实验三　图像特征检测

1　实验目的

① 掌握图像中边缘、直线、圆形的检测原理

② 设计算法实现图像的边缘检测和基于霍夫变换的直线及圆检测

2　实验仪器

① 机器人硬件：开源移动机器人、笔记本

② 笔记本软件：ros-kinetic-full、opencv（仅用于实现图像的读取操作）

3　实验原理

3.1 边缘检测

边缘检测算法主要是基于图像强度的一阶和二阶导数，但是导数通常对噪声很敏感，因此必须采用滤波器来抑制噪声影响。为了进一步提高边缘检测的准确性，需要对图像进行增强，既将图像灰度点邻域强度值有显著变化的点凸显出来，经过增强的图像，往往在邻域中由很多点的梯度值较大，所以需要采用某些方法做取舍。

这里边缘检测以 Canny 算法为例，主要步骤有如下 5 步：

1) 高斯滤波

$$G(x,y)=\sum_{x-m}^{x+m}\sum_{y-m}^{y+m}\exp|-\frac{x^2+y^2}{2\sigma^2}|,m=\frac{n-1}{2}$$

n 为高斯滤波器的尺寸大小。

2) 计算梯度的幅值和方向

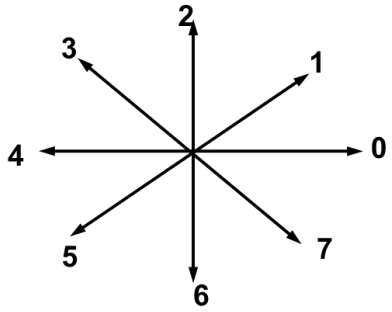
$$G_x=\frac{1}{2}\bullet\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix},G_y=\frac{1}{2}\bullet\begin{bmatrix}1 & 1\\ -1 & -1\end{bmatrix}$$

$$|B|=\sqrt{B_x^2+B_y^2},\theta=\arctan\frac{B_y}{B_x}$$

3) 非极大值抑制

如下图所示，M 的邻域中，如果 M 不是最大值，则 M=0；否则，保持 M 不变，这样可以保持局部最大梯度来找到边缘。

哈工大深圳自动化2018级本科班



4	3	2
1	M	1
2	3	4

Fig 1 非极大值抑制图

4) 滞后阈值

滞后阈值需要两个阈值（高阈值和低阈值）：

① 若某一像素的幅值超过高阈值，则该像素保留为边缘像素；

② 若某一像素的幅值小于低阈值，则该像素被排除；

③ 若某一像素的幅值处在两个阈值之间，则该像素仅仅在连接到一个高于高阈值的像素时被保留。

5) 边缘连接

利用 canny 算子对 lena 图进行边缘检测，得到的二值边缘图如图 2 所示，图中不是边缘的点灰度为 0，是边缘的点灰度为 255。

Fig 2 canny 边缘检测示意图

3.2 霍夫变换

霍夫变换是图像处理中的一种特征提取技术，是图像处理领域内从图像中检测几何形状的基本方法之一。经典霍夫变换用来检测图像中的直线，后来霍夫变

换扩展到任意形状物体的识别，多为圆和椭圆。霍夫变换运用两个坐标空间之间的变换将在一个空间中具有相同的曲线或直线映射到另一个坐标空间的一个点形成峰值，从而把检测任意形状的问题转化为统计峰值的问题。该方法的一个突出优点是分割结果的鲁棒性，即对数据的不完全或噪声不是非常敏感。

3.2.1 霍夫线变换

图像二维空间中的一条直线在极坐标系下，如图 3 所示，可由极径(r)和极角(θ)表示，即：

$$r = x\cos\theta + y\sin\theta$$

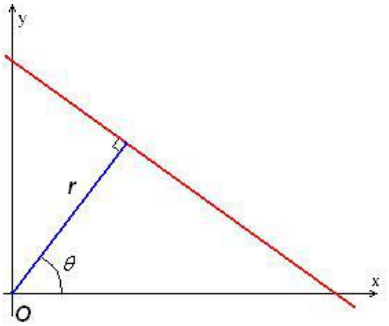


Fig 3　直线的极坐标表示方法

对于图像空间中的某个点，可以将通过这个点的一簇直线统一定义为：

$$r = x\sin\theta + y\cos\theta$$

对于给定点(x, y)，所有通过该点的直线，在参数空间中对应同一条正弦曲线。图像空间中的每个共线点，在参数空间对应的正弦曲线相交于一点(r', θ')，如图 4 所示：

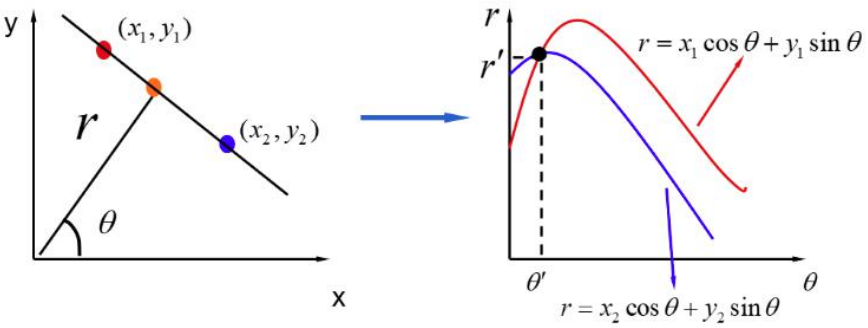


Fig 4　图像中的共线点在参数空间下的表示

霍夫线变换就是把图像空间中的直线变换到参数空间中的点，通过统计特性

来解决检测问题。在极坐标参数空间中，寻找检测正弦曲线的交点，若通过某点(r,θ)的正弦曲线数最多，则该点即为所求直线的参数。霍夫空间中，曲线的交点次数越多，所代表的参数越确定。一般通过设置直线上像素点的阈值来定义多少条曲线交于一点认为是检测到了一条直线。霍夫变换检测直线的主要步骤有如下几步：

1) 将彩色图像转化为灰度图，并对灰度图做边缘检测得到二值边缘图（同3.1）；
2) 参数空间离散化：对直线方程的参数(r,θ)离散化，并给出(r_{min},r_{max})和$(\theta_{min},\theta_{max})$，划分为有限个等间距的离散值，使参数空间量化为一个个等大小网格。
3) 设置累加器 A：为每个网格单元设置累加器。A 表示$(r_{min}:r_{max},\ \theta_{min}:\theta_{max})$，每个网格的累加器初始设置为 0；
4) 对图像空间中每个像素点坐标值(x,y)，计算参数空间对应的曲线方程，将该曲线穿过的格子的计数值加一。
5) 最后，*遍历*$A(i,j)$中的寻找累加计数大于某阈值 M 格子，其坐标(r_m,θ_m)即为检测到的直线参数。利用 cvtColor()将二值边缘图转换为 RGB 图，并将检测到的所有直线在图中画出来。

直线检测结果如图 5 所示：

哈工大深圳自动化2018级本科生

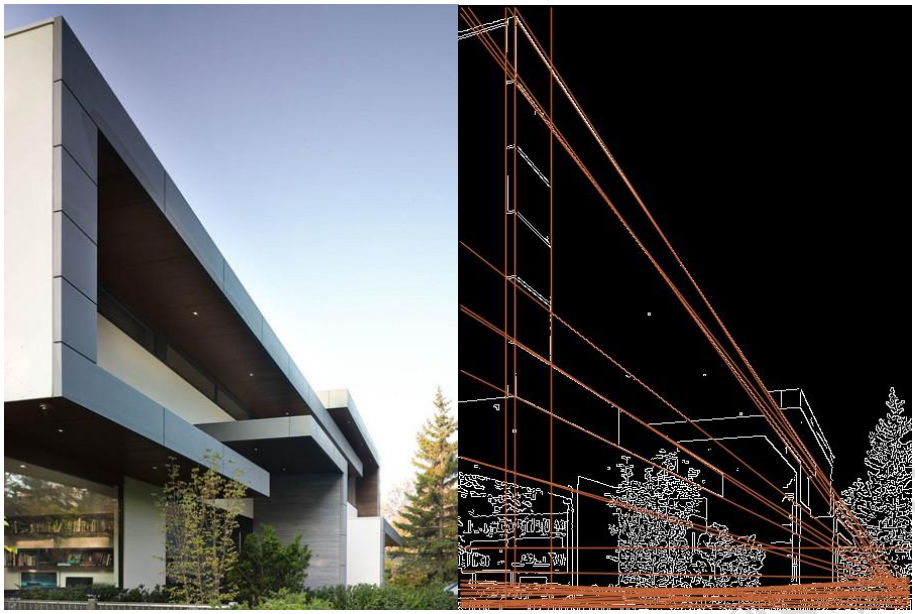


Fig 5 霍夫线变换检测图

3.2.2 霍夫圆变换

霍夫圆变换的基本原理与霍夫线变换的原理类似，只是点对应的二维极径极角空间被三维的圆心点(x_i, y_i)和半径 r 空间所取代，即利用三个参数表示圆：

$$(x - x_{center})^2 + (y - y_{center})^2 = r^2$$

利用霍夫梯度法检测圆的基本步骤有如下几步：

1) 将彩色图像转化为灰度图,然后对灰度图做边缘检测得到二值化的边缘图；

2) 设定检测半径和角度范围(r_{min}: r_{max}, θ_{min}: θ_{max}),设置累加器 A(x,y,r)

3) 将边缘点从图像空间(x,y)对应到参数空间(x_center, y_center, r) 大致步骤如下：

for ***edge_point*** in {all edge_points} //遍历边缘点

for ***r*** in (***r_min: r_max***) //在指定范围内遍历半径

for ***theta*** in (***theta_min: theta_max***) //在指定范围内遍历圆心角

计算（***edge_point, r, theta***）所对应的圆心(***x_center, y_center***)

累加器 ***A***(***x_center, y_center, r***)加一

end

end

end

4) 设定阈值，遍历累加器中大于阈值的格子，(x_center, y_center, r) 机位检测到的圆**参数**，利用 cvtColor()将二值边缘图转换为 RGB 图，并将检测到的所有圆在图中画出来。

圆检测结果如图 6 所示：

Fig 6 霍夫圆变换检测图

4 实验内容

4.1 边缘检测

① 使用 imread 读取图片

② 将 RGB 图转化为灰度图

③ 设计边缘检测函数(任选一种)

④ imshow 输出边缘检测结果

边缘检测结果示意图如下：

哈工大深圳自动化学院 技术科班

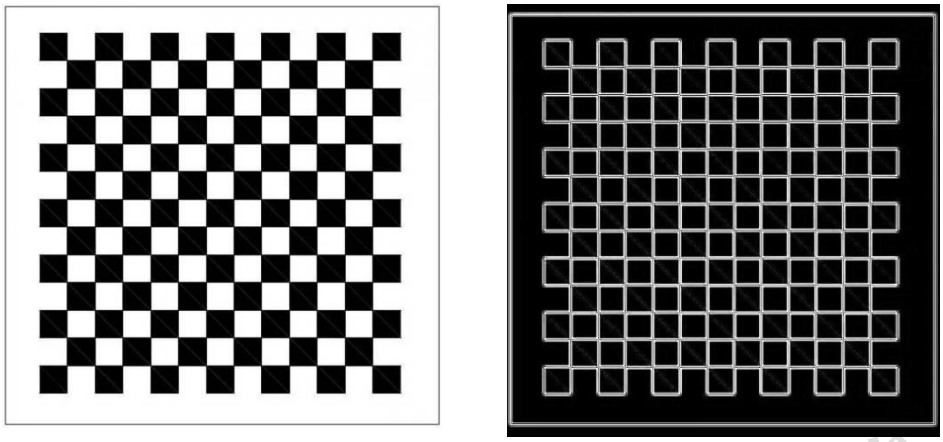


Fig 7 边缘检测结果

4.2 霍夫线变换

① 使用 imread 读取图片，进行图像预处理，得到二值边缘图（同 4.1）

② 设计霍夫变换函数对图片进行直线检测

③ imshow 输出直线检测结果

直线检测结果如下图所示：

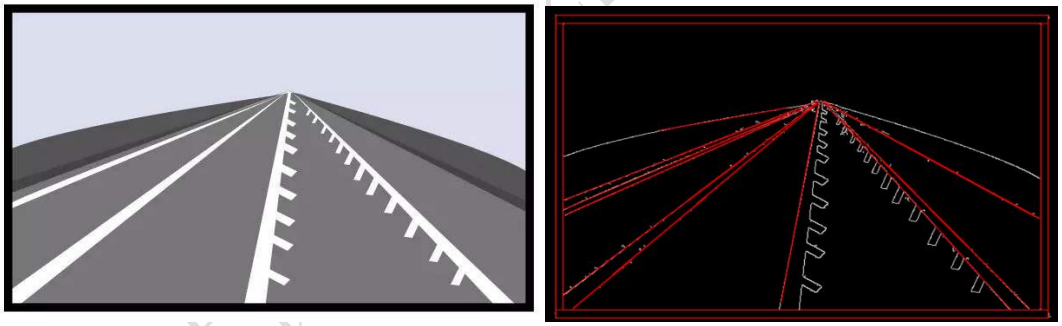


Fig 8 线检测结果

4.3 霍夫圆变换

① 使用 imread 读取图片，进行图像预处理，得到二值边缘图（同 4.1）

② 设计霍夫变换函数对图片进行圆检测

③ Imshow 输出圆检测结果

圆检测结果如下图所示：

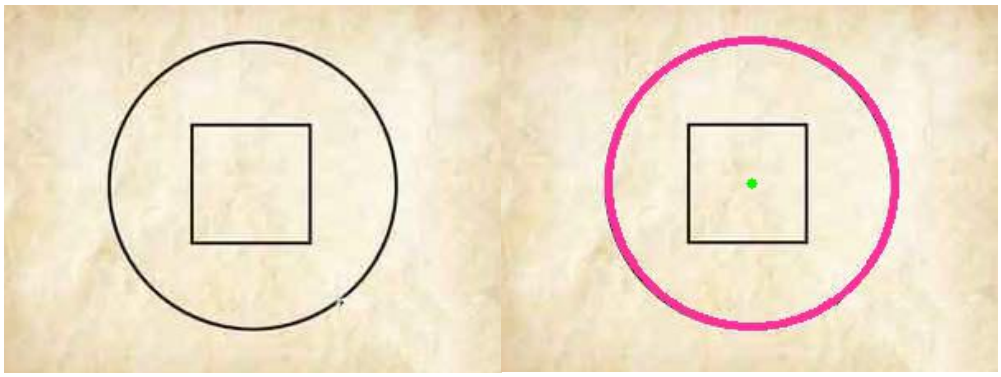


Fig 9 圆检测结果

哈工大深圳自动化2018级本科班

附录：（代码框架）

```
#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;
```

```
//////////////////////边缘检测/////////////////
//边缘检测函数
void EdgeDetector(Mat input, Mat output){

}
//////////////////////霍夫线变换/////////////////
Mat Hough_Line(Mat output){

}

//////////////////////霍夫圆变换/////////////////
Mat Hough_Circle(Mat output){

}
int main(int argc, char **argv)
{
    VideoCapture capture;
     capture.open(1);//打开 zed 相机


    ROS_WARN("*****START");
    ros::init(argc,argv,"trafficLaneTrack");//初始化 ROS 节点
        ros::NodeHandle n;

        // ros::Rate loop_rate(10);//定义速度发布频率
        ros::Publisher                                pub                                =
n.advertise<geometry_msgs::Twist>("/smoother_cmd_vel", 5);//定义速度发布器


    if (!capture.isOpened())
    {
        printf("摄像头没有正常打开，重新插拔工控机上当摄像头\n");
        return 0;
    }
    waitKey(1000);
    Mat frame;//当前帧图片
    int nFrames = 0;//图片帧数
    int frameWidth = capture.get(CV_CAP_PROP_FRAME_WIDTH);//图片宽
    int frameHeight = capture.get(CV_CAP_PROP_FRAME_HEIGHT);//图片高


    while (ros::ok())
    {
        capture.read(frame);
```

哈工大深圳自动化2018级本科班

```
        if(frame.empty())
        {
            break;
        }

        Mat frIn = frame(cv::Rect(0, 0, frame.cols / 2, frame.rows));//截取 zed 的左目图片

        // 灰度图转换
         CvtColor()
         // 边缘检测函数
         EdgeDetector();

         // 线检测
         Hough_Line();
         // 圆检测
         Hough_Circle() ;
         imshow("1",output);

        geometry_msgs::Twist cmd_red;

        // 车的速度值设置
        cmd_red.linear.x = LINEAR_X;
        cmd_red.linear.y = 0;
        cmd_red.linear.z = 0;
        cmd_red.angular.x = 0;
        cmd_red.angular.y = 0;
        cmd_red.angular.z = 0.2;

        pub.publish(cmd_red);

        ros::spinOnce();
        waitKey(5);

    }
    return 0;
}
```

哈工大深圳自动化2018级本科班