Navigation概念

ROS2

ROS2是Nav2中使用的核心中间件，详情可以参考https://docs.ros.org/en/rolling/。

Action Server

- action server是控制长周期任务例如导航的一种常用方法。它是这样的服务器，客户端请求它完成某个任务，但是它执行该任务需要很长时间。
- action server和client允许我们在另一个进程或线程中执行需要长时间运行的任务，并在将来返回一个结果，当然此时也允许阻塞直到动作完成。在执行任务的过程中，action server会为client端提供反馈，反馈定义在ROS的.action文件中，以feedback标识，同时.action文件中还定义了request和result。以导航为例，request可以是一个目标位置信息，feedback可以是已经导航的时间或者距离目标点的距离，result则是一个bool值表示是否成功到达目标点。
- feedback和result可以通过向client端注册回调函数的方式去同步完成收集工作，也可以从未来共享结果的方式异步的实现。两种方式都需要spin client的节点去处理回调组。
- 我们使用action servers通过名为NavigateToPose的action消息与最高级别的BT（Behavior Tree） navigator交互；这些servers还用于BT navigator与之后较小的action servers通信，这些较小的服务器将执行规划，控制和恢复等动作，他们都有各自的.action类型的消息与servers交互，消息类型定义在nav2_msgs中。

Lifecycle Nodes and Bond

- 生命周期节点（也称为管理节点）是ROS2独有的。他们是这样一类节点，实现了启动和销毁ROS2服务器的状态机转换机制。这有助于确定ROS系统在启动和关闭阶段的行为。它还能帮助用户以合理的方式去组织他们的程序，支撑商业目的和调试过程。
- 具体的生命周期过程为：
    - 一个节点启动时，它处于未被配置unconfigured的状态，只能处理节点的构造过程，而不应该包含任何ROS网络的建立或者参数读取的过程。
    - 通过launch系统或者提供生命周期管理之后，节点需要根据配置转换为非活动状态inactive。
    - 之后，节点才有可能通过激活阶段来转换到激活状态active。在这个状态，节点才被允许去处理信息并被设置为运行状态run。
    - 在配置阶段，通过触发on_configure()方法，将设置所有的参数，ROS网络接口和所有动态分配的内存以保证系统安全性。
    - 在激活阶段，通过触发on_activate()方法，将激活ROS网络接口并设置程序中的任何状态以开始处理信息。
    - 为了关闭节点，我们将依次过渡到停用、清理、关闭状态，并且以最终状态结束。在以上各个状态下，网络接口将被停用，停止处理消息，释放内存，干净的退出。
- 生命周期节点框架被广泛用于整个项目中（navigation），所有的servers都使用它。如果可能，所有的ROS系统最好也去使用生命周期节点。
- nav2_util LifecycleNode是LifecycleNodes的包装器。它隐藏了生命周期节点的大部分复杂度。同时，它也包括了一个生命周期管理器的绑定连接，用于确保一个服务器server发生转换后，它也能维持活动的状态。如果一个server崩溃了，它会让声明周期管理器知晓并关闭系统以防止严重错误的发生。

Behavior Tree

- 行为树用树形结构去描述待完成的任务，它创建了一个更具扩展性和更容易被理解的框架来帮助定义多步骤或者多状态的应用程序。
- 有限状态机（FSM）则与之相反，可能有几十个或更多的状态以及数百种转换。比如一个踢足球的机器人，如果将足球比赛的逻辑嵌入到一个FSM中将极具挑战性，还容易出错，有太多的状态和规则。此外建模选择如从左侧、右侧或中间射门将非常不清楚。
- BT和FSM相反，它将为许多行为创建和重用基本原语，如“踢”、“走”、“追球”等。
- 在Nav2项目中，我们使用BehaviorTree CPP V3 作为BT库。我们在BT Navigator模块中创建了节点插件。这些节点插件被加载到BT中，并且当树的XML文件被解析时，注册的名称也会和插件关联。此时我们就可以通过BT进行导航了。
- 我们使用这个库的原因之一就是它能够嵌套子树。这意味着一个Nav2的行为树能白加载到另一个更高级别的BT上，只须我们将当前的行为树看做节点插件。以足球比赛为例，使用Nav2行为树建立一个“追球”节点，而将对足球的检测作为此任务的一部分。
- 此外，我们为BT提供了名为NavigateToPoseAction的插件，这样就可以通过常用的action接口操作从客户端调用Nav2。

Navigation Servers

- 规划器和控制器是导航任务的核心。恢复器则被用于让机器人摆脱不良状态或者处理各种形式的问题，以提升整个系统的容错能力。
- 规划器、控制器和恢复器是Nav2中的三个action server。作用一是承载一个算法插件的地图以完成不同的任务；作用二是承载被算法插件用来计算其输出的环境表示。
- 规划器和控制器服务器将在运行时配置自己的名称（别名）和算法的类型。名称是当前任务的别名，算法类型就是算法插件库的注册名称。比如一个DWB控制器被成为FollowPath，是因为它遵循参考路径FollowPath。在这种情况下，DWB的所有参数都将被放在命名空间FollowPath.下。
- 执行过程：规划器和控制器将与任务相关的接口暴露出来。当一棵行为树唤醒与任务相应的BT节点时，该行为树将调用action server去处理这个任务。action server回调将根据映射到指定算法的名称（如FollowPath）调用被选中的算法。这样将允许一个用户把行为树中使用的算法抽象为算法类。如，你有n个插件控制器去跟踪路径、与充电器对接、躲避动态障碍物或者与工具交互。在同一个server中拥有以上所有的插件让用户可以使用单个环境表示对象，通常这个环境表示对象的复制成本很高。
- 对于恢复服务器，每个恢复器都有自己的名字，然而每个插件也会公开各自特定的action server。这样做是因为各种不同的恢复操作无法共享同一个简单的操作接口。恢复服务器也包含一个对本地代价地图的订阅者，用于实时从控制器接收更新，以完成任务的计算。我们这样做避免了局部代价地图拥有多个实例，因为高昂的复制成本。
- 由于BT节点是调用一个动作的的简单插件，因此可以创建新的BT节点来调用具有其他动作类型的动作服务器。如果可能的话，建议始终使用所提供的服务器。如果由于插件或者action接口需要，则应该创建一个适应于当前框架的新服务器。这个新服务器应该使用新类型和插件接口，类似于已经提供的服务器。

规划器

- 规划器的任务是计算一条路径以完成目标函数的功能。路径也称为路线，它取决于所选的命名方法和算法。两个典型的例子，一个是计算一条前往目标点的路径，另一个是规划覆盖所有空闲区域的路径。规划器将访问全局环境表示和缓存在其中的传感器数据。规划器将被设计为：
    - 计算最短路径
    - 计算完整覆盖路径
    - 计算稀疏或预定义的路径
- Nav2中的规划器是计算从当前位置到目标位置的有效且最佳路径。

控制器

- 控制器，也就是ROS1中的局部规划器，是我们遵循全局计算的路径或者完成本地任务的方式。
- 控制器将访问局部的环境表示，以尝试计算机器人本体应该遵循的可行控制。很多控制器会将机器人在空间中向前投射，并在每次更新迭代时计算一个局部的可行路径。
- 控制器将被设计为：
    - 跟随一条路径
    - 在里程计框架中与一个充电站对接
    - 登上电梯
    - 与工具交互
- 在Nav2中控制器的任务通常是计算出有效的控制过程来遵循全局规划。

恢复器

- 恢复器是容错系统的支柱。它的目标是处理系统中的未知或错误状态并自动解决这些问题。
- 使用场景
    - 感知系统的错误导致环境的描述中充斥着虚假的障碍物，此时会触发清理代价地图的恢复器，以便机器人继续运动
    - 机器人由于动态的障碍物出现或者不良操作而卡住了，如果可能的话，通过后退或者在原地旋转，机器人能够从恶劣的位置进入到一个自由运动的空间
    - 如果系统完全故障，恢复器将被用于寻求帮助，如通过email、短信、Slack、Matrix等

航点跟随

- 航点跟随是导航系统的基本功能，它告诉我们的系统如何利用导航到达多个目标点。
- nav2_waypoint_follower功能包拥有一个带特定任务执行器插件的航点跟随程序。目的是前往一个指定的位置并执行特定任务，比如取照片、搬箱子或者等待用户输入其他操作。
- 两种设计思路
    - 愚蠢的机器人，智能化的中央调度

      在这种模式下，nav2_waypoint_follower足以创建产品级的机器人解决方案。由于自动化系统或调度系统在分配任务时会考虑机器人的姿态、电池电量、当前任务等因素，机器人应用程序只需要负责手头的任务，而不必顾及系统的复杂性。此时，只需将向航点跟随器发出的请求视为一个工作单元（如仓库中的一个拣选、一个安全巡逻等）来执行任务，然后返回到给调度器以执行下一个任务等等。在这种思路下，航点跟随应用比导航高一步，但是低于系统自主应用。
    - 智能化的机器人，愚蠢的中央调度

      在这思路下，nav2_waypoint_follwer是一个示例应用/概念验证程序，但是开发者确实需要机器人上的航点跟踪或自动化系统承在制定稳定的解决方案时发挥更大作用。。此时，需要使用nav2_behavior_tree包创建自定义应用程序的行为树，并使用导航来完成任务。BT树的子树可以执行其他任务，如在运动过程中检车电池电量。很快nav2_bt_waypoint_follower就能上线来提供这个功能。

状态估计

- 在导航工程中，需要提供两个主要的坐标转换过程
    - map to odom转换由定位系统（localization, mapping, SLAM）来提供
    - odom to base_link由odometry系统提供
- 标准Standards

    - REP 105定义了导航和ROS生态系统所需的框架和约定。使用社区中丰富的定位、里程计和slam相关的工程都应该一直遵守这个标准。
    - 简而言之，REP-105标准规定你必须为你的机器人至少创建一棵TF树去完成一个完整的map->odom->base_link->[sensor frames]的变换过程。TF2是ROS2中的实时变化的坐标变换库。全局的定位系统如GPS、SLAM等的工作是提供map->odom的转换。里程计系统odometry提供odom->base_link的转换。其他相对于base_link的变换是静态的并定义在URDF中。
- 全局定位：Localization and SLAM
    - amcl方法：为静态地图的定位服务的一种例子滤波技术
    - SLAM toolbox：默认的SLAM算法，用于定位和生成静态地图
- Odometry
    - 里程计的来源：LIDAR, RADAR, 车轮编码器，VIO, IMU等
    - 里程计目的是提供基于机器人运动的平滑而连续的局部坐标系。全局的定位系统将更新全局坐标系来解决里程计漂移的问题
    - robot localization功能包用于进行多传感器的数据融合。他采用n个不同类型传感器的数据计算出一个连续且平滑的里程计（测距），并提供给TF和相应的topic。一个典型的移动机器人的里程计来自于车轮编码器、IMU和视觉数据的融合。
- 环境表示Environmental Representation
    - 他是机器人感知环境的方式
    - 它充当着各种算法和数据源的中心定位器，负责将他们的数据信息整合到一个空间中
    - 控制器、规划器和恢复器使用这个空间来高效且安全地完成各自的任务。
- 代价地图和分层 costmap and layers
    - 当前环境表示就是一个代价地图，它是一个规则的2D网格，网格的每个单元都包含着一个代价来反应它是未知空间、空闲空间、被占据空间或者膨胀空间。代价地图会被搜索遍历去计算出一个全局的规划，或者采样去计算一个局部的控制。
    - 代价地图被分为不同的层次，每个层次都被实现为一个插件用来缓存代价地图中的信息，其中包括来自LIDAR，RADAR，声呐，深度相机等的信息。比较聪明的做法是在数据被导入到代价地图图层之前完成传感器数据的处理，但这取决于开发者。
    - 代价地图的图层被创建出来用于侦测和跟踪场景中的障碍物，以实现避障功能。此外，可以基于某些规则或启发算法创建图层来更改底层的代价地图。最后这些图层可以被用来缓存2D或3D世界中的实时数据，以进行二元障碍物的标记。
- 代价地图过滤器 costmap filters
    - 假设，你正在为地图文件或者任意图片文件做注释，以便根据被注释的地图中的位置去执行特定操作。这个注释或标记可以是防止进入区域以避免规划路径到其中，也可以是让一些像素属于标记区域的最大速度。这个被注释的地图被称为“过滤器掩码”。就好像是一个表面上的一副面具，它的大小，位置和规模相比于主地图可以相同，也可以不同。
    - 代价地图过滤器是基于代价地图图层的方法，它将过滤器器掩码中注释的空间相关的行为变化应用到Nav2栈中。这些过滤器被实现为代价地图的插件。这些插件被成为过滤器是因为他们通过标记在过滤器掩码上的口弄件注释来过滤一个代价地图。为了制作一个过滤代价地图并在注释区域改变机器人的行为，过滤插件从过滤器掩码读取数据，这个数据在过滤空间被线性地转换为特征地图。拥有了这个转换后的特征地图以及代价地图，任何传感器数据和当前机器人坐标过滤器都能更新底层的代价地图，并根据机器人的位置改变其行为。以下功能可以通过使用代价地图过滤器来实现：
        - 机器人永远不会进入的区域（安全区）
        - 限速区域，进入这些区域的机器人的最大速度将受到限制
        - 机器人在工业环境和仓库中移动的首选通道
- 其他形式
    - 梯度图，类似于代价地图
    - 3D代价地图
    - Mesh map

- 矢量空间