Working Notes

qiaoyx

May 29, 2019

Contents

Part I

ISaac 笔记

Chapter 1

概述

1.1 简书:bazel & bazel office & bazel install on ubuntu

注意：当通过 install_dependences.sh 安装依赖环境时，出现 bazel 安装不成功的情况，请参考上述第三个连接，推荐其中的下载安装。

1.2 第三方包引入

以下三个函数用于引入第三方源码包：

1. isaac_new_http_archive
2. isaac_git_repository
3. isaac_new_local_repository

另外，尝试 isaac_git_repository 的方式不成功，配置比较繁琐，同时开发文档中不推荐 git 方式，下载比较慢，而且要考虑更多的版本控制问题。知乎上的介绍： new_local_repository 方法以及 IDE 配置方式。

1.3 几个重要的概念

isaac 框架中，有几种组织结构。他们与 isaac.bzl 中的构建函数对应关系为:

结构名称	对应的构建函数	其他
application	isaac_app	最高层次的结构
module	isaac_cc_module	c++, 其他语言的函数名中缀不同
component	isaac_component	
codelet	/	代码包, 这是 isaac 框架中的元单元

上表中由高到低列出各组织结构,isaac 架构是严格按照由下向上逐级构建的, 大致如下:

- 编写代码模块, 称为 Codelet
  需要继承自 isaac::alice::Codelet, 类似于 ros::node 和 ros::nodelet
- 生成 component
  由一个或者多个 Codelet 生成一个 component. 其过程应该是多个.o 文件连接成一个 so 文件, 具体还需要进一步了解.
- 构建 module
  由一个或者多个 components 生成一个 module, 以 so 库文件的形式提供给 application 使用.
- 构建 application.
  由一个或者多个 modules 生成一个可执行文件, 且可以为其指定 json 格式的配置文件. 通常, 我们将 modules 放在 isaac/packages 目录下, 而 application 则在 isaac/apps 目录或者子目录下. 也就是,packages 下的包, 主要提供 so 库文件给 application 调用. 但这些库必须严格遵守 isaac 的消息机制, 详细可参考说明文档 Message API Overview 及 Component API Overview.

  isaac 中，application 的构建非常优雅，只要在配置文件中指定各 node，以及他们间的消息流图和参数配置就完成了，基于 message 接口，就像是在搭积木.
- Codelet 最重要的接口: tick() -> void
  该接口的机制类似于 ros::spinOnce(), 相当于该代码块的主逻辑。而 tick 调用机制一共有三种，详见开发手册。

Chapter 2

完整流程示例

以 ydlidar 包构建一个 C++ 第三方工程为例, 默认 WORKSPCE 目录为: isaac/

2.1 isaac/WORKSPCE

```
cd isaac && vim WORKSPCE
────────────────────────────────────────────────────────
workspace(name = "isaac")

load("//engine/build:isaac.bzl", "isaac_git_repository", "isaac_new_http_archive")
load("//third_party:engine.bzl", "isaac_engine_workspace")
load("//third_party:packages.bzl", "isaac_packages_workspace")
load("//third_party:ros.bzl", "isaac_ros_workspace")
load("//third_party:zed.bzl", "isaac_zed_workspace")

## added by qiaoyx, so named *qiaoyx*, you know!
load("//third_party:qiaoyx.bzl", "isaac_qiaoyx_workspace")
isaac_qiaoyx_workspace() # why a funtion call here? just to see qiaoyx.bzl.
## end adding here by qiaoyx. below is the raw code ...

isaac_engine_workspace()

isaac_packages_workspace()

isaac_ros_workspace()

isaac_zed_workspace()

####################################################################################################
# Load cartographer

isaac_git_repository(
name = "com_github_googlecartographer_cartographer",
commit = "b6b41e9b173ea2e49e606f1e0d54d6d57ed421e3",
licenses = ["@com_github_googlecartographer_cartographer//:LICENSE"],
remote = "https://github.com/googlecartographer/cartographer.git",
)
```

```
...
```

2.2 创建 Sub Workspace: isaac/third_party/qiaoyx.bzl

```
cd isaac/third_party && touch qiaoyx.bzl
vim qiaoyx.bzl
──────────────────────────────────────────────────────────────
load("//engine/build:isaac.bzl",
     "isaac_http_archive", "isaac_new_http_archive")

def clean_dep(dep):
    return str(Label(dep))

# loads dependencies for various modules
## 注意:这个函数就是自定义的workspace, 在isaac/WORKSPCE中被调用
def isaac_qiaoyx_workspace():
    isaac_new_http_archive(
        name = "ydlidar",
        ## 注: 要在同级目录下创建ydlidar.BUILD文件, 编写构建规则!
        ## 这里只负责下载和按照一定的要求来解压
        build_file = clean_dep("//third_party:ydlidar.BUILD"),
        sha256 = "3e1d9b56e59c2720fd23561e531f9b3d6282e4052f5c66c3ab5aab0984ea1fee",
        url = "https://github.com/YDLIDAR/sdk/archive/v1.3.9.tar.gz",
        type = "tar.gz",
        strip_prefix = "sdk-1.3.9",
        licenses = ["//:LICENSE"],
        )
```

2.2.1 sha256 字段说明

刚指定下载地址时, 可能不知道加密编码, 有几种方式来解决:

1. 单独下载源码包，使用 sha 加密工具计算编码。或者,
2. 去掉这个字段，傲慢的忽略掉 (验证过，可行)。编译界面会提示下载进度以及 sha 编码。
3. 按照第 2 步的方式，此时可选将 sha256 码复制到 bzl 中或者忽略。

2.2.2 strip_prefix 字段

2.2.2.1 指定: strip_prefix = "sdk-1.3.9"

```
tree -d
.
├── doc
│   └── html
│       └── search
├── include
├── samples
└── src
    └── impl
        ├── unix
        └── windows
```

2.2.2.2 不指定: strip_prefix

```
tree -d
.
└── sdk-1.3.9
    ├── doc
    │   └── html
    │       └── search
    ├── include
    ├── samples
    └── src
        └── impl
            ├── unix
            └── windows
```

2.3 在 isaac/third_party 中创建对应的 BUILD 文件: ydlidar.BUILD

```
cc_library(
    name = "ydlidar",
    srcs = glob(["*.cpp", "*.h"],
                exclude=["ydlidar_driver.h",
                         "CYdLidar.h",],
    ),
    hdrs = glob([
        "include/*.h*",
    ]),

    copts = [
                "-Wno-shift-negative-value",
                "-Wno-implicit-function-declaration",
                ],
    visibility = ["//visibility:public"],
    includes = ["include"],
)
```

该文件是对应于 isaac/third_party/qiaoyx.bzl:strip_prefix = ”sdk-1.3.9” 而写的.
当不指定 strip_prefix 字段时:

```
cc_library(
        ...

        hdrs = glob([
            "sdk-1.3.9/include/*.h*",
            ## 还有更高级的通配符,适合用于多子目录情况,如下:
            ## "sdk-1.3.9/**/*.h*",
        ]),

        ...

        includes = ["sdk-1.3.9/include"],
    )
```

特别注意: includes 字段的设定, 是为了引用头文件时更加方便. 如果不设定该字段, 在 IsaacYdlidar.cpp 中, 要这样:

```
#include "sdk-1.3.9/include/ydlidar_driver.h"
#include "sdk-1.3.9/include/CYdLidar.h"
#include "IsaacYdlidar.hpp"

...
```

如上, 当指定 includes = [”sdk-1.3.9/include”] 后, 就不需要关心路径了:

```
#include "ydlidar_driver.h"
#include "CYdLidar.h"
#include "IsaacYdlidar.hpp"

...
```

特别注意: visibility 字段的设定, 如果想要在其他地方引用这个库, 就要要让对方可见. bazel 依赖库策略: 外部库文件或者非本地库文件的呼叫方式为: ”@libname”, 本地库文件的呼叫方式为: ”:libname”. 详见下面的: isaac/packages/ydlidar/BUILD

2.4 在 isaac/packages 中创建包

```
$cd isaac/packages
$mkdir ydlidar && cd ydlidar
$touch BUILD __init__.py IsaacYdlidar.cpp IsaacYdlidar.hpp
```

python 文件创建了就不用关心了, 编辑 BUILD 文件如下:

```
## load为常用函数,用于引入需要的指令.
## 第一个参数表示引入的源文件://engine/build:isaac.bzl
## 后面的参数列表, 表示从上述文件中引入那些指令/函数
## 也就是说, 在isaac/engine/build/isaac.bzl文件中, 定义了后面的这几个函数
## 轻轻的看一看isaac.bzl就知道是怎么回事了。
load("//engine/build:isaac.bzl", "isaac_cc_module", "isaac_component")

isaac_component(
    name = "isaac_ydlidar", # 给这个component取个好听的名字
    ## 注意: 这里我们没有显式指定源文件和头文件, bazel构建, 默认会包含
    ## IsaacYdlidar.cpp和IsaacYdlidar.hpp, 也就是在前面创建的两个文件。
    linkopts = [
        "-ldl",
        "-lgomp",
        "-lrt",
    ],
    deps = [
        "//engine/gems/geometry",
        "//engine/gems/system:cuda_context",
        "@ydlidar", ## 看到了吧, 我们在上面引入的第三方库, 现在可以用它了
        ## 由于这个库在当前包的外部, 所以由@呼叫。如果没有设置ydlidar库的
        ## visibility字段为public, 就比较尴尬了。很幸运的,我没有设置好.
    ],
)

isaac_cc_module(
    name = "ydlidar", ## 给这个module取个好听的名字, 在isaac_app中引用。
    deps = [":isaac_ydlidar"], ## 看到了吧, 这是内部呼叫方法: ":libname"
    ## isaac_ydlidar就是上面的component
)
```

2.4.1 编写 codelet

这部分比较清晰, 参考示例代码以及 Message API 和 Component API 就可以啦。

2.5 在 isaac/apps 目录的任一层级, 构建 application

2.6 在 isaac 下, 进行构建和执行

我的 ydlidar 库, 由 bazel 自动放置在: ~/.cache/bazel/__bazel__qiaoyx/8f79aac8f927e9008adbe64f87782c64/external/ydlid
目录下. 找到这个目录是有必要的.

bazel 在本地缓存目录为:~/.cache/bazel/_bazel_hostname/xxx/. 而所有需要 download 的第三方库, 都会放在上述目录下的 external 目录中. 不论是在 isaac/WORKSPACE,isaac/third_party/xxx.bzl 或者是 isaac 下的任一位置的 bzl 文件中指定一个有效的第三方库, 在执行

```
bazel build ...
```

or

```
## 注: _ydlidar是系统默认的命名规则, 在isaac.bzl中定义。
bazel build //packages/ydlidar:_ydlidar
```

时, 都会显示 download 进度和构建步骤. 最终编译生成的库名称为: libydlidar_module.so

Chapter 3

Map Represent and Obstacle Represent

3.1 Messages

3.1.1 BinaryMapProto

A binary occupancy grid map which indicates for every grid cell if the cell is free or occupied.

3.1.2 DistanceMapProto

A distance grid map which provides the distance to the nearest obstacle for every grid cell.

3.1.3 OccupancyMapProto

An occupancy grid map which provides mean and standard devia on for every grid cell. This represents the average number of mes a cell was observed as occupied.

3.2 Map

Definition: Provides the map of the environment to various other nodes.

3.3 LocalMap

Defination: Creates a representation of obstacles around the robot using the range scan measurements.

3.4 GlobalLocaliza on

Definition: Es mates the pose of the robot in a map only using the current range scan measurement without prior informa on.

3.5 GlobalPlanner

Defination: Computes a path to a target using the currently known map

3.6 MapEditor

支持下面三种图层类型

3.6.1 isaac.map.OccupancyGridMapLayer

A dense rectangular map which stores certain informa on per pixel.
A grid map layer for a map node. It provides access to an occupancy grid map which stores for each cell whether the cell is blocked or free. It also holds a distance map computed based on occupancy grid map which contains the distance to the nearest obstacle for each cell computed based on a given threshold.

3.6.2 isaac.map.PolygonMapLayer

A layer which stores mul ple polygons in the local layer coordinate frame.
A map layer which holds annotated polygons and provides various methods to access them 可用于设定虚拟路线和虚拟墙

3.6.3 isaac.map.WaypointMapLayer

A simple layer which stores annotatable points of interest.
A map layer which holds annotated waypoints and provides various methods to access them 用于设定兴趣点

3.7 与 Map 相关的功能

- Mapping
- Localization
- Obstacle Avoidance
- Planning

Chapter 4

Obstacle World

Obstacle: sphere [isaac::navigation::ObstacleWorld]

4.1 isaac.navigation.BinaryToDistanceMap

Converts a cost map into a binary map based on thresholds and computes a distance map from it. The resul ng distance map is added as an obstacle into an linked ObstacleWorld component.

- incoming: BinaryMapProto
- outgoing: DistanceMapProto

4.2 isaac.naviga on.OccupancyToBinaryMap

Converts an occupancy map into a binary map based on thresholds.

4.3 isaac.navigation.ObstacleWorld

A component which holds a virtual representa on of obstacles detected around the robot. Currently distance maps and spherical obstacles are available. This component is thread safe and can be accessed from other components without message passing.
Type: Component - This component does not ck and only provides certain helper func ons.

4.4 isaac.map.Map

This component is used to mark a node as a map and gives convenient access to the various map layers and also some cross-layer func onality.

4.5 isaac.navigation.LocalMap

Creates and maintains a dynamic obstacle grid map centered around the robot. The dynamic grid map is always rela ve to the robot with the robot at a fixed loca on in the upper part of the robot. The previous state of the grid map is con nuously propagated into the presence using the robot odometry. Good odometry is cri cal to maintaining a sharp, high-quality grid map. New flatscan measurements are integrated into the local map and mixed with the current local map accumulated based on the past. The local map “forgets” informa on over me to allow gradual dynamic updates. This enables it to be useful in the presence of dynamic obstacles. However thresholding might be challenging and addi onal object detec on and tracking should be used for dynamic obstacles.

- incoming: FlatscanProto
- outgoing: OccupancyMapProto

4.6 isaac.naviga on.MapWaypointAsGoal

Selects a waypoint from a map and publishes it as a goal

4.7 isaac.naviga on.MapWaypointAsGoalSimulator

Simulates moving to a desired map waypoint. The status of the movement will be published as variables.

4.8 isaac.naviga on.Naviga onMap

A map layer for a naviga on map. Holds various conveniences func ons for quick access of map data for naviga on tasks. This layer can work with a mul -floor map which holds mul ple layers for the same type for the different floors of a building. If mul -floor mode is enabled map layers are stored as prefix‘_n where ‘prefix is the base name of the layer and n is the floor index. The first floor has the index 0, the second floor the index 1, etc. If mul -floor mode is disabled by se ng num_floors to 0 only the prefix will be used to access the single layer of that type.
支持多楼层的导航地图。

4.9 isaac.naviga on.PoseHeatmapGenerator

4.10 isaac.map.MapBridge

A bridge for communica on between map container and WebsightServer

4.11 isaac.naviga on.PoseHeatmapGenerator

Divides given map into user-defined grid sizes. Reads the robot state (posi on, speed and the displacement since the last update) and determines which grid the robot was in at the me of acquisi on of the state.

- incoming: RobotStateProto Input robot state containing posi on, speed and the displacement since the last update
- outgoing: HeatmapProto Output HeatmapProto containing heatmap of probabili es, grid cell size and map frame

4.12 isaac.naviga on.TravellingSalesman

Es mates a set of waypoints over the reachable loca ons of given map and computes the shortest path through them. Returns a set of points ordered by the shortest path through them.

- incoming: none
- outgoing: waypoints [Plan2Proto] Output plan, which is a list of poses that the robot can move to

Chapter 5

json configure file

configuration through json file. how to define nodes and edges?