第四届国际海洋工程装备科技创新大赛

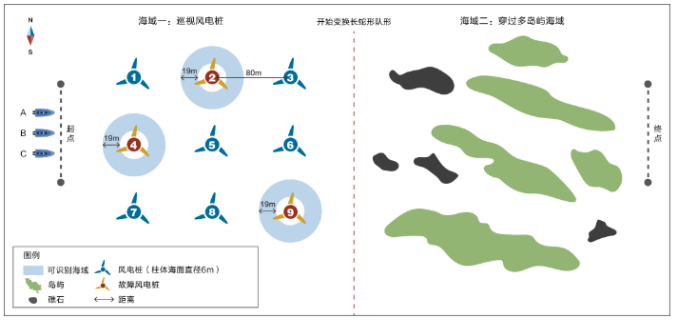
仿真类—技术手册

(初赛版本)

目录

前言：

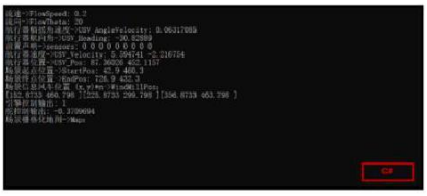
参赛者对**海上风电运维仿真平台**中的**航行器**进行**编程控制**，在**指定时间**内实现**规定动作**，完成**海上巡检**和**自主避障**等任务。(如下图)



【本次比赛支持 Python、C++、Java、C# 等编程语言链入竞赛平台】

参数说明

所有**参数信息**将会在参赛者**控制台**进行**打印**(如下四图)，以便于前期调试。不需要时，可将相关代码**注释**掉。



下图为**竞赛场景**俯视图，其 x、y 轴如下所示：

参数【接收】

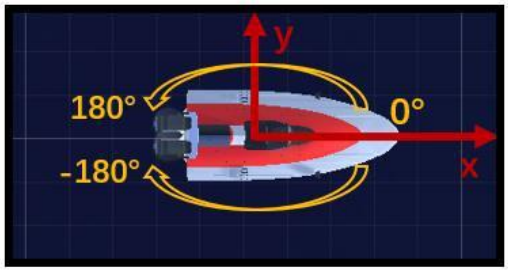
参赛者会收到以下环境参数和航行器参数：

float **FlowSpeed**; 【获取】海流流速，单位为 m/s

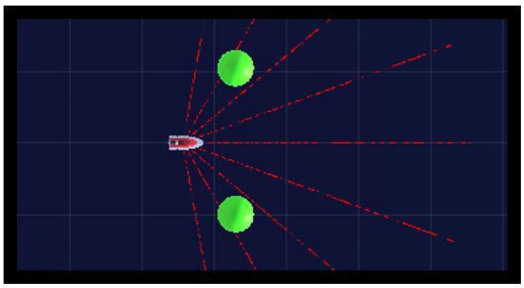
float **FlowTheta**; 【获取】海流流向，与 x 轴正方向同向为 0(同 Heading)，单位为度

float **USV_AngleVelocity**; 【获取】航行器艏摇角速度，单位为 rad/s

float **USV_Heading**; 【获取】航行器航向角，如下图,单位为度



float[9] **sensors**; 【获取】前置声呐数组，如下图所示，探测范围为 40，单位为米



float[2] **USV_Velocity**; 【获取】航行器速度，最大值为 6，单位为 m/s；
USV_Velocity[0] 为 x 轴分量；USV_Velocity[1] 为 y 轴分量

float[2] **USV_Pos**; 【获取】航行器位置：USV_Pos[0] 为 x 轴坐标；
USV_Pos[1] 为 y 轴坐标；单位为米

float[2] **StartPos**; 【获取】起点位置信息：StartPos[0] 为 x 轴坐标；
StartPos[1] 为 y 轴坐标；单位为米

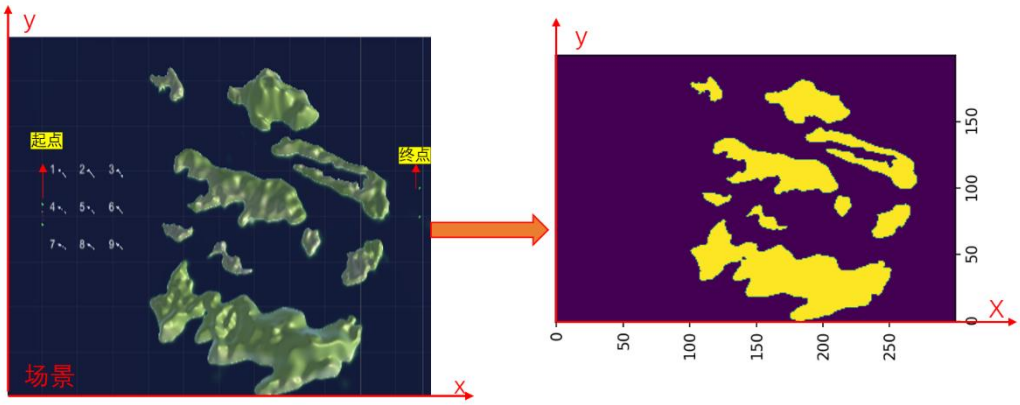
float[2] **EndPos**; 【获取】终点位置信息：EndPos[0] 为 x 轴坐标；
EndPos[1] 为 y 轴坐标；单位为米

float[3,2] **WindMillPos**; 【获取】风车位置信息：012 行分别对应 123 号风车；
0 列为 x 轴坐标，1 列为 y 轴坐标；单位为米

int[300,200] **Map**; 【获取】栅格化地图数据：

将竞赛地图[1200 * 800]通过二值数组[300 * 200]发送给用户，以用于路径规划。

其中，1 为岛屿，0 为水域，如下图：



参数【发送】

参赛者将会发送以下参数给竞赛平台：

```
float EnginesControl; 【发送】航行器引擎控制参数：范围[-1,1]
float RudderControl; 【发送】航行器舵控制参数：范围[-1,1]
```

例程介绍

例程算法介绍

领航者-跟随者算法

领航者-跟随者"算法是一种在群体机器人或无人车等多智能体系统中应用的算法，用于实现分布式协作和导航。在这种算法中，一些智能体被定义为领航者，它们负责引导和指导其他智能体（跟随者）以实现共同的目标。

跟随者需与领航者保持一定的距离与角度，此处因为需要进行避障，所以跟随者只需与领航者保持一定的距离。

AStar 路径规划

AStar（又称 A*），它结合了 Dijkstra 算法的节点信息（倾向于距离起点较近的节点）和贪心算法的最好优先搜索算法信息（倾向于距离目标较近的节点）。可以像 Dijkstra 算法一样保证找到最短路径，同时也像贪心最好优先搜索算法一样使用**启发值**对算法进行引导。

参数：

g: # 表示从起点移动到当前节点的实际代价函数。
h: # 表示当前节点至目标节点的估值函数，例程用的估值方法为欧式距离。
f = g + h : # 总代价
open_list = [] # 记录被考虑来寻找最短路径的数组
close_list = [] # 已遍历过的点

类或函数：

find_min_grid() # 获取最小临近点
find_neighbors() # 获取所有临近点
is_valid_grid() # 计算临近点是否合理
a_star_search() # 搜索函数
Grid: # 节点类型，包含 x、y、patent 及 f、g、h 的计算

A*主逻辑

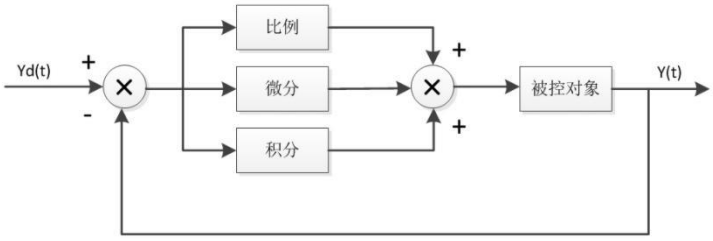
获取栅格地图 Map
设置起点、终点搜索地图

*初始化 open_list[]和 close_list[]
*将起点放入 open_list[]中
While 循环:
*在 open_list 中查找 F 值最小的节点作为当前节点
*当前节点从 open_list[]中移除；*当前节点放入close_list[];
*通过 find_neighbors()函数，获取所有相邻的节点(自主设置)； *for 循环:
*标记父亲节点、G、H、F，并放入open_list[] *for 循环:
*如果终点在 open_list[]中,直接返回终点节点

回溯地图路径
输出栅格地图及最优路径(控制台打印)

PID 控制算法

PID：一种应用非常广泛的控制算法，其原理如下:



参数：

dt: # 时间间隔。
kp: # 比例控制。
ki: # 积分控制。
kd: # 微分控制。
max_out: # 最大输出限制，规避过冲。

min_out: # 最小输出限制。
now_value: # 当前值。
Target: # 目标值。
err_last: # 上一个偏差值。
integral: # 累积偏差。
Output: # 输出。
Is_angle: # 判断引擎控制还是舵控制。

类或函数：

pid_realize() # PID 实现

PID 主逻辑

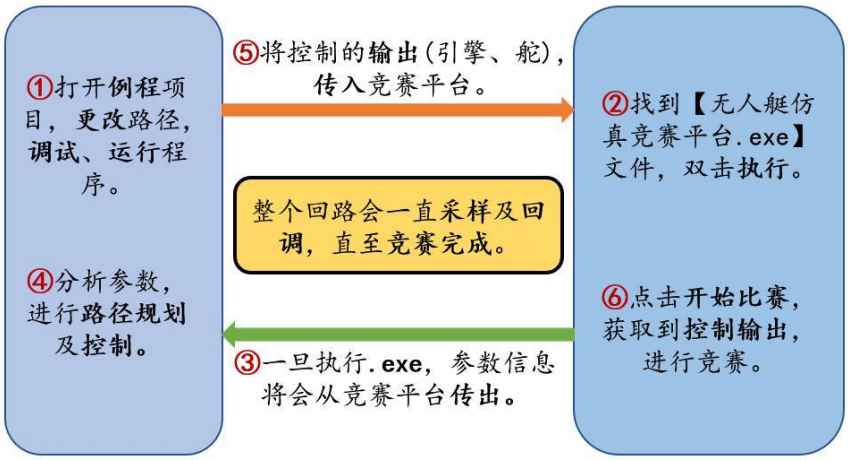
传入 target, now_value, dt, kp, ki, kd, is_angle
判断引擎控制 or 舵控制，并获得 err
Integral += err
Output= kp * err + ki * integral * dt + kd * (err - err_last) / dt
判断 Output 是否过冲

    self.err_last = self.err
    返回 Output

例程概括

四种例程均采用：① **AStar 算法** 进行 **路径规划**。
② 采用 **PID 算法** 分别对 **引擎**和**舵** 进行控制。

例程项目与**竞赛平台**交互逻辑如下：

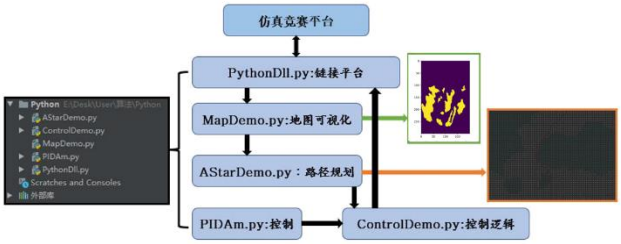


【例程运行前，需要根据竞赛平台所在位置**更改绝对路径**(在脚本中已说明)】

【例程代码已做了详尽的注释，以下四篇只说明例程架构】

Python 篇

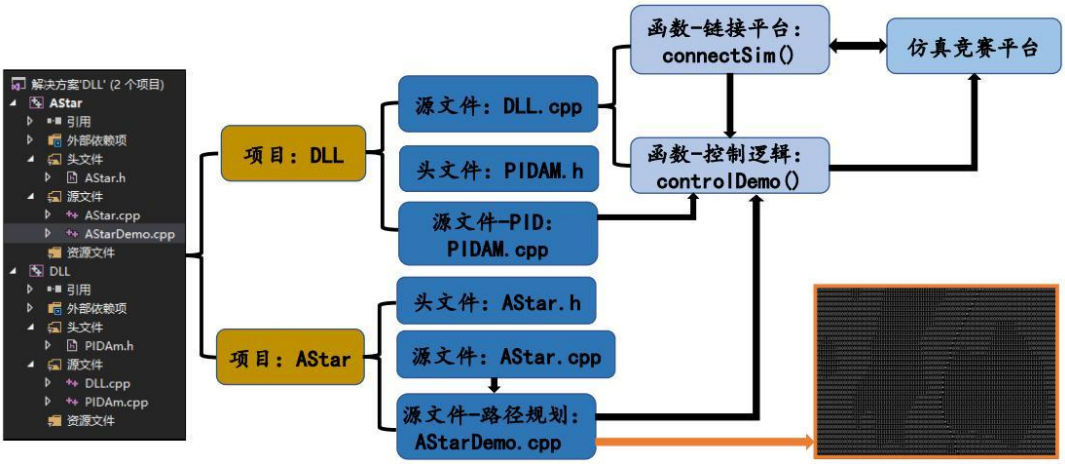
Python 例程文件构成如下：



(1)上面一直强调的**绝对路径**在 **PythonDll.py** 脚本中进行更改，想要获取**相关参数**直接调用该脚本中的**相关方法**即可。
(2)通过 **AStarDemo.py** 对岛屿区域进行**路径规划**，获得**最优路径**。
(3)参考**风车位置**及**岛屿地区最优路径**，设置**全局最优路径二维数组**(路径节点)。
(4)通过 **PID** 完成**点到点**(到达指定范围，更新下一目标点)的控制。
(5)**【调参】**，参赛者可在 **ControlDemo.py** 中，通过**调试 PID 参数**及优化**最优路径数组**，达到缩减竞赛时间的效果。
(6)**【进阶】，参赛者可自主获取参数、并对所有算法进行自主设计。**

C++ 篇

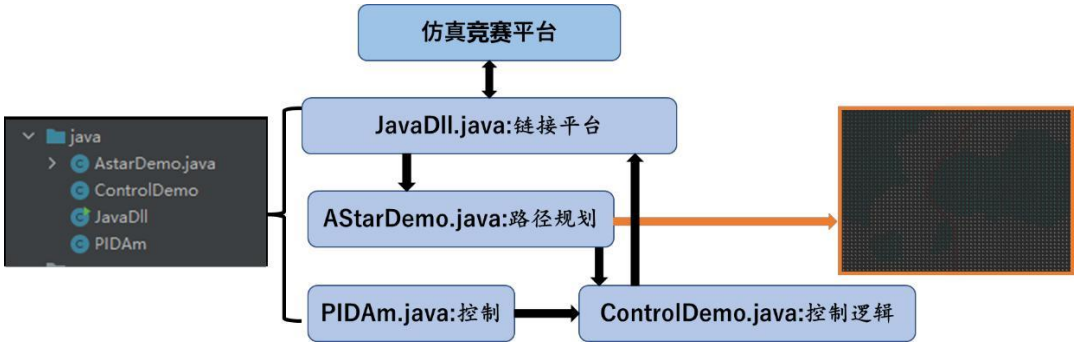
C++ 例程文件构成如下：



(1) C++解决方案包括两项目：**AStar**、**DLL**，鼠标右键项目名**设置启动项目**。
(2) **绝对路径**需要分别在 **AStarDemo.cpp**、**connection()**中进行更改，想要获取**相关参数**直接调用 DLL.cpp 相对应的**参数**即可。
(2)通过启动 DLL 项目(**AStarDemo.cpp**)对岛屿区域进行**路径规划**，获得**最优路径**。
(3)参考**风车位置**及**岛屿地区最优路径**，设置**全局最优路径二维数组**(路径节点)。
(4)通过 **PID** 完成**点到点**(到达指定范围，更新下一目标点)的控制。
(5)**【调参】**，参赛者可在 **controlDemo()**函数中，通过**调试 PID 参数**及优化**最优路径数组**，达到缩减竞赛时间的效果。
(6)**【进阶】，参赛者可自主获取参数、并对所有算法进行自主设计。**

Java 篇

Java 例程文件构成如下：



(1) 上面一直强调的**绝对路径**在 **JavaDll.java** 脚本中进行更改，想要获取**相关参数**直接调用该脚本中的**相关方法**即可。

(2) 通过 **AStarDemo.java** 对岛屿区域进行**路径规划**，获得**最优路径**。

(3) 参考**风车位置**及**岛屿地区最优路径**，设置**全局最优路径二维数组**(路径节点)。

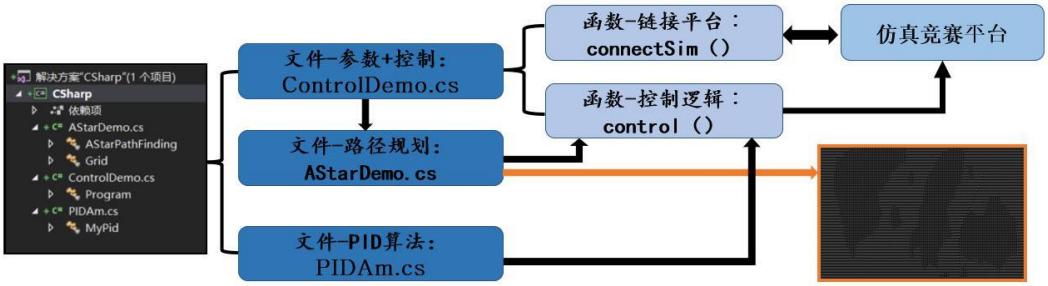
(4) 通过 **PID** 完成**点到点**(到达指定范围，更新下一目标点)的控制。

(5) **【调参】**，参赛者可在 **ControlDemo.java** 中，通过**调试 PID 参数**及优化**最优路径数组**，达到缩减竞赛时间的效果。

(6) **【进阶】，参赛者可自主获取参数、并对所有算法进行自主设计。**

C# 篇

C# 例程文件构成如下：



(1) 上面一直强调的**绝对路径**在 **ControlDemo.cs** 脚本中进行更改，想要获取**相关参数**直接调用该脚本中的**相应参数**即可。

(2) 通过 **AStarDemo.cs** 对岛屿区域进行**路径规划**，获得**最优路径**。

(3) 参考**风车位置**及**岛屿地区最优路径**，设置**全局最优路径二维数组**(路径节点)。

(4) 通过 **PID** 完成**点到点**(到达指定范围，更新下一目标点)的控制。

(5) **【调参】**，参赛者可在 **ControlDemo.cs** 的 **control()**中，通过**调试 PID 参数**及优化**最优路径数组**，达到缩减竞赛时间的效果。

(6) **【进阶】，参赛者可自主获取参数、并对所有算法进行自主设计。**

常见问题解答

主要包括开发者 IDE 与环境版本及常见问题。

Python

开发者 IDE：PyCharm2020.1

版本号：Python 3.9.7

C++

开发者 IDE：Visual Studio 2017

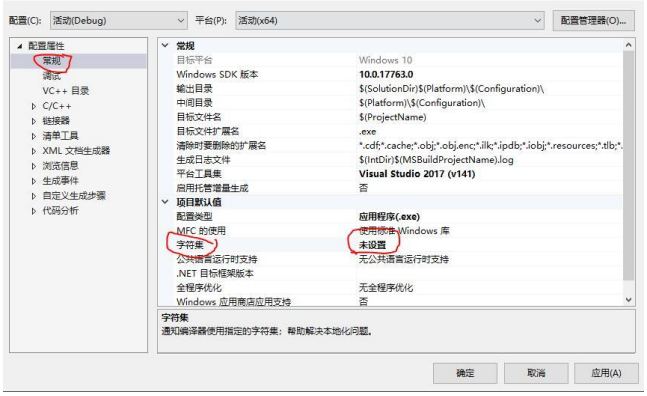
版本号：C++98

常见问题：出现下图所示红色下划线

```
HMODULE  hDll = LoadLibrary(dllAddress);
```

解决措施：

打开 项目-DLL 属性-常规-字符集，改为“未设置”



C#

开发者 IDE：Visual Studio 2017

版本号：C#7.0 & .NET Core 2.0

常见问题：启动报错：无法启动程序 dotnet.exe

解决措施：

(1) 检测“我的电脑-属性-高级-环境变量”中 Path 变量内是否存在 C:\Program Files\dotnet，没有的话加上，问题若未解决，请尝试(2)

(2)安装相应版本框架

```
It was not possible to find any compatible framework version
The specified framework 'Microsoft.NETCore.App', version '1.1.2' was not found.
```

或，打开项目-Csharp 属性-应用程序中，更改目标框架，设置为更高版本：

Java

开发者 IDE：IntelliJ IDEA 2020.3.1
版本号：java 1.8.0_181

技术支持

比赛过程中遇到问题，可扫码入群咨询相关技术人员。

QQ扫描二维码可加入赛事交流群