ME 132a - Implement a planner

Released: Wednesday 02/08/2017 Due: Friday 02/17/2017

This guide describes how to create a new planner in the OMPL framework named the 'Random Tree' method, which is a simplified version of RRT. This will require adapting and compiling code from the OMPL codebase on the Ubuntu 14.04 virtual machine installed in the last lab. You will then benchmark your new planner against several other planners to see comparisons of performance, as well as visualize the paths that it produced.

The Random Tree Method

1. Select a random configuration *a* from the existing Random Tree
2. Sample a random configuration *b* in the configuration space. With a small probability select the goal configuration instead of a random one
3. Check whether the straight-line path between *a* and *b* in the C-space is valid (i.e., collision free). If the path is valid, add the path to the Random Tree
4. Repeat steps 1-3 until the goal state is added to the tree

Creating new planner

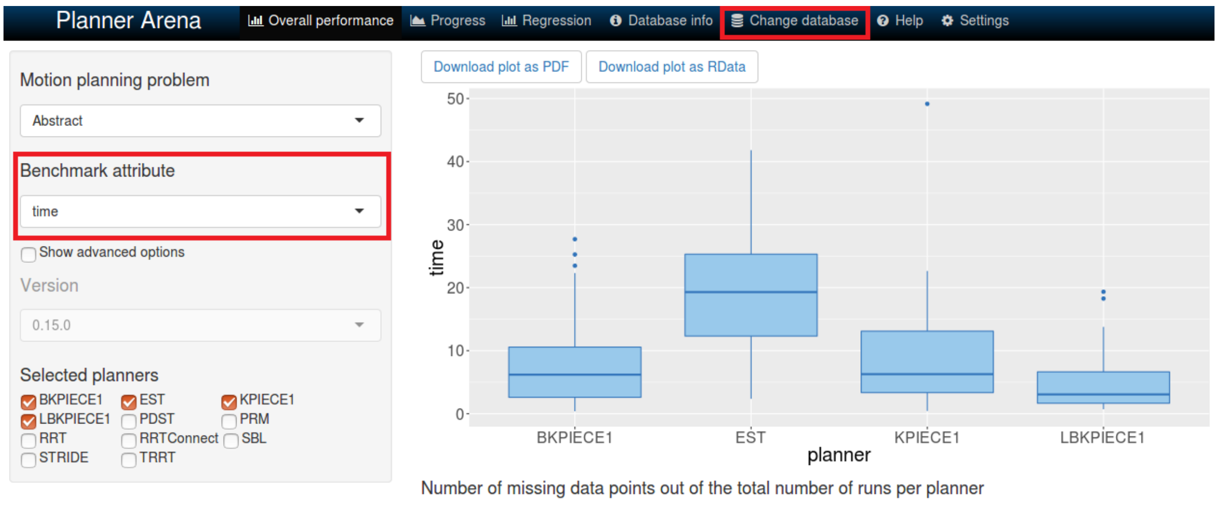
Figure out how to visualize the results of the tests

1. Log into the VM and download the 'me132_lab2.tar.gz' file from the class wiki. Extract all of the files somewhere easy to access such as the Desktop. Open a terminal window and navigate to the folder using the command 'cd' (Google how to if necessary)
2. Run the command 'chmod +x initialInstall.sh' to give this script permission to run, then use the command './initialInstall.sh'. This will place all of the files for your new planner in the correct locations and update the build lists so you can compile the program
3. Open the file 'RT.cpp' at location *omplapp-1.2.1-source/ompl/src/ompl/geometric/planners/rrt/src* and fill in the code to implement the basic Random Tree planner. You can refer to the 'RRT.cpp' file for an example of the classes to use and how to format your planner
4. Once you think you have something working in the planner, or in order to check if your code syntax is correct, you can compile the program using the second script you downloaded named 'buildProgram.sh'. Do this in the same manner as with 'initialInstall.sh'
5. After both your planner and the benchmark program compile successfully you can run the benchmarking program by opening a terminal, navigating to *~/Documents/OMPL/omplapp-1.2.1-source* and running the command './build/Release/bin/demo_me132benchmark'
   Note: this may take several minutes as we are running many iterations of each solver which can be slow (particularly PRM as you will see in the results)

6. Before checking the performance of each of the planners we can see some of the paths they generated. Open the OMPL gui with the terminal command 'ompl_app', select *File -> Open Problem Configuration* then select 'resources/2D/BugTrap_planar.cfg'. Next click *File -> Open Path* then select any of the 'planner_path.txt' files in the ompl base directory to see the path it produced

7. The benchmark program produces a log file (in the omplapp-1.2.1-Source folder) which we can process to get performance metrics on the different planners. To do this run the command: 'python ompl/scripts/ompl_benchmark_statistics.py logfile.log -d mydatabase.db' replacing 'logfile' with the name of the file corresponding to the test you just ran

8. Lastly, upload the mydatabase.db file we just made to the website plannerarena.org under the *Change Database* tab. You can now use this interface to explore all of the performance metrics for each of the different planners in the maze scenario we ran by selecting them from the dropdown box 'Benchmark attribute'



Motion planning problem: Abstract

Benchmark attribute: time

Show advanced options

Version: 0.15.0

Selected planners

✓ BKPIECE1	✓ EST	✓ KPIECE1
✓ LBKPIECE1	PDST	PRM
RRT	RRTConnect	SBL
STRIDE	TRRT	

Number of missing data points out of the total number of runs per planner

Homework Deliverables

Write two to three paragraphs comparing the performance metrics of the different planners that you were able to see on the plannerarena.org website. Make particular note of (computation) time, solution length, and the number of states sampled (graph states), and include a screenshot of the boxplot for time.

Attach this to your homework along with a printed copy of the code you added to the 'RT.cpp' file.