

ROS 作业：

1 准备工作

1.1 登录 ompl (The Open Motion Planning Library) 官网：

<https://ompl.kavrakilab.org/index.html>

1.2 进入 Download 页面，下载保存脚本文件 install-ompl-ubuntu.sh

Download

Releases

- **OMPL.app 1.4.2**, released Jul 29, 2019 ([release notes](#)):
[TGZ](#) [ZIP](#)
- **OMPL 1.4.2**, released Jul 29, 2019 ([release notes](#)):
Just the **core OMPL library** (no GUI, no bindings to **FCL**, **PQP**, and **Assimp**):
[TGZ](#) [ZIP](#)

OMPL is also available through through several package managers:

- [Debian](#),
- [Ubuntu \(14.04 and higher\)](#),
- [Fedora](#),
- [MacPorts](#), and
- [Homebrew](#).

Note that these package managers may not always have the latest release.

- **Installation script for Ubuntu 14.04, 15.10, 16.04, and 17.10**
- [Installation instructions](#).
- **Older releases**. See the [release notes](#) for a brief a description of changes for each release.

1.3 运行脚本文件

在脚本文件保存的路径下，右键打开终端，运行命令

1. `sudo chmod +x install-ompl-ubuntu.sh`
2. `./install-ompl-ubuntu.sh`

1.4 创建工作空间，编译作业里的功能包（首次编译无法通过，需要完善代码才能编译通过）

2 ROS 查找依赖包 ompl

2.1 修改 src/grid_path_searcher/CMakeLists.txt, 使用 `find_package()` 查找 ompl 的头文件、库路径等信息

1. `find_package(Eigen3 REQUIRED)`
2. `find_package(PCL REQUIRED)`

```
3. # add your code here: find_package(xxx REQUIRED)
```

2.2 在代码中添加使用到的 ompl 的头文件（该部分代码中已经添加）

见文件 src/grid_path_search/src/demo_node.cpp

```
1. #include <ompl/config.h>
2. #include <ompl/base/StateSpace.h>
3. #include <ompl/base/Path.h>
4. #include <ompl/base/spaces/RealVectorBounds.h>
5. #include <ompl/base/spaces/RealVectorStateSpace.h>
6. #include <ompl/base/StateValidityChecker.h>
7. #include <ompl/base/OptimizationObjective.h>
8. #include <ompl/base/objectives/PathLengthOptimizationObjective.h>
9. #include <ompl/geometric/planners/rrt/RRTstar.h>
10. #include <ompl/geometric/SimpleSetup.h>
```

3 学习调用ompl 实现 RRT*

要学会调用ompl 实现 RRT*，需要实现的功能如下：

- 把用户定义的起点、终点、地图用ompl 库定义的数据结构表示
- 了解 ompl 调用RRT*的方法和步骤
- 把 ompl 库求解得到的路径转换为用户定义的数据结构

本次作业需要添加的代码集中在文件 src/grid_path_searcher.cpp/src/demo_node.cpp 中的一个函数 `void pathFinding(const Vector3d start_pt, const Vector3d target_pt)` 和一个类 `class ValidityChecker : public ob::StateValidityChecker`。其中，pathFinding()交代了完整的代码流程，需要重点关注。

需要添加的代码在文件中以注释的形式标出，共有 7 处。

e.g.

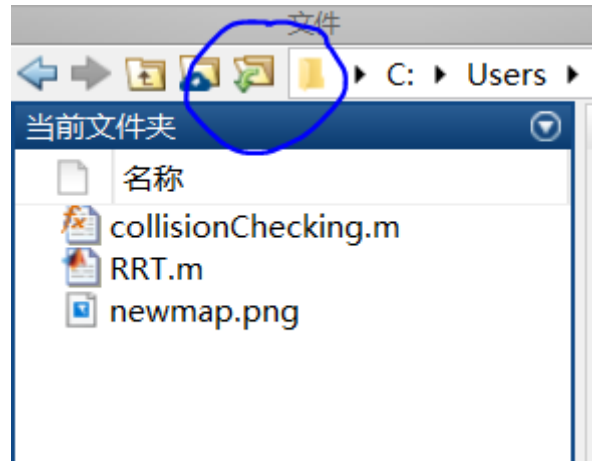
```
1. class ValidityChecker : public ob::StateValidityChecker
2. {
```

```

3.     public:
4.         ValidityChecker(const ob::SpaceInformationPtr& si) :
5.             ob::StateValidityChecker(si) {}
6.         // Returns whether the given state's position overlaps the circular
           obstacle
7.         bool isValid(const ob::State* state) const
8.         {
9.             // We know we're working with a RealVectorStateSpace in this
10.            // example, so we downcast state into the specific type.
11.            const ob::RealVectorStateSpace::StateType* state3D =
12.                state->as<ob::RealVectorStateSpace::StateType>();
13.            /**
14.             *
15.             *
16.             STEP 1: Extract the robot's (x,y,z) position from its state
17.             *
18.             *
19.             */
20.            return _RRTstar_preparatory->isObsFree(x, y, z);
21.        }
22.    };

```

Matlab 作业



打开工作文件夹，按照 STEP 提示完成 RRT.m