云台

@李昕达

index

术语与一些关键词

特征点：指在图像中突出且具有代表意义的一些点，这些点可以用来表征、识别图像、进行图像配准、进行3D重建等，主要将角点视为有效的特征点。

角点通常为在灰度化图像中水平、竖直方向上变化均较大的点，通常为两条边缘的交点，更严格的说，角点的局部邻域应该具有两个不同区域的不同方向的边界[3]。

Harris角点算法[4,6]：简单来讲，就是拿一个小窗在图像中移动，通过考察这个小窗口内图像灰度的平均变换值来确定角点。

- 如果窗口内区域图像的灰度值恒定，那么所有不同方向的偏移几乎不发生变化；
- 如果窗口跨越一条边，那么沿着这条边的偏移几乎不发生变化， 但是与边垂直的偏移会发生很大的变化；
- 如果窗口包含一个孤立的点或者角点，那么所有不同方向的偏移会发生很大的变化。

当窗口发生[u,v]移动时，那么滑动前与滑动后对应的窗口中的像素点灰度变化描述如下：

$$E(u,v)=\sum w(x,y)[I(x_u,y+v)-I(x,y)]^2$$

> [u,v]是窗口的偏移量
> (x,y)是窗口内所对应的像素坐标位置，窗口有多大，就有多少个位置
> w(x,y)是窗口函数,窗口函数内所设置权值可以为平均分布或高斯分布，当窗口中心存在孤立的角点时，该点在高斯分布下的灰度变化贡献远大于平均分布下的贡献
> 此外，将窗口函数描绘为一个椭圆高斯分布还可以指明角点的指向。

几个常用算法的介绍与对比

SIFT算法：尺度不变性特征转换[5,8-9]

SIFT是一种通过在空间中寻找极值点来侦测与描述图像中的局部性特征的算法。

SIFT算子是把图像中检测到的特征点用一个128维的特征向量进行描述，因此一幅图像经过SIFT算法后表示为一个128维的特征向量集，该特征向量集具有对图像缩放，平移，旋转不变的特征，对于光照、仿射和投影变换也有一定的不变性，是一种非常优秀的局部特征描述算法[7]。

SIFT算法大致的实现流程为：

1. 尺度空间极点检测
2. 关键点精确定位
3. 关键点的方向确定
4. 特征向量的生成

尺度空间极点检测

真实世界中的物体只有在一定的尺度上才有意义，我们所要寻找的特征点就是在连续尺度空间下位置不改变的点。

构建尺度空间的目的就是找到在尺度变化中具有不变性的位置，可以使用连续的尺度变化，即在尺度空间中所有可能的尺度变化中找到稳定的特征点，通过这种方式找到的极点可以保证在图像缩放和旋转变化中具有不变性。

> 构建尺度空间的目的就是构建一个尺度金字塔，在金字塔的每一层都有该特征点不同尺度的图像，通过这些图像对应的特征点来进行匹配，进而得到特征的匹配

尺度空间的内核是高斯函数.假设 $I(x,y)$是原始图像,$G(x,y,\sigma)$是尺度空间可变的高斯函数,则 一个函数的尺度空间可以定义为

$$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y) \tag{1}$$

其中$*$表示的是卷积运算，σ表示尺度空间的大小，σ越大则表示越模糊，表示图像的概貌，σ越小则表示越清晰，表示图像的细节。高斯函数$G(x,y,\sigma)$定义为

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2} \tag{2}$$

经过一系列的尺度空间变化和二倍下采样,最终得到高斯金字塔.

为了在尺度空间中找到稳定不变的极值点，在SIFT算法中使用了高斯差分(DOG)函数 $D(x,y,\sigma)$ ，定义为

$$\begin{aligned} D(x,y,\sigma)&=[G(x,y,k\sigma)-G(x,y,\sigma)]*I(x,y) \\ &=L(x,y,k\sigma)-L(x,y,\sigma) \end{aligned} \tag{3}$$

选择高斯差分函数的原因如下:

1. 计算简单 L函数需要一定计算,D函数只需要减法
2. LoG的运算效率不高

关键点(特征点)精确定位

计算机中存储的图像数据是离散的，而我们之前找到的极值点也就是离散空间中的极值点，但是离散空间中的极值点并不是真实的连续空间中的极值点。所以需要对DoG空间进行拟合处理，以找到极值点的精确位置和尺度。另外，我们还需要去除那些在边缘位置的极值点，以提高关键点的稳定性。

在Lowe的论文[5]中，使用的是泰勒展开式作为拟合函数。我们得到的极值点是一个三维向量，包括它所在的尺度σ以及所在尺度图像中的位置坐标，即$X=(x,y,\sigma)$.因此需要三维的泰勒展开式进行展开，设$X_0=(x_0,y_0,\sigma_0)$，则其展开式的矢量形式为：

$$f(X)=f(X_0)+\frac{\partial f^T}{\partial X}(X-X_0)+\frac{1}{2}(X-X_0)^T\frac{\partial^2 f}{\partial X^2}(X-X_0) \tag{4}$$

去除边缘影响:

到这一步,我们已经得到了许多的极值点,但是其中有一部分不是我们想要的,当中就有一大部分是边缘区域产生的极值点。因为物体的边缘轮廓在灰度图中，存在着灰度值的突变，这样的点在计算中就被“误以为”是特征值。

由于图像中的物体的边缘位置的点的主曲率一般会比较高，因此我们可以通过主曲率来判断该点是否在物体的边缘位置。某像素点位置处的主曲率可以由二维的Hessian矩阵H计算得到

$$H=\begin{bmatrix}D_{xx}(x,y) & D_{xy}(x,y)\\ D_{xy}(x,y) & D_{yy}(x,y)\end{bmatrix}$$

具体后续计算可参见**Harris角点检测算法**.

特征点方向的计算:

为了实现特征点的旋转不变性，因此需要计算特征点的角度。在计算特征点的方向时是根据特征点所在的高斯尺度图像中的局部特征计算出的。该高斯尺度σ是已知的，并且该尺度是相对于该图像所在的组的基准图像的。所谓的局部特征就是特征点的邻域区域内所有像素的梯度幅角和梯度幅值，这里邻域区域定义为在该图像中以特征点为圆心，以r为半径的圆形区域

$$r=3*1.5\sigma \tag{5}$$

像素的梯度幅值/幅角计算公式为:

$$m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2} \tag{6}$$

$$\theta(x,y)=arctan\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right) \tag{7}$$

取幅值最高的方向为主方向,超过峰值能量的百分之80的方向，称为辅方向。

特征向量的生成

到这里,我们已经得到了一组SIFT特征点,包含了位置,尺度,方向等信息.

接下来,我们用一组向量将这些特性包括关键点周围的特征描绘出来.

先将坐标轴旋转为关键点的方向，以确保旋转不变性。以关键点为中心取8×8的窗口。

像素点的旋转过程如下:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{8}$$

在这里，我们仍然需要计算像素点的梯度幅值和梯度幅角，用以生成直方图，同样，这里根据像素点与特征点的距离，像素点的加权值也是不一样的，这里仍然采用高斯加权，高斯处理的方法为$\frac{d^2}{2}$。

一个显而易见的问题出现了：一个正方形中有很多像素点，这些像素点对于这个正方形的整体的贡献也是有权值的，因此还需要在此进行高斯加权。

经过上面的计算，就可以得到128柱的直方图$p1, p2, \ldots, p128$，为了去除光照的影响，需要进行归一化处理:

$$q_i = \frac{p_i}{\sqrt{p_1^2 + p_2^2 + \ldots + p_{128}^2}}, i = 1, 2, 3, \ldots, 128 \tag{10}$$

至此,我们通过了SIFT得到了一个128维的向量来描述每个特征点.

SURF算法:加速稳健特征[10-13]

SURF是一种稳健的局部特征点检测和描述算法[11-12],2006年发布于ECCV,是对Lowe在1999年提出的SIFT算法[5]的改进,提高了算法的执行效率,为算法在实时计算机视觉系统中应用提供了可能.

与SIFT相比,SURF在其基础上有几个改进:

1. 构建Hessian(海森矩阵)
2. 构建尺度空间
3. 特征点定位
4. 特征点主方向分配
5. 生成特征点描述子
6. 特征点匹配

以下将根据流程中标红的部分,对比性地分析SIFT和SURF的不同,分析SURF如何提高计算速度与稳定性:

构建Hessian矩阵

严格的来说,构建海森矩阵与构建尺度空间两个步骤,SIFT与SURF都进行了,区别在于顺序的不同.

SIFT将图片先构建尺度金字塔,再使用尺度(大小)不变的高斯核来构建Hessian矩阵,进而使用DOG进行特征点的提取,而**SURF**使用Hessian进行特征点的提取,在这一过程中使用了大小变化的标准二维高斯矩阵来进行尺度变换.

到这里似乎SURF并没有很大的速度提升,但是在Hessian矩阵的求解过程中.SURF使用了盒式滤波器替代了高斯滤波器,具体如下:

由公式[8]知:当Hessian矩阵的判别式取得局部极大值时，判定当前点是比周围邻域内其他点更亮或更暗的点，由此来定位关键点的位置。

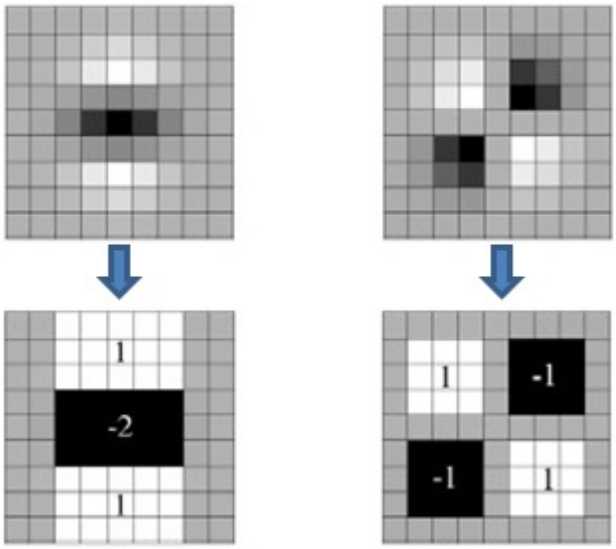
反过来看Hessian矩阵的判别式，其实就是当前点对水平方向二阶偏导乘以垂直方向的二阶偏导再减去当前点水平、垂直二阶偏导的二次方：

$$det(H) = D_{xx} * D_{yy} - D_{xy} * D_{xy} \tag{11}$$

由于高斯核实服从正态分布的，从中心点往外，系数越来越低，为了提高运算速度，Surf使用了盒式滤波器来近似替代高斯滤波器，所以在D_{xy}上乘了一个加权系数0.9，目的是为了平衡因使用盒式滤波器近似所带来的误差：

$$det(H) = D_{xx} * D_{yy} - (D_{xy} * 0.9)^2 \tag{12}$$

高斯滤波器与盒式滤波器的示意图如下:



上边两幅图是9*9高斯滤波器模板分别在图像上垂直方向上二阶导数Dyy和Dxy对应的值 下边两幅图是使用盒式滤波器对其近似，灰色部分的像素值为0，黑色为-2，白色为1。

盒式滤波器对图像的滤波转化成计算图像上不同区域间像素和的加减运算问题，这正是积分图的强项，只需要简单几次查找积分图就可以完成。

> 积分图: https://zh.wikipedia.org/wiki/%E7%A7%AF%E5%88%86%E5%9B%BE 积分图的每一点（x, y）的值是原图中对应位置的左上角区域的所有值得和

确定特征点的主方向

SIFT 在特征点邻域内进行最大梯度直方图统计,找到最大的梯度或者大于80%最大梯度方向作为特征的主方向.

与SIFT不同的是,SURF统计特征点邻域内的最大Haar小波相应直方图,从一个扇形方向开始，每次以固定步长进行旋转，找到其中最大响应的扇形，将其方向作为主方向.

> 哈尔特征: https://zh.wikipedia.org/wiki/%E5%93%88%E5%B0%94%E7%89%B9%E5%BE%81 哈尔特征（Haar-like features） 是用于物体识别的一种数字图像特征。它们因为与哈尔小波转换 极为相似而得名，是第一种即时的人脸检测運算。 例如，当前有一个人脸图像集合。通过观察可以发现，眼睛的颜色要比两颊的深。因此，用于人脸检测的哈尔特征是分别放置在眼睛和脸颊的两个相邻矩形。这些矩形的位置则通过类似于人脸图像的外接矩形的检测窗口进行定义。

确定特征描述子

SIFT:由于采用8方向作为主方向，所以描述子的维度为448=128；

SURF:只有水平和垂直两个方向的haar小波响应，但是考虑到the polarity of intensity images，同时还要计算两个方向haar响应的绝对值的累加，这样一来，每个小区域就有4维，所以最后得到的描述子维度为444=64；不难看出surf将描述子的长度由sift的128个浮点数精简到了64个浮点数，更加简洁了，因此计算速度就提升了

ORB算法:[14-15]

ORB算法的技术重点为:

1.基于FAST算法而改进的特征点提取方法:oFAST 2.基于BRIEF算法而改进的特征匹配描述方法:rBRIEF

以下就这两点进行具体阐述:

oFAST

首先介绍一下FAST特征点检测方法:

判别某一个点p是否为特征点可以通过以该点为中心画圆,半径为3个像素点,该圆过16个像素点,设在圆周上是否有n个连续的像素点都满足比I_p+t大，或者都比I_p-t小。（这里I_p指的点p的灰度值，t是一个阈值）如果满足这样的要求，则判断p是一个特征点，否则p不是。

由于在特征点检测过程中需要对所有点都进行检测,而大部分点都不是特征点,每一个点测16个圆周上的点,显然会浪费许多时间.

作者对每一个点都取1,5,9,13号像素点如果这4个点中至少有3个满足都满足要求,就继续测16个点,这极大地提高了算法的速度.

但是,传统的FAST算法有两个弊端:

1. 提取到的特征点没有方向
2. 提取到的特征点不能满足尺度变化

oFAST针对这两个问题进行了解决:

方向:

Orientation by Intensity Centroid方法:

这种方法的主要思想就是，首先把特征点的邻域范围看成一个组，然后求取这个组的质心，最后把该质心与特征点进行连线，求出该直线与横坐标轴的夹角，即为该特征点的方向。

求取质心的方法:

$$m_{pq}=\sum_{x,y}x^py^qI(x,y) \tag{13}$$

质心定义为:

$$C=(\frac{m_{10}}{m_{00}},\frac{m_{01}}{m_{00}}) \tag{14}$$

然后求取向量OC的方向,即为特征点的方向

尺度:

针对特征点不满足尺度变化，作者像SIFT算法中那样，建立尺度图像金字塔，通过在不同尺度下的图像中提取特征点以达到满足尺度变化的效果。

rBRIEF

BRIEF算法的目的是得到一个二进制串,对特征点的邻域建立一个邻域空间,对空间内每一像素进行判断:

$$\tau(p;x,y):=\left\{\begin{matrix}1:p(x)<p(y)\\0:p(x)\geq p(y)\end{matrix}\right\} \tag{15}$$

其中$p(x)$表示的是在点x处的图像灰度值，那么这样我们就可以得到一个n位的二进制串

这种方法在判断时主要进行亦或操作,这具有极佳的硬件友好性,运算速度非常快,但是BRIEF算法的缺点在于其不具有旋转不变性,当图像旋转后,其准确度快速下降,rBRIEF针对性地解决了这个问题.

想要让BRIEF算法具有旋转不变性，那么我们需要使特征点的邻域旋转一个角度，该角度就是我们上面求得的特征点的方向角θ。但是这样整体旋转一个邻域的开销是比较大的，一个更加高效的做法就是旋转我们前面得到的那些邻域中的匹配点x_i、y_i.

设生成特征点描述符的n个测试点对为(x_i, y_i)，定义一个$2 \times n$的矩阵

$$S = \begin{pmatrix} x_1, & \cdots & , x_n \\ y_1, & \cdots & , y_n \end{pmatrix} \tag{15}$$

利用角度θ形成的旋转矩阵为R_θ，那么旋转后匹配点的坐标为

$$S_\theta = R_\theta S \tag{15}$$

这样，在求取特征点描述符的时候，就使用S_θ中的像素点即可。

以上,对于SIFT,SURF,ORB三种算法进行了针对性的描述,下面将使用Python语言在Ubuntu16.04环境下实现FAST算法:

FAST算法的代码实现

目的

使用Python编程实现FAST算法,其中该算法的核心部分不使用既有库

源码

```
import cv2
import numpy as np
THRESHOLD = 40
N = 12

circle_list=[
    [2,2],[-2,2],[2,-2],[-2,-2],
    [3,-1],[3,0],[3,1],[-3,-1],
    [-3,0],[-3,1],[-1,3],[0,3],
    [1,3],[-1,-3],[0,-3],[1,-3]
    ]
src = cv2.imread('src.jpg')
gray = np.zeros(src.shape)
result = np.zeros(src.shape)

for i in range(gray.shape[0]):
    if i%100 == 0:
        print(i)
    for j in range(gray.shape[1]):
        g = sum(src[i,j,0:3])/3
        gray[i,j,0] = g
        gray[i,j,1] = g
        gray[i,j,2] = g

```

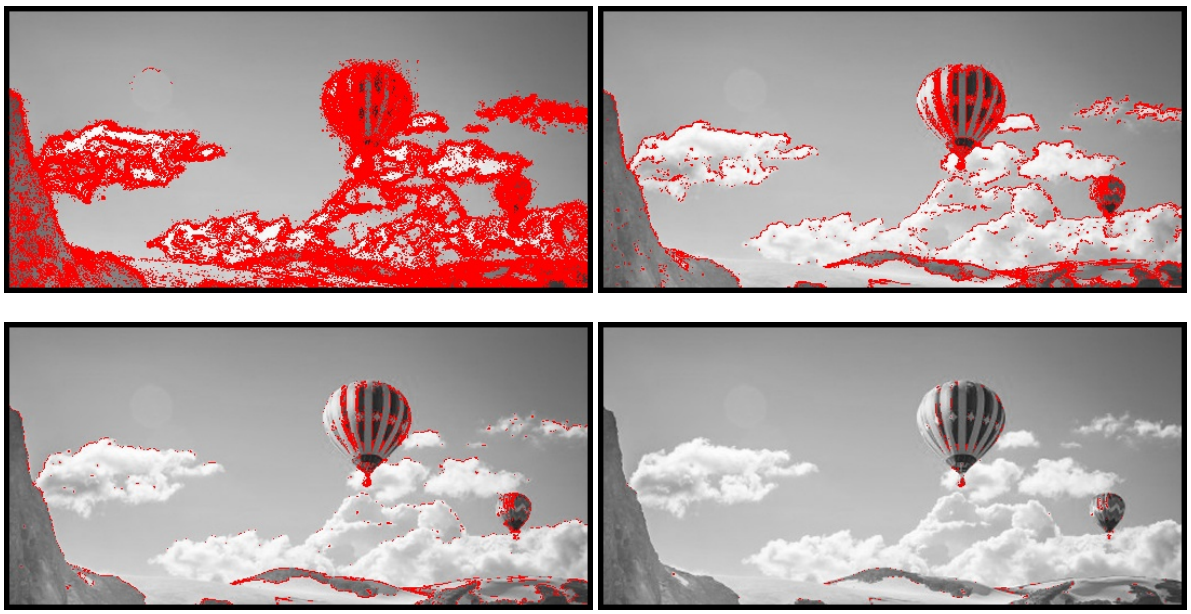
```
for i in range(5,gray.shape[0]-5):
    if i%100 == 0:
        print(i)
    for j in range(5,gray.shape[1]-5):
        s = 0
        for k in range(4):
            a=i+circle_list[k][0]
            b=j+circle_list[k][1]
            if abs(gray[a,b,1]-gray[i,j,1]) > THRESHOLD :
                s += 1
        if s > 2 :
            s = 0
            for k in range(16):
                a=i+circle_list[k][0]
                b=j+circle_list[k][1]
                if abs(gray[a,b,1]-gray[i,j,1]) > THRESHOLD :
                    s += 1
            if s > N or s == N :
                result[i,j,0]=0
                result[i,j,1]=0
                result[i,j,2]=255
            else :
                result[i,j,0]=gray[i,j,0]
                result[i,j,1]=gray[i,j,1]
                result[i,j,2]=gray[i,j,2]
        else :
            result[i,j,0]=gray[i,j,0]
            result[i,j,1]=gray[i,j,1]
            result[i,j,2]=gray[i,j,2]

cv2.imwrite('result'+str(THRESHOLD)+'.jpg',result)
```

代码块解释

行数	功能
1-2	引入相关库
4-5	设定阈值
18-25	将src灰度化后的图存入gray
34-39	初步判断四角,选择灰度值超过阈值的点的数量超过2的点
42-50	判断圆周16点,超过阈值的点大于N的点标红
51-58	不合格的点赋灰度值,此处也可赋为原RGB图
60	存图

结果示例

原图



两点灰度差值阈值:THRESHOLD	16个圆周点最低合格数量:N
阈值=2;N=12	阈值=10;N==12
阈值=20;N==12	阈值=40;N==12

Reference:

[3] https://blog.csdn.net/mmjwung/article/details/6748895
[4] https://blog.csdn.net/a429051366/article/details/51009438
[5]Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.
[6] https://blog.csdn.net/lwzkiller/article/details/54633670
[7] https://blog.csdn.net/lhanchao/article/details/52345845 [8] https://blog.csdn.net/weixin_38404120/articl

e/details/73740612 [9] https://www.cnblogs.com/YiXiaoZhou/p/5893835.html [10] https://blog.csdn.net/dcrmg/article/details/52601010 [11] Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features[C]//European conference on computer vision. Springer, Berlin, Heidelberg, 2006: 404-417. [12] https://blog.csdn.net/songzitea/article/details/16986423 [13] https://blog.csdn.net/eternity1118/article/details/51152162 [14] https://blog.csdn.net/lhanchao/article/details/52612954 [15] https://blog.csdn.net/zouzoupaopao229/article/details/52625678