Project Baseline: Learning Based RRT using a Visual Classifier as a Cost Function

Aaron Kampmeier akampmei@asu.edu
Brandon Evans bpevans@asu.edu
Braeden Woodard bmwoodar@asu.edu
Pavan Kumar Raja praja3@asu.edu

Abstract

In robotic navigation and exploration, there is a need to investigate efficent methods to handle continous state and control spaces. Many contributions have proposed variations of RRT algorithms that can more efficiently handle these enviroments over an Markov Decision Process (MDP). In this paper, we propose a method that expands on a proposed RRT variant that incorporates behvior cloning (BC) and utilizes a visual classifier to learn the cost function of the optimal trajectories.

1 Introduction

Rapidly-Exploring Random Trees (RRT) [**?**] have proven to be a very useful sample-based path-planning algorithm to efficiently search high-dimensional spaces. However, RRT can be hindered in environments with obstacles or specific pathways to a goal. The algorithm may waste time exploring down irrelevant or blocked paths. In this paper, we will survey various learning approaches to problems encountered by RRT in such environments, evaluate a baseline modified RRT algorithm based on behavior cloning, present our experiment results and discuss our future work.

2 Problem Statement

A basic RRT algorithm's performance can suffer in configuration spaces with obstacles by exploring down irrelevant or blocked paths. The surveyed learning improvements to RRT may help but they still need a way to discover where the goal states and relevant obstacles are in relation to the agent. This problem can arise a variety of different domains such as expansions through mazes, robotic control in constrained spaces [**?**], and systems testing of multi-step processes [**?**].

3 Related Work

In the last few years, several contributions to robotic navigation and exploration have been proposed that incorporate variations of sampling-based costmap planners, such as RRT or RRT*, and learning from demonstrations. Zuo et. al. [**?**] proposes a method, Clone Assisted RRT (CA-RRT), that learns heuristics for a search-based algorithm from a limited number of human demonstrations. CA-RRT is an expansion of the human-seeded RRT (HS-RRT) proposed by Chang et. al. [**?**], which uses states from human demonstrations as seed configurations states within an RRT. This expansion includes adding a first-person image-state-based (behavioral) cloning approach over the HS-RRT. Zuo et. al. [**?**] method consists of collecting a set of human-created trajectories and training a behavioral cloning policy using the first-person state-action pairs from those trajectories to seed the RRT. These heuristics allow the search algorithms to automatically test other states in the game thus removing the necessity to test all possible scenarios in the game through human game play.

Perez et. al. [**?**] also proposes an approach to learn navigation behaviors from demonstrations with the use of sampling-based costmap planners, though they propose the use Inverse Reinforcement

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Learning (IRL) concepts to identify the RRT cost function that best fits the example trajectories. The proposed method, Rapidly-exploring random Trees Inverse Reinforcement Learning (RTIRL), makes use of the RRT* instead of a Markov Decision Process (MDP). This is due to the advantages that RRT* has over MDP, in which RRT* can handle continuous state and control spaces. Perez et. al. [**?**] RTIRL method is used to extract the proper weights of the cost function from demonstration trajectories, which can then be used later in the RRT* process to allow the robot to reproduce the desired behavior at various scenarios.

Both papers propose learning algorithms that use sampling-based costmap planners for path planning and exploration. Our proposed method will also be expanding upon the RRT algorithm and involve collecting and utilizing human demonstrations to allow for learning from demonstrations. Our proposed algorithm will incorporate a similar behavior cloning (BC) approach to that of Zuo et. al. [**?**] as the BC policy will be act as a seed in the initial RRT configuration. In addition to BC, we aim to use a visual classifier to learn a cost function, by using an image to gather samples from various locations to determine the cost to reach the goal state. This approach is similar to Perez et. al. [**?**] proposed method, which adjusts and learns a cost function of the RRT* planner.

4 Problem Domain

Currently, the problem domain consists primarily of 2D pygame environment which contains static obstacles that create a maze that the agent must navigate. As of the baseline presentation, the static 2D environment consists of 9 static obstacles, a static start point and a static goal point. The start is in the upper left corner of the 2D environment and the current static goal is on the far right of the screen on the other side of all of the obstacles.

For our problem the agent is creating a path to navigate from start to goal. In order to do this, our agent uses an RRT algorithm.

Due to the static nature of the 2D environment as it stands now, our agent is being tested in a discrete space as the size and number of possible x,y coordinates the agent can be in is static at 500x800 pixels.

5 Learning Baseline

To create a baseline for our CARRT, we require a demonstration. As we planned to use static obstacle space, we ignored the location of obstacles in creating the demo. We used pygame to set up our environment and demonstrations were collected by us playing the game using WASD keys. Each demonstration is a trajectory with (state, action) pairs. Each state is represented as (the current position of the agent, goal position) and action is represented as ENUM of WASD keys. We used a simple MLP model as a prediction/action generator, a summary of the model is given in Table 1. For the next phase of the project, we'll keep this model as a baseline and expand it to accommodate dynamic world state space.

6 Experiment Metrics

Our baseline performance was measured using 3 metrics:

- Success Rate: The percentage of times out of n samples that the agent successfully reaches the goal.
- Time: The time (s) taken from the start of RRT to termination due to either reaching the iteration limit or reaching the goal.
- Number of Expansions: The number of nodes in the RRT graph at termination.

7 Experiment Results

As our final goal is to create a RRT algorithm that uses a visual classifier as a cost function for our baseline, we needed a learning RRT baseline rather than just standard RRT for the baseline.

Table 1: Behavior cloning MLP model summary

Layer (Type)	Output Shape	Parameter count
Linear-1	[-1, 8]	40
ReLU-2	[-1, 8]	0
BatchNorm1d-3	[-1, 8]	16
Linear-4	[-1, 8]	72
ReLU-5	[-1, 8]	0
BatchNorm1d-6	[-1, 8]	16
Linear-7	[-1, 4]	36
ReLU-8	[-1, 4]	0
BatchNorm1d-9	[-1, 4]	8
Linear-10	[-1, 4]	20
ReLU-11	[-1, 4]	0
BatchNorm1d-12	[-1, 4]	8
Total params	216	
Trainable params	216	
Non-trainable params	0	

Because of this, metrics were taken for both our baseline algorithm (our version of CA-RRT) and a normal RRT algorithm. In order to ensure our baseline was functioning correctly, we must see an improvement of our BCRRT over the base RRT. We can see the metric results for RRT in Table 2 as well as the BCRRT in Table 3, which show that as expected our baseline BCRRT algorithm performs better overall than the base RRT algorithm. In success rate, mean time, and mean expansion number, BCRRTperforms better over the different n sample tests than RRT with nothing added on. These results confirm our baseline is functioning well as we are seeing expected performance increases over RRT.

Table 2: Metrics from RRT Algorithm

Sample Size	Success Rate	Mean Time(s)	Std. Dev Time	Mean Expansions	Std. Dev Expansions
70	.79	3.6179	1.2385	745.3562	211.5847
100	.81	3.5861	1.1871	752.03	202.2828
120	.79	2.9090	1.0498	751.5603	209.2586
155	.77	3.4672	1.2880	734.6838	213.4561
185	.79	3.3586	1.2298	738.7589	205.6531

Table 3: Metrics from BCRRT Algorithm

Sample Size	Success Rate	Mean Time(s)	Std. Dev Time	Mean Expansions	Std. Dev Expansions
70	.89	1.9586	1.8819	440.6875	378.6503
100	.91	2.0340	1.7459	442.09	372.4981
120	.92	1.6560	1.4974	439.7563	375.8521
155	.90	2.0366	1.8273	428.4387	376.2023
185	.89	2.0287	1.6985	431.4369	373.4285

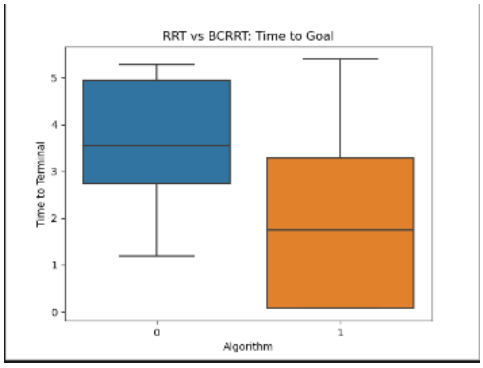

Figure 1: Time taken to reach Goal

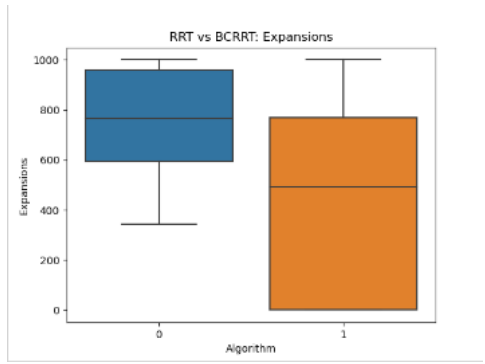

Figure 2: Number of nodes expanded.

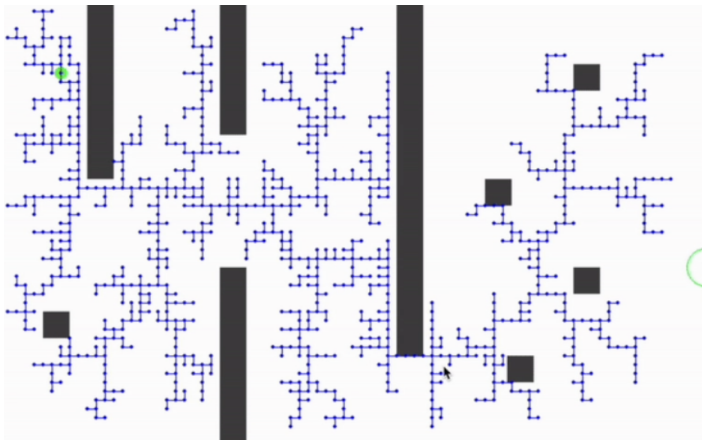

Figure 3: Expansion based solely on RRT.

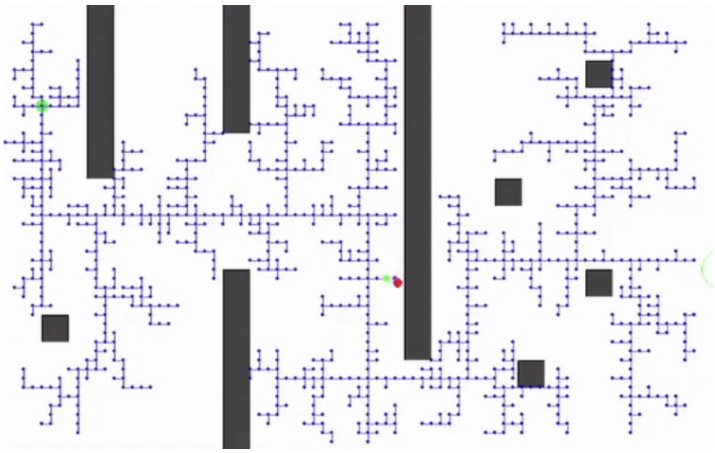

Figure 4: Expansion after behavior cloned roll-out.

8 Future Work

We plan to build on our baseline and accomplish these listed tasks.

- Creating and training a visual classifier to learn a cost function in the RRT.
  - Collecting demonstration for variable goal and obstacle environment.
  - Training a visual classifier to recognize a "goal region" in a 2D image.
  - Use the visual classifier as a cost function to aid in guiding the agent's expansions towards the goal faster.
- Perform testing on the Visual-RRT against the CA-RRT behavior cloning baseline.
- Prepare data visualizations of the data collected in the comparison testing.
- Perform student's t-test on the two algorithm results.