RBE550: Motion Planning

Project 0: Getting Familiar with OMPL

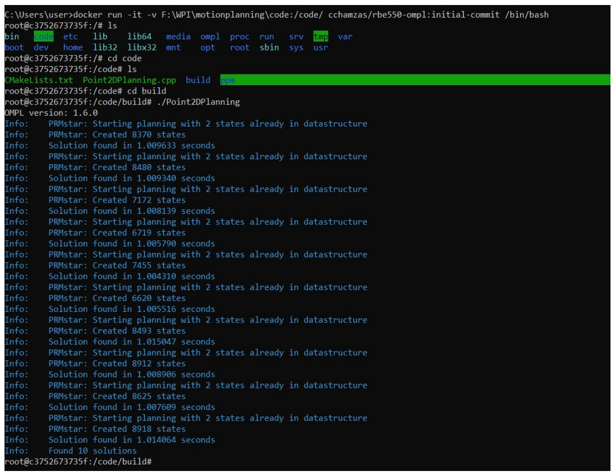
Student: Dhruv, Agrawal

Project Goals: Getting familiar and using the Open Motion Planning Library (OMPL)

1 Project Exercises

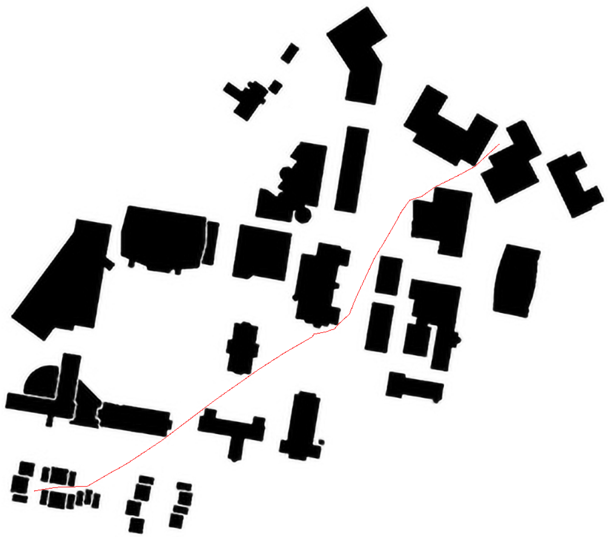
- **What method did you use to install OMPL? How long did it take you approximately?**
  The first method has been used to install OMPL.
  On Windows I first installed WSL2 and then I installed Docker for Desktop.
  It took around an hour to search, understand and implement the steps required for installing Docker.
  I was able to understand, learn and apply the basic Docker commands for pulling the image from Docker Hub and running a container in around 30 minutes.

- **Demonstrate that you have successfully installed OMPL. In your report, include the terminal output after running the Point2DPlanning.cpp demo, and also include the result_demo.ppm image that was created.**
  The installation of OMPL has been done successfully.
  This is evident from the below mentioned pictures of the terminal output and path image after running the demo Point2DPlanning.cpp.



```
C:\Users\user>docker run -it -v F:\WPI\motionplanning\code:/code/ cchamzas/rbe550-ompl:initial-commit /bin/bash
root@c3752673735f:/# ls
bin   code  etc   lib    lib64   media  ompl  proc  run   srv  tmp  var
boot  dev   home  lib32  libx32  mnt    opt   root  sbin  sys  usr
root@c3752673735f:/# cd code
root@c3752673735f:/code# ls
CMakeLists.txt  Point2DPlanning.cpp  build  ppm
root@c3752673735f:/code# cd build
root@c3752673735f:/code/build# ./Point2DPlanning
OMPL version: 1.6.0
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 8370 states
Info:    Solution found in 1.009633 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 8480 states
Info:    Solution found in 1.009340 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 7172 states
Info:    Solution found in 1.008139 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 6719 states
Info:    Solution found in 1.005790 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 7455 states
Info:    Solution found in 1.004310 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 6620 states
Info:    Solution found in 1.005516 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 8493 states
Info:    Solution found in 1.015047 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 8912 states
Info:    Solution found in 1.008906 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 8625 states
Info:    Solution found in 1.007609 seconds
Info:    PRMstar: Starting planning with 2 states already in datastructure
Info:    PRMstar: Created 8918 states
Info:    Solution found in 1.014064 seconds
Info:    Found 10 solutions
root@c3752673735f:/code/build#
```

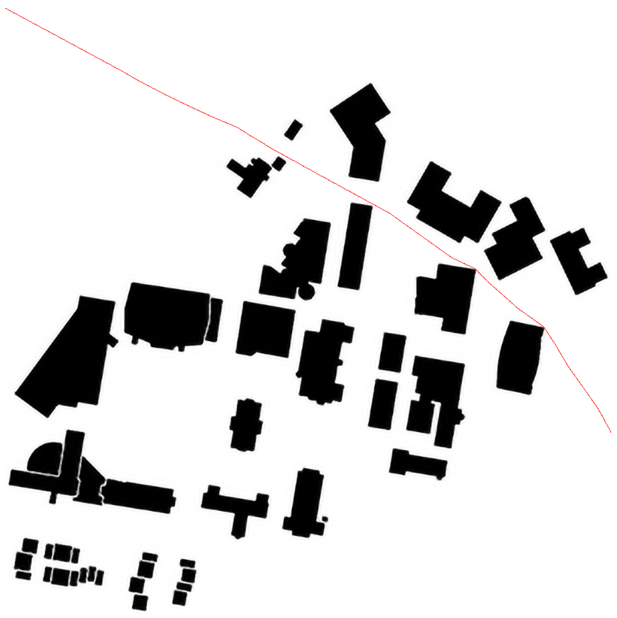
(a) Terminal Output



(b) Path Image

Figure 1: Results obtained after running the Point2DPlanning.cpp script

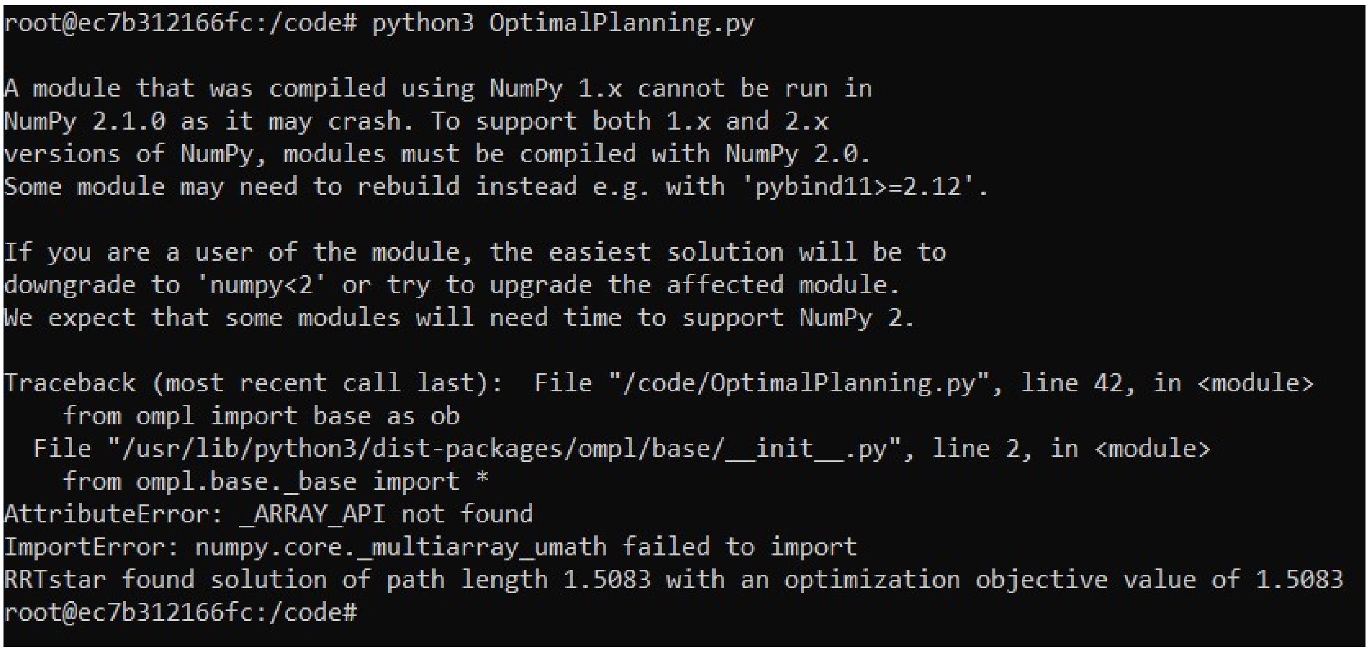
- **Understand and modify the demo. Skim the source code of Point2DPlanning.cpp, change the start and goal locations of the problem, and include the new .ppm image in your report. Try to make it sufficiently different from the previous one**.
  The start and goal configurations in the original script were (54,924) and (821,351) respectively.
  The start and goal configuration for this problem have been changed to (0,0) and (1000,700) respectively.
  The resulting path for the new configurations has been shown below.



(a) Path Image

Figure 2: Result obtained after running the modified Point2DPlanning.cpp script

- **Find one C++ demo and one Python demo from here, and run them. For the C++ demo, you will need to modify the CMakeLists.txt file accordingly. In your report, include the name of the demo you ran and the terminal output. It is possible that some of the more complicated demos do not run, so try to find one that does.**
  The C++ demo chosen is **Diagonal.cpp**.
  The CMakeLists.txt file has been modified by replacing the **Point2DPlanning.cpp** filename with **Diagonal.cpp** filename.
  The original commands have been commented.
  The Python demo chosen is **OptimalPlanning.py**.
  The terminal outputs for the above C++ and Python files is shown below.



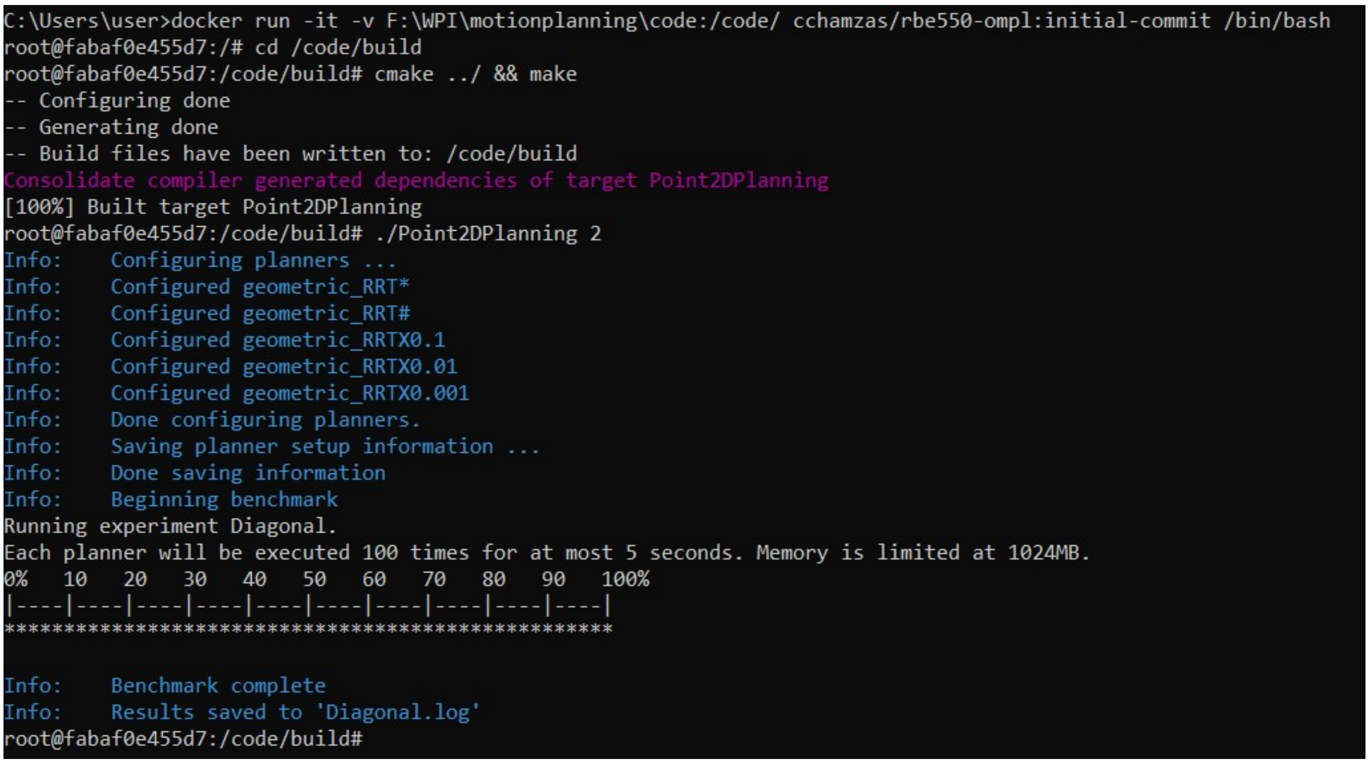
```
root@ec7b312166fc:/code# python3 OptimalPlanning.py

A module that was compiled using NumPy 1.x cannot be run in
NumPy 2.1.0 as it may crash. To support both 1.x and 2.x
versions of NumPy, modules must be compiled with NumPy 2.0.
Some module may need to rebuild instead e.g. with 'pybind11>=2.12'.

If you are a user of the module, the easiest solution will be to
downgrade to 'numpy<2' or try to upgrade the affected module.
We expect that some modules will need time to support NumPy 2.

Traceback (most recent call last):  File "/code/OptimalPlanning.py", line 42, in <module>
    from ompl import base as ob
  File "/usr/lib/python3/dist-packages/ompl/base/__init__.py", line 2, in <module>
    from ompl.base._base import *
AttributeError: _ARRAY_API not found
ImportError: numpy.core._multiarray_umath failed to import
RRTstar found solution of path length 1.5083 with an optimization objective value of 1.5083
root@ec7b312166fc:/code#
```

Figure 3: Optimal Planning

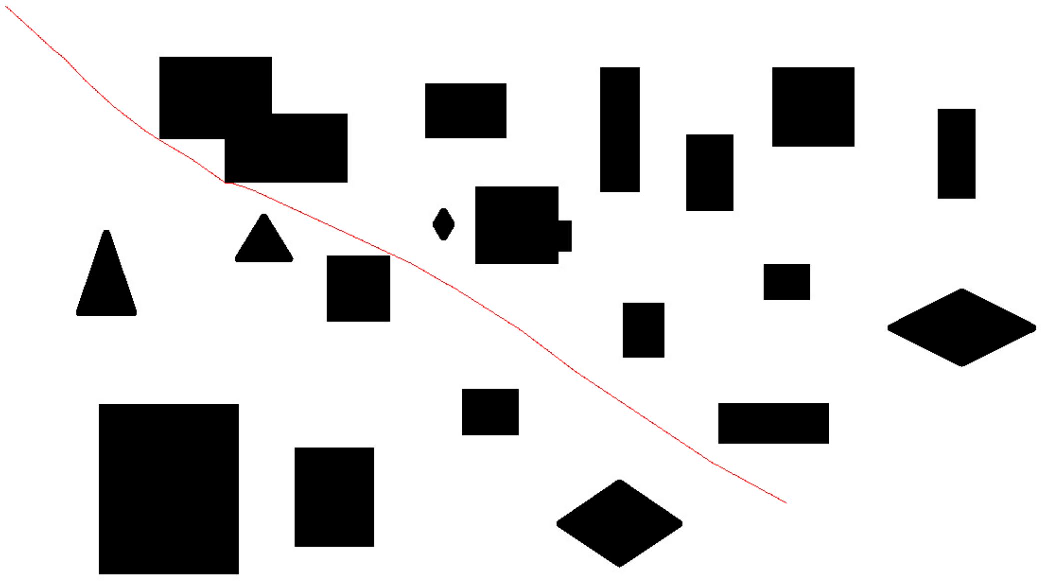


```
C:\Users\user>docker run -it -v F:\WPI\motionplanning\code:/code/ cchamzas/rbe550-ompl:initial-commit /bin/bash
root@fabaf0e455d7:/# cd /code/build
root@fabaf0e455d7:/code/build# cmake ../ && make
-- Configuring done
-- Generating done
-- Build files have been written to: /code/build
Consolidate compiler generated dependencies of target Point2DPlanning
[100%] Built target Point2DPlanning
root@fabaf0e455d7:/code/build# ./Point2DPlanning 2
Info:    Configuring planners ...
Info:    Configured geometric_RRT*
Info:    Configured geometric_RRT#
Info:    Configured geometric_RRTX0.1
Info:    Configured geometric_RRTX0.01
Info:    Configured geometric_RRTX0.001
Info:    Done configuring planners.
Info:    Saving planner setup information ...
Info:    Done saving information
Info:    Beginning benchmark
Running experiment Diagonal.
Each planner will be executed 100 times for at most 5 seconds. Memory is limited at 1024MB.
0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
***************************************************

Info:    Benchmark complete
Info:    Results saved to 'Diagonal.log'
root@fabaf0e455d7:/code/build#
```

Figure 4: Optimal Planning

- **Create your own .ppm environment and solve a motion planning problem there. You can use maps from the web or draw your own images. Include the path and the created map in your report.**
  The custom map was initially drawn in **Paint** and saved as a jpg file.
  The **ImageMagick** tool was then used to convert the custom map jpg file to a ppm file.
  A new script named **Point2DPlanning_custom_path.cpp** was created where the path to the new custom ppm file was provided.
  Since this ppm image size is different than the **result_demo** image, the goal configurations were changed to be inside image bounds.
  The output path was obtained after compiling and running the script and is shown below.



(a) Custom Path Image

Figure 5: Result obtained after running the Point2DPlanning_custom_path.cpp script