第二十四课 HiBot 车道线检测和行驶

1. 前言

本节课程学习基于 opencv 的图像处理技术，将机器人放置在车道内通过摄像头采集车道数据，对车道图像进行处理和检测。然后控制机器人在车道线内移动，同时实现大弯、小弯、S 弯、十字路口等路道检测和运动。

本节课程的源码包位于树莓派的/home/huike/robot_ws/src/track_detection/路径下，包名是 track_detection，在学习过程中可以查看源码了解车道检测和控制机器人的原理。

2. 源码讲解

1> 构造函数说明

在 Track_Detection::Track_Detection 构造函数中，通过 param 加载配置参数文件，同时创建三个消息发布器，用来发布原始 RGB 数据和车道二值化中线数据，二者都是 image 消息类型。然后打开一个摄像头，同时配置摄像头的参数：

```
nh_private.param<bool>("StartMove", this->StartMove, false);
nh_private.param<double>("MixKP", this->MixKP, 0.025f);
nh_private.param<double>("MaxKP", this->MaxKP, 0.04f);
nh_private.param<double>("HiBotSpeed", this->HiBotSpeed, 0.2);
```

"StartMove"当参数为 true 是，机器人开始沿车道运动。

"MixKP"机器人通过小弯时的 PID 参数值。

"MaxKP"机器人通过大弯时的 PID 参数值。

"HiBotSpeed"机器人沿车道运动时，X 方向的线速度。单位 m/s。

```
this->Image_pub = it.advertise("camera/image", 1);
this->RgbImage_pub = it.advertise("camera/rgb", 1);
this->cmd_pub = n.advertise<geometry_msgs::Twist>("/cmd_vel",1);
```

Image_pub：创建消息发布器，发布处理后的 image 数据，主要包含车道和边沿分割，中线显示。

RgbImage_pub：创建消息发布器，发布原始的摄像头数据，在 rviz 中可以查看。

cmd_pub：创建消息发布器，发布机器人运动指令。

```
capture.open(0);
if(!capture.isOpened())
{
    ROS_INFO("Open video0 is error!");
    ros::shutdown();
}

capture.set(CV_CAP_PROP_FRAME_HEIGHT, 240);
capture.set(CV_CAP_PROP_FRAME_WIDTH, 320);
capture.set(CV_CAP_PROP_FPS, 30.0);
if (!capture.set(CV_CAP_PROP_FOURCC, CV_FOURCC('M', 'J', 'P', 'G')))
{
    ROS_INFO("set format failed \n");
}
```

打开摄像头，并通过 capture 类来配置摄像头参数。

2> main 函数说明

```
ros::init(argc, argv, "HiBot_track_detection");
ROS_INFO("[ZHOUXUEWEI] Track_Detection_node start!");

Track_Detection HiBotTrackDetection;
```

初始化 ROS 节点，同时创建一个 HiBotTrackDetection 对象。

```
ros::Rate loop_rate(20);
while(ros::ok())
{
    HiBotTrackDetection.Open_Imread();
    HiBotTrackDetection.Image_Filter_Process();
    HiBotTrackDetection.ImageMiddleDetection();

    ros::spinOnce();
    loop_rate.sleep();
}
```

定义一个 ros::Rate 对象，用来控制程序的循环频率。此处为 20HZ.

3> Open_Imread 方法说明

```
capture >> frame->image; //流的转换

if(frame->image.empty())
{
    ROS_INFO( "Failed to capture frame!" );
    ros::shutdown();
}

frame->header.stamp = ros::Time::now();
this->RgbImage_pub.publish(frame->toImageMsg());

this->SrcImageRead = frame->image;
```

通过 capture 读取图像数据流，然后发布"camera/rgb"话题，话题包含原始 RGB 图像数据。然后将数据流透传到 Mat 格式的 SrcImageRead 变量中，

4>Image_Filter_Process 方法说明

```
Mat GyraImage, GaussianImage, autoImage;

resize(this->SrcImageRead, this->SrcImageRead, Size(), 0.5, 0.5);
cvtColor(this->SrcImageRead, GyraImage, COLOR_RGB2GRAY);

ros::param::get("/Track_Detection_node/GyraThreshold", GyraThreshold);

//adaptiveThreshold(GyraImage, DestImageOut, 255, CV_ADAPTIVE_THRESH_MEAN_C, CV_THRESH_BINARY_INV, 45, 3);
threshold(GyraImage, DestImageOut, 0, 255, THRESH_OTSU | THRESH_BINARY);
```

Resize：重新调整图像大小。

cvtColor：将图像转化为灰度图像。

Threshold：将灰度图像转化为阈值自适应的二值化图像。

5>ImageMiddleDetection 方法说明

首先了解一下 `unsigned short MiddleArray[120][4];` 二位数组，数组为 120 行 4 列数组，

主要存储每一行图像的左右车道和车道中线。

```
for (int j = MIDDLE_VALUE; j > 1; j--)       // left line found
{
    if((PtrRows[j] - PtrRows[j-1]) != 0)
    {
        this->MiddleArray[i][0] = j;
        break;
    }
    this->MiddleArray[i][0] = 0;
}

for (int j = MIDDLE_VALUE; j < DestImageOut.cols; j++)       // left line found
{
    if((PtrRows[j] - PtrRows[j-1]) != 0)
    {
        this->MiddleArray[i][1] = j;
        break;
    }
    this->MiddleArray[i][1] = DestImageOut.cols;
}
```

左边车道线和右边车道线的寻找，

```
this->MiddleArray[i][2] = (this->MiddleArray[i][0] + this->MiddleArray[i][1]) / 2;  //middle value
DestImageOut.at<uchar>(i, this->MiddleArray[i][2]) = 0;
```

计算中值，修改二值化图像的原始数据，修改的目的是在 rviz 中可以看到识别的中线。

```
this->HiBotMove_Control();

sensor_msgs::ImagePtr imageMsg = cv_bridge::CvImage(std_msgs::Header(), "mono8", DestImageOut).toImageMsg(); //mono8
this->Image_pub.publish(imageMsg);
```

调用 HiBotMove_Control 方法，同时发布二值化图像话题。

6>HiBotMove_Control 方法说明

```
for (int i = 1; i < PROSPECT_VALUE; i++)
{
    if(fabs(MiddleArray[DestImageOut.rows-i][2] - MiddleArray[DestImageOut.rows-i-1][2]) > 10)  //寻找有效的前瞻值
    {
        EffectiveValue = i;
        break;
    }
    if(i == PROSPECT_VALUE-1)
    {
        EffectiveValue = PROSPECT_VALUE-1;
    }
}
```

PROSPECT_VALUE：是机器人用于决策的前瞻值，通俗一点就是机器人向前看多远。

上面代码是在设定的前瞻范围内通过高斯滤波寻找有效的中值。

```
float SlopeValue = (this->MiddleArray[DestImageOut.rows-1][2] - this->MiddleArray[DestImageOut.rows - EffectiveValue][2]) / ((EffectiveValue) *1.0f);
```

计算有效中值的斜率，斜率在一定程度上决定了转弯的半径。

```
for (int i = 0; i < EffectiveValue; i++)
{
    MiddleAverageVal += this->MiddleArray[DestImageOut.rows-i-1][2];
}
MiddleAverageVal = MiddleAverageVal/EffectiveValue;

float Error = MIDDLE_VALUE - MiddleAverageVal;      //P
```

计算有效前瞻内中值的平均值，平均值用于 PID 控制器的反馈值。然后计算机器人当前偏离中线的误差。

```
ros::param::get("/Track_Detection_node/MixKP", this->MixKP);
ros::param::get("/Track_Detection_node/MaxKP", this->MaxKP);
```

加载最新的 P 参数，因为在代码运行过程中我们可以通过命令修改参数值。

```
if((EffectiveValue < 20) && (fabs(SlopeValue) > SLOPE_VALUE))   //斜率大于定值或者前瞻小于定值时，使用较大的PID参数
{
    OutPutVth = Error * this->MaxKP;
```

计算 PID 输出。

```
ros::param::get("/Track_Detection_node/StartMove", this->StartMove);
ros::param::get("/Track_Detection_node/HiBotSpeed", this->HiBotSpeed);
(this->StartMove)?(this->HiBotCmd_Vel(this->HiBotSpeed, OutPutVth)):(this->HiBotCmd_Vel(0.0, 0.0));
```

加载机器人的 StartMove 参数和 HiBotSpeed 参数，通过 HiBotCmd_Vel 方法控制机器人移动。

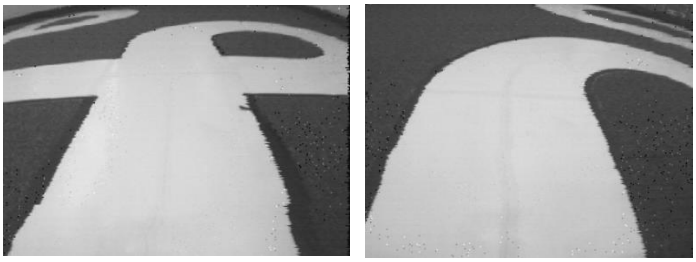
7>HiBotCmd_Vel 方法说明

```
geometry_msgs::Twist  twistMsg;

twistMsg.linear.x = vx;
twistMsg.angular.z = vz;

this->cmd_pub.publish(twistMsg);
```

创建一个 Twist 消息，赋值 x 方向的线速度和 z 方向的角速度，然后发布控制话题。

3. 测试代码

在测试之前确保已经在地面粘贴了车道，车道内部为白板边线为黑色，如下图所示。同时确保机器人安装了摄像头，摄像头朝正前方并且向下倾斜。



将机器人放置在车道的中间，打开HiBot的电源。等待一段时间后，Ubuntu主机连接树莓派开放的wifi。网络连接正常后，打开新的终端通过ssh连接到树莓派，同时运行机器人的启动节点。

```
huanyu@ubuntu:~$ ssh huike@192.168.12.1
huike@192.168.12.1's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.9.80-v7+ armv7l)
```

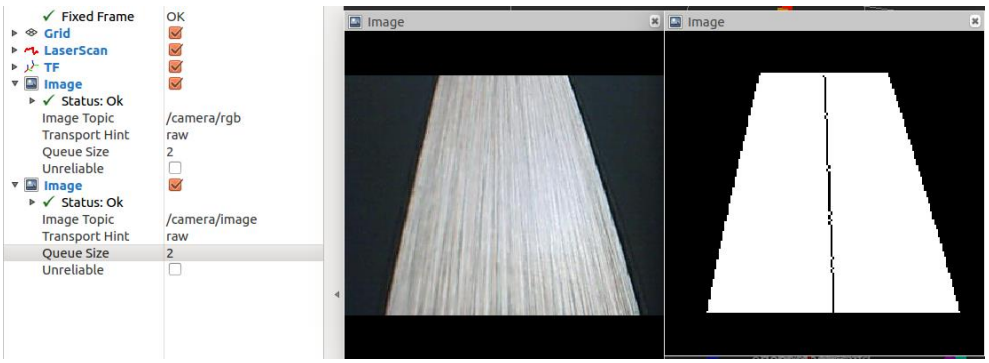
```
huike@huike-desktop:~$ roslaunch huanyu_robot_start Huanyu_robot_start.launch
```

```
[ INFO] [1455216978.297121854]: [ZHOUXUEWEI] Serial Port opened
[ INFO] [1455216978.352874348]: output frame: odom_combined
[ INFO] [1455216978.353334078]: base frame: base_footprint
[ INFO] [1455216980.324366366]: Initializing Odom sensor
[ INFO] [1455216980.325384885]: Initializing Imu sensor
[ INFO] [1455216980.374103208]: Odom sensor activated
[ INFO] [1455216980.374802100]: Imu sensor activated
[ INFO] [1455216980.380581194]: Kalman filter initialized with odom measurement
```

打开新的终端ssh连接到树莓派，运行车道检测的源码，不用担心机器人会马上移动，因为会有参数控制机器人是否开始运动。

```
huike@huike-desktop:~$ roslaunch track_detection Track_Detecation.launch
```

打新新的终端运行rviz可视化工具，加载两个image消息，选择相应的话题名称就可以看到下面显示的图像数据，左边是原始彩色图像，右边是二值化图。在二值化的图中可以看到车道的中线。HiBot会沿着中线移动。



当所有操作完成后，开始让机器人移动。打开新的终端，通过rosparam set设置参数，rosparam get查看参数。

```
huanyu@ubuntu:~$ rosparam get /Track_Detection_node/StartMove
false
huanyu@ubuntu:~$ rosparam set /Track_Detection_node/StartMove true
```

当StartMove设置为true时，机器人开始沿着车道移动。在移动过程中我们可以调节大弯和小弯的MaxKP，MixKP参数。不同的参数机器人转弯或发生变化，参数越大机器人的调节就会更快，转弯半径也会增大。

```
huanyu@ubuntu:~$ rosparam get /Track_Detection_node/MaxKP
0.019
huanyu@ubuntu:~$ rosparam get /Track_Detection_node/MixKP
0.016
huanyu@ubuntu:~$
```

```
huanyu@ubuntu:~$ rosparam set /Track_Detection_node/MixKP 0.017
huanyu@ubuntu:~$ rosparam get /Track_Detection_node/MixKP
0.017
huanyu@ubuntu:~$
```

直到机器人运动效果很好时，记录下两个参数值，为了使参数永久有效，需要将写入到Track_Detecation.launch文件中，文件在树莓派中的路径如下：

```
huike@huike-desktop:~/robot_ws/src/track_detection/launch$ pwd
/home/huike/robot_ws/src/track_detection/launch
huike@huike-desktop:~/robot_ws/src/track_detection/launch$ ls
Track_Detecation.launch
huike@huike-desktop:~/robot_ws/src/track_detection/launch$
```

通过 vim 修改参数文件。打开新的终端 ssh 连接到树莓派，用 vim 打开上面的 launch 文件。或者 nfs 挂载到 Ubuntu 主机，通过任意文件编辑器修改参数：

```
huike@huike-desktop:~$ sudo vim robot_ws/src/track_detection/launch/Track_Deteca
tion.launch
```

```
<launch>

  <node pkg="track_detection" type="Track_Detection_node" name="Track_Detection_node" output="screen">
        <param name="GyraThreshold" type="int"          value="60"/>
        <param name="StartMove"     type="bool"         value="false"/>
        <param name="MixKP"             type="double"   value="0.016"/>
        <param name="MaxKP"             type="double"   value="0.019"/>
        <param name="HiBotSpeed"     type="double"    value="0.2"/>
        </node>


</launch>
```

只需要修改参数后面的数值即可。