山东大学 计算机科学与技术 学院

计算机视觉 课程实验报告

|  |  |  |
| --- | --- | --- |
| 学号：201600301304 | 姓名：贾乘兴 | 班级：人工智能16 |
| 实验题目：Harris角点检测 | | |
| 实验内容：   1. 实现harris角点检测算法 2. 该算法是较为经典的角点检测算法，基于某个像素的移动窗口在角点处沿各个方向移动时，根据数值的变化的，我们可以对像素点所在区域分为三类，角点、边、flat区域，对某一方向的计算公式如下     其中，u、v为x与y方向的偏移方向，我们对其中I(x+u,y+v)进行泰勒展开，得到近似结果如下    带入原式得到的变形如下    进行进一步整理得    可知在该区域内各个方向求变化时，不变的量为中间部分，我们称之为结构张量，用M表示    其中w使用均值或者高斯滤波器  我们的目标是，找到这样的像素点，该点具有的性质为，窗口沿各个方向移动时，函数E(u,v)的变化都比较大，该问题可看作是一个二次型问题，我们对矩阵M进行特征分解，得到的特征值和对应的特征向量，在特征向量的方向上，如果沿该方向变化都比较大，则可知该点为角点，如果只有某一方向较大，则为边，如果都较小，则为flat区域  但是该判别标准并不非常精确，需要确定各个方向较大的定义的阈值，需要调整多个数值，所以我们有如下定义    该R值为响应值，且无需特征分解，因为我们可以直接求出矩阵的行列式的数值和对角线元素之和，只需要确定一个阈值和k值（0.04-0.06）即可，当R为较大正数时，为cornor区域，当R为绝对值较大的负数时，为edge区域，当R较小时，我们视为flat区域。   1. 算法步骤如下：   对于输入图像，首先我们进行二值化  然后使用sobel算子求出图像的x，y方向梯度  通过梯度图计算得到x方向平方，y方向平方，xy的方向的乘积  使用高斯滤波对得到的三个矩阵进行滤波处理  计算响应值，得到的结果，在一定区域内进行非极大值抑制，然后筛选大于阈值的点（可根据计算量进行该步骤的交换）   1. 实验代码如下   #include **"opencv2/imgproc/imgproc.hpp"** #include **"opencv2/highgui/highgui.hpp"** #include **<iostream>** #include **<cmath> using namespace** cv; **using namespace** std;  **int** main() {  **const** Mat srcImage = imread(**"/Users/apple/Desktop/12.jpg"**);  **if** (!srcImage.data)  {  printf(**"could not load image...\n"**);  **return** -1;  }   Mat srcGray;  Mat imageSobelX;  Mat imageSobelY;   *//灰度图* cvtColor(srcImage,srcGray,***CV\_BGR2GRAY***);  *//梯度* Sobel(srcGray,imageSobelX,srcImage.depth(),1,0,3);  Sobel(srcGray,imageSobelY,srcImage.depth(),0,1,3);   Mat imageXX = Mat(srcGray.size(), **CV\_64F**);  Mat imageYY = Mat(srcGray.size(), **CV\_64F**);  Mat imageXY = Mat(srcGray.size(), **CV\_64F**);  Mat harris0 = Mat(srcGray.size(), **CV\_64F**);   *//计算各方向的梯度的平方与乘积* **for**(**int** i=0;i<srcGray.rows;i++)  {  **for**(**int** j=0;j<srcGray.cols;j++)  {  **double** XX = imageSobelX.at<uchar>(i, j) \* imageSobelX.at<uchar>(i, j);  **double** YY = imageSobelY.at<uchar>(i, j) \* imageSobelY.at<uchar>(i, j);  **double** XY = imageSobelX.at<uchar>(i, j) \* imageSobelY.at<uchar>(i, j);  imageXX.at<**double**>(i,j) = XX;  imageYY.at<**double**>(i,j) = YY;  imageXY.at<**double**>(i,j) = XY;  *//cout<<(float)imageXX.at<double>(i,j)<<" "<<(float)imageSobelX.at<uchar>(i,j)<<" "<<XX<<endl;* }  }   *//高斯滤波XX YY XY* GaussianBlur(imageXX, imageXX, Size(7,7), 0, 0);  GaussianBlur(imageYY, imageYY, Size(7,7), 0, 0);  GaussianBlur(imageXY, imageXY, Size(7,7), 0, 0);   *//取alpha参数为0.04* **double** k = 0.04;  **for** (**int** i=0;i<harris0.rows;i++){  **for** (**int** j=0;j<harris0.cols;j++){  **double** a = imageXX.at<**double**>(i, j);  **double** b = imageYY.at<**double**>(i, j);  **double** c = imageXY.at<**double**>(i, j);  **double** r = abs(a\*b - c\*c - k\*(a + b)\*(a + b));  harris0.at<**double**>(i, j) = r;  }  }  **int** t = 8e+7;  **int** r = 5;  **for** (**int** i = r; i < srcGray.rows - r; i++)  {  **for** (**int** j = r; j < srcGray.cols - r; j++)  {  **int** flag = 1;  **if**((**int**)harris0.at<**double**>(i, j) > t){  **for**(**int** y = i - r;y < i + r; y++){  **for**(**int** x = j - r;x < j + r; x++){  **if** (harris0.at<**double**>(i, j) < harris0.at<**double**>(y, x)){  flag = 0;  }  }  }  **if** (flag == 1){  circle(srcImage, Point(j, i), 3, Scalar(0,0,255), 2, 2, 0);  }  }  }  }  imshow(**"Result"**, srcImage);  imwrite(**"/Users/apple/Desktop/2ak.jpg"**,srcImage);  waitKey(0);  **return** 0; }   1. 实验结果对比   ../../13.jpg../../1ak.jpg  可见蓝色背景和蓝色大楼导致一些边缘角点没有被检测出来  ../../12.jpg../../2ak.jpg  可见部分点没有检测到，但一些边缘被检测为了cornor   1. 使用opencv的cornorHarris函数调用对比   ../../22.png../../23.png  在调整到合适的阈值后，效果会变好，但对第一张图的检测上，opencv的代码实现的要更好 | | |
| 实验过程中遇到和解决的问题：  （记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）   1. 计算过程中，没有转为double形式导致向上截断 2. 最后非极大值抑制时，通过对区域的调整，发现在增大判别区域时，先判断阈值和先判断极大值有不同的时间，在区域较大时，先判断极大值较好，在区域较小时，先判断阈值较好 | | |
| 结论分析与体会：通过本次实验，在解决问题中更好的理解了检测算法，同时对时间的实验和分析得到的较好的顺序 | | |