圆形物体特征计算

SZ170410221-朱方程

OpenCV实现

Code

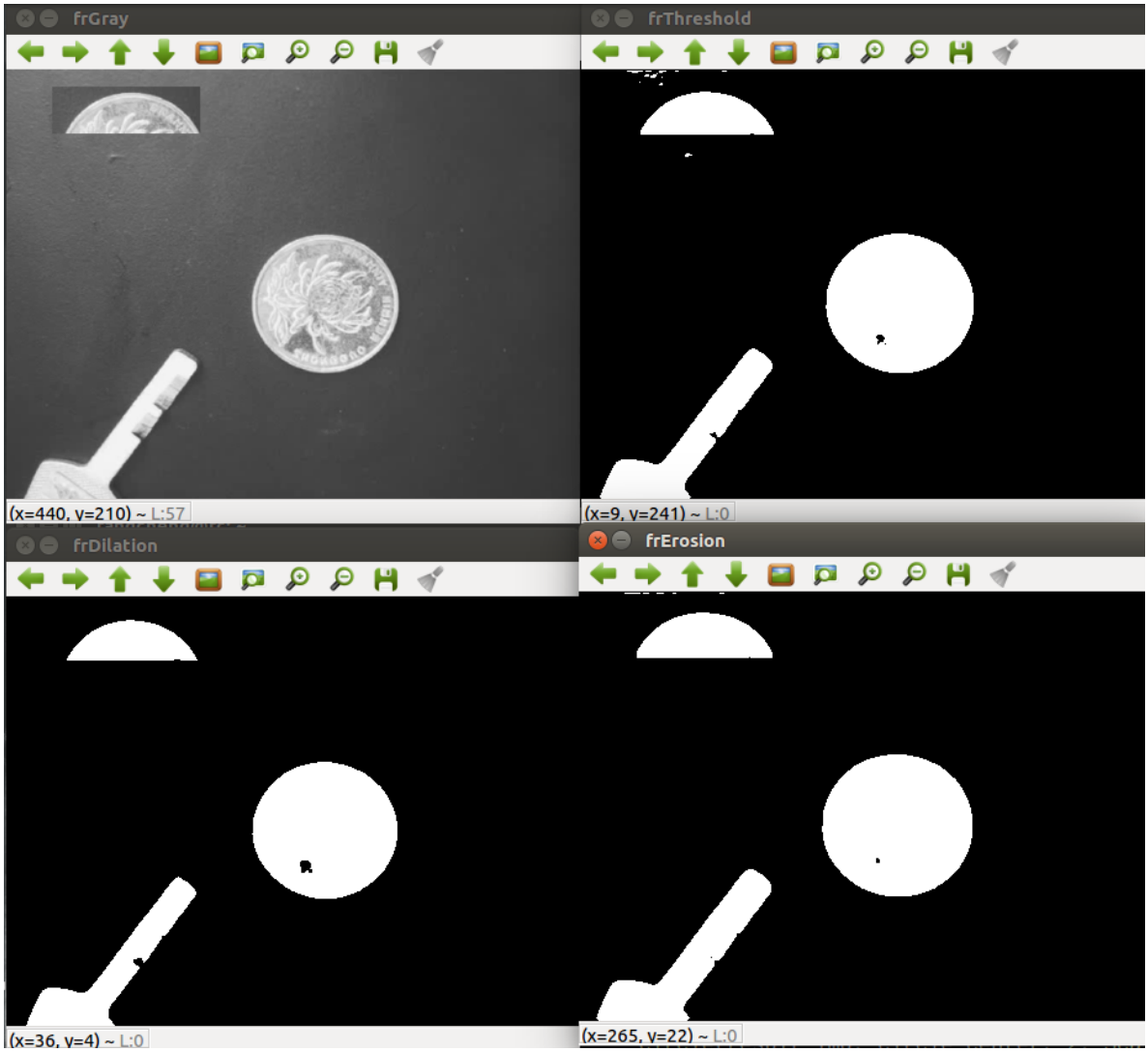
```
#include <stdlib.h>
#include <cv.h>
#include <highgui.h>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "std_msgs/Bool.h"
#include "std_msgs/Float32.h"
#include <geometry_msgs/Twist.h>
#include "sensor_msgs/Image.h"
#define LINEAR_X 0
using namespace cv;
using namespace std;
//筛选圆形区域，并计算面积，周长，圆心位置等
//对图像进行高斯模糊处理
void Gaussian(Mat input, Mat output, double sigma)
{
    //设计高斯滤波模板。当sigma固定时，模板尺寸越大，图像越模糊
    int MaskSize=3;
    int center_k=MaskSize/2;
    int center_l=MaskSize/2;
    double sum=0;   //模板权值之和
    double mask[MaskSize][MaskSize];
    //生成高斯滤波模板，对mask数组进行赋值操作
    for (int k = 0; k < MaskSize; k++ ){
        for (int l = 0; l < MaskSize; l++){
            mask[k][l] = exp( -(1.0)*( ((k-center_k)*(k-center_k)+(l-
center_l)*(l-center_l))/(2.0*sigma*sigma)) );
            sum += mask[k][l];
        }
    }
    //归一化模板权值
    for (int i = 0; i < MaskSize; i++){
        for (int j = 0; j < MaskSize; j++){
            mask[i][j] /= sum;
        }
    }
    //将模板函数与图像进行卷积
    for(int i=0;i<input.rows;i++){
        for(int j=0;j<input.cols;j++){
            double sum = 0;
            for (int k = 0; k <MaskSize; k++){
                for (int l = 0; l < MaskSize; l++){
```

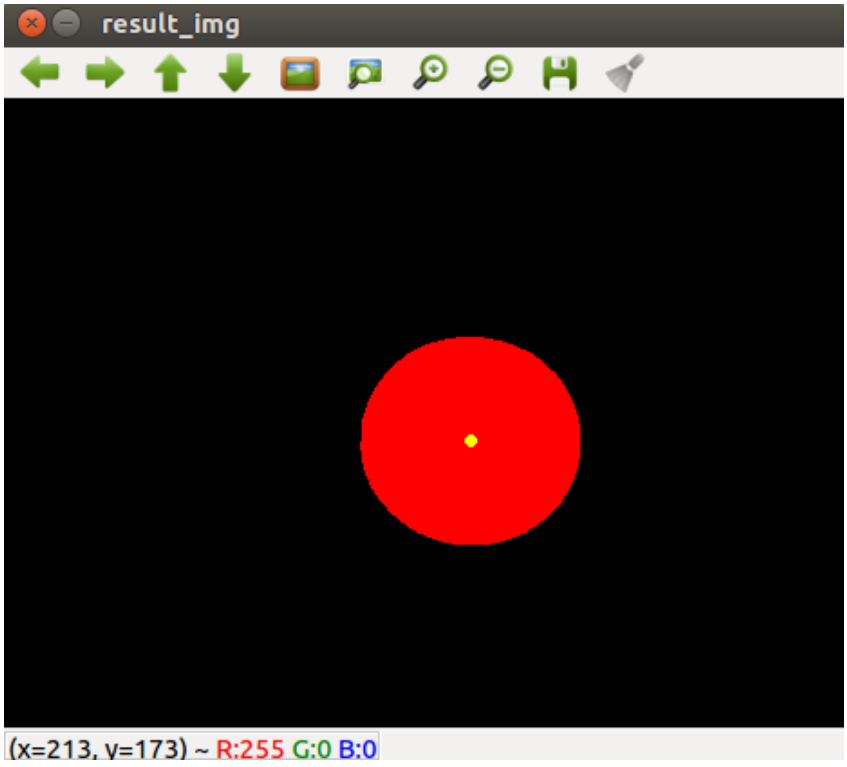
```
                    if(i+(k-center_k)>=0 && j+(l-center_l)>=0)    //判断。相当于将图像外面一圈像素的灰度值置零
                    sum = sum+input.at<uchar>(i+(k-center_k),j+(l-center_l))*mask[k][l];
                }
            }
            //对输出图像重新赋值
            output.at<uchar>(i,j)=sum;
        }
    }
}

// 膨胀(针对黑色)函数
void Dilate(Mat Src, Mat Dst)
{
    //定义膨胀结构元
    int Di_Size=3;
    int DilationMask[Di_Size][Di_Size]={{0,0,0},{0,0,0},{0,0,0}};
    for(int i=Di_Size/2;i<Src.rows-Di_Size/2;i++){
        for(int j=Di_Size/2;j<Src.cols-Di_Size/2;j++){
            int sum=0;
            for (int k = 0; k <Di_Size; k++){
                for (int l = 0; l < Di_Size; l++){
                    //膨胀要求图像区域的九个像素灰度值和模板有交集，有一个为0即可
                    //令图像区域九个像素块灰度值的和再除以255为sum，当sum<9，就表示不全为1，即有一个为0，击中
                    sum=sum+Src.at<uchar>(i-(k-Di_Size/2),j-(l-Di_Size/2))/255;
                }
            }
            if(sum<9)
                Dst.at<uchar>(i,j)=0;
            else
                Dst.at<uchar>(i,j)=255;
        }
    }
}

// 腐蚀(针对黑色)函数
void Erode(Mat Src, Mat Dst)
{
    //定义腐蚀结构元
    int Er_Size=5;
    for(int i=Er_Size/2;i<Src.rows-Er_Size/2;i++){
        for(int j=Er_Size/2;j<Src.cols-Er_Size/2;j++){
            int sum=0;
            for (int k = 0; k <Er_Size; k++){
                for (int l = 0; l < Er_Size; l++){
                    //腐蚀要求图像区域的九个像素灰度值和模板完全匹配，即全为0
                    //令图像区域九个像素块灰度值的和为sum，当且仅当sum=0，才有完全匹配
                    sum=sum+Src.at<uchar>(i-(k-Er_Size/2),j-(l-Er_Size/2));
                }
            }
            if (sum==0)
                Dst.at<uchar>(i,j)=0;
            else
```

```
                Dst.at<uchar>(i,j)=255;
        }
    }
}
void CircleExtraction(Mat input)
{
    // 寻找轮廓
    vector<vector<Point> > contours;
    vector<Vec4i> hireachy;
    findContours(input, contours, hireachy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE, Point());

    Mat result_img = Mat::zeros(input.size(), CV_8UC3);     // 创建与原图同大小的黑色背景
    Point circle_center;               //定义圆心坐标
    for (int t = 0; t < contours.size(); ++t)
    {
        // 面积过滤
        double area = contourArea(contours[t]);     //计算点集所围区域的面积
        if (area < 100)             //选出轮廓面积大于100的轮廓
            continue;
        // 横纵比过滤
        Rect rect = boundingRect(contours[t]);             // 求点集的最小直立外包矩形
        float ratio = float(rect.width) / float(rect.height);         //求出宽高比

        if (ratio < 1.1 && ratio > 0.9)       //因为圆的外接直立矩形肯定近似于一个正方形，因此宽高比接近1.0
        {
            drawContours(result_img, contours, t, Scalar(0, 0, 255), -1, 8, Mat(), 0, Point());    //画出圆
            printf("Area of the circle: %f\n", area);  //面积
            double perimeter = arcLength(contours[t], true);          //计算点集所围区域的周长
            printf("Perimeter of the circle : %f\n", perimeter);
            int x = rect.x + rect.width / 2;
            int y = rect.y + rect.height / 2;
            circle_center = Point(x, y);          //得到圆心坐标
            double diameter_area=2*sqrt(area/3.1415);
            printf("Diameter calculated with area : %f\n", diameter_area);//由面积计算直径
            double diameter_peri=perimeter/3.1514;
            printf("Diameter calculated with perimeter : %f\n", diameter_peri);//由周长计算直径
            double diameter=(diameter_area+diameter_peri)*0.5;
            printf("Diameter of the circle : %f\n", diameter);//二者平均值作为直径
            printf("Coordinates of the center point: width %d, height %d\n \n \n",circle_center.x,circle_center.y);
            circle(result_img, circle_center, 2, Scalar(0, 255, 255), 2, 8, 0);
        }
    }
    imshow("result_img", result_img);

}

```

```
int main(int argc, char **argv)
{

    VideoCapture capture;
        capture.open(0);//打开 zed 相机

    ROS_WARN("*****START");
    ros::init(argc,argv,"trafficLaneTrack");
        ros::NodeHandle n;

    ros::Rate loop_rate(10);
        ros::Publisher pub = n.advertise<geometry_msgs::Twist>
("/smoother_cmd_vel", 5);
    if (!capture.isOpened())
    {
        printf("摄像头没有正常打开\n");
        return 0;
    }
    waitKey(1000);
    Mat frame;
    while (ros::ok())
    {
        Mat frIn = imread("/home/fangcheng/Library/MV4.bmp",1);
            if(frIn.empty())
        {
            break;
        }
        Mat frGray;
        cvtColor(frIn, frGray, CV_RGB2GRAY);//RGB彩色图转换成Gray灰度图
        imshow("frGray",frGray);
        //高斯模糊处理
        Mat frGaussian = frGray.clone();
        Gaussian(frGray,frGaussian,9);
        Mat frThreshold=frGray.clone();
        cv::threshold(frGaussian, frThreshold, 125, 255, THRESH_BINARY);//
第三个参数为阈值
        imshow("frThreshold",frThreshold);
        //开操作
        Mat frDilation=frThreshold.clone();
        Dilate(frThreshold,frDilation);
        imshow("frDilation",frDilation);
        Mat frErosion=frThreshold.clone();
        Erode(frDilation,frErosion);
        imshow("frErosion",frErosion);
        //特征提取与计算
        CircleExtraction(frErosion);
        ros::spinOnce();
        waitKey(5);
    }
    return 0;
}

```

Result


frGray
(x=440, y=210) ~ L:57
frThreshold
(x=9, y=241) ~ L:0
frDilation
(x=36, y=4) ~ L:0
frErosion
(x=265, y=22) ~ L:0


result_img
(x=213, y=173) ~ R:255 G:0 B:0

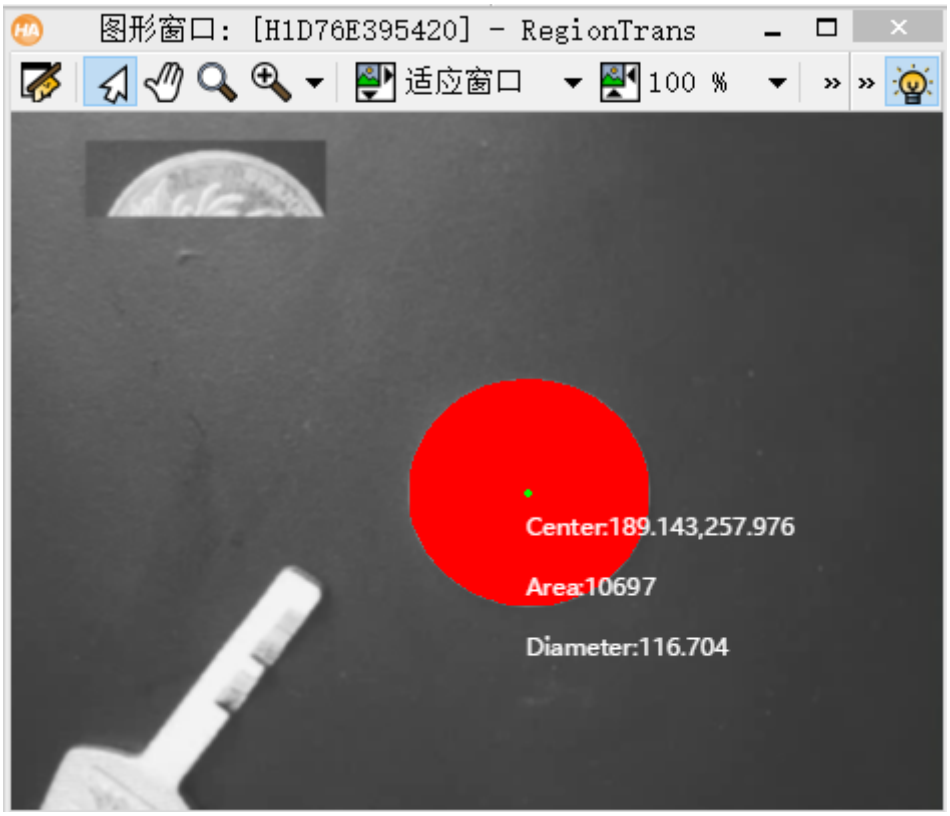
```
Area of the circle: 10947.000000
Perimeter of the circle : 391.504614
Diameter calculated with area : 118.061697
Diameter calculated with perimeter : 124.231965
Diameter of the circle : 121.146831
Coordinates of the center point: width 258, height 189
```

Halcon实现

Code

```
read_image (Image, 'C:/Users/62525/Desktop/The_Second_Semester_of_Junior_Year/Machine Vision/实验图像.bmp')
//高斯滤波
gauss_filter (Image, Image, 3)
//获取圆形ROI
gen_circle (ROI_0, 185.063, 256.074, 76.5054)
//减去不需要的区域
reduce_domain (Image, ROI_0,ImageReduced)
get_image_size (ImageReduced, Width,Height)
dev_open_window_fit_image (ImageReduced, 0,0, Width/3, Height/3, WindowHandle)
dev_display (ImageReduced)
//二值化
threshold (ImageReduced, Regions, 125, 255)
dev_set_color ('red')
//区域填充
shape_trans (Regions, RegionTrans, 'convex')
//获取图像中心点的位置
area_center (RegionTrans, Area, Row, Column)
//计算直径
Diameter:=2*sqrt(Area/3.1415926)
//显示圆心，半径为2并用绿色表示
dev_set_color ('green')
disp_circle(WindowHandle,Row, Column, 2)
//将提取到的体征信息显示在图像窗口中，文字颜色为白色
dev_set_color('white')
set_tposition (WindowHandle, Row+10, Column)
write_string (WindowHandle, 'Center:'+Row+','+ Column)
set_tposition (WindowHandle, Row+40, Column)
write_string (WindowHandle, 'Area:'+Area)
set_tposition (WindowHandle, Row+70, Column)
write_string (WindowHandle, 'Diameter:'+Diameter)
```

Result


图形窗口: [H1D76E395420] - RegionTrans
适应窗口
100 %
Center:189.143,257.976
Area:10697
Diameter:116.704