

计算机视觉

实验二 Harris Corner Detector

黄昭阳 3130000696- 12/14/15



个人信息



3130000696

黃昭阳

开发软件说明

提供的可执行文件在Mac下编译运行，使用命令格式为：

“./Lab2 输入图片名 k参数 Aperture参数”

eg.

“./Lab1 /Users/drinkingcoder/Documents/university/Vision/Labs/Lab2/Lab2/input.jpg 0.04 3”

程序将对指定图片做：灰度值转换、sobel算子求偏导得到偏导分布图、R值图、角点图，在当前目录下生成相应的png图片。当k参数不被输入时，默认值为0.04，Aperture不被输入时，默认值为3。

由于提供的可执行文件在Mac下编译的，提供源代码与makefile文件，可以自行编译。

算法具体步骤及实现要点

一、算法步骤：

将输入图片从彩图转成灰度图，并保存的相应图片名中，然后sobel求偏导，计算R值。

```
cvtColor(inputImg, grayImg,CV_RGB2GRAY);
imshow("img", grayImg);
imwrite(grayScaleFileName, grayImg);
waitKey();
//转灰度图
SobelOperator(grayImg);
//使用sobel算子求偏导
GetRValue();
//计算R值
imshow("img", inputImg);
imwrite(result, inputImg);
waitKey();
//将角点标到了inputImg上以便观察
```

二、实现要点：

1. 根据给定的Aperture，生成pascal三角形来计算指定大小的sobel算子

```

pascal1[1][1] = 1;
pascal1[2][1] = 1;
pascal1[2][2] = 1;
pascal2[1][1] = 1;
pascal2[2][1] = 1;
pascal2[2][2] = -1;
for(int i=3;i<21;i++)
{
    pascal1[i][1] = 1;
    pascal2[i][1] = 1;
    for(int j=2;j<=i;j++)
    {
        pascal1[i][j] = pascal1[i-1][j-1]+pascal1[i-1][j];
        pascal2[i][j] = pascal2[i-1][j-1]+pascal2[i-1][j];
    }
} //生成pascal三角形
for(int i=1;i<=Aperture;i++)
{
    for(int j=1;j<=Aperture;j++)
    {
        sobelx[i-1][j-1]+= pascal1[Aperture][i]*pascal2[Aperture]
[Aperture-j+1];
        sobely[i-1][j-1]+= pascal1[Aperture][j]*pascal2[Aperture][i];
        if(sobelx[i-1][j-1]>0) sumSobel+=sobelx[i-1][j-1];
    } //生成指定Aperture的sobel算子
}

```

2. 使用sobel算子实现差分求偏导

```

    for(int i=0;i<grayImg.rows-Aperature+1;i++)
        for(int j=0;j<grayImg.cols-Aperature+1;j++)
    {
//根据给定光圈大小使用sobel算子进行全图卷积
    d = 0;
    for(int r = 0; r<Aperture ; r++)
        for(int c = 0; c<Aperture ; c++)
            d+=sobelx[r][c]*gray[i+r][j+c];
//sobel算子的卷积操作求x方向偏导数
    d = fabs(d/sumSobel);
//结果取绝对值并归一化
    xpd[i+1][j+1] = d;
    d = 0;
    for(int r = 0; r<Aperture ; r++)
        for(int c = 0; c<Aperture ; c++)
            d+=sobely[r][c]*gray[i+r][j+c];
//sobel算子的卷积操作求y方向偏导数
    d = fabs(d/sumSobel);
//结果取绝对值并归一化
    ypd[i+1][j+1] = d;
    if(max < xpd[i+1][j+1]+ypd[i+1][j+1]) max = xpd[i+1][j+1]+ypd[i+1][j+1];
    if(min > xpd[i+1][j+1]+ypd[i+1][j+1]) min = xpd[i+1][j+1]+ypd[i+1][j+1];
//更新最大值和最小值,以便之后将其分布到0~255生成灰度图显示
    }
    double gap = max-min;
    for(int i=0;i<grayImg.rows-2;i++)
        for(int j=0;j<grayImg.cols-2;j++)
            image[i][j] = uchar((xpd[i][j]+ypd[i][j]-min)/gap*255);
//生成偏导数的灰度图

```

3. 计算R值.

```

for(int i=0;i<grayImg.rows-length;i++)
    for (int j=0;j<grayImg.cols-length; j++) {
//扫描全图,根据blocksize(即length)计算R值

    locatr = i+length/2;
    locatec = j+length/2;
    a = 0;
    b = 0;
    c = 0;

```

```

        for(int rr=0; rr<length;rr++)
            for(int cc=0; cc<length ;cc++)
            {
                a += xpd[i+rr] [j+cc]*xpd[i+rr] [j+cc];
                b += xpd[i+rr] [j+cc]*ypd[i+rr] [j+cc];
                c += ypd[i+rr] [j+cc]*ypd[i+rr] [j+cc];
            }
//a、b、c分别是hessian矩阵的对应元素
        lambda1 = ((a+c)+sqrt(4*b*b+(a-c)*(a-c)))/2;
        lambda2 = ((a+c)-sqrt(4*b*b+(a-c)*(a-c)))/2;
//根据特征方程求解特征值
        d = lambda1*lambda2-k*(lambda1+lambda2)*(lambda1+lambda2);
//计算R值
        r[locatr][locatc] = d;
        if(min > d) min = d;
        if(max < d) max = d;
//更新最大值与最小值,以便之后显示
        if(d>Threshold)
            image[locatr][locatc] = 255;
        else
            image[locatr][locatc] = 0;
//根据threshold来二值化
    }

```

3. 对得到的R值进行非极大值抑制计算

```

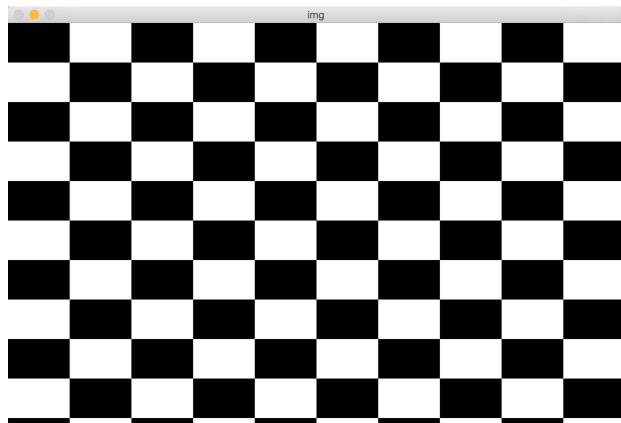
int rr,cc,x,y;
for(int i=0;i<featurePointImg.rows;i++)
    for(int j=0;j<featurePointImg.cols;j++)
    {
        if( r[i][j] < Threshold )
        {
            feature[i][j] = 0;
//检测到未达到threshold则放弃这个点
            continue;
        }
        feature[i][j] = 255;
//先选取这个点,在与周围点进行比较时如果发现非极大值则抛弃此点
        for(x = -5;x<5;x++)
        {
            rr = i+x;
            if(rr<0||rr>=rImg.rows) continue;
//当扫描到图片之外则进行下行的检测
            for (y = -5; y<5; y++) {
                cc = j+y;
                if(cc<0||cc>=rImg.cols) continue;
//当扫描到图片之外则进行下一个点的检测
                if(x == 0 && y == 0) continue;
//不与自己作比较
        }
    }
}

```

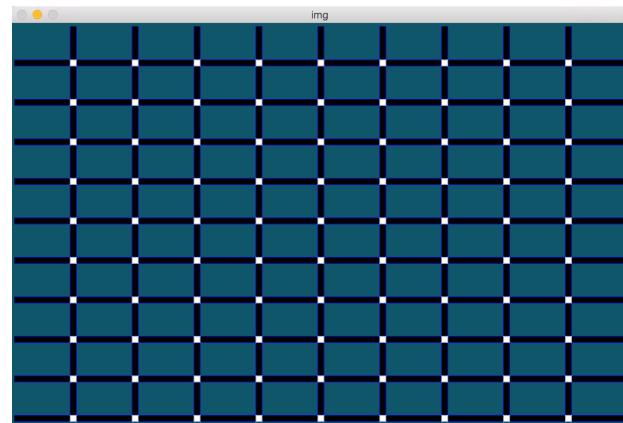
```
        if (r[rr][cc] >= r[i][j]) {
            feature[i][j] = 0;
            break;
        }
        if (feature[i][j] == 0) break;
    }
    if(feature[i][j] == 255)
        circle(inputImg, Point(j,i), 1, CV_RGB(255, 0, 0),2);
//将角点在原图中标记出来
}
```

实验结果展示及分析

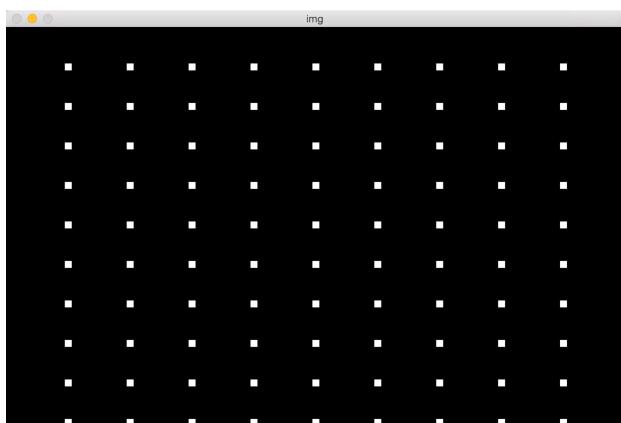
1. 黑白格子图测试



原图



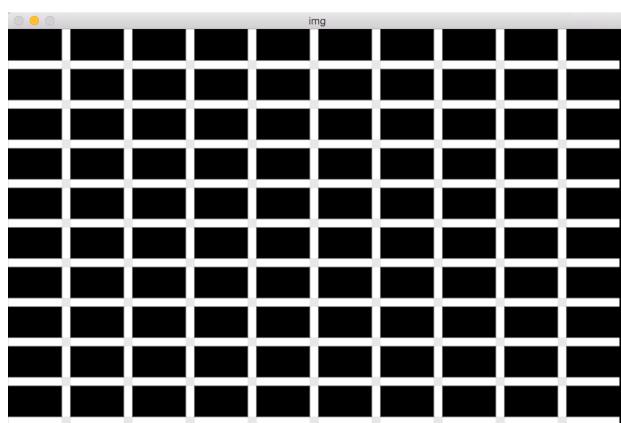
彩色R值图



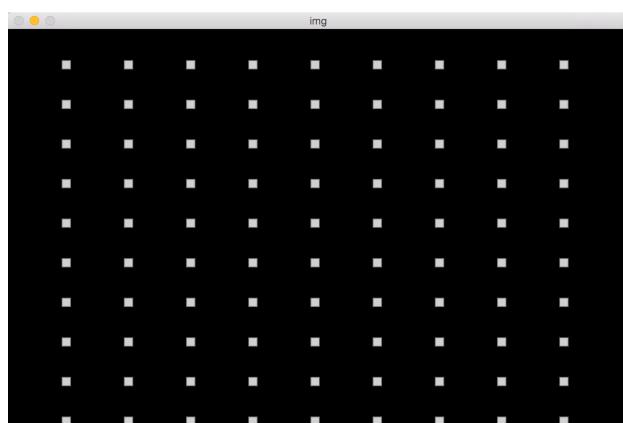
R值图



非极大值抑制图



较大特征值图

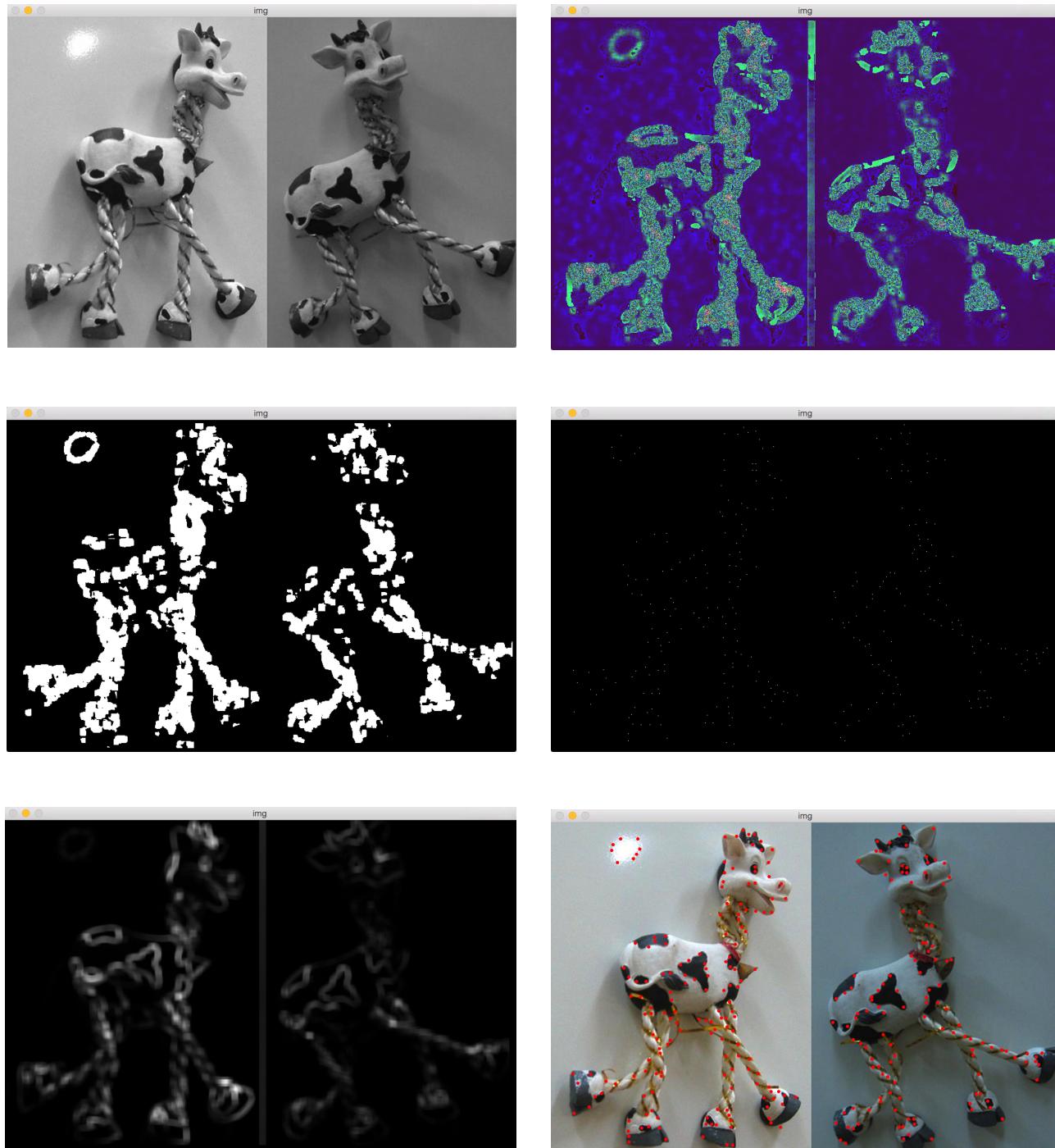


较小特征值图

分析：

彩色R值图可以观察得到边缘比平面颜色深，角点颜色十分亮，由于blocksize比较大（为10），所以中间一块值相等，非极大值抑制后全部消失。

2. 两张小鹿图



分别是：灰度图，彩色R值图，黑白R值图，非极大值抑制后得到的角点，较大特征值图，在原图中用红色小圆圈标出角点的图。

结果分析：

1. 从彩色R值图观察可以发现边缘R值较大（绿色），角点R值更大（红色），符合算法要求。
2. 与老师演示文稿中的角点检测图相比，此次试验结果角点数量偏多，但是分布与其大致相似，这里threshold、blocksize、Aperture等多个参数都与此相关

编程体会

1. OpenCV自带矩阵类，但是使用方法有些特殊，mat类中行向量的元素在地址上是连续的，但是行向量之间的元素的地址是分离的，这个特殊情况使得我之前程序出错调试好久都毫无头绪，最终发现是这个原因。

2. 而且直接使用效率十分之低，刚刚完成时访问[i][j]号像素点使用的是`xPDIImg.row(i).ptr<double>()[j]`，使用程序运行求上方小鹿图的角点时需要消耗将近7秒，因此我使用双重指针将所有矩阵的行向量取出来，直接使用指针访问，这样程序速度加快十分之多，一秒内可以出解。

3. sobel算子自己尝试了使用Aperture = 5的情况，但是效果不是很好。

4. OpenCV有C和C++两种语言风格，一开始不清楚，使得两种语言风格的变量和语言被我混着用，导致出错也分不清出在哪里，后来奇怪为什么有的函数前方有cv前缀有的没有，于是将所有的东西全部转换成C++风格，错误就没有了，同时也学会了mat类的使用方法，在图像信息处理课程中自己写了很多图像处理的函数，但是苦于导入数据只能使用bmp格式，接下来可以尝试将原来的bmpFile类重写，使用OpenCV导入数据，然后其余的方法全部由自己实现。