项目背景

1. ROS工具
    - gazebo

> gazebo是一款免费的机器人仿真软件，其提供高保真度的物理模拟，一整套传感器模型，以及对用户和程序非常友好的交互方式。能够在复杂的室内和室外环境中准确高效地模拟机器人工作的功能，通常与ROS联合使用，为开发者提供了优异的仿真环境。
>
> 打开gazebo的时候，默认是没有模型的，需要手动下载配置。
>
> ---
>
> **怎么处理**
>
> **1、下载模型文件**
>
> github链接：
>
> osrf/gazebo_modelsgithub.com/osrf/gazebo_models
>
> 该库包含许多常用模型，将其下载为压缩包格式。
>
> **2、创建models文件夹**
>
> ```
> $ cd ~/.gazebo/
> $ mkdir -p models
> ```
>
> ./gazebo文件夹默认被隐藏，需要按下“Ctrl+H”才能看到被隐藏的文件。
>
> **3、拷贝模型文件**
>
> 最后，将下载好的压缩包拷贝至新建的models文件夹下并**解压**。再次打开gazebo便可以加载我们下载好的models了。
>
> **4、测试**
>
> 打开gazebo，在“insert”面板中选择模型导入，查看效果。
>
> ```
> $ gazebo
> ```
>
> ---
>
> **两种形式：urdf与sdf**
>
> **1、urdf是什么?**
>
> **统一机器人描述格式**（**urdf**）是ROS用于描述机器人的所有元素的XML文件格式。要在gazebo中使用urdf文件，必须添加一些特定用于仿真的标签才能与gazebo一起正常使用。
>
> 尽管urdf在ROS中是一种有用且标准化的格式，但它们缺少许多功能，并且尚未进行更新以应对机器人技术的不断发展的需求。urdf只能单独指定单个机器人的运动学和动力学特性，无法指定世界中机器人本身的姿势。它也不是通用的描述格式，因为它不能指定关节环（平行连接），并且缺乏摩擦和其他特性。此外，它不能指定非机器人，例如灯光，高度图等。
>
> 在实现方面，urdf语法大量使用XML属性破坏了正确的格式设置，这反过来又使urdf更加不灵活。

2、sdf是什么?

为了解决此问题，创建了一种称为仿真描述格式（sdf）的新格式，供gazebo使用，以解决urdf的缺点。sdf是从世界级到机器人级的所有内容的完整描述，能够描述机器人、静态和动态物体、照明、地形甚至物理学的各方面的信息。sdf可以精确描述机器人的各类性质，除了传统的运动学特性之外，还可以为机器人定义传感器、表面属性、纹理、关节摩擦等；sdf还提供了定义各种环境的方法。包括环境光照、地形等。

sdf也使用XML格式进行描述。

总结而言，sdf是urdf的进化版，能够更好的描述真实的模拟条件。

3、使用哪种格式?

尽管目前有一些sdf与urdf的之间的转换方法，但往往十分复杂且易出错。因此，建议刚开始就根据自己的需求选择最合适的模型格式。

- 必须使用urdf的情况：要使用rviz进行可视化操作。
- 必须使用sdf的情况：研究并联机器人，或机器人中存在封闭链结构。
- 建议使用urdf的情况：要尽快做出仿真用以演示效果；使用Solidworks建模，想方便地导出ROS三维模型。
- 建议使用sdf的情况：想深入研究ROS-gazebo仿真，使仿真的动力学特性更加真实；想开发自己专用的Gazebo仿真插件。

4、urdf和sdf格式转换

在一些特殊情况中，只能使用urdf或sdf格式的模型。这时候，如果我们已有的模型格式不符要求，就需要转换。但值得注意的是，由于urdf和sdf的元素并不完全对应，因此下面列出的转换过程或多或少存在一些问题。**刚开始搭建模型，还是建议直接选用合适的格式，用xacro或rsdf参数化和模块化方法建模，而非使用转换功能。**

（1）urdf转sdf

由于sdf是gazebo的原生格式，因此urdf转sdf是比较简单的。采用的方法一般是：使用常规方法将*.urdf加载到gazebo中后，再将其另存为一个单独的*.world文件。此时sdf格式的模型就完整地保存在*.world文件的元素下了。

另外，gazebo官方还提供了另一种命令行方法，也可将实现urdf转sdf。在urdf所在目录下打开终端，执行如下命令：

```
$ gz sdf -p my_model.urdf > my_model.sdf
```

（2）sdf转urdf

虽然官方没有给出将sdf转为urdf的方法，但好在有大佬自己开发了可行的工具——pysdf功能包，github链接：

https://github.com/andreasBihlmaier/pysdfgithub.com/andreasBihlmaier/pysdf![img](https://pic2.zhimg.com/v2-e1c3125fa848db3b854a97c54d4ea6e9_180x120.jpg)

该包的版本过老，在ubuntu18.04和ros melodic上直接运行报错，对其做了一些细微的修改后可用：

https://github.com/chenjm1109/pysdfgithub.com/chenjm1109/pysdf

该功能包使用起来也很简单，分为4个步骤：

- 在github中下载pysdf功能包，放到ROS工作空间（以catkin_ws为例）下的src目录下；
- 编译工作空间

```
$ cd ~/catkin_ws/
$ catkin_make
```

- 从待转换的sdf模型文件所在的目录打开终端，执行如下命令

```
$ rosrun pysdf sdf2urdf.py my_model.sdf result_model.urdf
```

注意事项：

- sdf2urdf.py是python可执行文件，如果报错[rosrun] Couldn't find executable named sdf2urdf below...，就需要先通过`chmod +x *`指令为其赋予可执行权限，这是使用所有ROS-python可执行文件时都要注意的事情。
- sdf文件中不要有插件，也不要有ball关节等urdf无法识别的关节类型。
- 转换完成后可使用`check_urdf`工具检查urdf的合法性，命令如下：

```
$ sudo apt-get install liburdfdom-tools
$ check_urdf result_model.rudf
```

2. 对时间要求比较高的任务
    - 小车

> 无人驾驶中的避障（从收到信息到刹车的时间）

往年项目

1. 2019年的OSH项目

> 基于冯诺依曼架构的现代计算机，由于程序计数器 (Program Counter) 带来的硬件根本概念的串行性，处理大批量数据流的能力十分有限。尽管现代计算机利用指令级并行、多核多线程编程等带来了大幅的性能提升，但在例如服务器等的海量 IO 和数据并发场景下，冯氏结构与并行性之间的矛盾愈加显著。与此同时，CPU 与主存之间的速度不一致进一步限制了海量数据并发下的处理效率。
>
> 为了应对处理大量高速数据流需求，基于数据流驱动架构的智能网卡 (SmartNIC) 应运而生。区别于传统的控制流计算机，数据流计算机在原理上不存在 PC 寄存器，只有当一条或一组指令所需的操作数全部准备好时，才能激发相应指令的一次执行，执行结果又流向等待这一数据的下一条或一组指令，以驱动该条或该组指令的执行。因此，程序中各条指令的执行顺序仅仅是由指令间的数据依赖关系决定的。另一方面，数据流计算模型中没有传统的变量这一概念，它仅处理数据值，忽略存放数据的容器，从而具有纯函数的特点。
>
> //上面都是背景
>
> 于是他们采用了网卡进行并行处理，并对数据进行了方差分析，（虽然实验报告乱的跟一坨大便似的）卷积，函数分析等数学运算（虽然感觉他们也没有做什么，啊吧啊吧）

2. 其他的项目

> 看到了很多奇奇怪怪的项目，但是都是上古时期的了
>
> - 基于网卡的混合模式（虽然没什么用，但是起码是网卡的一种使用方式，不是吗）
>
> 在一个网卡，一条线路上多个帐号同时拨号（或多种不同的连接方式）以达到叠加的效果。 里边包含“静态IP”，“DHCP/动态IP”，“ADSL/PPPoE拨号”三种接入方式，可以混合同时接入。

- 基于智能网卡（Smart Nic）的Open vSwitch卸载方案（(20条消息) 基于智能网卡（Smart Nic）的Open vSwitch卸载方案简介_qingdao666666的博客-CSDN博客）

  这个也是网卡相关的

- 一个做数据实时同步的(20条消息) 两万字讲全数据实时同步方案(附代码及架构图)(建议收藏)_无精疯的博客-CSDN博客