任务调度与决策模块

模块介绍

任务调度与决策模块，提供感知输入调度模块、规划执行输出调度模块的接口，以及决策的核心框架。

模块位于 `roborts_decision` 包中，依赖 `roborts_common` 包中的参数读取模块、 `roborts_costmap` 包中的代价地图对象（可选）以及 `roborts_msgs` 包中相关消息类型。

模块文件目录如下所示

```bash
roborts_decision/
├── behavior_test.cpp               #行为示例的测试节点
├── behavior_tree                   #决策框架示例，行为树
|   ├── behavior_node.h             #行为树节点类定义
|   ├── behavior_state.h            #行为树状态定义
|   └── behavior_tree.h             #行为树运行类定义
├── blackboard
|   └── blackboard.h                #黑板定义（决策框架的输入）
├── CMakeLists.txt
├── cmake_module
|   ├── FindEigen3.cmake
|   └── FindProtoBuf.cmake
├── config
|   └── decision.prototxt           #行为示例的参数配置文件
├── example_behavior                #行为示例 （决策框架的输出）
|   ├── back_boot_area_behavior.h   #返回启动区行为定义
|   ├── chase_behavior.h            #追击敌方行为定义
|   ├── escape_behavior.h           #看到敌方执行逃跑行为定义
|   ├── goal_behavior.h             #指定目标导航行为定义
|   ├── line_iterator.h
|   ├── patrol_behavior.h           #定点巡逻行为定义
|   └── search_behavior.h           #在地方消失区域搜寻行为定义
├── executor                        #任务执行的调度 （不同模块的任务委托）
|   ├── chassis_executor.cpp
|   ├── chassis_executor.h          #底盘任务调度类定义
|   ├── gimbal_executor.cpp
|   └── gimbal_executor.h           #云台任务调度类定义
├── package.xml
└── proto
    ├── decision.pb.cc
    ├── decision.pb.h
    └── decision.proto              #行为示例的参数配置文件
```

其中主要包括决策和任务调度两个核心部分：

决策模块

决策模块主要包括几个部分：

- 决策框架

  决策框架以观察（Observation）的信息为输入，以动作（Action）为输出，辅助机器人制定决策。当前官方提供的示例框架为行为树，详见

`roborts_decision/behavior_tree`

- 黑板

  黑板与游戏设计中黑板（Blackboard）的概念相似，作为当前决策系统中观察（Observation）的输入，用于调度一系列感知任务并获取感知信息。示例详见 `roborts_decision/blackboard/blackboard.h` ，用户可根据自身获取的信息种类完成blackboard类的修改与完善

- 行为

  机器人的具体行为经过不同程度的抽象，即可作为当前决策系统中的动作（Action）。我们提供了一系列具体行为示例，详见 `roborts_decision/example_behavior` ，用户可模仿示例自定义行为

任务调度模块

行为依赖的任务调度模块主要负责各个模块的任务委托，以调度功能执行模块来完成具体任务。

对于每一个调度模块，其核心在于

- 任务执行（具体任务的输入）
- 任务状态更新（反馈任务实时状态）
- 任务取消（打断后状态重置和相关回收）

三个调用接口，基本可以完成对于不同任务的调度。

其中，任务状态包括

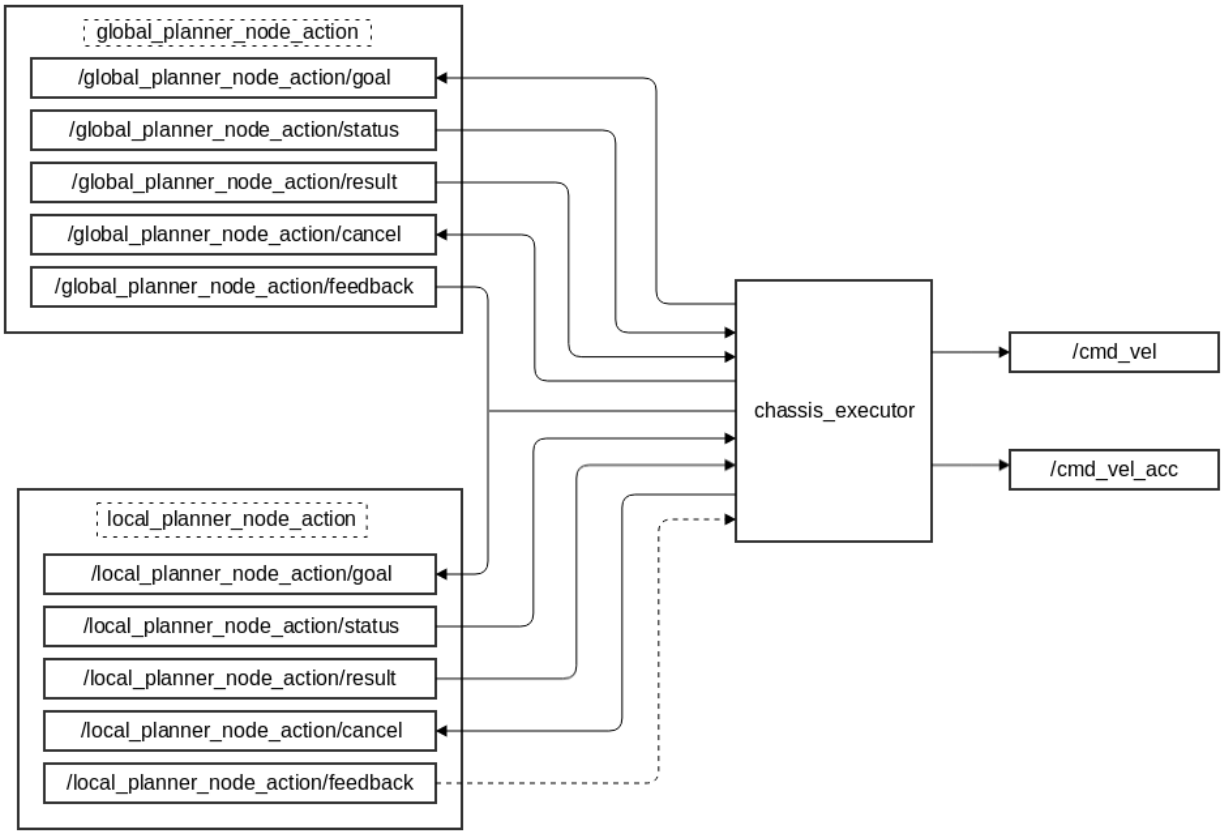
- 初始 IDLE
- 运行 RUNNING
- 成功 SUCCESS
- 失败 FAILURE

根据机器人模块主要分为

- 底盘调度模块
- 云台调度模块

底盘调度模块

底盘调度模块包括对底盘不同抽象程度的任务调度接口，运行框图如下所示



其中包括三个任务模式

- 运动规划控制

  输入目标点 goal (geometry_msgs/PoseStamped)调度全局路径规划和局部轨迹规划进行机器人底盘规划控制

- 速度控制

  输入速度 twist (geometry_msgs/Twist)直接进行机器人底盘匀速运动的速度控制

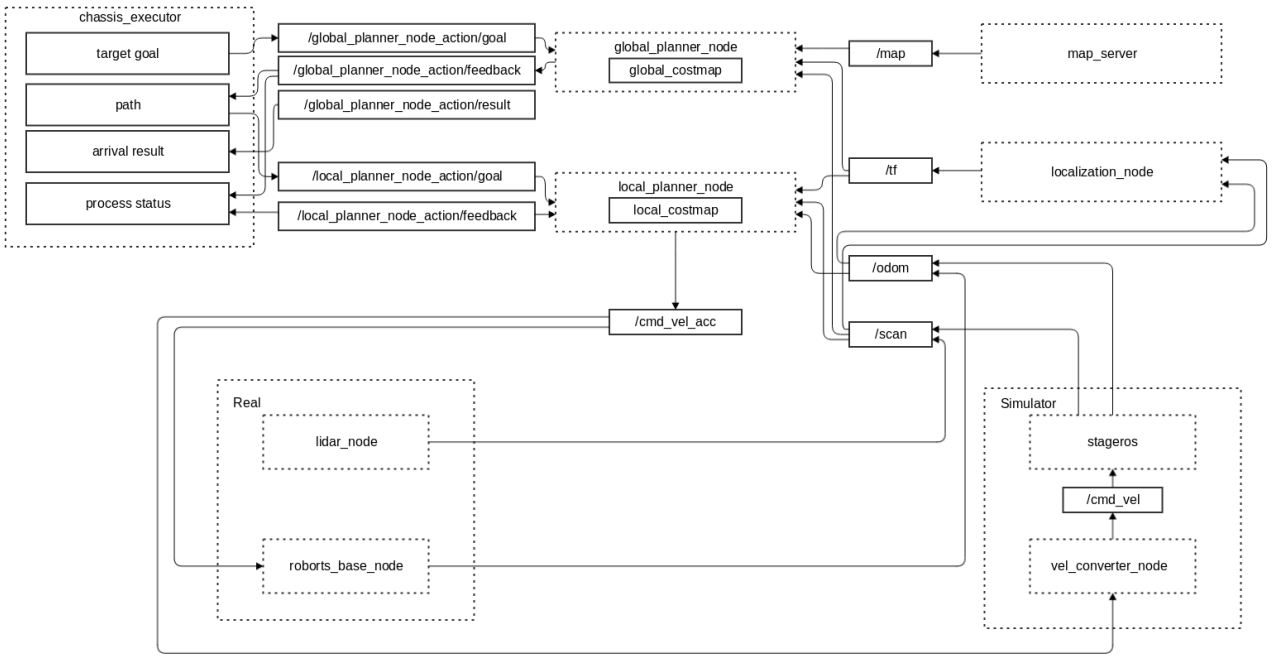
- 速度加速度控制

  输入速度与加速度 twist_accel (roborts_msgs/TwistAccel)直接进行机器人底盘匀加速度运动的速度控制

导航任务示例

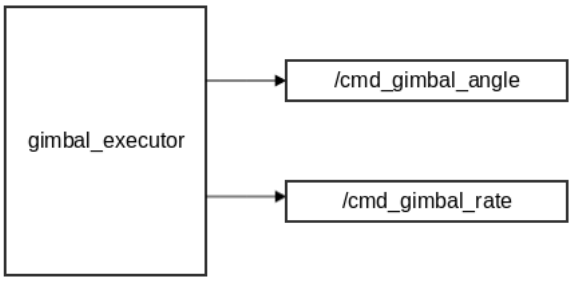
对于运动规划控制的导航任务，实际上是多模块节点相互配合的复杂任务，底盘调度模块将规划任务委托给全局规划模块和局部规划模块，最终输出速度和加速度控制量给底层主控板，而规划模块还依赖实时更新的里程计信息、定位信息与代价地图

在实际场景与虚拟环境中的导航系统框图如下所示



云台调度模块

云台调度模块包括对云台不同抽象程度的任务调度接口，运行框图如下所示



其中包括两个任务模式

- 角度控制

  输入目标角度 angle (roborts_msgs/GimbalAngle)直接进行机器人云台角度控制

- 速度控制

  输入目标角速度 angle (roborts_msgs/GimbalRate)直接进行机器人云台角速度控制

编译与运行

编译

在ROS的工作区内编译

```shell
catkin_make -j4 behavior_test_node
```

运行

在仿真环境中测试

```shell
roslaunch roborts_bringup roborts_stage.launch
```

启动行为测试节点

```shell
rosrun roborts_decision behavior_test_node
```

输入不同数字指令，可切换执行不同的行为