IEEE Robotics and Automation Letters (RAL) paper presented at the
2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments

Farzad Niroui, *Student Member*, *IEEE*, Kaicheng Zhang, *Student Member*, *IEEE*,
Zendai Kashino, *Student Member*, *IEEE*, and Goldie Nejat, *Member, IEEE*

***Abstract*— Rescue robots can be used in urban search and rescue (USAR) applications to perform the important task of exploring unknown cluttered environments. Due to the unpredictable nature of these environments, deep learning techniques can be used to perform these tasks. In this paper, we present the first use of deep learning to address the robot exploration task in USAR applications. In particular, we uniquely combine the traditional approach of frontier-based exploration with deep reinforcement learning to allow a robot to autonomously explore unknown cluttered environments. Experiments conducted with a mobile robot in unknown cluttered environments of varying sizes and layouts showed that the proposed exploration approach can effectively determine appropriate frontier locations to navigate to while being robust to different environment layouts and sizes. Furthermore, a comparison study with other frontier exploration approaches showed that our learning-based frontier exploration technique was able to explore more of an environment earlier on, allowing for potential identification of a larger number of victims at the beginning of the time-critical exploration task.**

I. INTRODUCTION

Mobile rescue robots can be deployed in harsh urban search and rescue (USAR) scenarios in order to assist in exploring unknown cluttered environments, while searching for trapped victims. The advantage of using such robots in USAR is to help reduce the workload of rescue workers and improve their situational awareness [1]. Moreover, rescue robots can be used as additional agents to help speed up the search for victims during such time sensitive missions.

To address the autonomous exploration problem for USAR environments, rescue robots need to navigate to different locations in an unknown environment in order to map the environment and locate potential victims [2]. Simultaneous localization and mapping (SLAM) techniques [3]–[5] can be used to map areas the robots are exploring. To-date the most common method used for exploring unknown USAR environments is frontier exploration [2], [6]–[9]. In this method, a robot is directed to explore the boundaries of an already-explored area, i.e., the *frontiers* [10].

Frontier exploration techniques are mainly based on cost [8], [9], [11], [12] or utility-based [2], [6], [7], [11] performance metrics. In general, these approaches are optimized for a pre-defined environment layout and an objective such as greedily minimizing travel distance or maximizing information gain. However, USAR environments affected by disasters are unpredictable as their layout can significantly change post-disaster, making the reliance on a known layout not possible. Furthermore, USAR environments vary from each other and are not identical.

The general robot exploration problem has also been represented as a traveling salesman problem (TSP), in which both a TSP solver using optimization and a frontier exploration technique have been used [9]. The advantage of incorporating a TSP solver in the exploration task is that a near optimal solution can be obtained. However, a priori information of the environment is still needed such as a topo-metric graph.

Machine learning (ML) techniques have become increasingly popular in computer vision and robotics applications. Traditional machine learning and deep learning techniques have been applied to image classification [13]–[16], target object detection [17], [18], robot-team work division [18], and robot visual odometry [19], [20] problems.

With respect to the robot exploration problem, reinforcement learning (RL) has been used for exploring *unknown* environments. In particular, RL approaches have been used to teach robots how to explore a variety of unknown environments over time [21]. However, these approaches require robot state features to be handcrafted and suffer from the curse of dimensionality [22]. This limits their application to fully observable environments with low-dimensional state spaces [23].

Recently, deep reinforcement learning (DRL) approaches have been proposed to address these issues. In DRL, state features are learned automatically, and dimensionality can be reduced through iterative interactions with the environment [22]. DRL techniques have mainly been proposed for robot navigation in unknown environments, which requires a robot to learn how to avoid obstacles while traversing the environment [24]–[26]. To-date, DRL has yet to be used to develop exploration strategies for unknown cluttered environments such as those found in USAR applications.

To address the limitations of existing frontier exploration and traditional machine learning techniques, our research focuses on the first use of deep learning for the robot exploration problem in USAR applications. Namely, our contribution consists of the design of a novel DRL network architecture for frontier exploration to allow a robot to autonomously explore unknown environments. In particular, we develop a unique exploration approach which integrates DRL with frontier exploration to allow a robot to learn from

This research was funded in part by the Natural Sciences and Engineering Council of Canada, and the Canada Research Chairs program.
All authors are with the Autonomous Systems and Biomechatronics Laboratory in the Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON M5S 3G8, Canada. (email: {farzad.niroui, kc.zhang} @mail.utoronto.ca, {zendkash, nejat} @mie.utoronto.ca).

its own experiences while being robust enough to generate exploration strategies for varying unknown environments. The objective of our approach is to maximize the robot's information gain early on during exploration. This is a desired behavior to allow the robot to find trapped victims within an environment as quickly as possible.

II. Related Work

Exploration in unknown environments is generally achieved using frontier exploration. Frontier-based techniques can be categorized into: 1) utility-based [2], [6], [7], [11], or 2) cost-based [8], [9], [11], [12] approaches. In USAR applications, utility-based frontier exploration approaches have mainly been used. For example, in our own previous work, we developed a direction-based frontier exploration technique [2], [6]. The approach utilized a utility function which weighted the three parameters of terrain type (e.g., open unvisited, unvisited climbable, unknown obstacle), neighboring cell states, and travel distance to determine an exploration direction for the robot. In both simulated and real experiments, a mobile robot was able to successfully explore unknown cluttered USAR-like environments.

In [7], a multi-criteria technique was used to determine which frontier to explore based on the distance to a frontier, information gain, and the robot's ability to transmit information back to a base station. The approach was compared to other approaches which evaluated frontiers based on just distance or a combination of distance and information gain. For simulated USAR-like indoor environments, it was found that, on average, the proposed multi-criteria approach outperformed the other strategies.

A cost-based frontier exploration technique was presented in [8] where frontier locations were selected based on the robot's orientation. Namely, a set of selection rules were defined in order to reduce the expended energy in the exploration task by trying to avoid repeatedly navigating in the same area. In simulated experiments, this approach outperformed a utility-based technique in terms of exploration distance, repeated coverage and energy.

For frontier exploration in USAR, what is common amongst the different approaches is that the parameters considered for formulating the strategies are fixed throughout the task and they each have a constant weight assigned to them. Therefore, the formulated strategies will only perform well in certain environments with specific layouts. Our goal with DRL is to formulate an exploration strategy which generalizes to more environment layouts. Moreover, we want the DRL exploration strategy to evolve as the exploration task progresses in order to satisfy our objective to maximize the robot's information gain early on during exploration.

A. DRL-based Exploration

DRL methods have been developed to solve complex problems such as determining robot motion primitives in unknown environments [24]–[26]. For USAR applications, DRL has been applied to the robot visual servoing problem based on the detection of a target object [18]. As previously mentioned, DRL addresses the limitations of traditional RL techniques which suffer from the curse of dimensionality due to large state and action spaces by learning low-dimensional state features of high-dimensional states from sensory data [22]. For example, Deep Q-networks (DQNs) utilize large neural networks as the function approximator for value-based RL. In [24], a DQN was used to learn robot movement directions (forward, right, left) using depth images as input to the network. Simulations performed in corridors showed that the robot could navigate straight and circular corridors while avoiding walls.

In general, a DQN works with a large input space but it can suffer from slow convergence speed. To improve upon the DQN approach, Asynchronous Advantage Actor-critic (A3C) was developed [27]. A3C uses asynchronous gradient descent to optimize deep neural network controllers. This approach can significantly accelerate both the optimization and training processes. A3C is an on-policy learning algorithm, which maintains a policy and an estimate of the value function. Multiple parallel actor-learner threads are involved simultaneously, each of which operate in an isolated environment and determine accumulated loss in order to update weights of a central deep neural network, where the latter is shared among all threads [27].

In [25], an A3C DRL approach which provided a robot with long-term memory was presented. The robot learned the representation of a global map from sensory data and used this information to explore unknown regions of an environment. 2D and 3D simulations were conducted in rooms with obstacles, where a robot successfully explored the unknown environment by implementing the primitive actions of standing still, turning left or right, and moving forwards. This approach was compared to both random walk and A3C with no long-term memory, and the proposed approach had a higher success rate for the exploration task.

In [26], an A3C network was designed for a robot to navigate rough terrain in USAR environments, where goal target locations were provided. The network was trained to formulate primitive robot motion actions such as moving forward, moving backwards, and turning. The overall navigation task was achieved by the robot performing a series of these primitive actions in order to reach the a priori defined target locations. The approach was successfully tested in varying 3D simulated USAR-like environments.

Existing DRL approaches have been investigated for traversing corridors [24], rooms [25], and USAR-like environments [26], where the output of the networks are robot movement commands used to either navigate around obstacle or to specific locations. However, to-date, DRL has not yet been used to directly determine an exploration strategy that will allow the robot to search an unknown environment in order to maximize its knowledge gain. This exploration strategy must be able to effectively decide the optimal locations for the robot to explore. This is different from the motion problems addressed in [24]–[26], which focus on determining robot motion primitives to either simply move in a certain direction or to already defined target locations.

III. Deep Learning Robot Architecture for Exploration in USAR

Our proposed learning-based exploration architecture is presented in Fig. 1. The *World Model* generates a 2D occupancy grid (i.e., map) of the environment while a robot explores the unknown environment. The *Exploration* module uses this occupancy grid as well as odometry data to determine a frontier location to explore. This frontier location is sent to the *Navigation* module which computes a navigation path for the robot to this location, that is implemented using the *Low-level Controller*.

A. World Model

The 2D occupancy grid of the environment is generated using depth information from the 3D sensor, and the robot odometry information. The terrain in the occupancy grid is classified as open, occupied, and unknown cells. As the robot explores an environment, its information gain is determined by identifying the number of new open or occupied cells in the occupancy grid.

B. Exploration Module

For the *Exploration* module, we have designed an A3C network for frontier exploration, as shown in Fig. 2.

1) Inputs

The input to the network consists of the 2D occupancy grid, the locations of all possible frontiers, and the robot location. Each input is represented as a 64 by 64 array.

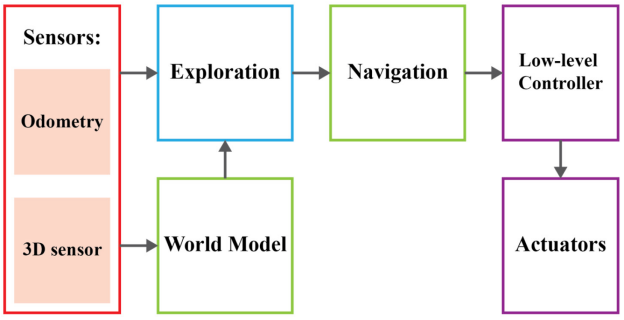

Figure 1. Robot exploration architecture for USAR.

The scaling of the occupancy grid to the network input dimension is achieved in a two-step process. First, the cells of the occupancy grid are grouped into 10×10 regions to remove redundant cells and to down scale the grid without losing useful information. For each region, the total number of cells in each cell category (open, occupied and unknown) is multiplied by its corresponding weight of w_{open}, $w_{occupied}$ and $w_{unknown}$, respectively. Then, the region is assigned to the cell category with the highest weighted sum. In the second step, the already scaled-down occupancy grid is resized to the desired input dimension using nearest-neighbor scaling.

For the frontier locations, all possible frontiers are identified by first extracting the open and unknown cells from the occupancy grid into two different layers. Then, the unknown cells layer is dilated in order to overlap with the open cells layer. The frontier boundaries are identified by performing element wise operations between the two layers. Frontier boundaries are then clustered using k-means to represent individual locations, with each location representing a single boundary.

2) Architecture

The network architecture contains multiple convolutional layers. After each convolutional operation, we use an exponential linear (ELU) activation function. In the design, the skip architecture is used to preserve the details of the input states which are found in earlier depths of the network [28]. Furthermore, a long short-term memory (LSTM) unit is used to ensure that the network also considers previous robot state features in order to make better decisions [29].

The output of the LSTM unit is directly used by the Actor and Critic layers. From the Actor layer, a single parameter representing a weight $w_{output} \in (0,1)$ is obtained. The *Filter* layer uses w_{output} in a cost-function to evaluate the cost of each frontier location. This cost is based on the distance, d, to each frontier location from the robot's current location as measured using the A* algorithm, and the potential information gain, g, at the frontier location determined by the number of unknown cells surrounding the frontier location in the robot's 2D occupancy grid. The overall cost function is

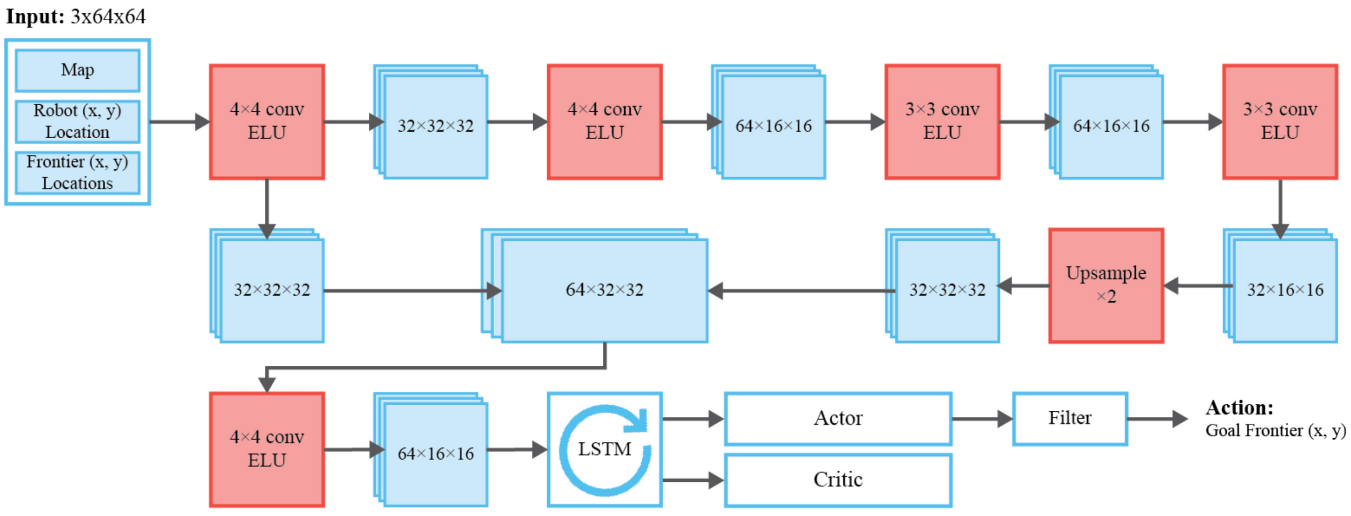

Figure 2. The proposed A3C network architecture for robot exploration of an unknown cluttered environment.

formulated as:

$$cost = w_{output}\,\bar{d} + (1 - w_{output})(1 - \bar{g})\,, \qquad (1)$$

where $\bar{d}$ and $\bar{g}$ are the normalized distance and information gain for a frontier location. Once all the frontier location costs have been determined, the location with the lowest cost is chosen for the robot to navigate to.

3) Objective Function

The objective of our approach is to maximize the total information gain along the robot's navigation path:

$$\max\left[\sum_{i=1}^{N}\left(\sum_{j=1}^{i} g_j\right)d_i + \left(D_h - \sum_{i=1}^{N} d_i\right)\sum_{i=1}^{N} g_i\right], \qquad (2)$$

where N is the total number of frontiers a robot navigates to in an exploration episode. For step $i, i \in N$, the robot travels a distance of d_i and obtains an information gain of g_i. D_h is a distance horizon determined by the environment size, namely, by defining an overall travel distance as a common scale for each environment to compare against.

C. Navigation Module

A frontier location chosen by the *Exploration* module is provided to the *Navigation* module, where the A* algorithm is then used to generate a path from the robot's current location to the frontier location. The Robot Operating System (ROS) move-base package [30] is used to generate movement commands along this path.

D. Training

To train the A3C network, a modified version of the 2D simulator, Turtlebot Stage [31], in ROS was used. A Turtlebot with both odometry and 3D sensory information was used to explore cluttered unknown environments. For grouping regions of the 2D occupancy grid, the weights of 0.08, 0.9 and 0.02 were used for w_{open}, $w_{occupied}$, and $w_{unknown}$, respectively. These weights were found through trial and error to be the best in preserving the useful information of the occupancy grid.

We used an AMD Ryzen Threadripper 1950x CPU for the training process. Ten actor-learner threads (e.g., agents) were utilized for training our A3C network. An additional agent, the test agent, was used for monitoring the progress of the training. Herein, an agent represents a single robot deployed in a unique randomly generated 1,600 m^2 cluttered environment. The agent's starting location within each environment was also random. This was done to avoid overfitting to a single environment. Fig. 3 shows an example of an unknown environment being explored, with possible and chosen frontier locations shown as red and green dots, respectively.

A single exploration of an environment was defined as an episode. At the start of each episode, the agents made a local copy of the global policy parameter, θ, and value function parameter, θ_v. Then, each agent at step i and with state s_i performed action a_i based on policy $\pi(a_i|s_i;\theta)$, and received a reward r_i for transitioning to a new state s_{i+1} [27]. The state consists of a combination of the 2D occupancy grid, agent location and frontier locations, and an action is the goal frontier location of the network. State transition occurs by the

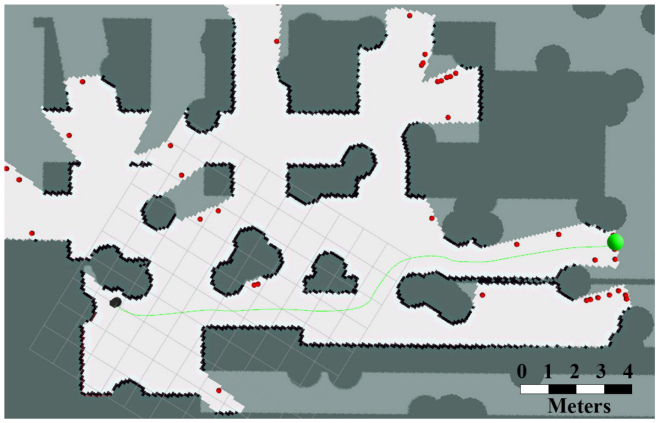

Figure 3. Example of a robot exploring a simulated environment ($40 \times 40\ m^2$) during training. Red points are potential frontier locations, and the green point is the chosen frontier.

agent navigating to a frontier location. This procedure was repeated until the exploration task was finished or the maximum step size, i_{max}, of 30 had been reached. Both θ and θ_v were then updated as follows [27], [32]:

$$\Delta\theta = \nabla_\theta \log \pi(a_i|s_i;\theta)\, A(s_i, a_i;\theta_v), \qquad (3)$$

$$\Delta\theta_v = A(s_i, a_i;\theta_v)\nabla_{\theta_v} V(s_i;\theta_v), \qquad (4)$$

where:

$$A(s_i, a_i;\theta_v) = \sum_{j=0}^{k-1}\gamma^i r_{i+j} + \gamma^k V(s_{i+k};\theta_v) - V(s_i;\theta_v), \qquad (5)$$

and $A(s_i, a_i;\theta_v)$ is an estimation of the advantage function that quantifies the additional benefit for taking an action in a certain state. k is the total number of steps from the current step i to the last step N in an episode, and V is the estimated value function that quantifies the additional benefit for taking an action in a certain state.

For training, a discount factor of γ=0.99 and a learning rate of α=0.0001 were used. An agent received the following reward:

$$r_i = \begin{cases} 0 & ,\ i \neq N \\ \dfrac{\sum_{i=1}^{N}\left(\sum_{j=0}^{i} g_j\right) d_i + \left(D_h - \sum_{i=1}^{N} d_i\right)\sum_{i=1}^{N} g_i}{D_h \sum_{i=1}^{N} g_i} & ,\ i = N \end{cases}. \qquad (6)$$

The reward is given when an agent completes an episode, and is dependant on the information gain and travel distance of the agent at every step during the episode. The objective is to provide higher rewards to an agent which has obtained more information about the environment during the earlier steps of an episode. Therefore, the weights of the network are tuned to prefer actions (frontier locations) which maximize the total information gain of an agent in the shortest distance possible.

Fig. 4 shows the progress of the training as recorded by the test agent, which shows the reward initially increasing and distance decreasing, and then both parameters converging. An episode, on average, took 2 minutes to complete. The test agent completed 1,760 episodes and the training agents, on average, completed 3,570 episodes. Overall, our model took 119 hours to train.

IV. EXPERIMENTS

To investigate the feasibility of our proposed architecture for exploration in cluttered unknown environments, we conducted three sets of experiments: 1) a performance and scalability test for our learning-based frontier exploration approach in varying environments, 2) a comparison of our proposed learning-based frontier exploration approach with six other frontier exploration techniques, and 3) the deployment of our approach on a physical mobile robot exploring an unknown cluttered environment.

A. Performance and Scalability Study

The Turtlebot Stage simulator was used with 30 randomly generated environments sizes of $20 \times 20\ m^2$, $60 \times 60\ m^2$ and $80 \times 80\ m^2$, respectively. The robot was able to explore, on average, 98.4%, 98.1% and 97.6% of these environments, respectively. There were some small unmapped regions (a few pixels in size) that were within the level of sensory noise. Fig. 5 shows an example environment for each environment size. Even though our A3C network was trained using $40 \times 40\ m^2$ environments, it was able to explore these randomly generated environments of different sizes. It is important to note, that the (down and up) scaling of the 2D occupancy grids to the necessary network input dimension did not affect the ability of the exploration approach in determining appropriate frontier locations for the robot to navigate to. All the important features of the occupancy grid were preserved.

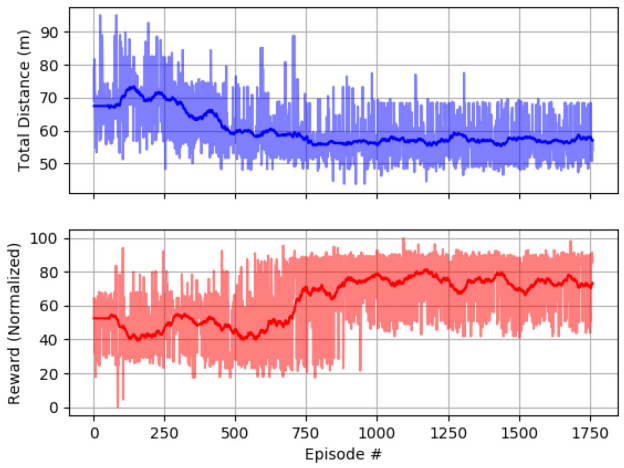

Figure 4. Total distance traveled (the distance traversed by the agent in a single episode) and reward per episode for the test agent.

B. Comparison Study

The performance of our learning-based frontier exploration approach was then compared with utility-based and cost-based frontier exploration methods presented in [7] that are applicable to our exploration problem, and a policy-based exploration approach that was shown to outperform traditional frontier exploration approaches in many scenarios [8]. These techniques are based on the following strategies:

1) *Cost*: This approach always selects the nearest frontier during exploration.
2) *Utility*: This approach chooses the frontier location with the largest potential information gain.
3) *Hybrid*: This approach uses a combination of distance cost and information gain using different weights in choosing a frontier location. We investigated three variations of this hybrid (H) approach with the following weight values for distance cost and information gain: a) 0.75-0.25, b) 0.50-0.50, and c) 0.25-0.75, respectively.
4) *Policy*: For this approach, a robot followed a policy which was a predefined set of exploration rules. Starting with the robot's heading direction, the first frontier in the clockwise direction and within a 10 m radius is chosen. If no such frontier exists, a random frontier is selected.

All approaches were implemented for 50 randomly generated cluttered environments. After each trial, the objective function (2) was evaluated for the specific techniques. Herein, we set the horizon, D_h, to be the total distance the robot traveled in the environment when it used the A3C network in order to be able to compare the performance of other exploration techniques to our proposed method.

Fig. 6 shows the average normalized objective function value for each technique in our comparison. The A3C network had the highest average value of 0.67, followed by the hybrid 0.75-0.25 weighting method with a value of 0.65.

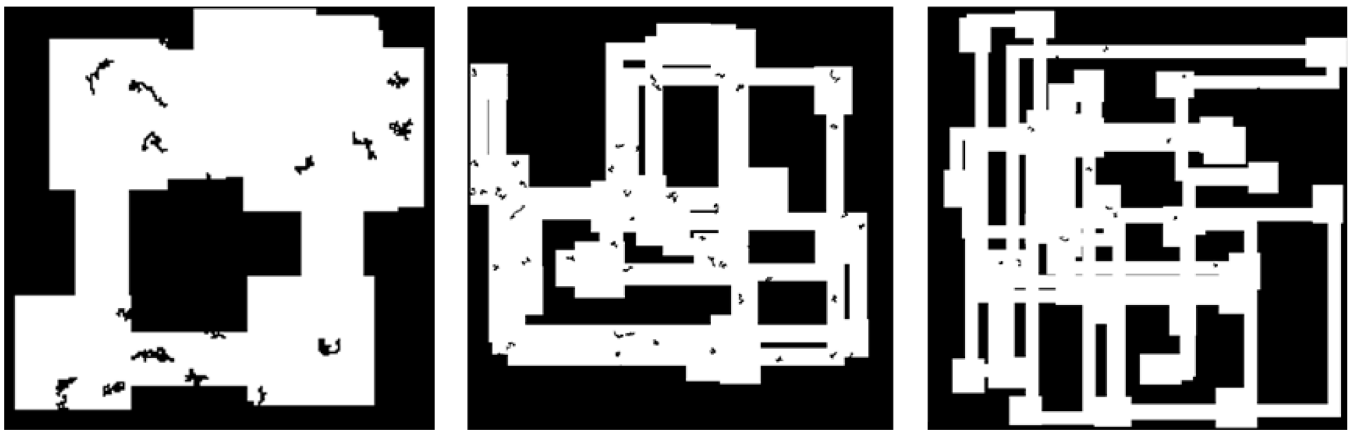

Figure 5. Example of a $20 \times 20\ m^2$ (left), $60 \times 60\ m^2$ (middle) and $80 \times 80\ m^2$ (right) environment used for the performance and scalibility study.

Fig. 7 shows the exploration progress of the robot as it traversed the same environment using the different techniques. In the majority of the trials, our learning-based exploration technique had more information gain early on, as is also shown in Fig. 7. For time-critical USAR missions, this behavior allows for a robot to explore more of the environment sooner in order to locate trapped victims faster.

A statistically significant difference was determined across all the tests using the non-parametric Friedman test in SPSS ($\chi^2(6) = 250.32, p < 0.0001$). Post hoc analysis was conducted using Wilcoxon signed-rank tests for pairs of exploration techniques with Bonferroni correction applied, changing the significance level to $\alpha_{revised} = 0.001$, with an original α of 0.05 and 50 tests. It was identified that there is a statistically significant higher average objective function value when using the A3C network when compared to the 6 other exploration techniques, Table I. The z test statistic and two-sided p-values of the Wilcoxon signed-rank tests are reported in Table I, where all p-values are less than 0.001.

C. *Physical Experiments in USAR-like Environments*

Experiments in an unknown cluttered USAR-like environment were performed with our overall architecture using a physical Turtlebot 2 with an onboard Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder. The A3C network used the weights obtained from the simulation training stage. In order for the robot to simultaneously map and explore the environment, the ROS gmapping SLAM package [5] was used. The inputs for mapping included wheel odometry and laser rangefinder data. The move-base package [30] was used for navigation to frontier locations. The environment, Fig. 8, was $15 \times 15\ m^2$ and consisted of three regions with obstacles and a single entry between these regions. We investigated the performance of the robot with respect to the total percentage of the environment explored. Overall, the robot was able to explore the environment (97% coverage) by visiting 44 frontier locations after traveling 95 m in 835 seconds. The map created by the robot, along with its path and the visited frontier locations are presented in Fig. 9. The average computation time to choose a frontier location was 1.2 s. A video showing the robot implementing our exploration strategy is presented here on our YouTube channel.

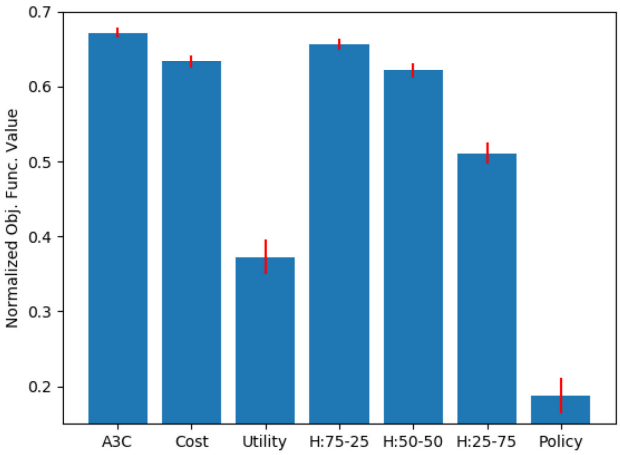

Figure 6. Average objective function values for all 7 techniques, normalized according to the A3C total traveled distance.

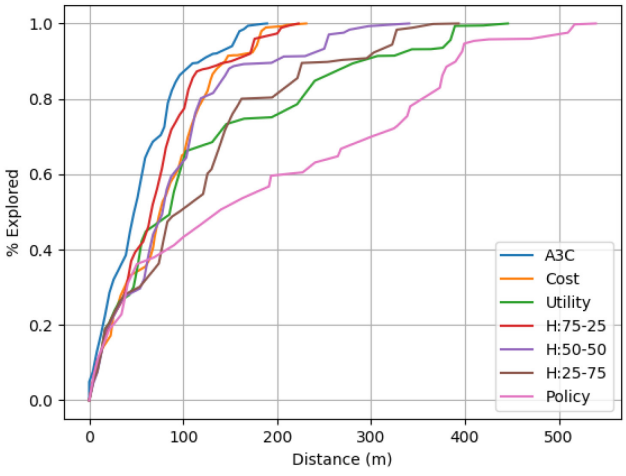

Figure 7. Percentage of environment explored with respect to distance traveled for all techniques for an example environment.

TABLE I. WILCOXON SIGNED-RANK TEST RESULTS FOR PAIRWISE COMPARISONS WITH THE A3C NETWORK (WITH Z TEST STATISTIC AND THE TWO-SIDED P-VALUE).

	z	p
Cost	-5.121	< 0.0001
Utility	-6.154	< 0.0001
H:75-25	-3.77	0.000164
H:50-50	-5.343	< 0.0001
H:25-75	-6.096	< 0.0001
Policy	-6.154	< 0.0001

V. CONCLUSIONS

In this paper, we have developed a unique approach which combines an A3C network with frontier exploration in order to learn an efficient exploration strategy based on high-dimensional robot states. Experiments with a mobile robot showed that the robot was able to effectively explore different unknown environments with varying sizes while

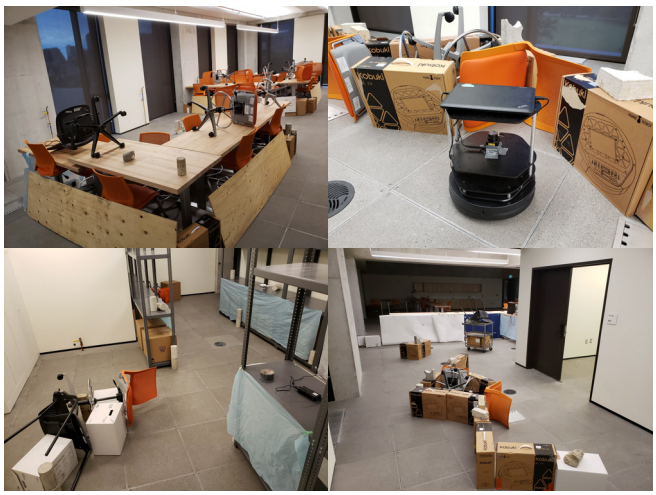

Figure 8. Experimental environment showing the three regions and the Turtlebot.

generating appropriate frontier locations to navigate to. A comparison study with our proposed learning-based exploration method and traditional exploration techniques showed that our method was able to explore more of the environment early on and had a statistically significant higher average objective function value. Our future work consists of extending our exploration approach to incorporate varying cluttered terrain that the robot needs to traverse such as climbable obstacles, and then testing our architecture in larger varying environments with both dynamic and static obstacles.

REFERENCES

[1] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective," *J. Intell. Robot. Syst.*, vol. 72, no. 2, pp. 147–165, Nov. 2013.

[2] F. Niroui, B. Sprenger, and G. Nejat, "Robot exploration in unknown cluttered environments when dealing with uncertainty," in *Proc. IEEE Int. Symp. Robot. Intell. Sensors*, Ottawa, Canada, 2017, pp. 224–229.

[3] H. Wang, C. Zhang, Y. Song, and B. Pang, "Robot arm perceptive exploration based significant SLAM in search and rescue environment," *Int. J. Robot. Automat.*, vol. 33, pp. 394-406, 2018.

[4] T. D. Barfoot, *State Estimation for Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[5] *gmapping - ROS Wiki*. Feb. 14, 2018. [Online]. Available: http://wiki.ros.org/gmapping. Accessed on: Feb. 24, 2018.

[6] B. Doroodgar, Y. Liu, and G. Nejat, "A learning-based semi-autonomous controller for robotic exploration of unknown disaster scenes while searching for victims," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2719–2732, Dec. 2014.

[7] N. Basilico and F. Amigoni, "Exploration strategies based on multi-criteria decision making for searching environments in rescue operations," *Auton. Robots*, vol. 31, no. 4, p. 401, Sep. 2011.

[8] Y. Mei, Y.-H. Lu, C. S. G. Lee, and Y. C. Hu, "Energy-efficient mobile robot exploration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, USA, 2006, pp. 505–511.

[9] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss, "Speeding-up robot exploration by exploiting background information," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 716–723, Jul. 2016.

[10] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, Monterey, CA, USA, 1997, pp. 146–151.

[11] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Auton. Robots*, vol. 33, no. 4, pp. 427–444, Nov. 2012.

[12] S. Wirth and J. Pellenz, "Exploration transform: A stable exploring algorithm for robots in rescue environments," in *Proc. IEEE Int. Workshop Saf., Secur. Rescue Robot.*, Rome, Italy, 2007, pp. 1–5.

[13] N. Ali *et al.*, "A hybrid geometric spatial image representation for scene classification," *PLoS One*, vol. 13, no. 9, Sep. 2018, Art. no. e0203339.

[14] N. Ali *et al.*, "A novel image retrieval based on visual words integration of SIFT and SURF," *PLoS One*, vol. 11, no. 6, Jun. 2016, Art. no. e0157428.

[15] B. Zafar, R. Ashraf, N. Ali, M. Ahmed, S. Jabbar, and S. A. Chatzichristofis, "Image classification by addition of spatial information based on histograms of orthogonal vectors," *PLoS One*, vol. 13, no. 6, Jun. 2018, Art. no. e0198175.

[16] N. Ali, K. B. Bajwa, R. Sablatnig, and Z. Mehmood, "Image retrieval by addition of spatial information based on histograms of triangular regions," *Comput. Electr. Eng.*, vol. 54, pp. 539–550, Aug. 2016.

[17] R. Wang, H. Lu, J. Xiao, Y. Li, and Q. Qiu, "The design of an augmented reality system for urban search and rescue," in *Proc. IEEE Int. Conf. Int. Saf. Robot.*, Shenyang, China, 2018, pp. 267–272.

[18] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *J. Intell. Robot. Syst.*, Jul. 2018.

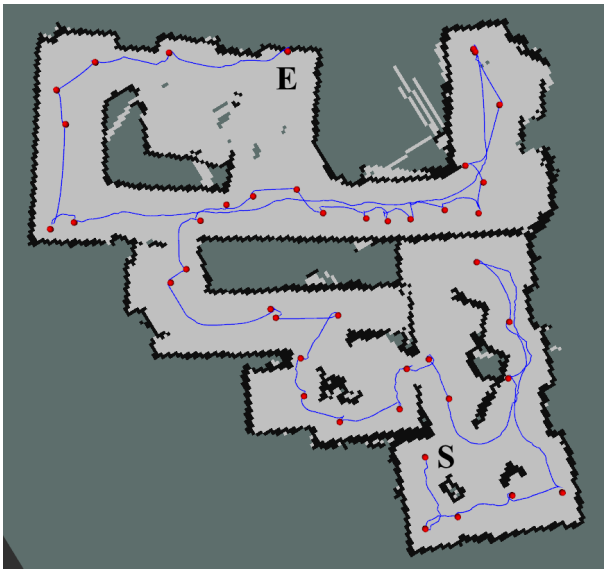

Figure 9. The 2D occupancy grid map generated by the robot. The black, light gray and dark areas are occupied, open, and unknown cells, respectively. The robot's path is represented by the blue line, and the red markers are the frontier locations that robot visited. The start and end locations are labeled with "S" and "E".

[19] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 2043–2050.

[20] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *Int. J. Robot. Res.*, vol. 37, no. 4/5, pp. 513–542, Apr. 2018.

[21] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.

[22] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[23] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[24] L. Tai and M. Liu, "A robot exploration strategy based on Q-learning network," in *Proc. IEEE Int. Conf. Real-time Comput. Robot.*, Angkor Wat, Cambodia, 2016, pp. 57–62.

[25] J. Zhang, L. Tai, J. Boedecker, W. Burgard, and M. Liu, "Neural SLAM: Learning to explore with external memory," Jun. 2017, ArXiv170609520 Cs.

[26] K. Zhang, F. Niroui, M. Ficocelli, and G. Nejat, "Robot navigation of environments with unknown rough terrain using deep reinforcement learning," in *Proc. IEEE Int. Symp. Safety, Secur., Rescue Robot.*, Philadelphia, PA, USA, 2018, pp. 1–7.

[27] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement Learning," in *Proc. Int. Conf. Mach. Learn.,* New York, USA, 2016, vol. 48, pp. 1928–1937.

[28] R. Garg, V. K. B.G., G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *Proc. Eur. Conf. Comput. Vis.,* Amsterdam, The Netherlands, 2016, pp. 740–756.

[29] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for partially observable MDPs," in *Proc. AAAI Fall Symp. Series*, Arlington, VA, USA, 2015, pp. 29–37.

[30] *move_base - ROS Wiki*. Sep. 27, 2018. [Online]. Available: http://wiki.ros.org/move_base. Accessed on: Feb. 24, 2018.

[31] *turtlebot_stage - ROS Wiki*. Sep. 25, 2018. [Online]. Available: http://wiki.ros.org/turtlebot_stage. Accessed on: Feb. 24, 2018.

[32] A. Gruslys, W. Dabney, M. G. Azar, B. Piot, M. Bellemare, and R. Munos, "The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning," Apr. 2017, arXiv170404651 Cs.