CS 593 Robotics Assignment 3
Learning-based Motion Planning

February 24, 2023

Familiarize yourself with the methods described in [1-2]. We will be considering simple 2D and complex 3D environments for following problems. In the given datasets the demonstration paths were generated by RRT* [3]. For more details on the dataset and their corresponding network architectures, refer to [1-2].

Recommended Package versions

- Python == 3.8.12
- pytorch == 1.10.2
- numpy == 1.21.2

1 Question 1: Simple 2D [3 pts]

In this setting, we provide the dataset in [4], as well as pre-trained neural models and code. Please familiarize yourself with the implementations and their various functions such as Lazy Vertex Contraction (LVC) and relate them with the theory presented in [1-2]. By using the given material, get the following:

- Run the test code and report i) the mean and standard deviation of computation time per planning problem; and ii) success rates in the Table 1. Also report your system specifications RAM/GPU/ CPU (e.g., in [1], we use s 3.40GHz× 8 Intel Core i7 processor with 32 GB RAM and GeForce GTX 1080 GPU). [0.5+0.5+0.5 points]
- Comparison of expert demonstrations against MPNet paths: Visualize and compare the Euclidean lengths (cost) of stored RRT* paths (from dataset) and MPNet's generated paths. Select three different environments from test dataset and for each selected

	NMP	**NMP-w/o-Dropout**	**NMP-w/o-LVC**
success rate	-	-	-
computation time	-	-	-

Table 1: Test results on 2D environment

environment display a stored path and MPNet's path in a randomly selected planning problem, i.e. a start-goal pair. Note that you do not need to run RRT* planner. An example is shown in Fig 1: [0.5 points]

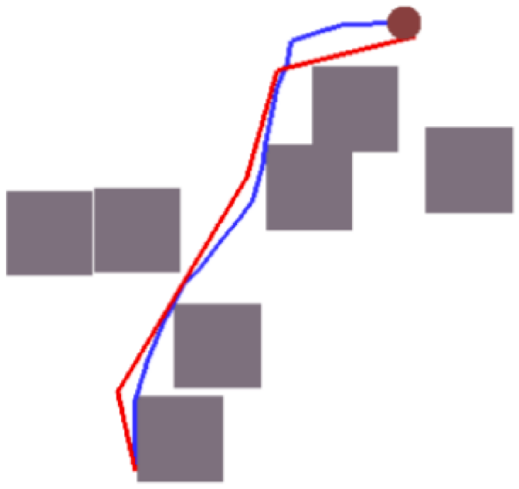

Figure 1: MPNet(Red): cost=34.3 ; RRT*(Blue): cost 34.1

- Since MPNet is stochastic, it can generate multiple paths for a fixed start and goal pair. In this part, display five runs of MPNet for a randomly selected planning problem from the test dataset (Example: Fig. 1). [0.5 pts]

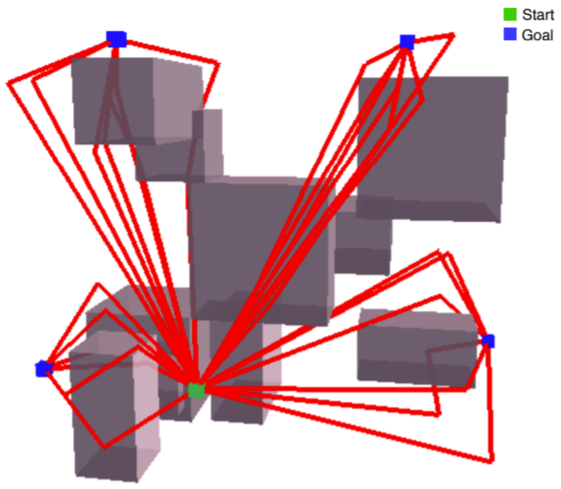

Figure 2: Multiple paths for fixed start and goal pairs

- Describe the significance of Dropout and lazy vertex contraction and their role in MPNet (max 100 words). [0.5 pts]

	NMP	**NMP-w/o-Dropout**	**NMP-w/o-LVC**
success rate	-	-	-
computation time	-	-	-

Table 2: Test results on 3D environment

2 Question 2: Complex 3D [7 Pts]

In this setting, we provide the training and testing datasets in [5], and a procedure to load them. Please build on the code provided for simple 2D environment and do the following:

- Train and tune end-to-end MPNet models on complex 3D dataset, and try to achieve success rates $> 90\%$. This part requires adding code/collision checker that can handle 3D cases. [3 pts]
- By running the test code, please report i) the mean and standard deviation of computation time per planning problem; and ii) success rates in the following Table 2. Also report your system specifications RAM/GPU/ CPU (e.g., in [1], we use s 3.40GHz$\times$ 8 Intel Core i7 processor with 32 GB RAM and GeForce GTX 1080 GPU). [1+1+1 pts]
- Comparison of expert demonstrations against MPNet paths: Visualize and compare the Euclidean lengths (cost) of stored RRT* paths (from dataset) and MPNet's generated paths. Select three different environments from test dataset and for each selected environment display a stored path and MPNet's path in a randomly picked planning problem, i.e. a start-goal pair. Note that you do not need to run RRT* planner. An example is shown in Fig 2: [0.5 points]
- Since MPNet is stochastic, it can generate multiple paths for a fixed start and goal pair. In this part, display five runs of MPNet for a randomly selected planning problem from a test dataset (Example: Fig. 2). [0.5 pts].

3 Additional Guidelines for 2D Environment

Download simple 2D dataset [4]. The simple 2D has 7 blocks each of size 5×5 that are placed randomly. In the dataset:

- e0-100 has the training and testing paths in 100 different environments. 0-100 environments and 0-4000 paths/environment are for training. Test dataset: 0-100 envs and 4001-4100=100 paths/env.
- obs_cloud is the point-cloud of randomly generated 30,000 environments. 0-100 corresponds to the same environments for which path data is provided.
- obs.dat contains the center location (x,y) of each obstacle in the environments.

4 Additional Guidelines for 3D environment

Download complex 3D dataset [5], which is organized as follows:

- Complex 3D contains 10 blocks with sizes as follow with shapes [5,5,10], [5,10,5], [5,10,10], [10,5,5], [10,5,10], [10,10,5], [10,10,10], [5,5,5], [10,10,10], and [5,5,5].
- e0-10 has the training and testing paths in 10 different environments. 0-10 environments and 0-2000 paths/environment are for training. Test dataset: 0-10 envs and 2001-2100=100 paths/env.
- obs_cloud is the point-cloud of randomly generated 30,000 environments. 0-10 corresponds to the same environments for which path data is provided.
- obs.dat contains the center location (x,y,z) of each obstacle in the environments.
- obs_perm2.dat contains the order in which the blocks should be placed in preset locations given by obs.dat file to setup environments. For instance, in complex 3D, the permutation 8342567910 indicates obstacle 0 of size 5x5x10 should be placed at the location 8 given by obs.dat.

5 Computational Resources: Access to Scholar

There are several different ways students can access Scholar. First, students can access Scholar via ssh – 'ssh scholar.rcac.purdue.edu'. (Note that Scholar and all RCAC systems require BoilerKey.). However, if students require GUIs. Purdue's resource team suggests using OpenOnDemand (https://gateway.scholar.rcac.purdue.edu). This not only allows students to easily run interactive Jupyter lab sessions, but it also helps guide them in submitting batch jobs. Another GUI option is ThinLinc, which provides a full virtual desktop (https://desktop.scholar.rcac.purdue.edu).

6 Submission Instructions

Your submission should consist of the following items:

- Python code. Submit all code files given to you, even the ones you didn't modify.
- A model (pkl file) that you trained in Question 2. *Please only include one trained model*, to limit the size of your submission. Do **not** include any of your results files, and do **not** submit all or part of the datasets provided to you.
- A report describing each of your steps. Be as detailed as possible, and include all the data and visualizations required by questions 1 and 2.
- Instructions for running your code. You may place these in your report itself, or in a separate README file. Be specific enough that we can reproduce your results, if needed.

Bundle all files into a zip or tarfile, and submit it via Brightspace. Name your submission “<your_username>.zip” or “<your_username>.tar.gz”. For example, I would name my submission “thonegge.zip”.

References

1. Qureshi, Ahmed H., et al. Motion planning networks. 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.

2. Qureshi, Ahmed H., et al. Motion Planning Networks: Bridging the Gap Between Learning-based and Classical Motion Planners.IEEE Transactions on Robotics (2020).

3. RRT*: Sampling-based Algorithms for Optimal Motion Planning.

4. Simple 2D dataset

5. Complex 3D dataset