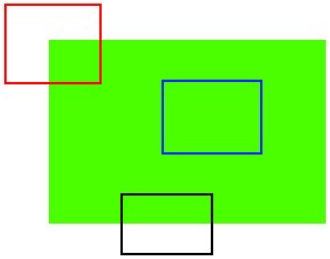
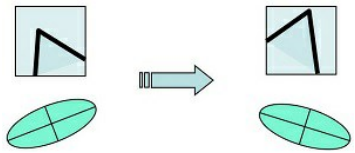
OpenCV 之 特征检测

飞鸢逐浪 2021-08-01 原文

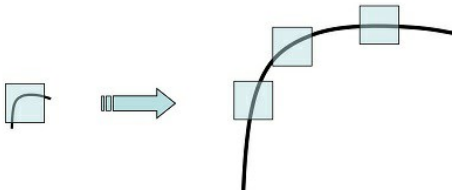
特征，也称 兴趣点 或 关键点，如下：蓝框内区域平坦，无特征；黑框内有“边缘”，红框内有“角点”，后二者都可视为“特征”



角点作为一种特征，它具有旋转不变性，如下：当图像旋转时，代表角点响应函数 R 的特征椭圆，其形状保持不变



但是，角点不具有尺度不变性，如下：左图中被检测为角点的特征，当放大到右图的尺度空间时，则会被检测为 边缘 或 曲线



下面介绍几种具有尺度不变性的特征检测算法，如 SIFT、SURF、ORB、BRISK、KAZE 和 AKAZE 等

1 特征检测

1.1 SIFT

SIFT 全称 Scale Invariant Feature Transform，是特征检测中里程碑式的算法，也是目前最有效的特征检测，该算法申请了专利，直到 2020年3月才过专利保护期

OpenCV 从 4.4.0 起，已经将 SIFT 移到了主模块 feature2d 中，SIFT 继承自 Feature2D 类，而 Feature2D 继承自 Algorithm 类，SIFT 的创建函数 create() 定义如下：

```
class SIFT : public Feature2D
{
public:
    static Ptr<SIFT> create(
        int nfeatures = 0,              // The number of best features to retain
        int nOctaveLayers = 3,          // The number of layers in each octave. 3 is the value used in D.Lowe paper
        double contrastThreshold = 0.04, // The contrast threshold used to filter out weak features in low-contrast regions
        double edgeThreshold = 10,      // The threshold used to filter out edge-like features
        double sigma = 1.6 );           // The sigma of the Gaussian applied to the input image at the octave 0
```

Algorithm 类中有两个虚函数：detect() 检测特征，compute() 计算描述符

```
class Feature2D : public virtual Algorithm
{
public:
    /* Detects keypoints in an image (first variant) or image set(second variant). */
    virtual void detect(InputArray image, std::vector<KeyPoint>& keypoints, InputArray mask=noArray() );

    /* Computes the descriptors for a set of keypoints detected in an image (first variant) or image set (second variant). */
    virtual void compute(InputArray image, std::vector<KeyPoint>& keypoints, OutputArray descriptors );
```

1.2 SURF

SIFT 算法虽好，但计算速度不够快，于是 SIFT 的近似版 SURF (Speeded Up Robust Features) 应运而生， SURF 的运行时间约为 SIFT 的 1/3

SURF 属于 xfeature2d 模块，也继承自 Feature2D 类，其 create() 函数定义如下：

```
namespace xfeatures2d
{
class SURF : public Feature2D
{
public:
    static Ptr<SURF> create(
        double hessianThreshold = 100, // Threshold for hessian keypoint detector used in SURF
        int nOctaves = 4,              // Number of pyramid octaves the keypoint detector will use
        int nOctaveLayers = 3,         // Number of octave layers within each octave
        bool extended = false,         // Extended descriptor flag (true, 128-element descriptors; false, 64-element descriptors)
        bool upright = false);         // Up-right or rotated features flag (true,do not compute orientation of features; false, compute orientation)
```

其中，hessianThreshold 为海森阈值，只有大于该阈值的特征才会被保留；海森阈值越大，对应检测到的特征越少；海森阈值取决于图像对比度，一般 300~500 之间的检测效果较好

1.3 CenSurE

CenSurE (Center Surround Extremas)，是在 SURF 基础上做的一种改进，基于 CenSurE 特征检测 和 M-SURF 特征描述符，号称比 SURF 更快，可用于实时处理领域

OpenCV 并没有完全实现 CenSurE 算法，而是借鉴衍生出了 StarDetector，其 create() 函数定义如下：

```
    static Ptr<StarDetector> create(
        int maxSize = 45,                   //
        int responseThreshold = 30,         //
        int lineThresholdProjected = 10,    //
        int lineThresholdBinarized = 8,     //
        int suppressNonmaxSize = 5          //
    );
```

2 实时特征检测

SURF 的运行速度比 SIFT 快 3 倍，但在一些实时处理系统 (视觉里程计) 或低功耗设备中，SURF 还是不够快，于是，便有了下面的两种算法

2.1 ORB

OpenCV Labs 实现了一种更快的算法 ORB - Oriented FAST and Rotated BRIEF，它是在 FAST 角点检测 和 BRIEF 特征描述符的基础上修改实现的

视觉 SLAM (Simultaneous Localization and Mapping 同步定位与建图) 领域中，著名的开源项目 ORB-SLAM，其中的特征提取就是基于 ORB 算法

OpenCV 中 ORB 的 create() 函数定义如下：

```
static Ptr<ORB> create (
    int nfeatures = 500,           // The maximum number of features to retain
    float scaleFactor = 1.2f,      // Pyramid decimation ratio, greater than 1
    int nlevels = 8,               // The number of pyramid levels
    int edgeThreshold = 31,        // This is size of the border where the features are not detected
    int firstLevel = 0,            // The level of pyramid to put source image to
    int WTA_K = 2,                 // The number of points that produce each element of the oriented BRIEF descriptor
    ORB::ScoreType scoreType = ORB::HARRIS_SCORE, // The default HARRIS_SCORE means that Harris algorithm is used to rank features
    int patchSize = 31,            // size of the patch used by the oriented BRIEF descriptor
    int fastThreshold = 20         // the fast threshold
 );
```

2.2 BRISK

BRISK 号称比 SURF 的运行速度快一个数量级，它基于 AGAST 角点检测 和 BRIEF 特征描述符，其中 AGAST 是比 FAST 更快的一种角点检测算法

BRISK 的 create() 函数如下：

```
    /* The BRISK constructor */
    static Ptr<BRISK> create(
        int thresh = 30,            // AGAST detection threshold score
        int octaves = 3,            // octaves detection octaves. Use 0 to do single scale
        float patternScale = 1.0f   // apply this scale to the pattern used for sampling the neighbourhood of a keypoint
    );

    /* The BRISK constructor for a custom pattern, detection thresholdand octaves */
    static Ptr<BRISK> create(
        int   thresh,                          // AGAST detection threshold score
        int   octaves,                         // detection octaves. Use 0 to do single scale.
        const std::vector<float> &radiusList,  // defines the radii(in pixels) where the samples around a keypoint are taken (for keypoint scale 1).
        const std::vector<int> &numberList,    // defines the number of sampling points on the sampling circle.Must be the same size as radiusList..
        float dMax = 5.85f,                    // threshold for the short pairings used for descriptor formation (in pixels for keypoint scale 1)
```

```
        float dMin = 8.2f,                      // threshold for the long pairings used for orientation determination (in pixels for keypoint scale 1)
        const std::vector<int>&indexChange = std::vector<int>()  // index remapping of the bits
    );
```

2.3 BRIEF 描述符

上述 ORB 和 BRISK 中，都提到了 BRIEF 特征描述符，BRIEF 全称 Binary Robust Independent Elementary Feature)，是用二进制串向量来描述特征的一种方式

SIFT 中的一个特征，对应着一个由128个浮点数组成的向量，占 512 个字节；而 SURF 的一个特征，对应着一个由 64个浮点数组成的向量，占 256 个字节

当有成千上万个特征时， 特征描述符会占用大量的内存，并且会增加匹配的时间，在一些资源受限的场合，尤其是嵌入式系统中，SIFT 和 SURF 并非最优选择

而 BRIEF 特征描述符，采用的是二进制串，可将所占字节缩减为 64 或 32 甚至 16，相比 SIFT 和 SURF，大大减少了对内存的占用，非常适合于实时处理系统

OpenCV 中 BRIEF 描述符的定义如下：

```
// Class for computing BRIEF descriptors described in @cite calon2010 .
class BriefDescriptorExtractor : public Feature2D
{
public:
    static Ptr<BriefDescriptorExtractor> create(
        int bytes = 32,                 // legth of the descriptor in bytes, valid values are: 16, 32 (default) or 64 .
        bool use_orientation = false);   // sample patterns using keypoints orientation, disabled by default.
};
```

3 非线性尺度空间

SIFT 和 SURF 是在线性尺度空间内的分析，在构建高斯尺度空间的过程中，高斯滤波会将图像中的边界和细节信息等，连同噪声一起模糊化掉，因此，会造成一定程度上特征定位精度的损失

为了克服高斯滤波的缺点，2012年，西班牙人 Pablo F. Alcantarilla 利用非线性扩散滤波代替高斯滤波，通过加性粒子分裂法 (Additive Operator Splitting) 构建了非线性尺度空间，提出了 KAZE 算法

KAZE 是为了纪念“尺度空间分析之父” Iijima 而取得名字，在日语中是 “风” 的意思；AKAZE 是 Accelerated KAZE，顾名思义是 KAZE 的加速版本

3.1 KAZE

KAZE 的 create() 函数如下：

```
    /* The KAZE constructor */
    static Ptr<KAZE> create (
        bool extended = false,          // Set to enable extraction of extended (128-byte) descriptor
        bool upright = false,           // Set to enable use of upright descriptors (non rotation-
invariant)
        float threshold = 0.001f,       // Detector response threshold to accept point
        int nOctaves = 4,               // Maximum octave evolution of the image
        int nOctaveLayers = 4,          // Default number of sublevels per scale level
        KAZE::DiffusivityType diffusivity = KAZE::DIFF_PM_G2  // Diffusivity type. DIFF_PM_G1,
DIFF_PM_G2, DIFF_WEICKERT or DIFF_CHARBONNIER
    );
```

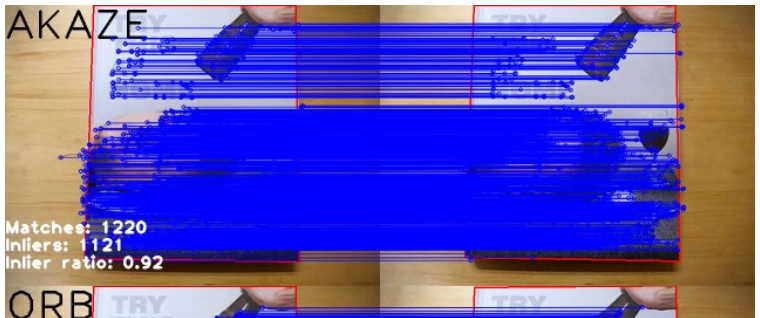
3.2 AKAZE

AKAZE 的 create() 函数如下：

```
    /* The AKAZE constructor */
    static Ptr<AKAZE> create(
        AKAZE::DescriptorType descriptor_type = AKAZE::DESCRIPTOR_MLDB, // Type of the extracted
descriptor: DESCRIPTOR_KAZE, DESCRIPTOR_KAZE_UPRIGHT, DESCRIPTOR_MLDB or DESCRIPTOR_MLDB_UPRIGHT.
        int descriptor_size = 0,                // Size of the descriptor in bits. 0 -> Full size
        int descriptor_channels = 3,            // Number of channels in the descriptor (1, 2, 3)
        float threshold = 0.001f,               // Detector response threshold to accept point
        int nOctaves = 4,                       // Maximum octave evolution of the image
        int nOctaveLayers = 4,                  // Default number of sublevels per scale level
        KAZE::DiffusivityType diffusivity = KAZE::DIFF_PM_G2  // Diffusivity type. DIFF_PM_G1,
DIFF_PM_G2, DIFF_WEICKERT or DIFF_CHARBONNIER
    );
```

3.3 AKAZE vs ORB

OpenCV Tutorials中，有 ORB 和 AKAZE 的对比实验，从所选取的图像数据集来看，AKAZE 的检测效果优于 ORB



4 代码例程

2004年 D. Lowe 提出 SIFT 算法后，在提高运算速度的方向上，先是诞生了比 SIFT 快3倍的 SURF，而后又在 SURF 的基础上改进出了 CenSurE，宣称可用于实时处理领域

BRIEF 特征描述符，利用二进制串描述符，减少了对内存的占用，提高了匹配的速度，特别适合资源受限的场合，如嵌入式系统

在 BRIEF 的基础上，ORB 结合 FAST 角点检测 和 BRIEF 描述符，BRISK 结合 AGAST 角点检测 和 BRIEF 描述符，真正实现了实时特征检测

KAZE 和 AKAZE 针对高斯滤波的缺点，另辟蹊径，直接从 线性尺度空间 跳转到 非线性尺度空间，变换尺度空间后，重新定义了特征检测

以上七种特征检测的算法，代码例程如下：

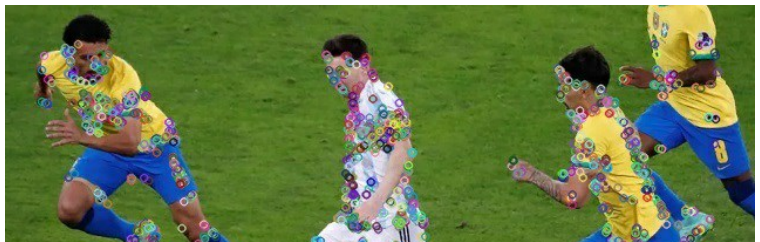
```
#include "opencv2/core.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/features2d.hpp"
#include "opencv2/xfeatures2d.hpp"

using namespace cv;

int main()
{
    // read
    Mat img = imread("messi.jpg");
    if (img.empty())
        return -1;

    // create and detect
    Ptr<SIFT> detector = SIFT::create();
    // Ptr<xfeatures2d::SURF> detector = xfeatures2d::SURF::create(400);
    // Ptr<xfeatures2d::StarDetector> detector = xfeatures2d::StarDetector::create(20, 20);
    // Ptr<ORB> detector = ORB::create(2000);
    // Ptr<BRISK> detector = BRISK::create();
    // Ptr<KAZE> detector = KAZE::create();
    // Ptr<AKAZE> detector = AKAZE::create();
    std::vector<KeyPoint>  keypoints;
    detector->detect(img, keypoints);

    // draw and show
    Mat img_keypoints;
    drawKeypoints(img, keypoints, img_keypoints);
    imshow("SIFT", img_keypoints);

    waitKey();
}
```

各算法的检测效果对比如下：


SIFT


SURF


CenSurE




ORB


BRISK


KAZE


AKAZE

后记

一开始酝酿本篇博客时，目标是将 OpenCV 中所有的特征检测算法，都阅读一遍原始论文，并弄懂 OpenCV 的代码实现，但随着阅读的深入，发现这几乎是不可能完成的任务。

第一，自己非学术科研人员，没这么多时间和精力投入；第二，数学知识的薄弱，尤其是读到 KAZE 算法，涉及到非线性尺度空间，深感数学的博大精深和自身能力的瓶颈。

“吾生也有涯，而知也无涯”，想到牛人如 David Lowe，一生最有名的也只是发明了 SIFT 算法，我等凡夫俗子更难以遑论，莫名间竟生出一些悲凉，继续写下去的动力消失殆尽 ...

好在这几天想通了，重新认清自己的水平和定位，调整当初太过宏大的目标，改目标为 “介绍 OpenCV 中的特征检测算法和使用例程”，于是，便有了本篇文章 ^_^

参考资料

SIFT 算法作者 David Lowe 的主页

OpenCV-Python Tutorials / Feature Detection and Description / Introduction to SIFT (Scale-Invariant Feature Transform)

OpenCV-Python Tutorials / Feature Detection and Description / Introduction to SURF (Speeded-Up Robust Features)

Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*

OpenCV-Python Tutorials / Feature Detection and Description / BRIEF (Binary Robust Independent Elementary Features)

OpenCV-Python Tutorials / Feature Detection and Description / ORB (Oriented FAST and Rotated BRIEF)

OpenCV Tutorials / 2D Features framework (feature2d module) / AKAZE and ORB planar tracking

KAZE 和 AKAZE 作者 Pablo F. Alcantarilla 的个人主页

OpenCV 之 特征检测的更多相关文章

1. OpenCV4.1.0实践(2) - Dlib+OpenCV人脸特征检测
   待更! 参考: python dlib opencv 人脸68点特征检测
2. OpenCV——SIFT特征检测与匹配
   SIFT特征和SURF特征比较 比较项目 SIFT SURF 尺度空间极值检测 使用高斯滤波器,根据不同尺度的高斯差(DOG)图像寻找局部极值 使用方形滤波器,利用海森矩阵的行列式值检测极值,并利用积 ...
3. opencv图像特征检测之斑点检测
   前面说过,图像特征点检测包括角点和斑点,今天来说说斑点,斑点是指二维图像中和周围颜色有颜色差异和灰度差异的区域,因为斑点代表的是一个区域,所以其相对于单纯的角点,具有更好的稳定性和更好的抗干扰能力. ...
4. OpenCV——Brisk特征检测、匹配与对象查找
   检测并绘制特征点: #include <opencv2/opencv.hpp> #include <opencv2/xfeatures2d.hpp> #include < ...
5. OpenCV——HOG特征检测
   API: HOGDescriptor(Size _winSize, ---:窗口大小,即检测的范围大小,前面的64*128 Size _blockSize,--- 前面的2*2的cell,即cell的 ...

6. OpenCV——ORB特征检测与匹配

   原文链接:https://mp.weixin.qq.com/s/S4b1OGjRWX1kktefyHAo8A #include <opencv2/opencv.hpp> #include ...

7. OpenCV——SURF特征检测、匹配与对象查找

   SURF原理详解:https://wenku.baidu.com/view/2f1e4d8ef705cc1754270945.html SURF算法工作原理 选择图像中的POI(Points of i ...

8. Opencv HOG特征检测

   HOGDescriptor hogDescriptor = HOGDescriptor(); hogDescriptor.setSVMDetector(hogDescriptor.getDefault ...

9. OpenCV入门指南----人脸检测

   本篇介绍图像处理与模式识别中最热门的一个领域——人脸检测(人脸识别).人脸检测可以说是学术界的宠儿,在不少EI,SCI高级别论文都能看到它的身影.甚至很多高校学生的毕业设计都会涉及到人脸检测.当然人脸 ...

随机推荐

1. RubyOnRails local_assigns

   http://api.rubyonrails.org/classes/ActionView/Template.html#method-i-local_assigns Returns a hash wi ...

2. Java并发集合的实现原理

   本文简要介绍Java并发编程方面常用的类和集合,并介绍下其实现原理. AtomicInteger 可以用原子方式更新int值.类 AtomicBoolean.AtomicInteger.AtomicL ...

3. LinckedhashMap原理

   http://zhangshixi.iteye.com/blog/673789 TreeMap的key是有顺序的,是自然顺序,也可以指定比较函数. 但默认不是按插入的顺序. 为了让Map JSON ...

4. C#高级二

   编程小虾米大侠之梦 软件环境:win7 开发工具:vs 2010 平台:.NET 语言:C# 类库版本:.NET Framework 2.0 语言特点:强类型语言 规划知识点: 1..net fram ...

5. matlab制造一个64*64的仿真数据

   fid = fopen('test_001.img','w'); r=random('Normal',100,0,64,64); z=random('Uniform',0,5,64,64); %%%% ...

6. Servlet过滤器——仿盗链过滤器

   1.概述 介绍如何使用过滤器技术,防止通过其他URL地址直接访问本站资源.运行本实例,当URL地址不是本站地址时,在网页中将显示错误提示信息. 2.技术要点 主要应用request对象的getHead ...

7. android用于打开各种文件的intent

   import android.app.Activity; import android.content.Intent; import android.net.Uri; import android.n ...

8. iOS之NSPredicate（正则表达式和UIBarController）

   本文转发至:https://segmentfault.com/a/1190000000623005 NSPredicate,这个类和我上一篇博文中提到的valueForKeyPath一样很强大.它的使 ...

9. 在C#中winform程序中应用nlog日志工具

   在C#中winform程序中应用nlog日志工具,配置文件简单应用. 文件名 nlog.config,请注意修改属性为"始终复制",发布时候容易遇到不存在文件的错误提示.通过Nu ...

10. 可视化分析工具Cytoscape使用记录

    最近项目要使用到可视化分析工具Cytoscape,所以会花费很多的时间跟精力来整理Cytoscape软件使用和开发的相关资料,希望写下的文章能减少有兴趣的同行学习跟开发所走的弯路时间.同时也是因为百度 ...

热门专题

SQL 正则表达式 不包含数字 | .NET反编译工具下载 | EXTRATREE优点 | LINUX HTTP服务 模拟接口超时 | STATIC FINAL在JAVA组合使用 | CHROME跨域插件

VS2019 C# WINFORM资源文件实现多语言切换 CKEDITOR 演示 MYSQL8.0停止使用BIN-LOG VUE轮播插件VUE-AWESOME-SWIPER点击放大

GIT GLOG安装CENTOS 安卓SHELL改变经纬度 DOTFUSCATOR混淆JOSN的引用不能自动添加 JAVA设置WEBSERVICE的过滤器 PYCHARM 2020.2.5永久激活

[NOIP2000]计算器的改良 PHP连接不上宝塔数据库 PYTHO实现的WEB漏洞扫描器 表格怎么把1.5变成一天12时 爬站工具TELEPORT PRO