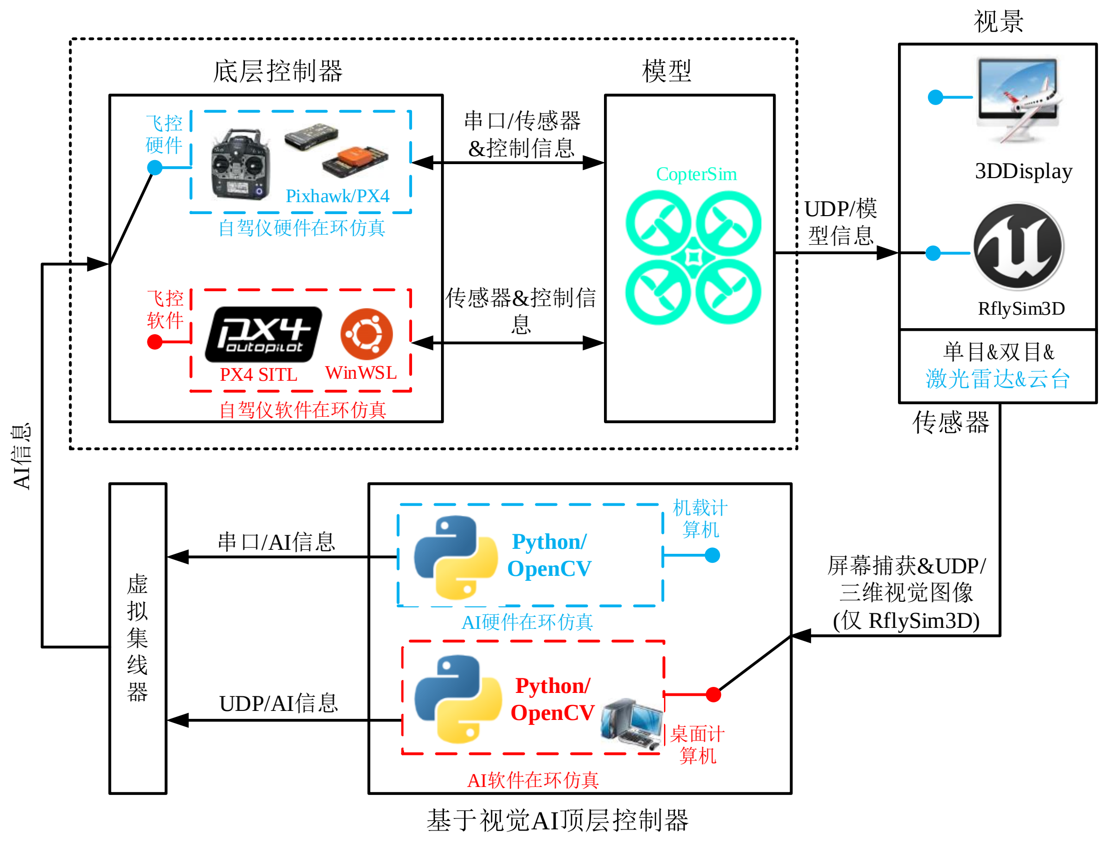
Contents

RflySim Chapter 10 Cluster Control Algorithm Development

This lecture focuses on the communication architecture and collaborative control technology of UAV cluster, which is systematically divided into two levels: basic experiment and advanced experiment, aiming at realizing efficient networking communication and precise formation control of UAV in local area network. The experimental platform integrates RflyUdpFast communication module, supports MATLAB Simulink, Python and other environments, and supports local single-machine simulation and cluster control experiments under multi-machine networking.



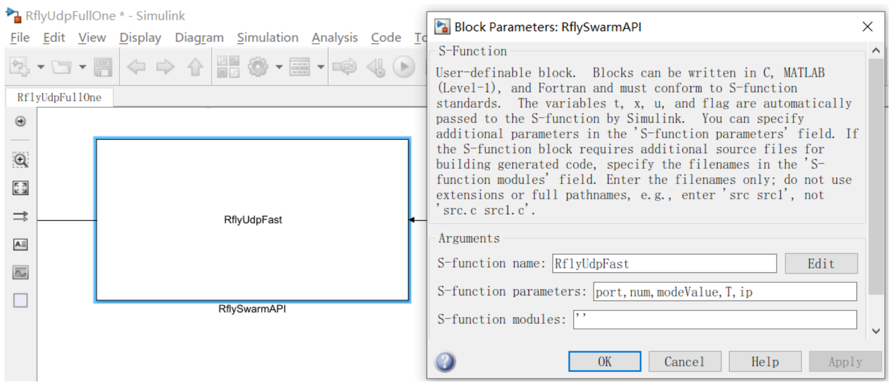
The concept of cluster originated from biological research. Based on the nesting behavior of termites, French zoologist Grasse first proposed the concept of consensus autonomy (Stigmergy): a mechanism of intermittent coordination between individuals, that is, without any centralized planning and direct communication to complete complex intelligent activities. This is the beginning of the concept of autonomous cluster coming into human vision and gradually developing. From biological systems to Multi-Agent System (MAS), the concept of cluster is also evolving and enriching. A multi-agent system is a system in which there are multiple agents that need to be controlled at the same time. Each agent needs to compete or cooperate with other agents while interacting with the environment. When a multi-agent system has a large scale to perform very complex tasks, it is called a clusterADDIN

CNKISM.Ref.{4920E6A5E8AA40219922D8E4403E6268}[1]. Cluster control mainly studies the consistency problem, that is, using distributed coordination control algorithm to make the state or output of each agent in the "network" achieve a certain consistency. The so-called distributed coordinated control algorithm is to form a coordinated global behavior through simple local rules.

Simulink Cluster Interface

Open the "RflySimAPIs \ SimulinkSwarmAPI \" RflyUdpFullOne. Slx "file. As shown in the lower left figure, the" RflySwarmAPI "module is the cluster communication module.

The Simulink S function of this module is realized by C + + mixed programming, and the source file is shown in "RflyUdpFast. CPP" ".


RflyUdpFullOne * - Simulink
File Edit View Display Diagram Simulation Analysis Code To
RflyUdpFullOne
RflyUdpFast
RflySwarmAPI
Block Parameters: RflySwarmAPI
OK
Cancel
Help
Apply
Edit

S-Function

User-definable block. Blocks can be written in C, MATLAB (Level-1), and Fortran and must conform to S-function standards. The variables t, x, u, and flag are automatically passed to the S-function by Simulink. You can specify additional parameters in the 'S-function parameters' field. If the S-function block requires additional source files for building generated code, specify the filenames in the 'S-function modules' field. Enter the filenames only; do not use extensions or full pathnames, e.g., enter 'src src1', not 'src.c src1.c'.

Arguments

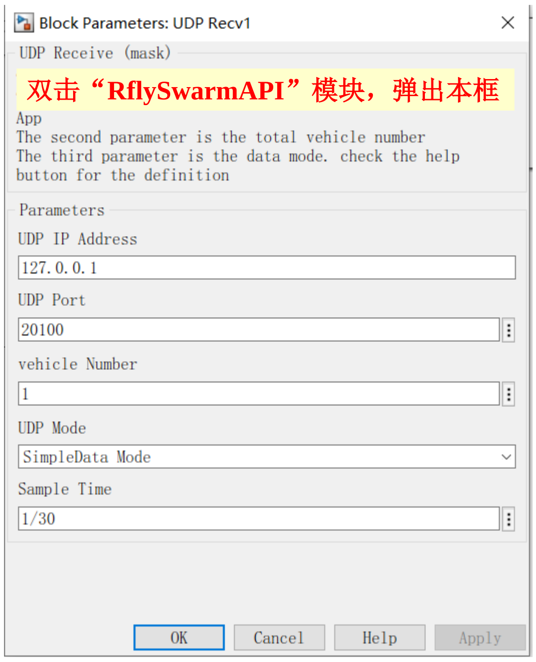
S-function name: RflyUdpFast

S-function parameters: port, num, modeValue, T, ip

S-function modules: ''

The "RflySwarmAPI" module needs to be in the same folder as "RflyUdpFast.mexw64" to be called. Therefore, when creating a new slx project, copy the "RflySwarmAPI" module and also copy the "RflyUdpFast.mexw64" file to the directory where the new slx file is located. RflyUdpFast.mexw64 is a compilation of the "source file" RflyUdpFast. CPP "in MATLAB.

Double-click on the "RflySwarm API" module in Simulink to see the details.


Block Parameters: UDP Recv1

UDP Receive (mask)

双击“RflySwarmAPI”模块，弹出本框

App
The second parameter is the total vehicle number
The third parameter is the data mode. check the help
button for the definition

Parameters

UDP IP Address: 127.0.0.1

UDP Port: 20100

vehicle Number: 1

UDP Mode: SimpleData Mode

Sample Time: 1/30

(A) The first item "UDP IP Address" is the IP address of the target computer. If "127.0.0.1" is input, it can only accept the Pixhawk autopilot status forwarded by the local CopterSim and control it; "255.255.255.255" can receive and control the CopterSim program running in all computers in the LAN (the CopterSim of other computers needs to check the "Online" button); the designated IP such as "192.168.1.12" will only send control instructions to the host of the IP address. Generally speaking, the "255.255.255.255" broadcast can meet the demand in a small-scale cluster. When the number of aircraft continues to increase, the designated IP needs to be enabled to reduce the network load and improve the communication speed and reliability.

(B) The second entry "UDP Port" "is the initial port number of the first aircraft, and the default initial port is the 20100. Each CopterSim needs to occupy one port to send and receive messages. For example, this module needs to simulate aircraft with aircraft ID of 10 ~ 15, and this item needs to be filled with 20100 + 10 * 2 = 20120, and the latter item "Vehicle number" aircraft quantity "needs to be filled with 5.

(C) The third item "Vehicle number" represents the number of CopterSim to be connected and controls the number of I/O ports of the module. If 10 is entered, the module will automatically generate 10 pairs of I/O ports.

(D) The fourth item "UDP mode" is the data mode protocol of the input and output interface, mainly including the FullData complete mode (the most complete data, but the amount of transmitted data is large); SimpleData reduced data mode (more aircraft > 8, to avoid network congestion with too large data) and UltraSimple ultra-reduced mode (more than 20 aircraft per computer), with less latency.

(E) The fifth term is the "Sample Time", which should correspond to the Simulink simulation time.

The RflyS warmAPI module has three UDP modes: FullData mode, SimpleData mode, and Ultra Simple mode.

Python Cluster Interface

Note: To read and run the Python code, you need to install VS Code and associate it with the platform Python38 environment. For the specific installation process, please refer to Lecture 6 of the Advanced Course.

The Python communication interface file for cluster control and the visual Python control interface are identical, both in the "PX4MavCtrlV4.py" interface file, which is basically used in the same way as in Lecture 6 of the advanced course.

The interface file used by the Python cluster control is "PX4MavCtrlV4.py", which is identical to the API file used by the PythonVision vision control.

According to the tutorial in Lesson 6, Python communication with CopterSim/PX4 includes two communication modes (UDP structure and MAVLink data flow) and two optimization modes (Full and Simple).

In MAVLink _ Full and MAVLink _ Simple communication modes (corresponding to InitMavLoop (2) and InitMavLoop (3)), MAVLink data stream is directly used for communication between Python and PX4 (transferred by CopterSim), which has a large amount of data and occupies a high bandwidth. However, it is closer to the real machine and has perfect functions, and it is easy to block broadband in large-scale clusters;

In UDP _ Full and UDP _ Simple modes (corresponding to InitMavLoop (0) and InitMavLoop (1)), Python sends a simplified UDP structure message to communicate with CopterSim, which compresses or decompresses it and then communicates with PX4 through MAVLink. This method can effectively reduce the amount of data in the LAN, and is suitable for cluster simulation control.

The Python script communicates with CopterSim using the 20100 + 2i series interface to receive flight control data (the InitMavLoop function enables listening, and the UAV * * * variable reads the data). 30100 + 2i serial interface to receive true data (InitTrueDataLoop function to enable data listening, true * * * variable to read data).

In addition to co-simulation with PX4 and CopterSim (6 DOF high granularity model), the cluster platform also supports Python and RflySim3D compact simulation (particle low granularity model), uses Python internal compact aircraft model, with small amount of calculation, and supports large-scale clusters.

Control model

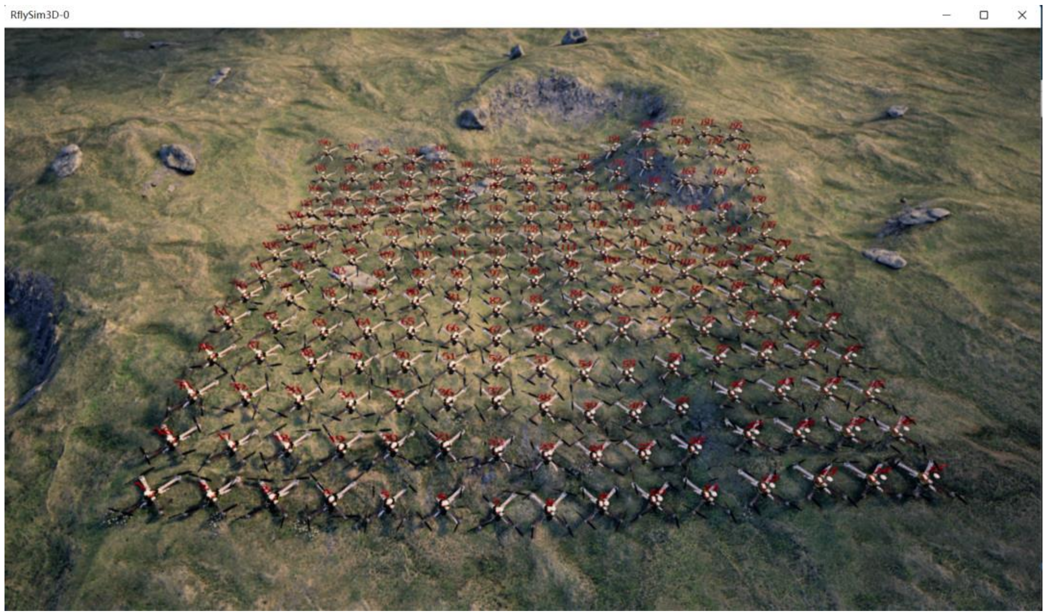
At the control model level, the experimental platform includes a variety of models suitable for cluster control, including but not limited to PX4 flight controller integration scheme based on high-precision dynamic model, and custom controller design in Simulink environment. At the same time, it also supports the simulation and control of many physical models, such as particle model, rotor model and fixed-wing model.

From the perspective of model accuracy, the software/hardware-in-the-loop simulation closed-loop method of using high-precision 6-Dof model (CopterSim) + real flight control system (PX4) can effectively improve the credibility of the model, thereby reducing the gap between simulation and real machine experiment.

However, the complexity of the code computation of the above architecture leads to the limitation of the number of UAVs running on a computer (usually < 20 for SITL software in the loop and < 30 for HITL hardware in the loop).

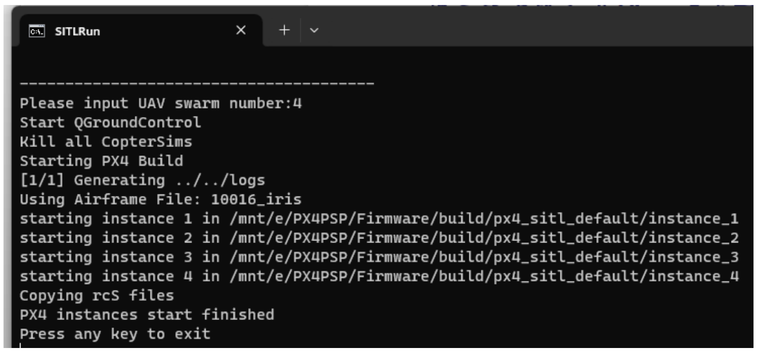
In order to increase the number of cluster aircraft simulated by a single computer, it is necessary to reduce the accuracy of the model and use a simplified flight control model.

Therefore, this platform has developed a particle multi-rotor model under Python, which can realize 100-driving-level UAV cluster simulation on a single computer only with Python and RflySim3D software (the free version only supports up to 12 routines), and the input and output interfaces of each aircraft are consistent with SITL and HITL simulation. To ensure a smooth transition to the real machine experiment.



Hardware and software are in the loop

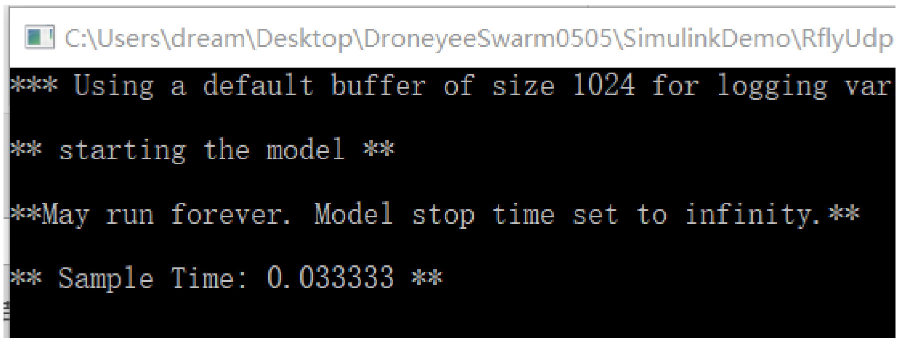
In order to ensure the diversity and depth of the experiment, the platform supports two simulation modes of software in the loop (SIL) and hardware in the loop (HIL) to fully verify and debug the performance of the cluster control algorithm under virtual and actual hardware conditions. In terms of control strategy, it covers a variety of control modes, such as position control, speed control and attitude control, so as to explore the communication coordination and control strategy of UAV cluster in complex task execution in an all-round way.



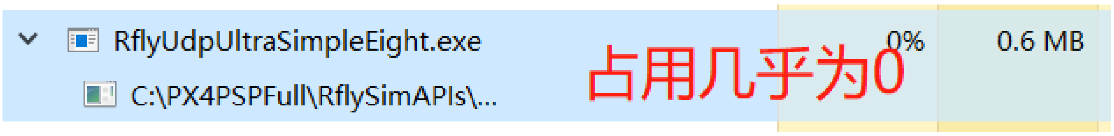
```
----------------------------------------
Please input UAV swarm number:4
Start QGroundControl
Kill all CopterSims
Starting PX4 Build
[1/1] Generating ../../logs
Using Airframe File: 10016_iris
starting instance 1 in /mnt/e/PX4PSP/Firmware/build/px4_sitl_default/instance_1
starting instance 2 in /mnt/e/PX4PSP/Firmware/build/px4_sitl_default/instance_2
starting instance 3 in /mnt/e/PX4PSP/Firmware/build/px4_sitl_default/instance_3
starting instance 4 in /mnt/e/PX4PSP/Firmware/build/px4_sitl_default/instance_4
Copying rcS files
PX4 instances start finished
Press any key to exit
```

Executable file generation

MATLAB itself will occupy a lot of CPU and memory resources. When running the complex Simulink control program, on the one hand, the amount of calculation is too large to cause the algorithm to run slowly, which can not meet the real-time requirements (Simulink runs 1 s larger than the real clock 1 s), so it can not control the cluster aircraft of the simulation system (or the real system) in real time; Secondly, if Simulink takes up a lot of computing resources in simulation, it will lead to less allocation of computing resources in RflySim3D and CopterSim, resulting in poor aircraft simulation, severe aircraft shaking and even crash.
After the Simulink controller is compiled into exe, the algorithm can run without MATLAB, and it is a binary executable file, so the running efficiency is very high, even if the large-scale control algorithm, it can also ensure real-time control.


C:\Users\dream\Desktop\DroneyeeSwarm0505\SimulinkDemo\RflyUdp

```
*** Using a default buffer of size 1024 for logging var
** starting the model **
**May run forever. Model stop time set to infinity.**
** Sample Time: 0.033333 **
```



```
RflyUdpUltraSimpleEight.exe                         0%      0.6 MB
  C:\PX4PSPFull\RflySimAPIs\...
```