DIP HW-9 part-1 19041002 方亮 自动化1班

10.3

0	0	0
1	-2	1
0	0	0

horizonal

0	1	0
0	-2	0
0	1	0

vertical

0	0	1
0	-2	0
1	0	0

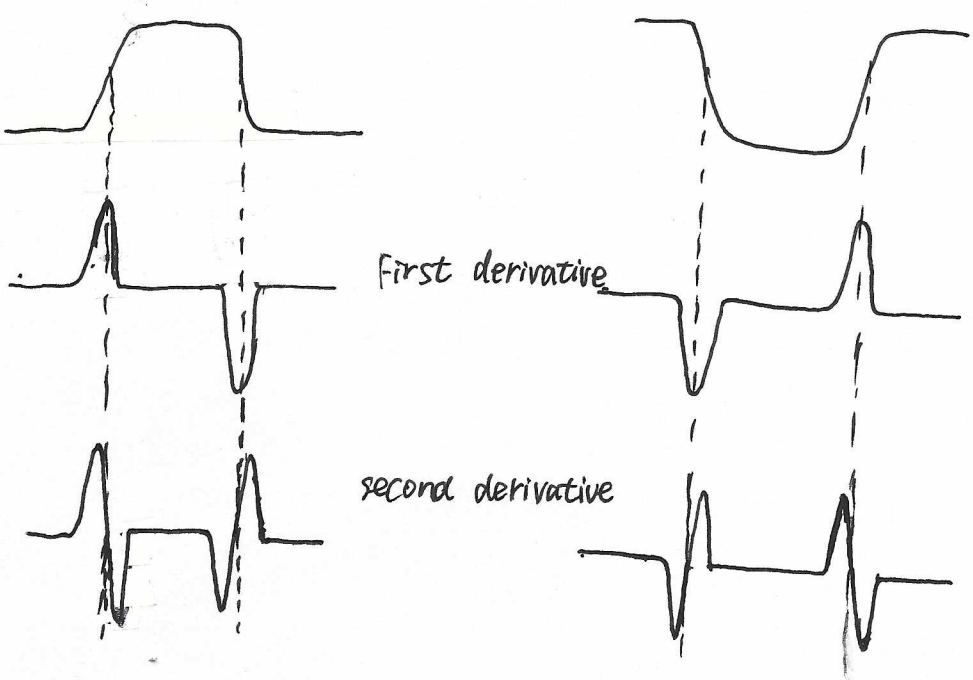
+45°

1	0	0
0	-2	0
0	0	1

-45°

若没有间断返回值为0，有间断返回值为2，可用于检测1像素间断

10.9



10.26 (a) (a)图中一点(1,1)为(0,0) 代入到 $\rho=x_0\cos\theta+y_0\sin\theta$ 中为 $\rho=0$ 是一条直线

(b) 通过 $\rho=x_0\cos\theta+y_0\sin\theta$ 可以看出，当仅当 $x_0=y_0=0$ 时 $\rho=x_0\cos\theta+y_0\sin\theta$ 为一条直线，故是0性一点

(c) 若 $\rho=x_0\cos\theta+y_0\sin\theta$ 与 $\rho=x_1\cos\theta+y_0\sin\theta$ 在 (ρ_0,θ_0) 相交.

即 $\begin{cases}\rho_0=x_0\cos\theta_0+y_0\sin\theta_0\\ \rho_0=x_1\cos\theta_0+y_1\sin\theta_0\end{cases}$ 则 $\theta=\theta_0+\pi$ 时，

$\rho_1=x_0\cos(\theta_0+\pi)+y_0\sin(\theta_0+\pi)=-\rho_0$

$\rho_2=x_1\cos(\theta_0+\pi)+y_1\sin(\theta_0+\pi)=-\rho_0=\rho_1$ 即也在 $(-\rho_0,\theta_0+\pi)$ 相交

此为反射邻接关系.

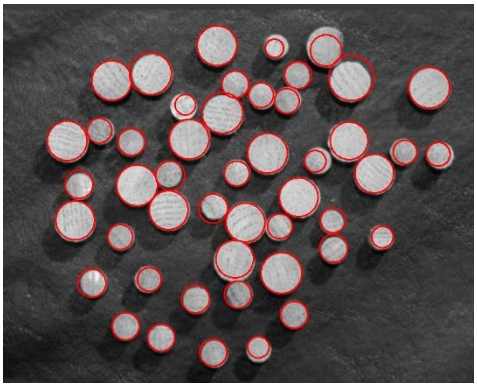
DIP HW-9 part-2 设计一种快速圆检测算法

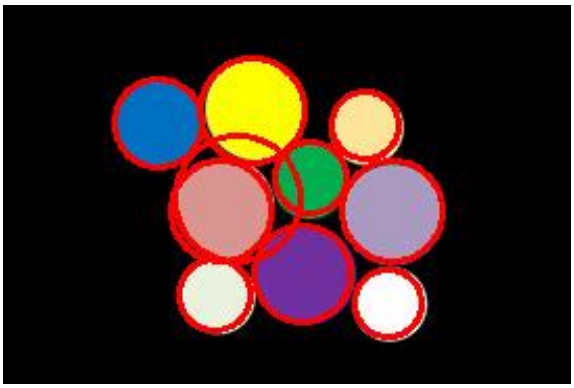
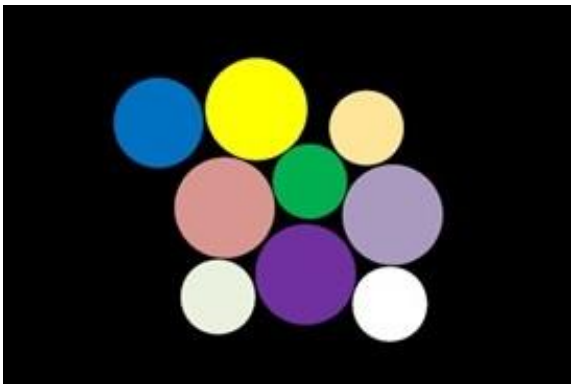
实验结果：

处理三张图片总用时不到 **0.3s,**且开始运行后处理速度越来越快。且精度较高，识别准确。

```
[ 87%] Built target exp_3
[100%] Linking CXX executable /home/hebuyong/dip_
[100%] Built target hw_9
Total_time_1:0.145275s
Total_time_2:0.094305s
Total_time_3:0.043202s
```





运行环境：Ubuntu16.04 LTS + OpenCV 3.0.4 + ROS-kinetic-full
代码语言：c++
过程：转灰度图->模糊处理->自适应全局阈值二值化处理->
边界检测->霍夫圆检测->绘制图像。
代码如下：

```
#include <stdlib.h>
#include <iostream>
#include <opencv2/opencv.hpp>
#include<ctime>

#define pi 3.14159265
using namespace cv;
using namespace std;

//基本全局阈值求最佳二值图像
Mat Global_threshold(Mat in){
    Mat out=in.clone();
    int sum=0;
    for (int i=0; i<in.rows; i++){
        for (int j=0; j<in.cols; j++){
            sum+=in.at<uchar>(i,j);
        }
    }
    int t=int(sum/in.cols/in.rows);
    int u1=0;int u2=255;
    while((u1+u2)/2-t<=5){
        int sum1=0,sum2=0;
        for (int i=0; i<in.rows; i++){
            for (int j=0; j<in.cols; j++){
                if(in.at<uchar>(i,j)<t){
                    sum1+=in.at<uchar>(i,j);
                }else{
                    sum2+=in.at<uchar>(i,j);
                }

            }
        }
        u1=int(sum1/in.cols/in.rows);
        u2=int(sum2/in.cols/in.rows);
        t=int((u1+u2)/2);
    }
    threshold(in, out, t, 255, CV_THRESH_BINARY);
    return out;
}
Mat Fast_circle_detect(Mat in){
    Mat out=in.clone();
    // imshow("original image",in);
    //转灰度图像
    Mat gray;
    cvtColor(in, gray, COLOR_BGR2GRAY);
    //模糊处理
    Mat gaussian;
    GaussianBlur(gray, gaussian, Size(7,7), 0, 0);
    //自适应全局阈值二值化处理
    Mat binary=Global_threshold(gaussian);
    // imshow("binary",binary);
```

```
    //边缘检测(canny)
    Mat canny;
    Canny(gaussian, canny, 100,180);//200,300
    // imshow("canny",canny);
    vector <Vec3f> circles;
    HoughCircles(canny, circles, HOUGH_GRADIENT, 1, canny.rows/17, 100, 20,
0, 45);
    for (int i = 0; i < circles.size(); i++) {
        circle(out, Point(circles[i][0], circles[i][1]), circles[i][2],
Scalar(0, 0, 255), 2);
    }
    // imshow("circles_detect",out);
    return out;
}
int main(int argc, char **argv)
{

    clock_t start,end;
    double endtime;
    Mat src,out;
    start=clock();
    src=imread("./src/hw_pkg_9/src/3.jpg");
    imshow("original_image_1",src);
    out=Fast_circle_detect(src);
    imshow("out_1",out);
    end=clock();
    endtime=(double)(end-start)/CLOCKS_PER_SEC;
    cout<<"Total_time_1:"<<endtime<<"s"<<endl;

    start=clock();
    src=imread("./src/hw_pkg_9/src/4.png");
    imshow("original_image_2",src);
    out=Fast_circle_detect(src);
    imshow("out_2",out);
    end=clock();
    endtime=(double)(end-start)/CLOCKS_PER_SEC;
    cout<<"Total_time_2:"<<endtime<<"s"<<endl;

    start=clock();
    src=imread("./src/hw_pkg_9/src/4.jpg");
    imshow("original_image_3",src);
    out=Fast_circle_detect(src);
    imshow("out_3",out);
    end=clock();
    endtime=(double)(end-start)/CLOCKS_PER_SEC;
    cout<<"Total_time_3:"<<endtime<<"s"<<endl;


    waitKey(0);
    return 0;
}
```