**ORB:An Alternative to SIFT or SURF**

Author: Ethan Rublee,Gary Bradski

Translator:练孙鸿

**Preface**

既然是分析ORB算法，那我干脆翻译一部分ORB论文好了。翻译比较口语化，偶尔带些自己的理解。

**Abstract**

特征匹配是很多cv问题的关键。例如物体识别/SFM。很多特征描述子(descriptor)的花销都挺大的。所以本文提出一个很快的、基于BRIEF的二维特征描述子，叫做ORB（这种方法的特点有：旋转不变性；抗干扰能力强）。**ORB比SIFT快了两个数量级（这看起来还行）**，但是表现依旧很好。把这东西用于现实世界都没什么问题。

**1-Introduction**

SIFT是上个10年发明的关键点的检测器(detector)与描述子(descriptor)，算是很成功了。但是它的计算量非常的大，特别是用在实时视觉测距、低功率设备上时，SIFT就显得很吃力了。所以SURF作为SIFT的改进就是最好的。也有一些研究是探讨GPU加速的SIFT。在这篇文章里面，我们提出一种小计算量的方法，匹配效果与SIFT差不多，但是计算量小、实时的、抗干扰能力强(less affected by noise)。

我们提出的特征是建立在大名鼎鼎的FAST关键点检测器上的；还有最近发明的BRIEF描述子；所以们就把这算法叫做ORB吧(Oriented FAST and Rotated BRIEF).我们的主要的贡献是blablablabla....之后会把ORB和SIFT&SURF进行对比，以说明ORB的表现。

**2-Related Work**

关键点Keypoints：FAST和他的变体可以用于在实时系统里面匹配视觉特征。它在发现合理的角点关键点(corner keypoints)领域是比较高效的，但是还是得配上金字塔机制(其实就是mipmap)来实现缩放。在我们的系统里面，用了Harris corner filter来剔除某些边，从而得以给角点合理地打分。很多的关键点检测器会包含一个“方向算子”(orientation operator)，SIFT和SURF是两个最经典的例子，但是FAST并没有这么做。

描述子Descriptor：BRIEF是最近发明的一种描述子，它使用了简单的像素间的二值化测试。BRIEF跟SIFT比起来，在对光照影响的、模糊、透视畸变影响的鲁棒性形方面没什么差别的。但是它对平面内旋转特别的敏感。卧槽兄弟，BRIEF需要用binary test来训练一个分类树。一旦用一定数量样本训练了分类树，就可以返回一个签名(signature)用于任意的点。用

类似的办法，我们也要找到一种对方向最不敏感的test。用于找到非相关测试(uncorrelated tests)的传统方法是主成分分析（PCA）。..............

**3-oFAST: FAST Keypoint Orientation**

因为它的计算特性，FAST特征是被广泛应用的。但是FAST特征是没有“方向”这个部件的。所以在这一章我们会加一个可以快速计算的方向。

**3.1FAST Detector[2]**

我们先从在图像里提取FAST点开始。FAST需要输入一个参数，就是中心像素与它周围一圈像素之间的亮度阈值(intensity threshold)。ORB使用了FAST-9（半径为9的圆）。

FAST不产生cornerness的度量，但是我们发现它在边缘处有很大的响应(large respondses along edges)。所以我们用了Harris角点度量值来对FAST关键点进行排序。对于目标的N个关键点，我们先把阈值设置到足够低去获取N个关键点，然后再按照Harris Measure来对它们排序，取最高的N个点。FAST不产生多尺度的特征。所以我们要构建image的mipmap chain来产生在每个mipmap level的FAST特征。

**3.2强度中心的方向**

我们的方法用了一个简单但是有效的角点度量，就是强度中心（intensity centroid）。这个强度中心假设一个corner的强度是偏离中心的，所以这个向量就可以用来估算(impute)一个方向。Rosin把一个patch的矩(moment)定义为：（看原论文）。然后方向orientation就可以简单地用矩的元素进行计算。

然后文章对这种估计特征点朝向的方式和基于双梯度(two gradient-based measure)的方法比较了一下。

**4.rBRIEF：Rotation-Aware BRIEF**

在这一节，我们先介绍下steered BRIEF描述子，然后说明一下他怎么高效地计算他，并说明他对旋转的处理是不容乐观的。然后提出个更好的算子,rBRIEF。

**4.1BRIEF算子的快速旋转**

**BRIEF[1]**

BRIEF算子是一个通过二元强度测试构建的像素块(image patch)的 比特串描述子(bit string descriptor)。看BRIEF的论文的意思是，从image patch p里面选n pair点X(u1,v1)&Y(u2,v2)，n对像素点就可以做n次对比，p(X)<p(Y)的话test result就是1，反之为0。于是n个1和0可以连成一个n bit的串。这个bit string就是BRIEF descriptor。**所以怪不得opencv里面的descriptor match用Hamming Distance（毕竟感觉是对应位置的bit比较才有意义）**

**Steered BRIEF**

我们想让BRIEF在平面内的旋转是不变的，但是普通的BRIEF对于旋转太过敏感，只要source与destination image旋转角度超过几度，匹配的表现就会迅速下降（sharply fall off）。但是如果对于某一个image patch都计算一系列旋转和透视扭曲影响下的BRIEF描述子，那么计算量将会非常的恐怖。原文构造了一种”Steered” version的BRIEF算子，就是把BRIEF算子的一系列旋转构造一个查找表（因为BRIEF是一系列pixel pairs的binary test嘛，所以他们旋转后的binary test的pixel pairs也可以确定下来。ORB原文是把360度分成12份来构造查找表）

**4.2方差与相关性**

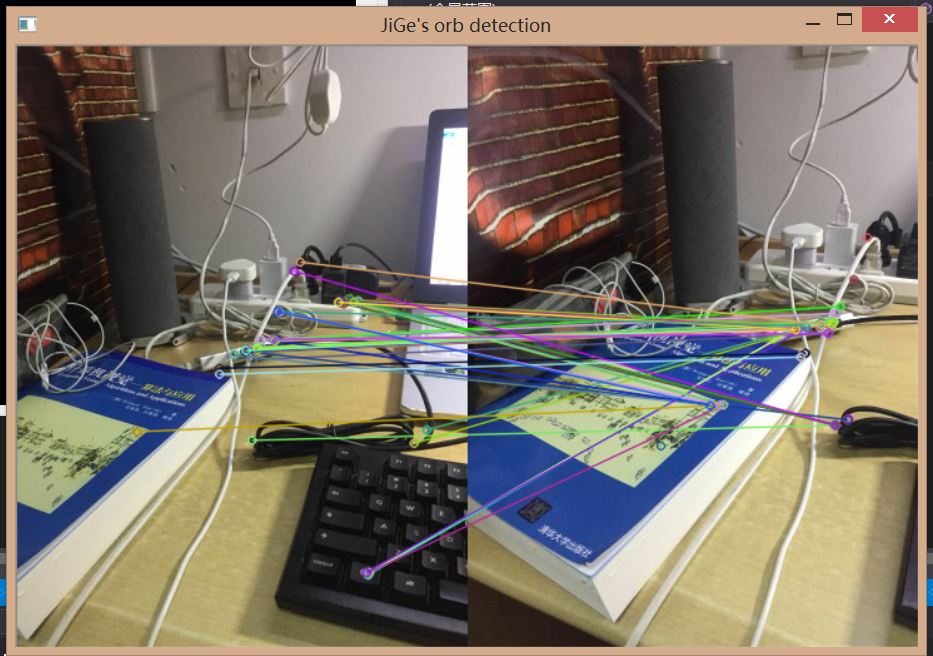
BRIEF算子有个很好的性质就是这个描述子的每一个bit，都有比较大的方差和约等于0.5的均值。这一节大致秀了一下BRIEF,steered BRIEF和rBRIEF描述子“距离”的分布。

**4.3 Learning Good Binary Feature**

为了应对steered BRIEF损失的方差，减少binary test之间的相关性(correlation)，我们提出了一种选择good binary test的方法。原本可以考虑PCA走一波，但是后来发现计算量也是不小。所以居然又上了训练集.....PASCAL 2006的数据集，300k关键点，全部枚举一次，用贪心算法来找出一组均值约为0.5的uncorrelated binary test.这就是rBRIEF

**C++代码(opencv3 based)与实验结果**

|  |
| --- |
| #include <opencv2/core/core.hpp>  #include <opencv2/features2d/features2d.hpp>  #include <opencv2/highgui/highgui.hpp>  #include <opencv2/calib3d/calib3d.hpp>  #include <opencv2/imgproc/imgproc.hpp>  int main()  {  cv::Mat image1 = cv::imread("1.jpg");  cv::Mat image2 = cv::imread("2.jpg");  // -- Step 1: Detect the keypoints using STAR Detector  std::vector<cv::KeyPoint> keyPointList1, keyPointList2;  int nKeyPoints = 50;//feature point count  cv::Ptr<cv::ORB> orb = cv::ORB::create(nKeyPoints);  orb->detect(image1, keyPointList1);  orb->detect(image2, keyPointList2);  // -- Stpe 2: Calculate descriptors (feature vectors)  cv::Mat descriptor1, descriptors2;  orb->compute(image1, keyPointList1, descriptor1);  orb->compute(image2, keyPointList2, descriptors2);  //-- Step 3: Matching descriptor vectors with a brute force matcher  cv::BFMatcher matcher(cv::NORM\_HAMMING2);  std::vector<cv::DMatch> matches;  matcher.match(descriptor1, descriptors2, matches);  std::vector<cv::DMatch> goodMatches;  for (int i = 0; i < descriptor1.rows; i++)  {  constexpr int minDist = 100;  if (matches[i].distance < 3 \* minDist)  {  goodMatches.push\_back(matches[i]);  }  }  // -- draw matches  cv::Mat drawnMatchLineImage;  cv::drawMatches(image1, keyPointList1, image2, keyPointList2, matches, drawnMatchLineImage);  // -- show  cv::imshow("JiGe's orb detection", drawnMatchLineImage);  cv::waitKey(0);  return 0;  } |



**引用**

[1]Calonder M, Lepetit V, Strecha C, et al. BRIEF: Binary Robust Independent Elementary Features[C]// European Conference on Computer Vision. 2010:778-792.

[2]Rosten E, Drummond T. Machine learning for high-speed corner detection[C]// European Conference on Computer Vision. Springer-Verlag, 2006:430-443.