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：图像代数运算 | | |
| 实验内容：   1. 图像对比度调整 2. 图像对比度代表的是图像像素的差异大小，大致相当于图像像素rgb数值的方差，本实验将图像在原始分布f（x）=x的基础上，使用sigmoid作为像素的rgb数值变换映射函数   原始的sigmoid函数为:    考虑到像素范围为0到255，本题采用的函数为：    其中x为原始图像像素值大小，avr代表了映射后的像素中值，与均值有关，其他数值分布以此对称，bvr用于调整sigmoid的收敛速度，与方差相关  一开始我们将avr取值为图像像素均值，bvr取一个适当的数值，对两张图片（如下）进行处理  ../../aaa.jpg ../../abc.jpg  处理的效果如下：  ../../aaa1.jpg ../../abc2.jpg  可见效果并不是很理想，但我们要调整参数较为麻烦，采用二分法调整也难以很好的得到结果，所以我们采用了slider控件，代码如下  #include**<opencv2/opencv.hpp>**  #include **<iostream>**  #include**<opencv2/core.hpp>**  #include **<cmath>**  **using namespace** std;  **using namespace** cv;   **void** on\_Constract(**int**, **void** \*);   Mat bmg0,img0;  **int** avr = 40;  **int** bvr = 4096;   **int** main() {  img0 = imread(**"/Users/apple/Desktop/abc.jpg"**);  img0.copyTo(bmg0);   namedWindow(**"slider"**);  createTrackbar(**"avr："**, **"slider"**, &avr, 145, on\_Constract);  setTrackbarMin(**"avr"**, **"slider"**, 1);*//设置滑块的最小值*  setTrackbarMax(**"avr"**, **"slider"**, 145);*//设置滑块的最大值*  createTrackbar(**"bvr"**,**"slider"**,&bvr, 8192, on\_Constract);  setTrackbarMin(**"bvr"**, **"slider"**, 1024);*//设置滑块的最小值*  setTrackbarMax(**"bvr"**, **"slider"**, 8192);*//设置滑块的最大值*  **while** (**char**(waitKey(1) != **'q'**));  **return** 0;  }   **void** on\_Constract(**int**, **void** \*) {   **for** (**int** i = 0; i < img0.rows; i++) {  **for** (**int** j = 0; j < img0.cols; j++) {  **for** (**int** c = 0; c < 3; c++)  bmg0.at<Vec3b>(i, j)[c] =saturate\_cast<uchar>(255/(1+exp(-(avr\*img0.at<Vec3b>(i,j)[c] )/bvr)));  }  }  imshow(**"slider"**, bmg0);  }  图像处理更方便，效果如下：  ../../../Library/Containers/com.tencent.qq/Data/Library/Application%20Support/QQ/Users/1131225623/QQ/Temp.db/30E2C15F-2F86-47DE-8219-EF347158CB8A.png   1. 背景相减   很多情况下我们需要将数帧图像中的第一张为背景图，然后提取出之后人物或者其他物体作为前景图或者前景图的mask再作处理，本次实验首先采用了最简单的背景相减的方式，原图如下：  ../../h1.png ../../h2.png  首先进行背景相减，得到结果如下：  ../../hh.png  然后设置阈值，提取出灰度图的mask，mask如下：  ../../h.png  可见处理效果并不是很好，因为图像和图像之间可能由于时间导致树叶等  物体发生了变化，为噪声数据所以进行改进，使用滤波器进行处理，本实  验采用了两种滤波器，5\*5的核与9\*1的核，考虑到图像的特征，衣服条  纹等问题，不适合采用1\*n的核进行处理，在经过处理后得到两种mask  ../../hp.png ../../hp1.png  然后对这些二值化的mask采用了与的操作，得到最终结果如下：  ../../hp2.png  改进后的实验代码如下：  #include **<iostream>** #include **<opencv2/opencv.hpp>  using namespace** std; **using namespace** cv;  **int** main(){  Mat fm0 = imread(**"/Users/apple/Desktop/h1.png"**);  Mat bm0 = imread(**"/Users/apple/Desktop/h2.png"**);  Mat imga = Mat::zeros(fm0.size(),fm0.type());   **for**(**int** y=0;y<fm0.rows;y++)  **for**(**int** x=0;x<fm0.cols;x++)  **for**(**int** c=0;c<3;c++){  imga.at<Vec3b>(y,x)[c]=abs(fm0.at<Vec3b>(y,x)[c]-bm0.at<Vec3b>(y,x)[c]);  }  cvtColor(imga,imga,***CV\_BGR2GRAY***);  threshold(imga,imga,32,255,***THRESH\_BINARY***);   Mat imgb = imga;  **int** sum = 0;  **int** conv[5][5] = {{1,4,16,4,1},{4,16,32,16,4},{16,32,64,32,16},{4,16,32,16,4},{1,4,16,4,1}};  **int** csum = 0;  **for**(**int** i=0;i<5;i++) {  **for**(**int** j=0;j<5;j++) {  csum = csum + conv[i][j];  }  }   **for**(**int** y=2;y<fm0.rows-2;y++) {  **for**(**int** x=2;x<fm0.cols-2;x++) {  sum = 0;  **for**(**int** i=y-2;i<=y+2;i++){  **for**(**int** j=x-2;j<=x+2;j++) {  sum = sum + conv[i+2-y][j+2-x]\*imga.at<uchar>(i,j);  }  }  imgb.at<uchar>(y,x) = 255\*((sum/csum)>96?1:0);  }  }  *//imshow("out",imgb);  //waitKey(0);  //imwrite("/Users/apple/Desktop/hp.png",imgb);* Mat imgc = imga;  **int** conv\_h[9] = {1,4,16,32,64,32,16,4,1};  sum = 0;  **int** csum\_h = 0;  **for**(**int** i=0;i<9;i++) {  csum\_h = csum\_h + conv\_h[i];  }   **for**(**int** y=0;y<fm0.rows;y++) {  **for**(**int** x=8;x<fm0.cols-8;x++) {  sum = 0;  **for**(**int** i=y-4;i<y+4;i++){  sum = sum + conv\_h[i+4-y]\*imga.at<uchar>(i,x);  }  imgc.at<uchar>(y,x) = 255\*((sum/csum\_h)>128?1:0);  }  }  *//imshow("out",imgc);  //waitKey(0);  //imwrite("/Users/apple/Desktop/hp1.png",imgc);* Mat imgd = imga;  **for**(**int** y=0;y<fm0.rows;y++) {  **for**(**int** x=0;x<fm0.cols;x++) {  imgd.at<uchar>(y,x) = 255\*((imga.at<uchar>(y,x)+imgb.at<uchar>(y,x)+imgc.at<uchar>(y,x))>=255?1:0);  }  }   imshow(**"out"**,imgd);  waitKey(0);  imwrite(**"/Users/apple/Desktop/hp2.png"**,imgd);  }  通过查阅资料，本问题还可以采用高斯混合的运算进行处理，但需要较多帧，并不适合本题。对于影子问题，认为影子的特征为在某一片区域有了相同的亮度减小，但程度并不大，所以可以设置一个数值范围，然后搜索区域，在一片较大的且相连的区域内设置mask进行遮挡 | | |
| 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. 在滤波过程中遇到了边界问题，考虑过padding或者越过的方法解决，最终认为直接越过的方法效果更好。 2. 在对比度调整中，确定参数范围直接影响效果，最终使用slider控件动态控制 3. 在背景相减运算中，横向的衣服条纹一度是个问题，但采用了纵向的滤波器之后这一问题便得到了解决 | | |
| 结论分析与体会： 本次实验初步接触了图像处理和计算机视觉中一些小问题，除了背景相减、亮度调整之外，里面的肤色识别、影子问题都是一些较小的但解决起来较为麻烦的问题，所以需要我们自己去设计一些特征提取，从而应用于这些问题，更要懂得发现问题并解决 | | |