备上这样是不可行的。匹配时还会消耗更多的内存和时间。

但是在实际的匹配过程中如此多的维度是没有必要的。我们可以使用 PCA，LDA 等方法来进行降维。甚至可以使用 LSH（局部敏感哈希）将 SIFT 浮点数的描述符转换成二进制字符串。对这些字符串再使用汉明距离进行匹配。汉明距离的计算只需要进行 XOR 位运算以及位计数，这种计算很适合在现代的CPU 上进行。但我们还是要先找到描述符才能使用哈希，这不能解决最初的内存消耗问题。

BRIEF 应运而生。它不去计算描述符而是直接找到一个二进制字符串。这种算法使用的是已经平滑后的图像，它会按照一种特定的方式选取一组像素点对 𝑛𝑑(x，y)，然后在这些像素点对之间进行灰度值对比。例如，第一个点对的灰度值分别为 p 和 q。如果 p 小于 q，结果就是 1，否则就是 0。就这样对 n d个点对进行对比得到一个$n\_d$维的二进制字符串。

n d 可以是 128，256，512。OpenCV 对这些都提供了支持，但在默认情况下是 256（OpenC 是使用字节表示它们的，所以这些值分别对应与 16，32，64）。当我们获得这些二进制字符串之后就可以使用汉明距离对它们进行匹配了。

非常重要的一点是：BRIEF 是一种特征描述符，它不提供查找特征的方法。所以我们不得不使用其他特征检测器，比如 SIFT 和 SURF 等。原始文献推荐使用 CenSurE 特征检测器，这种算法很快。而且 BRIEF 算法对 CenSurE关键点的描述效果要比 SURF 关键点的描述更好。 简单来说 BRIEF 是一种对特征点描述符计算和匹配的快速方法。这种算法可以实现很高的识别率，除非出现平面内的大旋转。

**OpenCV中的BRIEF**

下面的代码显示了在CenSurE检测器的帮助下计算BRIEF描述符。（CenSurE探测器在OpenCV中称为STAR探测器）

请注意，你需要opencv contrib才能使用它。

## 实例

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |
|  |