动捕接入指南

注：如遇无法解决的问题，请联系幻思创新的工程师寻求协助。

修订日志:

版本	日期	更新内容
V1.2	2025.10.8	1、更新了“1.配置 ROS 工程”章节中的“（4）配置通信节点参数”——>参数配置（set_goal）（P2，点击跳转）

1.配置 ROS 工程

A、配置单架飞机接入动捕

（1）配置组网模块

该步骤确定的 IP 地址将在 配置通信节点参数 中使用。

（1）如果仅飞一架飞机，无需进行额外配置。在后续步骤中，可直接使用个人电脑连接飞机的默认 IP 地址：192.168.4.1。

（2）如果飞多架飞机（飞集群），请参照官网教程—>“集群开发指南—>③配置组网模块”（点击跳转）。

（2）配置电脑环境

该步骤用于配置 个人电脑（远程电脑）环境，而非无人机的机载电脑。请注意区分两者，避免误操作。

配置教程请参照官网教程—>“集群开发指南—>④ 配置电脑环境”（点击跳转）。

（3）导入 ROS 工程

请参照官网教程—>“集群开发指南—>⑤ 导入 ROS 工程”（点击跳转）。

（4）配置通信节点参数

① 飞单机

首先，打开刚导入的 ROS 工程（fcu_core 文件）的 fcu_bridge_001.cpp（点击跳转）文件。在文件顶部可以看到参数配置变量，请将这些变量按照下图所示进行配置：

```
#define BUF_SIZE 32768//数据缓存区大小
#define BAUDRATE 460800 //虚拟串口波特率
#define DRONE_PORT 333 //port
static char* DRONE_IP = "192.168.4.1"; //ip
static char* USB_PORT = "/dev/ttyACM0"; //usb虚拟串口文件描述符
static mavlink_channel_t mav_chan=MAVLINK_COMM_1;//MAVLINK_COMM_0虚拟串口发送，MAVLINK_COMM_1网口发送
static bool offboard=false;//是否使用机载电脑
static bool use_uwb=false; //是否使用UWB基站
static bool set_goal=false //远程电脑用于设置轨迹规划的目标，机载电脑应为false
static bool simple_target=true;//仅机载电脑配置：是否为简单目标点,simple_target表示目标只有位置，没有速度和加速度
```

其中，根据 配置组网模块——飞单机 的说明，参数“DRONE_IP”应填写为“192.168.4.1”， 即飞机的默认 IP 地址。

② 多无人机通信节点参数配置说明（以三机为例）

同时飞 3 架飞机，需要运行 3 个通信节点：

（a）**通信节点 001**（对应于 ROS 工程中的 fcu_bridge_001.cpp 文件）；

（b）**通信节点 002**（对应于 ROS 工程中的 fcu_bridge_002.cpp 文件）；

（c）**通信节点 003**（对应于 ROS 工程中的 fcu_bridge_003.cpp 文件）。

因此需要对 3 个通信节点的参数进行配置。

以配置通信节点 001（fcu_bridge_001.cpp 文件）为例进行讲解，配置过程请参照**① 飞单机** 中的步骤，其中参数“DRONE_IP”应填入 **（1）配置组网模块——飞多架飞机中配置的组网模块的 IP 地址**。

注意：

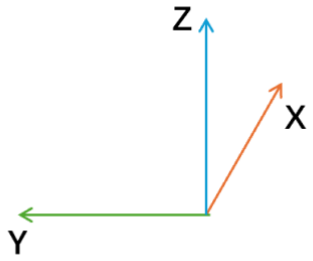
fcu_bridge_001.cpp、fcu_bridge_002.cpp、fcu_bridge_003.cpp 中的参数“DRONE_IP”为 FanciSwarm 无人机网络通信的 IP，用户需要填入 **（1）配置组网模块——飞多架飞机** 中配置的组网模块的 IP 地址，三个通信节点（3 台无人机）对应三个 ip 地址，请一一对应。

（5）动捕配置

第一步，配置动捕话题，在 fcu_bridge_001.cpp 文件中找到动捕话题，修改为自己动捕系统中的话题名，如下图所示默认话题名为/motion_001。

```
gnss_global = nh.advertise<sensor_msgs::NavSatFix>("gnss_global_001",100);
imu_global = nh.advertise<sensor_msgs::Imu>("imu_global_001",100);
odom_global = nh.advertise<nav_msgs::Odometry>("odom_global_001",100);
odom=nh.subscribe<nav_msgs::Odometry>("/vins_estimator/odometry_001", 100, odomHandler, ros::Transport
cmd=nh.subscribe<std_msgs::Int16>("/fcu_bridge/command", 100, cmdHandler, ros::TransportHints().tcpNoD
mission=nh.subscribe<geometry_msgs::InertiaStamped>("/fcu_bridge/mission_001", 100, missionHandler, rc
path_global = nh.advertise<nav_msgs::Path>("/path_global_001", 100);
goal=nh.advertise<geometry_msgs::PoseStamped>("/move_base_simple/goal", 100);
motion=nh.subscribe<geometry_msgs::PoseStamped>("/motion_001", 100, motionHandler, ros::TransportHints
command = nh.advertise<std_msgs::Int16>("/fcu_bridge/command",100);
```

第二步，确定动捕坐标系的坐标轴方向，动捕坐标系应为“前左上”坐标，如下图所示：



第三步，用 rostopic echo 指令查看动捕话题的位置数据，明确位置单位为毫米、厘米、或者米。

第四步，在 fcu_bridge_001.cpp 文件中找到动捕回调函数，在回调函数中修改坐标单位，默认程序中坐标单位为米。

例如：假如我们实际动捕输出话题的坐标单位为毫米，则需要转换为米，转换方式如下图所示：

```
void motionHandler(const geometry_msgs::PoseStamped::ConstPtr& odom)
{
  if(time_odom<=time_start||mav_chan == MAVLINK_COMM_0){//USB传输无需降频
    time_odom=ros::Time::now().toSec();
  }else{
    if(ros::Time::now().toSec()-time_odom<0.1){//用网络通信，如果odom频率过高进行降频，降至10hz以下
      return;
    }
    time_odom=ros::Time::now().toSec();
  }

  Eigen::Vector3f position_map ((float)odom->pose.position.x, (float)odom->pose.position.y, (float)odom->
  float quaternion_odom[4]={(float)odom->pose.orientation.w,
                            (float)odom->pose.orientation.x,
                            (float)odom->pose.orientation.y,
                            (float)odom->pose.orientation.z};

  float roll, pitch, yaw;
  mavlink_quaternion_to_euler(quaternion_odom, &roll, &pitch, &yaw);
  printf("x:%f,y:%f,z:%f,yaw:%f\n",position_map.x(),position_map.y(),position_map.z(),yaw);
  //动捕一般为前左上坐标系，需要改为前右下坐标系发给飞控
  mavlink_message_t msg_local_position_ned_cov, msg_attitude;
  mavlink_attitude_t attitude;
  mavlink_local_position_ned_cov_t local_position_ned_cov;

  attitude.yaw = -yaw;
  mavlink_msg_attitude_encode(mavlink_system.sysid, mavlink_system.compid, &msg_attitude, &attitude);
  mavlink_send_msg(mav_chan, &msg_attitude);

  local_position_ned_cov.x=position_map.x();
  local_position_ned_cov.y=-position_map.y();
  local_position_ned_cov.z=-position_map.z();
  mavlink_msg_local_position_ned_cov_encode(mavlink_system.sysid, mavlink_system.compid, &msg_local_posit
  mavlink_send_msg(mav_chan, &msg_local_position_ned_cov);
}
```

红框部分等号右侧应乘以 0.001，修改为：

```
local_position_ned_cov.x=position_map.x()*0.001;
local_position_ned_cov.y=-position_map.y()*0.001;
local_position_ned_cov.z=-position_map.z()*0.001;
```

第五步，进一步确定动捕数据的偏航角是否正确。

第六步，编译并运行此 fcu_core 工程，如果前面步骤已正确执行，并且动捕系统已经正常启用，那么 fcu_core 工程运行后会在终端输出动捕数据，如下图所示，在 2 条心跳数据之间应出现 10 条动捕数据，注意如果不够 10 条说明局域网质量不好存在丢包问题，飞行存在风险。

```
x:8922.002930,y:712.713806,z:165.902893,yaw:0.053785
x:8921.969727,y:712.682312,z:165.753189,yaw:0.054537
x:8922.242188,y:712.895142,z:165.801010,yaw:0.053772
x:8922.079102,y:712.826355,z:165.754562,yaw:0.053295
x:8922.044922,y:712.741028,z:165.833344,yaw:0.056608
x:8921.876953,y:712.826355,z:165.726959,yaw:0.055130
x:8921.711914,y:712.792480,z:165.912460,yaw:0.055174
001 Received heartbeat time: 34.105137s, voltage:8.011000V, curr
x:8921.770508,y:712.791992,z:165.909866,yaw:0.055556
x:8921.999023,y:712.795410,z:165.821335,yaw:0.054171
x:8922.101562,y:712.649109,z:165.830231,yaw:0.054778
x:8921.968750,y:712.832458,z:165.827484,yaw:0.051649
x:8921.903320,y:712.765320,z:165.848419,yaw:0.051909
x:8922.082031,y:712.875916,z:165.789001,yaw:0.054927
x:8922.052734,y:712.797363,z:165.756195,yaw:0.056071
x:8922.037109,y:712.702820,z:165.786774,yaw:0.052738
x:8921.929688,y:712.782532,z:165.797867,yaw:0.052661
001 Received heartbeat time: 35.115080s, voltage:8.010000V, curr
x:8922.055664,y:712.776550,z:165.680389,yaw:0.055064
x:8921.981445,y:712.771606,z:165.824860,yaw:0.052628
```

上图中第四个 yaw 数据就是偏航角，变化范围-pi~pi.

需要确定：

当无人机机头沿动捕坐标 X 轴摆放的时候，yaw 值为 0；

当无人机机头沿动捕坐标 Y 轴摆放的时候，yaw 值为 pi/2；

B、配置多架飞机接入动捕

参考单架飞机步骤，每个飞机对应一个 fcu_bridge 文件。

2.配置飞控固件

打开飞控固件的 clibrary/include/confg.h

```
87  //配置LED
88  #define FMU_LED_CONTROLL_ENABLE 1 // if use system default led control, set 1; if you want
89
90  //配置磁罗盘
91  #define USE_MAG 0 // if use mag, set 1; if not use mag, set 0;
92
93  //配置GNSS
94  #define USE_GNSS 1 // if use gnss, set 1; if not use gnss, set 0;
95
96  //配置UWB
97  #define USE_UWB 1 // if use uwb, set 1; if don't use uwb, set 0;
98
99  //配置光流
100 #define USE_FLOW 1 // if use optical flow, set 1; if don't use optical flow, set 0;
101
102 //配置里程计
103 #define USE_ODOMETRY 1 // if use odometry, set 1; if don't use odometry, set 0;
104
105 //VINS
106 #define USE_VINS 0 // if use vins, set 1; if use lidar-slam, set 0;
107
108 //SLAM定高
109 #define USE_ODOM_Z 0 // if use slam z, set 1; else set 0;
110
111 //动捕
112 #define USE_MOTION 1 // if use motion capture, set 1; else set 0;
113
```

按上图修改对应参数，保存编译烧录后，无人机进入动捕定位模式。

首先在该模式中完成试飞。

试飞无问题后可以进一步启用动捕定高

```
//配置LED
#define FMU_LED_CONTROLL_ENABLE 1 // if use system default led control, set 1; if you

//配置磁罗盘
#define USE_MAG 0 // if use mag, set 1; if not use mag, set 0;

//配置GNSS
#define USE_GNSS 1 // if use gnss, set 1; if not use gnss, set 0;

//配置UWB
#define USE_UWB 1 // if use uwb, set 1; if don't use uwb, set 0;

//配置光流
#define USE_FLOW 1 // if use optical flow, set 1; if don't use optical flow, set 0;

//配置里程计
#define USE_ODOMETRY 1 // if use odometry, set 1; if don't use odometry, set 0;

//VINS
#define USE_VINS 0 // if use vins, set 1; if use lidar-slam, set 0;

//SLAM定高
#define USE_ODOM_Z 1 // if use slam z, set 1; else set 0;

//动捕
#define USE_MOTION 1 // if use motion capture, set 1; else set 0;
```

按上图启用动捕定高。保存编译烧录后，无人机进入动捕定位模式+动捕定高模式。