

视觉巡线小车比赛实验报告

姓名	学号	专业	任务
陈安德	18340009	人工智能与大数据	控制、地图设计
陈邦铠	18340010	计算机科学与技术	控制、撰写报告
陈琮昊	18340013	人工智能与大数据	通信、图像处理

一、实验内容：

在 V-REP 仿真环境下，每个队伍制作一条主场路径，加上给出的统一路径，利用图像某些特征的区别，通过视觉巡线来进行反馈控制，让给定的小车能够顺利抵达终点。

二、实验过程：

1.实验环境：





Windows 下，用 Python (Version 3.6) 进行远程控制，opencv (Version 3.4.2.16) 进行图像处理。

2.算法介绍：

1.通信部分：

通过建立 Python 端和 V-REP 端的通信来进行视觉巡线、并控制小车。

为了能够成功建立通信，需要将 V-REP 自带的一些端口复制至工作目录下：

 remoteApi.dll	2020/1/10 16:30	应用程序扩展	76 KB
 Robot_Control.py	2020/12/29 22:40	JetBrains PyCharm	8 KB
 sim.py	2020/1/10 16:30	JetBrains PyCharm	71 KB
 simConst.py	2019/11/12 14:25	JetBrains PyCharm	43 KB

Robot_Control.py 为 Python 端实现，其功能为建立与 V-REP 端的连接、获取图像，通过对图像进行处理从而完成巡线，以及对小车的控制；其他三个文件则是远程遥控所需接口。

通信的实现比较简单，到 V-REP 官网查看相应函数即可，主要内容如下：

```
sim.simxFinish(-1)          # 每次建立连接前需关掉之前的连接
clientId = sim.simxStart("127.0.0.1", 19999, True, True, 5000, 5) # 建立连接
if clientId != -1:           # 连接成功
    .....                   # 执行相关操作
sim.simxFinish(clientId)    # 结束连接
```

还需要在 V-REP 端脚本里面添加这样一句话：

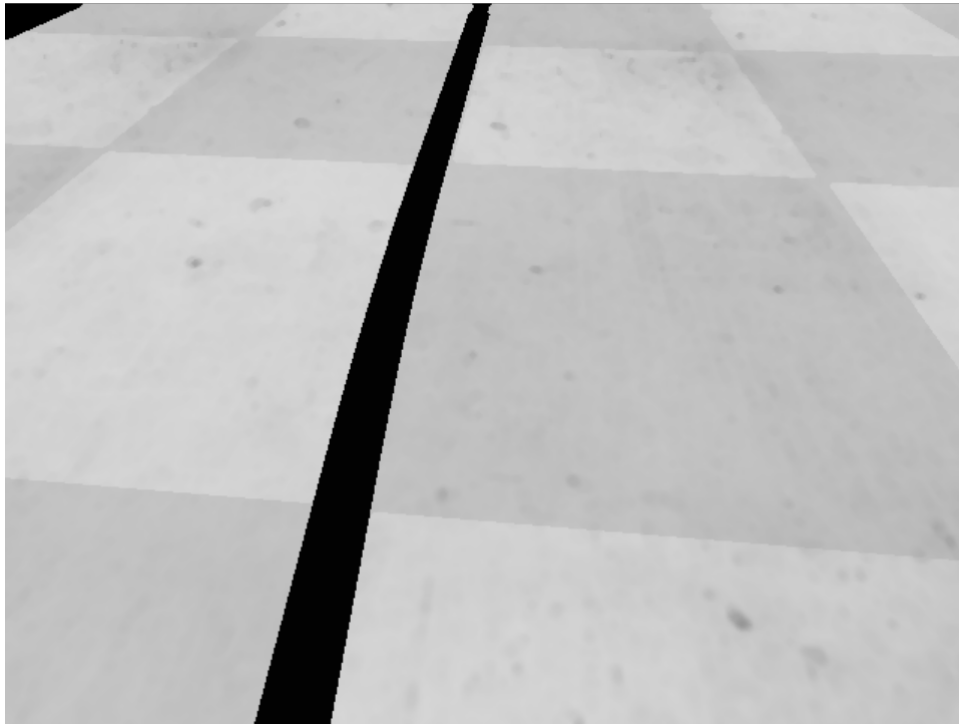
```
simRemoteApi.start(19999)    --V-REP端建立连接
```

2.图像处理部分：

使用 opencv 进行图像处理，调用 opencv 库函数对从传感器得到的图像进行灰化、腐蚀、滤波等操作：

```
errprCode, resolution, image = sim.simxGetVisionSensorImage(clientId, visionSensorHandle, 0, sim.simx_opmode_buffer)
sensorImage = np.array(image, dtype=np.uint8)
sensorImage.resize([resolution[1], resolution[0], 3]) # 480行 640列
grayImage = cv2.cvtColor(sensorImage, cv2.COLOR_BGR2GRAY)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
eroded = cv2.erode(grayImage, kernel)
dst = cv2.medianBlur(eroded, 5)
cv2.flip(dst, 0, dst)
cv2.imshow('result', dst) # 显示图像
```

最终得到的效果如下（该图像的分辨率为 640×480 ）：



3.控制规划部分：

我们考虑利用获取的路线与视角中心线的偏离度来对小车轮子的差速进行控制，从而实现小车沿着路线前进。

我们首先定义两个取样行 `row = 420` 和 `high = 240`，和一个取样列 `mid = 320`，对获取的图像的这两行进行检测获取路线。`row` 行作为主检测行，从图像的中心列开始向两边遍历，找到更近的黑色像素点，就是我们要找的路线（这样也可以保证在交叉路口，小车会沿着转角更小的路线前进）。通过简单的反三角函数计算获取路线的中心点与图像的下边沿中心点（480, 320）的连线与竖中心线的夹角 `angle`，作为小车轮子差速控制的自变量，同时我们也获取了转向的方向，用变量 `direction` 表示，其中 -1 表示左转（路线偏左），1 表示右转（路线偏右），0 表示直行。

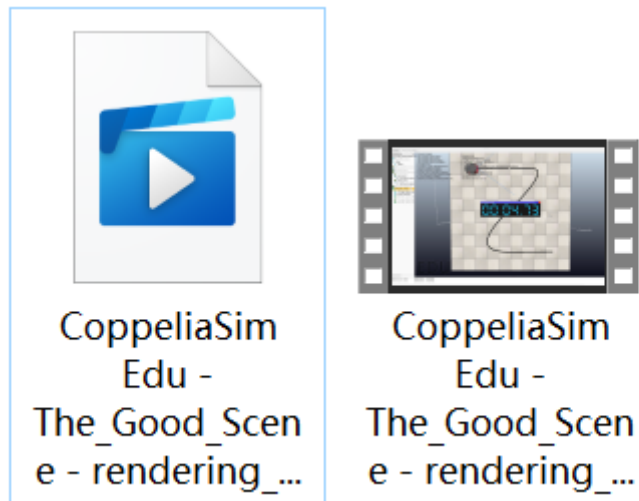
另外，以同样的方法计算 `high` 行的夹角，利用其与 `row` 行夹角的商作为差速修正系数。这是由于主检测行偏下，在弯度比较大的弯道可能转向速度不够，从而偏离轨道。因此采用一个较高的检测行，当弯度较大时，`high` 行的角度会比 `row` 行大得多，较高的商值修正了差速，实现提前转向。将计算后的商乘以 `direction` 作为函数返回值即可。

在得到 `direction` 与 `angle` 以后，利用二次函数模型，计算出左轮和右轮各自的速度，并向小车下达指令。函数中的参数经过多次的仿真调整，是目前表现效果较好的参数。

三、实验结果：

注：实验采用的仿真步长是 $dt=10ms$ ，需要进行调整。

实验结果可见录制的视频，已放至文件夹 report 内：



注：视频里的simulation time模块有点小问题，还烦请助教调整一下！！

其中 src 文件夹内 The_Good_Scene_1.ttt 为TA提供路径； The_Good_Scene_2.ttt 为我们所给出的路径。

四、实验源码：

代码结构如下：

```
import ...

# 获取相关信息
def GetInfo(clientId):...

# 处理
def pro(result):...

# 用于计算偏差角度
def cal_ang(point_1, point_2, point_3):...

# 连接v-rep
def connection(num):...

if __name__ == "__main__":
    connection(0)
```

getInfo 为获取图像，并通过调用 pro 函数来对获取的图像进行巡线； cal_ang 函数是用来计算三个点的偏差角度； connecion 函数则是建立 Python 端和 V-REP 端的通信。

Python 端完整代码如下（可测试文件为 Robot_Control.py）：

```
import sim
import time
import numpy as np
import cv2
import math

# 获取传感器采集的路径相关信息
def GetInfo(clientId):
    errorCode, visionSensorHandle = sim.simxGetObjectHandle(clientId,
    'vision_sensor',
    sim.simx_opmode_oneshot_wait) # 获取相机（传感器）句柄
    # 第一次获取数据无效
    errprCode, resolution, image = sim.simxGetVisionSensorImage(clientId,
    visionSensorHandle, 0, sim.simx_opmode_streaming) # 获取图像
    _, pos = sim.simxGetObjectPosition(clientId, visionSensorHandle, -1,
    sim.simx_opmode_streaming)
    time.sleep(0.1)
    while True: # 循环获取图像数据
        errprCode, resolution, image = sim.simxGetVisionSensorImage(clientId,
        visionSensorHandle, 0, sim.simx_opmode_buffer)
        sensorImage = np.array(image, dtype=np.uint8)
        sensorImage.resize([resolution[1], resolution[0], 3]) # 480行 640列
        grayImage = cv2.cvtColor(sensorImage, cv2.COLOR_BGR2GRAY)
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
        eroded = cv2.erode(grayImage, kernel)
        dst = cv2.medianBlur(eroded, 5)
        cv2.flip(dst, 0, dst)
        cv2.imshow('result', dst) # 显示图像
        direction, angle = pro(dst)
        print(direction, angle)
        v_left = 1 + direction * angle * angle / 8100 # 设定左右速度
        v_right = 1 - direction * angle * angle / 8100

        _, l_motor = sim.simxGetObjectHandle(clientId, "Pioneer_p3dx_leftMotor",
        sim.simx_opmode_oneshot_wait) # 获取轮子控制句柄
        _, r_motor = sim.simxGetObjectHandle(clientId,
        "Pioneer_p3dx_rightMotor", sim.simx_opmode_oneshot_wait)
        errorCode = sim.simxSetJointTargetVelocity(clientId, l_motor, 1.4 *
        v_left, sim.simx_opmode_oneshot)
        # 发送指令给左右轮子，控制速度，这里不等待返回结果，减小控制命令间隔，增大控制精度
        errorCode = sim.simxSetJointTargetVelocity(clientId, r_motor, 1.4 *
        v_right, sim.simx_opmode_oneshot)
        if cv2.waitKey(1) == 27: # 可以等待ESC指令退出
            break
    return dst

# 处理图像数据来获取控制的信息
def pro(result):
    row = 420 # 第一个采样行
    high = 240 # 第二个采样行
    mid = 320 # 中间列
    endpoint_high = 320 # 中间列
    endpoint_left1 = 0 # 左侧第一个黑色像素点（黑白像素交界位置，即路径最左侧）
```

```

endpoint_right1 = 0    # 距离中心线最近的黑色像素点（左）
endpoint_left2 = 640   # 距离中心线最近的黑色像素点（右）
endpoint_right2 = 640  # 右侧第一个黑色像素点（黑白像素交界位置，即路径最右侧）
endpoint = 320         # 找到的路径的中间位置
direction = 0          # 方向（-1左，1右，0直行）
for j in range(mid, 1, -1): # 对于第420行，循环变历检查图像中路径（黑色像素点）位置
    if (result[row, j] < np.array([30, 30, 30])).all() and endpoint_right1 <
j:
    endpoint_right1 = j
    for jj in range(j, 1, -1):
        if (result[row, jj] > np.array([30, 30, 30])).all() and
endpoint_left1 < jj:
        endpoint_left1 = jj
        break
    break
for j in range(mid, 640, 1): # 同上
    if (result[row, j] < np.array([30, 30, 30])).all() and endpoint_left2 >
j:
    endpoint_left2 = j
    for jj in range(j, 640, 1):
        if (result[row, jj] > np.array([30, 30, 30])).all() and
endpoint_right2 > jj:
        endpoint_right2 = jj
        break
    break
print(endpoint_left1, endpoint_right1, endpoint_left2, endpoint_right2)
if endpoint_right1 + endpoint_left2 > mid * 2:
    endpoint = (endpoint_left1 + endpoint_right1) / 2
    direction = -1 # left
elif endpoint_right1 + endpoint_left2 < mid * 2:
    endpoint = (endpoint_left2 + endpoint_right2) / 2
    direction = 1  # right
else:
    endpoint = mid
    direction = 0  # mid
if endpoint != 320: # 此时不在中心线上，算角度
    angle = cal_ang(np.array([row, mid]), np.array([480, mid]),
np.array([row, endpoint]))
else:
    angle = 0      # 此时已经在路径中心线上，偏离角度为0
# 下面的代码为对于第二个采样行进行分析，由于第二个采样行只是起到辅助作用解决一些特殊情况，所
以在扫路径的时候只扫一个方向，而不是对图像所有列进行遍历
# 只考虑左侧
if (direction == -1):
    endpoint_left = 0
    endpoint_right = 0
    for j in range(mid, 1, -1):
        if ((result[high, j] < np.array([10, 10, 10])).all() and
endpoint_right < j):
        endpoint_right = j
        for jj in range(j, 1, -1):
            if ((result[high, jj] > np.array([10, 10, 10])).all() and
endpoint_left < jj):
            endpoint_left = jj
            break
        break
# 只考虑右侧
else:

```

```

        endpoint_left = 0
        endpoint_right = 0
        for j in range(mid, 640):
            if (result[high, j] < np.array([10, 10, 10])).all() and
endpoint_left > j:
                endpoint_left = j
                for jj in range(j, 640):
                    if (result[high, jj] > np.array([10, 10, 10])).all() and
endpoint_right > jj:
                        endpoint_right = jj
                        break
                break

# 同上
endpoint_high = (endpoint_left + endpoint_right) / 2
if endpoint_high != 320:
    angle_high = cal_ang(np.array([high, mid]), np.array([480, mid]),
np.array([high, endpoint_high]))
else:
    angle_high = 0
# 差速修正系数
if angle != 0:
    direction *= 5 * (angle_high / angle)
return direction, angle

# 用于计算偏差角度
def cal_ang(point_1, point_2, point_3):
    """
    根据三点坐标计算夹角
    :param point_1: 点1坐标
    :param point_2: 点2坐标
    :param point_3: 点3坐标
    :return: 返回任意角的夹角值，这里只是返回点2的夹角
    """
    a = math.sqrt((point_2[0] - point_3[0]) * (point_2[0] - point_3[0]) +
(point_2[1] - point_3[1]) * (point_2[1] - point_3[1]))
    b = math.sqrt((point_1[0] - point_3[0]) * (point_1[0] - point_3[0]) +
(point_1[1] - point_3[1]) * (point_1[1] - point_3[1]))
    c = math.sqrt((point_1[0] - point_2[0]) * (point_1[0] - point_2[0]) +
(point_1[1] - point_2[1]) * (point_1[1] - point_2[1]))
    A = math.degrees(math.acos((a * a - b * b - c * c) / (-2 * b * c)))
    B = math.degrees(math.acos((b * b - a * a - c * c) / (-2 * a * c)))
    C = math.degrees(math.acos((c * c - a * a - b * b) / (-2 * a * b)))
    return B

# 连接v-rep
def connection(num):
    print('Program Start!')
    sim.simxFinish(-1) # 关掉之前的连接
    clientId = sim.simxStart("127.0.0.1", 19999, True, True, 5000, 5) # 建立和服务器的连接
    wrong_nums = 100
    while True:
        if clientId != -1: # 连接成功
            print('Successful Connect!')
            GetInfo(clientId) # 获取传输的图像
            wrong_nums = 100

```

```

        break
    else:
        time.sleep(0.2)
        clientId = sim.simxStart("127.0.0.1", 19999, True, True, 5000, 5)
        # 重新建立和服务器的连接
        wrong_nums = wrong_nums - 1
    if wrong_nums == 0:
        print('Fail Connect!') # 尝试100次未果则不再连接
        break
    sim.simxFinish(clientId) # 结束连接
    print('Program Ended!')

if __name__ == "__main__":
    connection(0)

```

V-REP端脚本文件内容如下：

```

function sysCall_init()
    --motorLeft=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
    --motorRight=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
    simRemoteApi.start(19999)
end

```