Deep Reinforcement Learning for Autonomous Search

2021-03-06

Andrew T. Herdering, Hugo F. Quintero, Sara E. Centeno, Mishell L. Beylik, Jason T. Isaacs

California State University Channel Islands

Deep Reinforcement Learning for Autonomous Search

Andrew T. Herdering, Hugo F. Quintero, Sara E. Centeno, Mishell L. Beylik, Jason T. Isaacs

California State University Channel Islands, Camarillo, USA

Abstract

We propose a technique for autonomous exploration and search of unknown environments with applications to urban search and rescue. By utilizing a deep reinforcement learning approach we maximize the search coverage while avoiding unknown obstacles. Allowing autonomous platforms to rapidly search and provide feedback to rescue teams has the potential to aid both the victim and rescue personnel. Through a simulation study, we find that this approach searches significantly more area over time than a conventional map-based frontier exploration.

1. Introduction

Mobile robotics has become a major point of interest, particularly in search and rescue applications. Robotic agents can enter areas inaccessible or too hazardous for humans and gather data on an environment being explored. They can also find specific objects, points of inspection, and potential survivors. For such operations to be effective, the robot agents must be able to navigate autonomously, therefore we propose to use reinforcement learning for this application. The main problem we address is the utilization of Deep Reinforcement Learning (DRL) to autonomously navigate real-world environments, specifically subterranean (e.g. mine tunnels, natural caves, or urban underground structures).

The Defense Advanced Research Projects Agency (DARPA) Subterranean (SubT) Challenge is a multi-stage robotics competition designed to stimulate research and innovations in the areas of mapping, navigation, and search in the complex underground environments described above. The objective of the competition is to explore an unknown environment in search of artifacts representing survivors and other items associated with human presence. Points are rewarded for correctly identifying artifacts and accurately reporting their position [1].

Tai et al. establish a method of Reinforcement Learning (RL) for mapless navigation in complex environments by utilizing asynchronous deep deterministic policy gradients (ADDPG) to plan the motion of mobile robots [2]. This method is intended to be a low-cost tool for indoor robots in smaller environments. Larger environments would require a map of the environment intended to be explored. Babaeizadeh et al. make improvements to the Asynchronous Advantage Actor Critic (A3C) used in RL by creating a hybrid architecture placing more emphasis on the Graphics Processing Unit (GPU) for computations, freeing room for the Central Processing Unit (CPU) to run other tasks during training [3]. The final product, GA3C, makes faster progress in learning as opposed to the original A3C. The authors used GA3C to do RL training for video games instead of real-world environments.

Zhang et al. propose a method that utilizes external memory to allow the Neural Network (NN) layer to compute the pose and map simultaneously with continuous control exploration commands, allowing it to map, localize, and make navigation decisions in a single unit [4]. This provides accurate frontier exploration within the map which the external memory has created. Tai et al. [5] provide a sparse 10 element LiDAR input and relative target location as input to DRL for continuous control autonomous navigation, allowing for more effective and stable path planning than the move base method [6]. This provides a significantly less computationally expensive point-to-point navigation system without the use of a world map. Everett et al. use DRL to allow for path planning with continuous control to efficiently navigate through dynamic obstacles such as crowds of people [7]. They utilize Long Short Term Memory as a fixed-size input to a NN to internally map the static elements of the environment and track the dynamic elements effectively. Lillicrap et al. provide a general framework for taking in arbitrary input such as LiDAR or a camera and using it to control an arbitrary drive train through continuous control, such as Ackermann steer, skid steer, seven DoF arm, and many more with a finite action space [8]. They use Q-Learning to efficiently utilize an unknown control system in a simulated environment.

For our training environment, we make use of RobLearn developed by Surmann et al. [9], which conducts RL training more quickly than other simulators running in conjunction with ROS. The Q-step algorithm used in their RL training stems from the RL asynchronous parallel training algorithms developed by Mnih et al. [10] as an alternative to the costlier deep learning method experience replay memory. This parallel training frees resources for the CPU to learn more quickly and maintain control in a continuous exploration task. The RobLearn simulator was successful, but the primary focus of Surmann et al. [9] was on reaching a desired waypoint in an unknown environment and did not extend into exploring and searching the area.

Niroui et al. present a Deep Reinforcement Learning approach to Urban Search and Rescue (USAR) [11]. The sensor inputs to the system are a 2D Lidar and Odometry. These sensors are used to generate a 2D Occupancy Grid Map using the gmapping library for ROS [12]. The 2D occupancy map and odometry information is then combined with potential frontier points to form the inputs to a Deep RL A3C network. The output of this network is a selection of the best frontier point to visit next. They then utilize the ROS move-base package to navigate the robot to the frontier point [6]. The information gain used in their objective function is determined by the number of new open or occupied cells in the occupancy grid map. We build off this idea in the proposed approach to link cells searched by a camera to the information gain instead. The results presented in [11] show that the DRL approach to be slightly better than a deterministic approach where the objective function was a simple convex combination of information gain and distance. Motivated by this performance we investigate this approach for the autonomous search problem.

The main contribution of this paper is a preliminary investigation of using deep reinforcement learning for the application of autonomous exploration and search. First, we extend the work of [11] to optimize for searching as opposed to mapping an unknown area. Initial simulation tests with the simulator used in the DARPA SubT Challenge demonstrate an improvement in area searched as a function of time. Second, we build upon the work of Surmann et al. by expanding RobLearn's use beyond waypoint navigation and into search applications [9]. We modify the simulator to train in realistic underground environments found in the SubT Challenge. Preliminary simulation tests performed in the RobLearn simulator demonstrate the potential effectiveness of using Deep reinforcement learning for this application.

The remainder of this paper is organized as follows. In Section 2, we describe the problem of Autonomous Exploration and Search, and Section 3 presents our proposed solutions to the problem. The methods used in our simulation study are explained in Section 4. Next, the results of the simulation study are discussed in Section 5. Finally, the conclusion and future work are covered in Section 6.

2. Autonomous Exploration and Search

Performing search and rescue operations in an unknown environment where natural or other disasters have occurred is dangerous and taxing for even the most skilled human search and rescue teams. Autonomous platforms equipped with LiDAR and cameras can be used to search and provide feedback to rescue personnel. The block diagram in figure 1 describes the typical inputs and outputs for the autonomous exploration and search problem that we address in this work. An effective solution to this problem should use the LiDAR and camera readings to guide the robot in order to maximize the area searched within a limited time. Given the urgency of emergency situations, a rapid and efficient search of the area is highly valued as it has the potential to save lives.

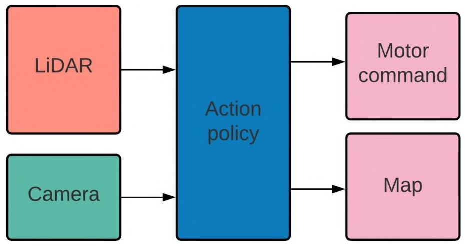

Figure 1: A block diagram that describes the input/output relationship of the autonomous exploration and search problem.

3. Navigation Algorithm

In this section, we describe an extension of the work of Niroui et al. [11] to optimize for searching unknown areas rather than exploration. We refer to this approach as Camera Frontier Exploration. Later in this section, we discuss a second approach that builds upon the work of Surmann et al. [9] by expanding RobLearn's use beyond waypoint navigation and into search applications. We refer to this approach as Deep Reinforcement Learning Search.

3.1. Camera Frontier Exploration

We propose a new camera view frontier exploration algorithm that was developed and tested in ROS. The open-source package LIO-SAM [13] was used for Simultaneous Localization and Mapping (SLAM), Move_Base [6] for navigating to points from the frontier exploration algorithm, a custom node was developed for the conversion of a 3D point cloud map from LIO-SAM to a 2D occupancy grid with dimensions of 300×300 meters and 0.5 m cells for the cost map for Move_Base using the Z-axis of points in the map to judge their cost, and finally, a ROS adapted variation of the class to record what the camera has seen generating the same size and resolution for the output grid as the cost map. The algorithms were given the pose from SLAM, the 2D cost map of the world, and the grid from the camera view. The proposed algorithm then generates a

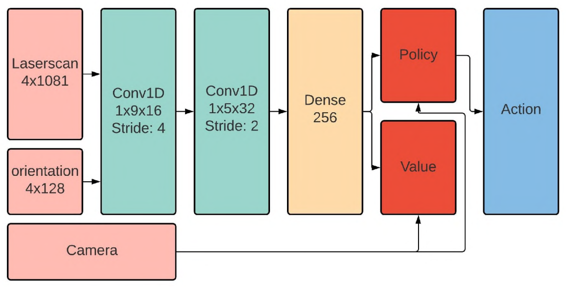

Figure 2: Architecture of the Actor Critic DRL network modified from [9] to include a camera input.

list of candidate waypoints based on open cells in the 2D occupancy grip that are on the boundary of searched cells in the camera view map. These candidate waypoints are then evaluated to determine the utility of visiting each of these cells. This utility is determined by the number of unsearched cells within a 5 m radius of the candidate cell. The cost of each candidate waypoint is evaluated based on the estimated distance from the current pose of the robot to the candidate cell as determined by an A* path planning algorithm. Then utility, u, and cost, d, are combined by a convex combination to form the optimization function, f_{opt}, using a weighting factor α.

$$f_{opt} = \alpha d + (1-\alpha)(1-u) \tag{1}$$

The waypoint resulting in the minimum value of f_{opt} is then chosen as the optimal waypoint to visit next.

3.2. Deep Reinforcement Learning Search

In addition to the Camera Frontier Exploration approach, we present a second approach that modifies the RL algorithm from [9] to include a search capacity. The objective of the GA3C network from [9] seeks to navigate to a desired goal location using the following reward function,

$$f_o(D, O, collision) = \begin{cases} -200, & collision \\ \beta + \delta \end{cases} \tag{2}$$

where β is a small constant reward factor given in accordance to the change in the robot's distance to the goal, D, over the current iteration, and δ is a small constant reward factor given according to the robot's change in orientation, O, relative to the goal heading.

Figure 2 shows the architecture of the DRL network which we modified from [9] for the search objective by including a class to record the area searched, A, using the horizontal field of view and range of the camera, logging the information in a ROS-like occupancy grid. The contents of this grid are used to compute the reward function for the DRL Search algorithm as differential square meters ($\frac{\Delta A}{\Delta t}$) by counting the number of searched cells in the grid over one iteration, Δt, converting those to square meters using the resolution of the grid. This modified reward function is,

$$f_o(A, collision) = \begin{cases} -200, & collision \\ \frac{\Delta A}{\Delta t} \end{cases} \tag{3}$$

In the case of a collision, the reward function f_o produces a large negative penalty, similar to (2). The $collision$ parameter is a Boolean representing whether there has been a collision with an obstacle in the simulator.

4. Simulation Methods

4.1. Camera Frontier Exploration: DARPA SubT Simulator

The camera frontier exploration simulations were run on the Open Source Robotics Foundation's (OSRF) SubT Simulator [14] in the Urban Circuit Practice 03 World. A preliminary set of simulations was run with the aforementioned simulators and algorithms assuming a 5 m range and 90^o field of view for the camera. The Camera Frontier Exploration algorithm was tested with random values in $0.25 < \alpha < 0.75$. The Camera Frontier Exploration approach was compared to the more traditional Map Frontier Exploration from [11] which seeks to efficiently build a map of the environment using LiDAR measurements without considering the area searched by the camera. The same optimization function, (1), was used with a weighting factor chosen randomly in the range $0.0 < \alpha < 1.0$, and utility u was replaced by the number of unmapped cells in the vicinity of the candidate frontier. The value of α was randomly selected to allow comparison between camera and map frontier exploration based on different weight values.

4.2. Deep Reinforcement Learning Search: RobLearn Simulator

The DRL Search algorithm was developed, trained, and tested using the RobLearn simulator. Adjustments to the simulator included changing the LiDAR field of view to 360^o and the range to 100 m, changing the diameter of the robot to 70 cm to match the SubT simulator, setting the starting point to be the origin, and adding three complex worlds to the simulation environment to represent the urban environments found in the DARPA SubT Challenge. These are shown in Figures 3, 4, and 5. The NN was trained on these 3 worlds simultaneously for just under 550,000 episodes total, and the trainer was configured to run 1000 step episodes with steps configured for 2 steps per second. The modified simulator is shown in Figure 6

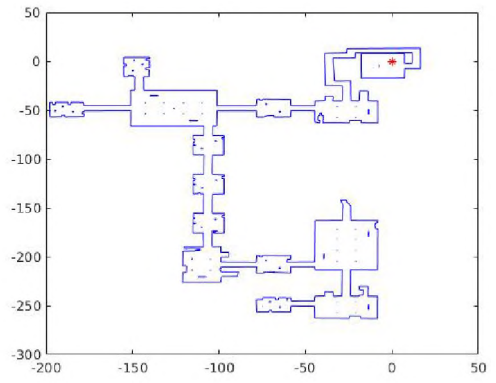

Figure 3: DARPA's Urban Circuit Practice 03 World adapted to the Roblearn simulator.

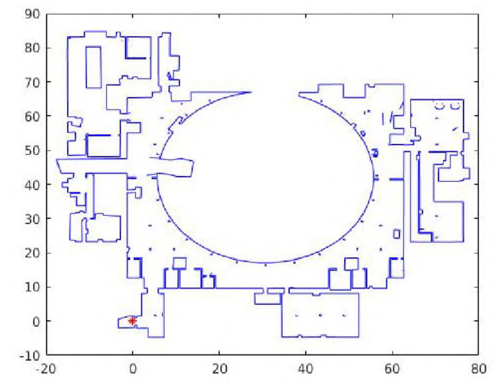

Figure 4: DARPA Urban Circuit Alpha Course.

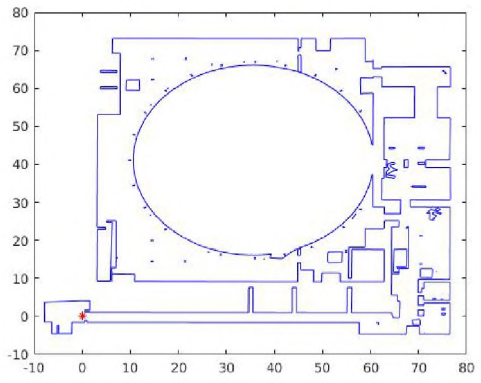

Figure 5: DARPA Urban Circuit Beta Course.

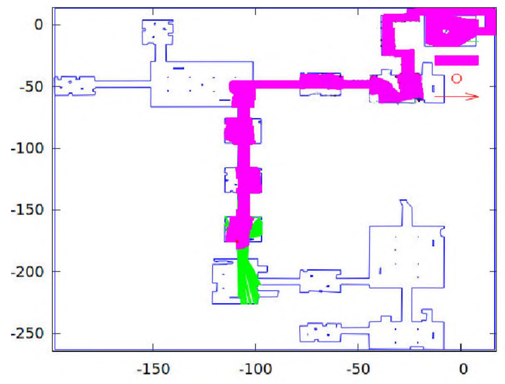

Figure 6: Roblearn simulator with camera view (magenta) and LiDAR (green).

in the world from Figure 3 with the camera view grid overlayed in magenta and the last LiDAR frame in green.

Both simulators recorded at the end of each simulation the elapsed time in simulated time and the number of cells searched. Using this information, the algorithm's effectiveness could be evaluated by calculating the area searched in square meters per second for each trial and comparing the results.

5. Simulation Results

By modifying the map frontier exploration approach from [11] to optimize for camera-based search, more area can be searched in less time compared to the original approach as demonstrated by figure 7. Camera frontier exploration averaged 2.38 m^2/s and a maximum of 3.34 m^2/s, and map frontier exploration averaged 1.01 m^2/s and a maximum of 1.85 m^2/s. Figure 7 shows our approach, camera frontier exploration, outperforms the traditional map frontier exploration for nearly all values of α.

The results of the DRL RobLearn simulations are contained in Figure 8 and show the progression of learning with an increasing number of episodes. The average search speed increases with the number of episodes, with the average rate of search of the last 50,000 episodes of the DRL training being 4.72 m^2/s and a maximum of 7.63 m^2/s. The average of the DRL training is significantly lower than its maximum due to the simulations being run on three rather different worlds simultaneously. These preliminary results demonstrate that the DRL method yields significantly faster search rates than either of the frontier exploration methods and once implemented in like environments this comparison could be validated.

6. Conclusions

Our results demonstrate that when considering autonomous exploration and search of an unknown environment one can improve upon traditional frontier exploration

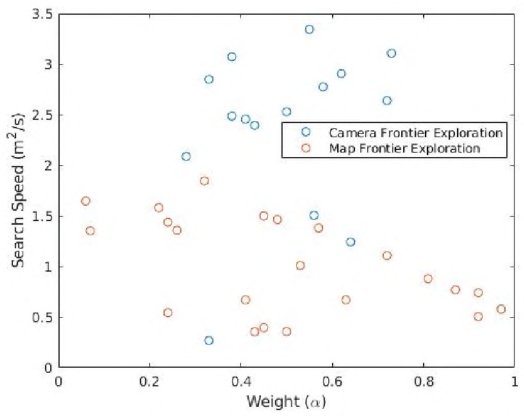

Figure 7: Search speed of the frontier exploration algorithms vs. α as measured in DARPA SubT Simulator.

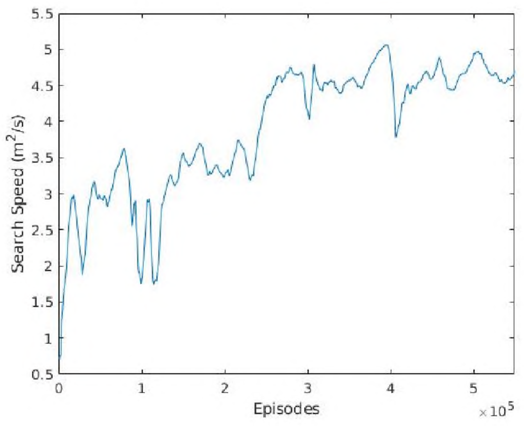

Figure 8: Average search speed of the DRL algorithm vs. training episodes as measured in the RobLearn Simulator.

techniques by considering camera-based frontiers. Furthermore, we have demonstrated the potential of using a deep reinforcement learning approach for autonomous exploration and search. Figure 8 shows an upward trend by the DRL method, gaining a higher score as more episodes are completed leaving more room to grow. However, our DRL method must be implemented in a like environment as frontier exploration for a complete comparison.

Going forward we plan to train and test our DRL algorithm with the ROBLearn software and transfer the trained network to the ROS simulator to compare this approach to the camera frontier exploration in a head-to-head comparison. Additionally, we plan to implement these algorithms on physical platforms and thoroughly test them in different environments. Ultimately, we aim to use these algorithms to compete in the finals of the DARPA SubT Challenge.

References

[1] SubT challenge competition rules urban circuit, 2019. URL `https://subtchallenge.com/`. Retrieved on 12/01/2020 from `https://www.subtchallenge.com/resources/`.

[2] Tai L, Liu M. A robot exploration strategy based on q-learning network. In 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR). IEEE, 2016; 57–62.

[3] Babaeizadeh M, Frosio I, Tyree S, Clemons J, Kautz J. Reinforcement learning through asynchronous advantage actor-critic on a GPU. Preprint arXiv:1611.06256 (2016).

[4] Zhang J, Tai L, Boedecker J, Burgard W, Liu M. Neural SLAM: Learning to explore with external memory. Preprint arXiv:1706.09520 (2017).

[5] Tai L, Paolo G, Liu M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In 2017 International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017; 31–36.

[6] Marder-Eppstein E. move_base ROS Wiki, 2020. URL `http://wiki.ros.org/move_base`.

[7] Everett M, Chen YF, How JP. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In 2018 International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018; 3052–3059.

[8] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. Preprint arXiv:1509.02971 (2105).

[9] Surmann H, Jestel C, Marchel R, Musberg F, Elhadj H, Ardani M. Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments. Preprint arXiv:2005.13857 (2020).

[10] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K. Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning. 2016; 1928–1937.

[11] Niroui F, Zhang K, Kashino Z, Nejat G. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. IEEE Robotics and Automation Letters 2019;4(2):610–617.

[12] Gerkey B. gmapping - ROS Wiki. URL `http://wiki.ros.org/gmapping`.

[13] Shan T, Englot B, Meyers D, Wang W, Ratti C, Rus D. LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping. Preprint arXiv:2007.00258 (2020).

[14] Koenig N. DARPA SubT Virtual Competition Software. `https://github.com/osrf/subt/wiki`, 2020.

Address for correspondence:

Andrew Herdering
CSU Channel Islands
One University Drive
Camarillo, CA 93012
andrew.herdering347@csuci.edu