无人机开源项目学习

无人机的发展离不开开源社区的贡献，在Github上也有很多值得学习的开源算法。

目录

[1. Fast Planner 1](#_Toc159925033)

[1.1. 安装配置于Ubuntu 20.04 1](#_Toc159925034)

[1.1.1. 手动安装NLopt库 2](#_Toc159925035)

[1.1.2. 改为使用C++14编译 2](#_Toc159925036)

[1.1.3. 修改源码中函数返回值 3](#_Toc159925037)

[1.1.4. “/world”参数改为“world” 3](#_Toc159925038)

[1.2. 启动规划器Rviz仿真 3](#_Toc159925039)

[1.2.1. 手动选择目标点 4](#_Toc159925040)

[1.2.2. 自动设置目标点 4](#_Toc159925041)

[1.2.3. 启动逻辑简述 5](#_Toc159925042)

[1.3. 项目结构分析 6](#_Toc159925043)

[1.3.1. 重要节点功能分析 6](#_Toc159925044)

[1.3.2. 重要话题功能分析 7](#_Toc159925045)

[1.4. 源码分析 7](#_Toc159925046)

[1.4.1. FP核心节点的实现 7](#_Toc159925047)

[1.5. 实机飞行 9](#_Toc159925048)

# Fast Planner

·Github：<https://github.com/HKUST-Aerial-Robotics/Fast-Planner>

Fast Planner是香港科技大学空中机器人团队推出的基于深度视觉的开源路径规划器，以ROS功能包的形式发布，专门针对Aggressive Flight（高速敏捷飞行）场景。Fast Planner需要ROS环境才能运行。

下面简称FP

## 安装配置于Ubuntu 20.04

FP可以在Ubuntu 16.04及18.04上正常运行，但在Ubuntu 20.04上编译、运行时会报错，需要更改源码以及依赖项的配置才能正常运行。官方文档最后也提及部分问题的解决方案。

新建一个ROS的catkin工作空间，进入src文件夹中git clone拉取项目文件。

|  |
| --- |
| git clone https://github.com/HKUST-Aerial-Robotics/Fast-Planner.git |

### 手动安装NLopt库

·NLopt官方下载教程：<https://nlopt.readthedocs.io/en/latest/NLopt_Installation/>

NLopt是一个跨语言的非线性优化库，正常可以通过ROS安装，但因为未知原因，在Ubuntu 20.04下必须手动下载编译这个库项目文件，否则在Rviz仿真选择目标点时会导致规划器进程“kino\_replan.launch”崩溃。

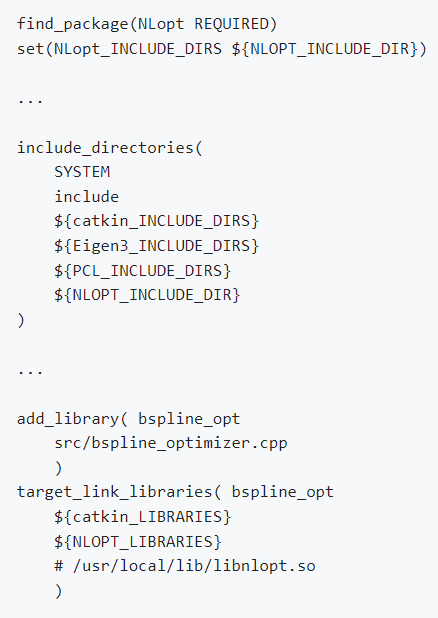
首先安装一个C++的线性代数库。

|  |
| --- |
| sudo apt install libarmadillo-dev |

git clone拉取NLopt的源码，安装到“/home/<username>/Document”文件夹中；进入项目文件夹开始CMake编译。

|  |
| --- |
| git clone <https://github.com/stevengj/nlopt.git>  cd nlopt  mkdir build  cd build  cmake ..  make  sudo make install |

按照FP官方指导修改B样条曲线优化器“bspline\_opt”的“CMakeLists.txt”配置文件，使其链接到NLopt的安装路径。



### 改为使用C++14编译

FP在Ubuntu 20.04上需要指定使用C++14标准编译，否则编译时会报错。

在项目文件夹“Fast-Planner”运行查找指令，找到所有功能包的“CMakeList.txt”编译配置文件，逐一在关键词“project()”的下面一行补上“set(CMAKE\_CXX\_STANDARD 14)”

|  |
| --- |
| find -name CMakeList.txt |
| project(…)  set(CMAKE\_CXX\_STANDARD 14) |

### 修改源码中函数返回值

这里需要修改三个函数的返回值，解决运行规划器“kino\_replan.launch”时出现的返回值报错。

找到“./fast\_planner/path\_searching/src/kinodynamic\_astar.cpp”中的  
“KinodynamicAstar::timeToIndex”函数，补充返回“return idx;”

|  |
| --- |
| int KinodynamicAstar::timeToIndex(double time)  {  int idx = floor((time - time\_origin\_) \* inv\_time\_resolution\_);  return idx;  } |

找到“./fast\_planner/plan\_env/include/edt\_environment.h”中“interpolateTrilinear”和“evaluateEDTWithGrad”函数，修改返回值类型为void

同样，在“./fast\_planner/plan\_env/src/edt\_environment.cpp”中修改以上两个函数的返回值为void。

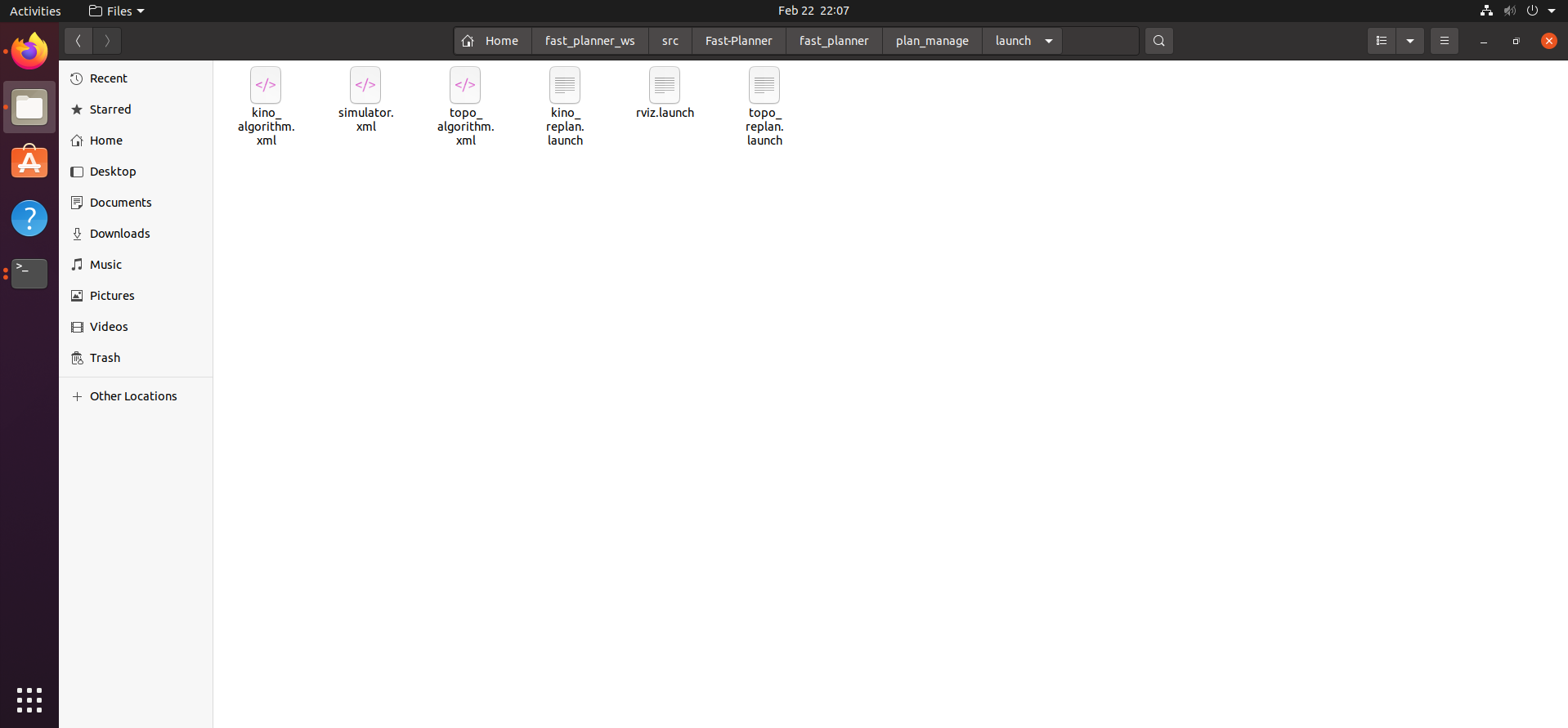
### “/world”参数改为“world”

tf2对斜杠敏感，这里主要为了解决运行“rviz.launch”无人机不显示的问题。

进入“/uav\_simulator/Utils/odom\_visualization/src/odom\_visualization.cpp”，修改所有传入函数的“/world”参数为“world”就可以了。

## 启动规划器Rviz仿真

FP的功能包“plan\_manage”下面有三个可执行的.launch文件，其中rviz.launch用于启动Rviz仿真，另外两个以“Kino”和“Topo”开头的分别对应FP提供的两种路径规划的方法，Kinodynamic和Topological，实际运行时选用一个即可。



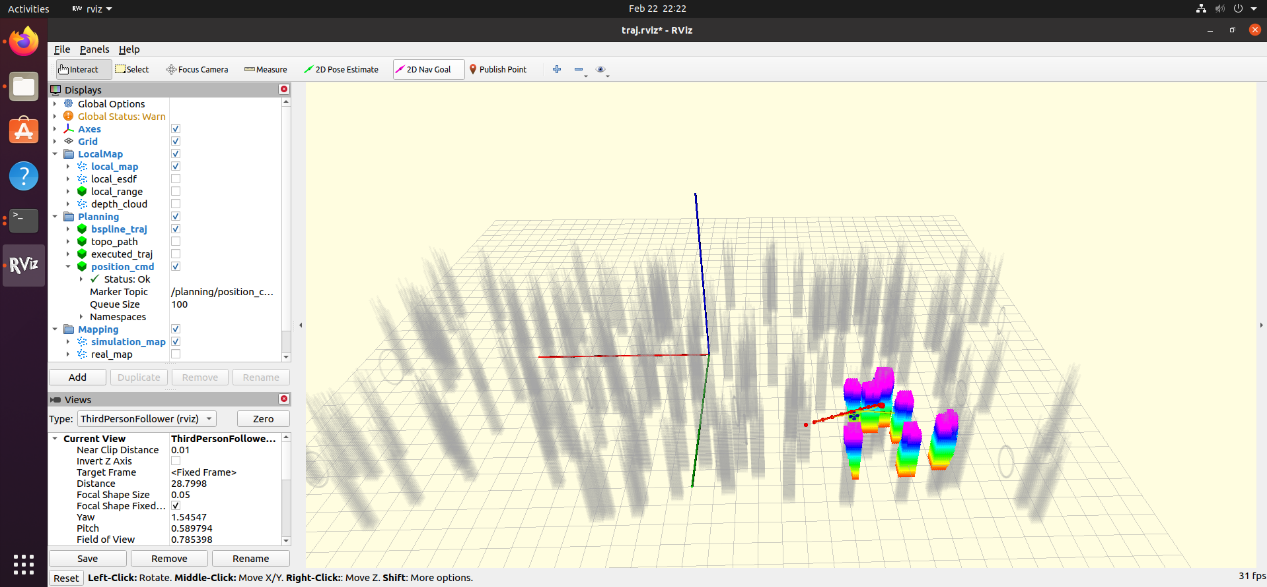
分析.launch内部的结构可以发现，Topo不仅支持设置多个目标点Waypoint，也支持多点生成一条连续的预设轨迹Reference Path，因此我们下面用Topo作为例子。

分别在两个命令行窗口中启动Rviz仿真器和FP规划器就成功运行了仿真。

|  |
| --- |
| roslaunch rviz.launch |
| roslaunch topo\_replan.launch |

### 手动选择目标点

FP默认设置为手动选择目标点，在打开的Rviz仿真窗口中选择左上角工具栏的“2D Nav Goal”，在图中点击就设置了一个目标点，无人机会自动规划路径沿目标点移动并躲避前进轨迹上的障碍物。



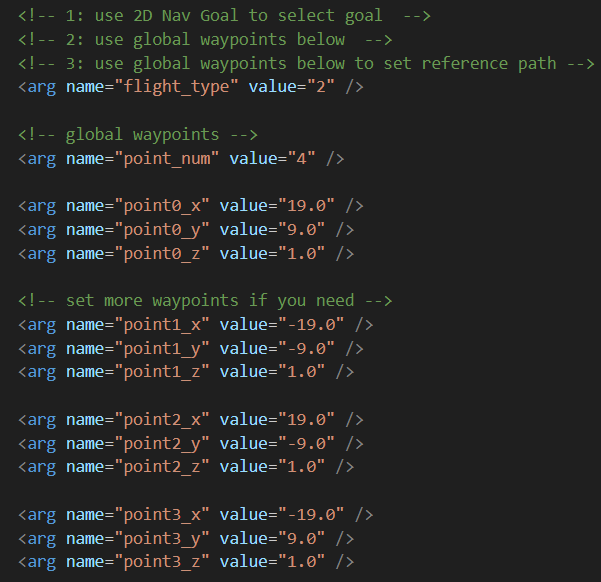
### 自动设置目标点

要想在规划器内部自动设置目标点，需要修改topo\_replan.launch和topo\_algorithm.xml两个文件的参数。FP中手动设置的目标点由/waypoint\_generator节点发布，自动设置的目标点保存在参数服务器中。

·修改飞行模式：

在topo\_replan.launch文件第37-60行，修改飞行模式“flight\_type”为3，在Rviz中用“2D Nav Goal”点击任意位置，无人机就会按顺序依次往各个目标点移动。

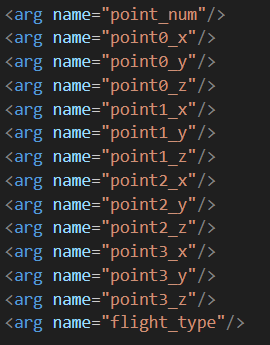
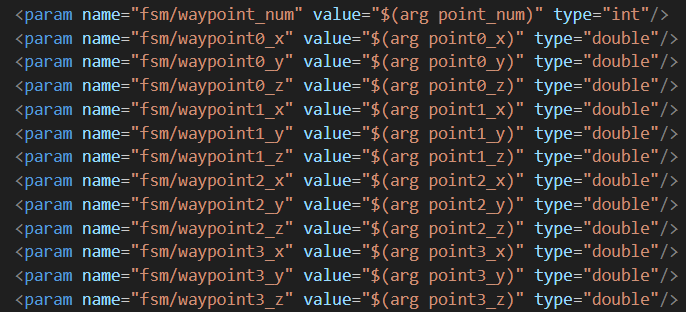
飞行模式2和3都是自动设置目标点，其区别是，模式2（源代码称“PRESET\_TARGET”）中无人机到达一个目标点以后需要重新用“2D Nav Goal”发布命令才会到前往下一个目标点。



·增加目标点：

如果需要增加目标点“point3”，不仅需要更改目标点数“point\_num”的值以及在下方增加新的“point3”点坐标，还需要在topo\_algorithm.xml中增加新目标点的声明和传参。

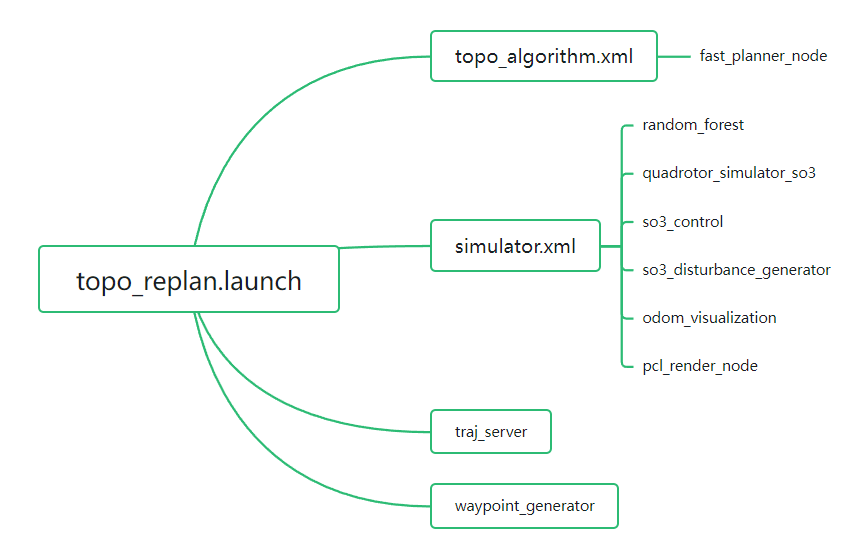
在topo\_algorithm开头一段加上“point3”的坐标变量声明，同时在打开节点“fast\_planner\_node”的部分仿照其他目标点的格式增加“waypoint3”的参数。

### 启动逻辑简述

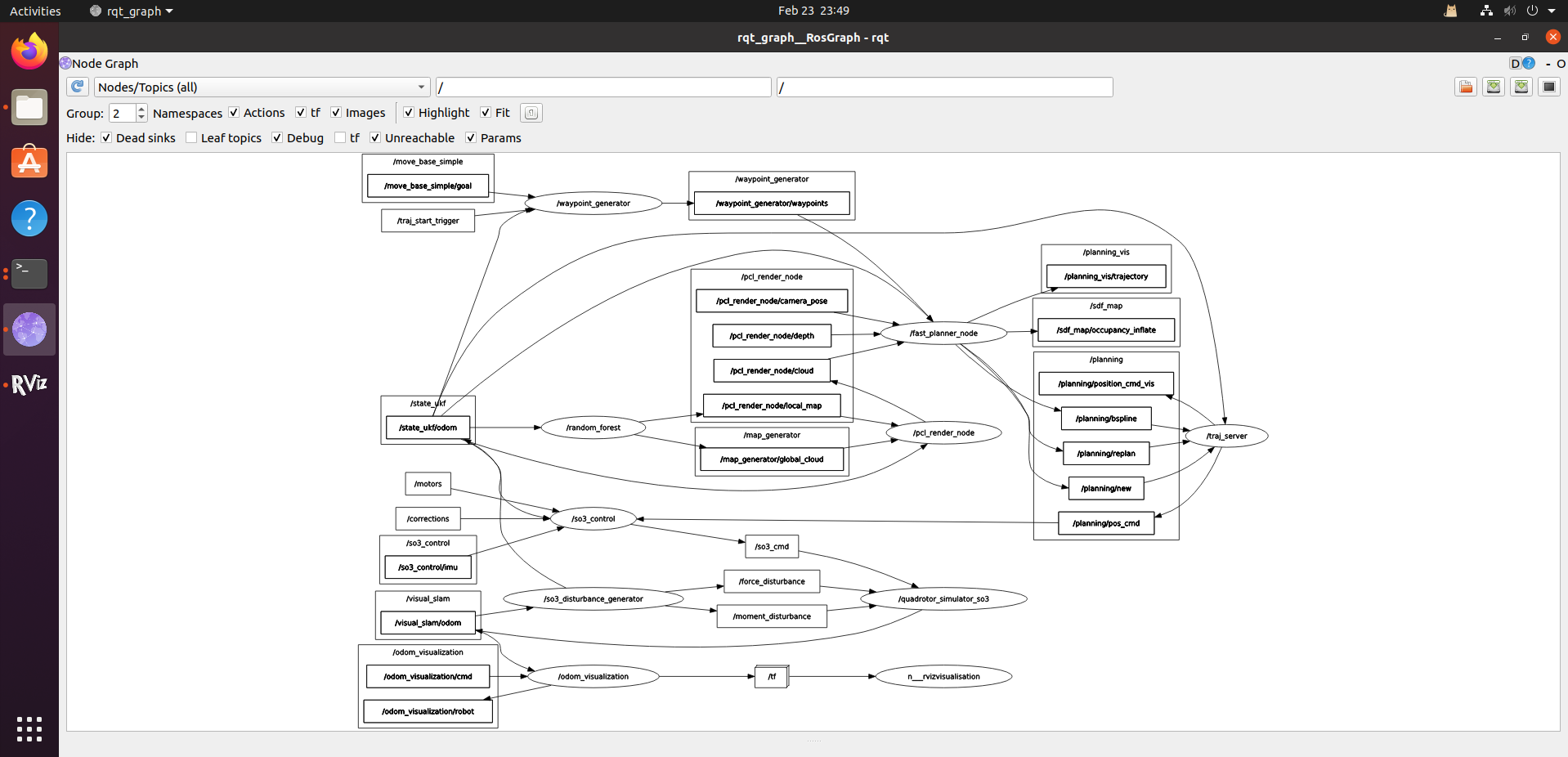
·rviz.launch：只有一行，使用traj.rviz配置文件启动Rviz节点

·topo\_replan.launch：直接启动两个节点，include了两个xml后缀的launch文件。 它主要的职能是在节点启动时传入预设的参数，以及实现节点内发布者、订阅者的相关话题重映射remap



## 项目结构分析

Topological规划器运行仿真后的完整节点和话题逻辑图如下，其中可以看到10个活跃节点，其中有一些是规划器的核心节点，有些则是仿真器的节点。为了能将项目用在实机上，我们应该关注核心节点的接口。



### 重要节点功能分析

|  |  |  |
| --- | --- | --- |
| **节点** | **功能类别** | **具体功能** |
| n\_rvizvisualisation | 仿真 | Rviz仿真器的节点 |
| /waypoint\_generator | 仿真 | 检测在Rviz中手动点击的点，发布手动目标点 |
| **/fast\_planner\_node** | 规划 | 规划器核心节点，运行FSM有限状态机 |
| /pcl\_render\_node | 仿真 | 模拟深度摄像机：根据仿真地图生成的障碍物和仿真无人机的位置生成虚拟的障碍物点云 |
| /traj\_server | 规划 |  |
|  |  |  |
|  |  |  |

### 重要话题功能分析

|  |  |  |
| --- | --- | --- |
| **话题** | **消息类型** | **具体功能** |
| /state\_ukf/odom | nav\_msgs/Odometry | 无人机的位姿、速度反馈话题，Odometry指的是“从传感器数据得到对机器人状态的估计” |
| /waypoint\_generator/waypoints | nav\_msgs/Path | 手动目标点话题，在Rviz中点击时会被发布一次，即使是通过参数服务器自动设置目标点也需要通过它触发规划器 |
| /planning/pos\_cmd | quadrotor\_msgs/PositionCommand | 无人机控制信号话题，包含位置、速度、加速度和偏航角 |
| /pcl\_render\_node/cloud | sensor\_msgs/PointCloud2 |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 源码分析

### FP核心节点的实现

·节点：“/fast\_planner\_node”

·所属功能包：plan\_manage

·源代码路径：“Fast-Planner/fast\_planner/plan\_manage/src/fast\_planner\_node.cpp”

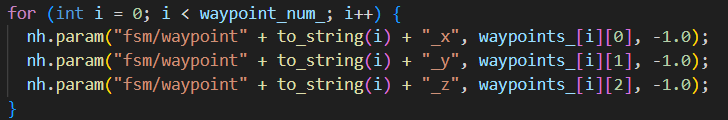
·关键头文件：“<plan\_manage/topo\_replan\_fsm.h>”、“<plan\_manage/planner\_manager.h>”

核心节点接收虚拟深度相机发送的点云消息“/pcl\_render\_node/cloud”和无人机的位置反馈消息“/state\_ukf/odom”，通过调用TopoReplanFSM类的成员函数实现轨迹计算，TopoReplanFSM类的声明位于“topo\_replan\_fsm.h”，定义位于“topo\_replan\_fsm.cpp”。

1. **目标点的接收**

·函数：“TopoReplanFSM::init()”、“TopoReplanFSM::waypointCallback()”

在对象初始化init()时，通过访问参数服务器中由launch文件设置的参数“fsm/waypoint”获取自动目标点，存放在二维数组“waypoints\_[][]”中



Rviz中的手动目标点则通过回调订阅话题“/waypoint\_generator/waypoints”获得。

注意，由于判断点的来源以及将目标点传入“planner\_manager\_”对象的代码位于订阅者的回调函数“waypointCallback()”中，即使是自动设置的目标点也需要触发回调函数（向“/waypoint\_generator/waypoints”发布一条消息）才能成功传入规划器。



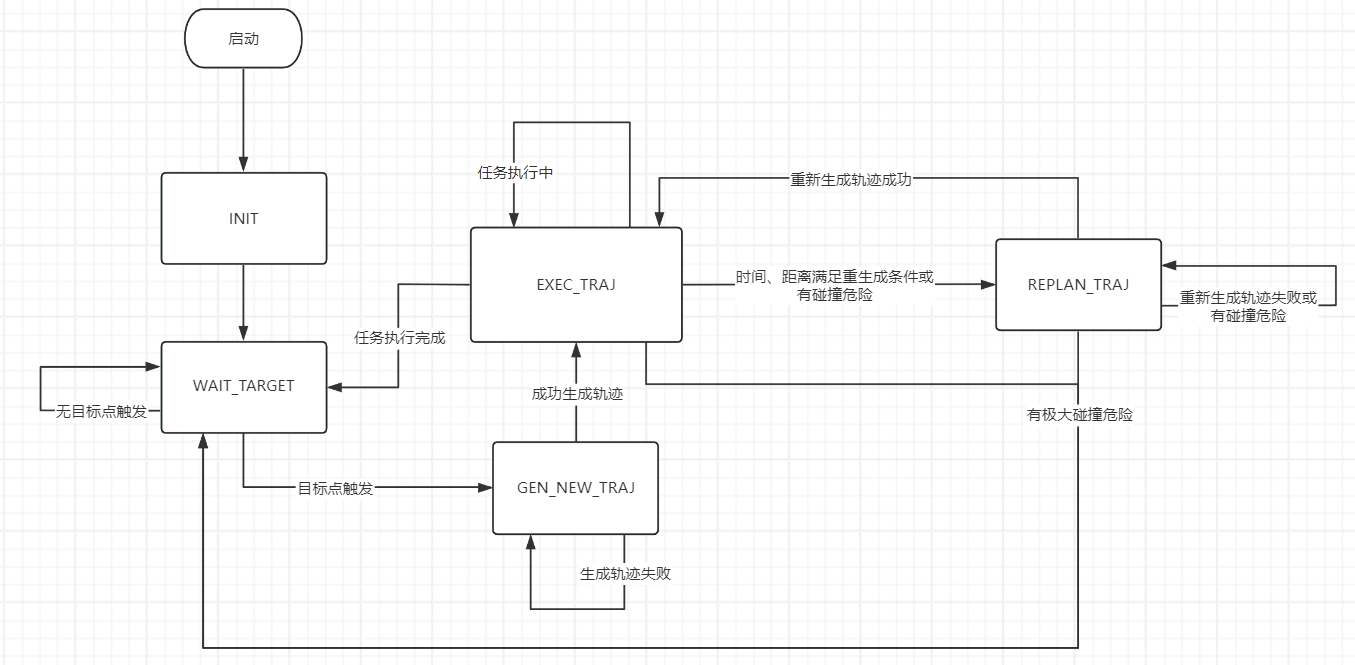
1. **有限状态机FSM**

·函数：“TopoReplanFSM:: execFSMCallback ()”、“TopoReplanFSM:: changeFSMExecState ()”

规划器内部的状态切换由一个有限状态机FSM控制，FSM则通过一个计时器的回调函数execFSMCallback ()实现，该函数每0.01秒触发一次。

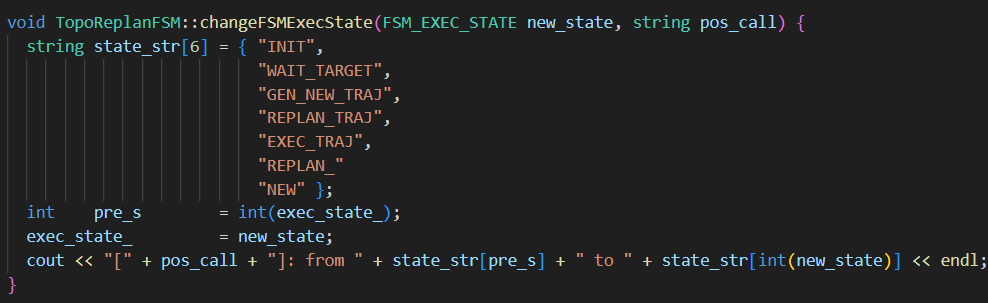
FSM一共定义了六个状态，当前激活的状态由枚举变量“exec\_state\_”记录，“REPLAN\_NEW”状态实际未使用。





1. INIT：规划器启动初始化；
2. WAIT\_TARGET：等待目标，表示当前无活跃任务且waypointCallback()未触发；
3. GEN\_NEW\_TRAJ：生成新轨迹；
4. REPLAN\_TRAJ：重新规划轨迹；
5. EXEC\_TRAJ：按规划好的轨迹执行任务；

状态的转换由changeFSMExecState()函数实现。



在checkCollisionCallback()函数检测到无人机存在碰撞危险的情况下，也会控制FSM进入重规划“REPLAN\_TRAJ”或者等待模式“WAIT\_TARGET”（急停）。

1. **防碰撞机制**

## 实机飞行

FP默认提供的启动文件topo\_replan.launch适用于Rviz仿真展示算法的效果，若想将FP用于实机飞行，则需要将FP的关键输入输出话题与MAVROS建立连接。