Robot Navigation of Environments with Unknown Rough Terrain Using Deep Reinforcement Learning

Kaicheng Zhang, *Student Member*, *IEEE*, Farzad Niroui, *Student Member*, *IEEE*, Maurizio Ficocelli, and Goldie Nejat, *Member, IEEE*

***Abstract*— In Urban Search and Rescue (USAR) missions, mobile rescue robots need to search cluttered disaster environments in order to find victims. However, these environments can be very challenging due to the unknown rough terrain that the robots must be able to navigate. In this paper, we uniquely explore the first use of deep reinforcement learning (DRL) to address the robot navigation problem in such cluttered environments with unknown rough terrain. We have developed and trained a DRL network that uses raw sensory data from the robot's onboard sensors to determine a series of local navigation actions for a mobile robot to execute. The performance of our approach was successfully tested in several unique 3D simulated environments with varying sizes and levels of traversability.**

I. Introduction

Mobile rescue robots deployed in Urban Search and Rescue (USAR) missions must navigate unknown rough terrain in order to explore cluttered environments to search for potential victims [1]. However, the traversability of the rough terrain can vary greatly, consisting of different rubble piles with various shapes and sizes. In order to be able to perform semi-autonomously or fully autonomously, these robots need to find navigation paths to safely navigate in these cluttered environments with unknown terrain with no a priori map of the environment.

Previous work in robot navigation of rough terrain has mainly focused on known terrain [2]. A feasible path to a goal location in the environment can be determined using such techniques as graph search [3], rapidly exploring random trees [4] and potential field methods [5], [6]. In cases when the terrain is unknown, the robot can navigate to multiple local target locations using a defined utility function [7]–[11]. By navigating to these local locations, the robot can therefore progress towards the final goal location. For a number of these approaches, the robot also obtains a model of the environment, e.g. [7]–[10]. The challenge with such approaches is that they can require substantial expert input for parameter tuning [12].

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Canada Research Chairs (CRC) Program.

K. Zhang, F. Niroui and G. Nejat are with the Autonomous Systems and Biomechatronics Laboratory (ASBLab) in the Department of Mechanical and Industrial Engineering, University of Toronto, (email: kc.zhang@mail.utoronto.ca), (email: farzad.niroui@mail.utoronto.ca), (e-mail: nejat@mie.utoronto.ca). *(The first two authors contributed equally to this work.)*

M. Ficocelli is with the Guidance, Navigation & Control group at MacDonald, Dettwiler and Associates (MDA) Inc., Brampton, Canada (e-mail: Maurizio.Ficocelli@mdacorporation.com)

In order to address this issue, learning techniques have been proposed for robot navigation in rough terrain [12]–[18]. These techniques focus on learning to classify the traversability of terrain from environment features. In particular, learning is used to 1) classify the surrounding terrain which is then represented as a costmap [13], [16]–[18] or 2) to learn the overall cost function [12], [14], [15], in order to plan optimal paths to goal locations.

Our own previous research has focused on utilizing traditional learning methods (e.g. MAXQ hierarchical reinforcement learning, support vector machines) and utility function based approaches to address such tasks as exploration, rough terrain navigation, and victim identification [19]–[22].

However, to effectively train learning techniques, usually a large number of labeled data is required, which can be time consuming to obtain [23]. A handful of techniques [13], [18] have automated the process of data collection and labeling by having the robot directly interact with the environment in order to assign a class to a set of online captured data.

In this research, we investigate the use of deep reinforcement learning (DRL) to address the robot navigation problem in environments with unknown rough terrain, in particular in USAR scenarios. DRL can directly use raw sensory data to determine robot navigation actions without the need of pre-labeled data [24]. A handful of papers have applied DRL approaches for robot navigation using onboard sensory information in environments with known [25]–[27] and unknown [28] flat terrain. However, DRL has yet to be implemented for cluttered environments with unknown rough terrain.

In USAR missions, we have areas of interest with high likelihoods of victims being present. A rescue robot needs to navigate to these regions, in order to search for victims. For the robot navigation problem addressed in this paper, these areas are defined as goal target locations for the robot. Namely, we are addressing the local navigation problem, where the goal target locations can be given by a global exploration planner such as in [29], and the robot needs to locally navigate to these locations without previous knowledge of the unknown rough terrain. Such a scenario would be after a natural disaster such as an earthquake, when a building has collapsed and the terrain at this site is unknown a priori, however, a rescue robot needs to navigate the environment to help search for victims.

In this paper we present the first use of DRL to address the mobile robot navigation problem in unknown rough terrain such as in USAR environments. The main contribution of this work is in the design of a DRL network which uses raw

sensory data from the robot's onboard sensors to determine a series of primitive navigation actions for the robot to execute in order to traverse to a goal location in an environment with unknown rough terrain.

II. Robot Learning in Rough Terrain

Existing work on robot navigation in rough terrain can be categorized into learning approaches that focus on classifying the terrain [17], [18] as well as its traversability [12]–[16], and learning approaches to determine robot navigation actions [25], [26], [28].

A. Learning to Classify Terrain for Navigation

With respect to classifying terrain, in [17], a Support Vector Machine (SVM) classifier was used to improve robot navigation capabilities in forest environments. The classifier was trained to identify 3D points from LiDAR data features which represented the surface of the terrain. The 3D points were hand labeled as either ground or non-ground and used for training the classifier. Online learning was also utilized in order to adapt the classifier to changes in the terrain due to season and vegetation growth. This approach was validated using different experimental data sets.

A terrain modeling approach was introduced in [18] where noisy sensor data from stereo cameras and LiDAR were used to estimate ground and vegetation height and classify obstacles mainly in agricultural environments. Three prior assumptions were made: ground height varied smoothly, terrain in neighboring cells were likely to be in the same class, and class members have similar height. The assumptions were represented as a probabilistic generative model which included two Markov random fields and a latent variable. Training data was autonomously collected by driving a tractor in a known environment to train the model.

With respect to learning the traversability of rough terrain environments, in general, sensory information and prior knowledge of the environment have been used to learn cost functions [12], [14], [15] or compute cost maps [13], [16] which are then used in planning navigation paths. For example, in [13], learning from demonstration (LfD) was used to learn a cost function from terrain data which were labeled by a human expert for navigation. In particular, LfD was used to learn the mapping from perceptual information to planning costs. The data included prior overhead feature maps of the outdoor environment and onboard sensory information such as LiDAR data. The cost function was used in conjunction with the Field D* planning algorithm to navigate a robot to waypoints. This approach was compared to manually engineered cost maps in outdoor field tests where a robot navigated a series of courses. The learned cost function achieved more reliable results and improved performance than the engineered cost maps.

Similarly, in [14], LfD was used to learn non-linear cost functions from geometric features of satellite maps of rough terrain. The maps were labeled by tracking the paths walked by several human experts, and then conformal geometric algebra was used to describe the maps. The geometric features were used as inputs into a neural network to learn the non-linear cost function to be used by a navigation planner.

In [15], an approach was introduced where far-range data from overhead images and height maps were used along with near-range data from a robot's onboard LiDAR to compute traversal costs of the terrain. This was done by using an online Bayesian learning approach to combine far-range and near-range features. In experiments, a robot was able to use the traversal costs along with the D* planner to navigate large outdoor cluttered environments.

In [13], a robot used on-line learning to classify different terrain patches of outdoor environments as traversable or non-traversable. Geometric and appearance features from stereo images were used as inputs. An autonomous data collection system was used to gather labeled training data. The collected features, which included height estimates and surface texture, were clustered into traversable and non-traversable terrain. A planner used the online terrain classifier to compute cost maps in two different outdoor environments. The results showed that the classifier approach was able to outperform a traditional planner when the robot had to navigate through tall traversable vegetation.

In [16], terrain traversability was determined by using a neural network with geometric features from stereo cameras as input. Four cost classes of low, intermediate, high, and lethal were used to represent the traversability. The training was done offline first and then in an online learning process a robot learned to infer geometric features from RGB images when stereo information was not available. By combining the classification from geometry features and color images, a cost map was created for path planning. A comparison with different network configurations and other classifiers showed that the neural network configurations were the most suitable classifiers due to their low error score and fast computation for a wide range of environments with obstacles and different lighting conditions.

B. Learning to Navigate Terrain

Learning techniques used in order for a robot to learn navigation actions have mainly used DRL to determine continuous or discrete navigation actions to travel using onboard sensory information in environment with known [25]–[27] or unknown [28] flat terrain.

In [25] a robot motion controller used a Deep Q-network (DQN) with *successor features* to provide discrete robot actions to a goal object location in a known environment. These actions included standing still, turning 90^{o} left or right, or going straight for 1 m. The input to the controller was depth images from a Kinect sensor. The controller was trained in simulation within an environment with walls and obstacles. The trained controller was transferred to a new environment on a physical mobile robot, and additional training was conducted. Experiment in the same environment showed that the number of additional training is less than having to train a new network from scratch, and the proposed model with *successor feature* preserved the ability to navigate both the simulated and real environments.

In [26], a motion controller using DRL with generalized computation graphs was developed to provide continuous steering angle predictions for a mobile robot. The inputs to the system included grayscale images from an onboard camera, as

well as previous action estimations. Training and testing were conducted in the same single path corridor for the robot to learn to move at a fixed speed without collisions. The proposed planner was compared with planners using Double Q-learning and random policies. The results demonstrated that on average, the robot was able to travel 7 times longer distances and 15 times longer distances respectively, after 4 hours of real world training.

In [27], a robot localization and navigation module was developed with a modified Asynchronous Advantage Actor-Critic (A3C) approach. The objective was to localize a robot in an environment for which a map is given and determine the navigation actions such as move (forward, backward, left, right) and rotate (left, right) to a given target location. The inputs to the system included a 2D map of the environment, RGB-D images, robot heading and the previous estimated robot location. Both training and testing were conducted in 3D simulated maze environments. Environments used for testing were different from the ones used in the training. Results showed navigation success rates of 91% - 100% depending on map sizes.

With respect to environments with unknown flat terrain, in [28], a mapless motion planner was developed using the asynchronous deep deterministic policy gradients method to provide continuous linear and angular velocities for a mobile robot to navigate to a target location. The input to the system was sparse laser range data, the previous robot action, and the relative position of the target location. The system was trained in two simulated indoor environments with walls and obstacles. Experiments were conducted on a mobile robot in an office environment in both simulations and in a real setting. The proposed planner was compared with the *Move Base* motion planner, and the results showed that it was able to navigate the robot to all target locations in unseen environments, especially in narrow areas, whereas the *Move Base* planner was not able to.

The aforementioned literature shows the feasibility of using DRL to learn navigation policies using high dimensional sensory inputs from the environment. However, to-date, DRL has not yet been applied to robot navigation in cluttered environments with unknown rough terrain. The traversability of the rough terrain can be challenging due to the varying shapes and sizes of both climbable terrain and non-climbable terrain. Our research focuses on the first development and investigation of a DRL network to address the robot navigation problem in such difficult environments.

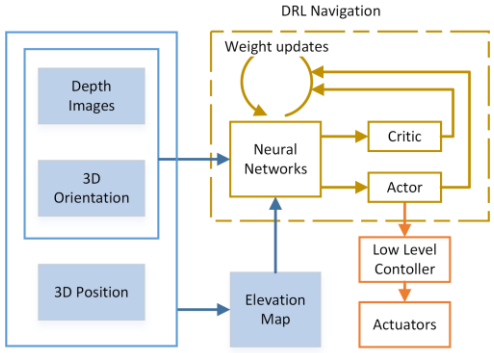


Figure 1. Proposed architecture for rough terrain robot navigation

III. Deep Reinforcement Learning for Robot Navigation in Unknown Rough Terrain

Our proposed DRL architecture for a robot to learn how to navigate an environment with unknown cluttered terrain is presented in Fig. 1. The architecture uses an end to end learning strategy, for which a robot learns to navigate rough terrain by using elevation maps and high dimensional sensory data from depth images as inputs into a *DRL Navigation* module. This module uses deep reinforcement learning to determine the navigation actions for the robot. Once a robot navigation action has been chosen, the robot's *Low-Level Controller* executes each action.

In the below subsections we discuss in more detail the main modules of our architecture.

A. Inputs

Inputs to the *DRL Navigation* module consist of depth images, the *Elevation Map*, as well as the robot's orientation (α, β, γ), where the robot heading γ is relative to a goal target location on the unknown terrain.

B. Elevation Map

An elevation map of the robot's surrounding environment is generated and updated as the robot navigates the environment. In order to obtain this elevation map we use the ROS OctoMap package [30]. Depth images as well as the robot 3D pose, containing both position (x, y, z) and orientation (α, β, γ), are used to create a 3D occupancy grid. This 3D occupancy grid is then converted into a 2D grid map using the ROS grid_map library [31], where each grid cell represents the elevation of a local square area.

C. DRL Navigation Module

The *DRL Navigation* module uses the Asynchronous Advantage Actor-Critic (A3C) approach [32] to learn a policy that determines the robot's navigation actions using depth information, the robot's 3D orientation and the elevation map. We use A3C as it can handle our high dimensional input space and has a faster learning speed compared to other DRL techniques such as DQN [32]. This is due to the approach deploying several agents to learn in parallel using multi-threads.

1) A3C

The A3C model uses the actor-critic method for reinforcement learning by combining the policy and state-value functions [32]. At each discrete time step t, the robot in state s_t executes a navigation action a_t which transitions the robot to state s_t' in order to maximize the expected future reward. These navigation actions consist of the robot moving forward, backward, or turning right or left. Herein, the actor maintains the policy $\pi(a_t|s_t, \theta)$ which is determined by a neural network with parameter θ. The critic estimates the value function $V(s_t; \theta_v)$ in order for the actor to adjust its policy accordingly. $V(s_t; \theta_v)$ is estimated by the neural network with parameter θ_v. Both the actor and the critic update their networks as new information is obtained by

the robot. The policy and the value function are updated after every t_{max} steps or when a terminal state is reached.

Rewards are used to compute and accumulate the gradients at every step. Stochastic gradient descent (SGD) is used to update the policy as follows [32], [33]:

$$\Delta\theta = \nabla_{\theta} \log \pi(a_t|s_t;\theta) A(s_t, a_t; \theta_v) + \zeta \nabla_{\theta} H\big(\pi(s_t;\theta)\big), \quad (1)$$

where,

$$A(s_t, a_t; \theta_v) = \sum_{i=0}^{n-1} \Gamma^i r_{t+i} + \Gamma^n V(s_{t+n}; \theta_v) - V(s_t; \theta_v), \quad (2)$$

and n represents the number of remaining steps until t_{max}, or until a terminal state. H is the entropy of the policy to encourage the robot to explore different navigation actions, and the hyperparameter ζ controls the strength of the entropy regularization term. $A(s_t, a_t; \theta, \theta_v)$ is an estimation of the advantage function, and Γ is the discount factor. SGD is also used to update the value function [32], [33]:

$$\Delta\theta_v = A(s_t, a_t; \theta_v) \nabla_{\theta_v} V(s_t; \theta_v). \quad (3)$$

Herein, θ and θ_v are optimized using the RMSprop approach [34].

As A3C deploys several agents acting independently in their own environment, each agent provides updates to the global policy and value function asynchronously.

2) Rewards

The reward r_t is given based on the robot: 1) getting closer to a target goal location, 2) reaching the target location within a certain tolerance (since the terrain is unknown), or 3) reaching an undesirable terminal state before reaching the goal.

The reward r_t is given after every executed navigation action and is represented as follows:

$$r_t = \begin{cases} \mu \dfrac{d_{\min} - d_t}{\Delta t}, & d_t < d_{min} \\ 0.5 + r_v, & d_t \le d_g \\ -0.5, & s_{tu} \\ 0, & elsewhere \end{cases}, \quad (4)$$

where d_{min} represents the closest distance the robot has navigated to with respect to the goal target location, up to step $t-1$. d_t is the distance from the robot to the goal target location at step t. Δt is the time duration from the previous time step $t-1$ to the current time step t. μ is a discount factor that can be used in order to choose the impact of this reward relative to the rewards provided for the terminal states. d_g represents the distance to the goal location and r_v is a dynamic variable that represents the distance from the robot's starting location to the goal location over the total time the robot took to reach the goal. s_{tu} represents an undesirable terminal state which includes repeatedly colliding with the same obstacle, the robot flipping over, the robot being stuck (i.e., attempting to climb a steep hill), or exceeding n_m number of steps. n_m is the maximum number of steps allowed for an episode, similar to the concept of *time-out*.

3) A3C Network Design

As previously mentioned, the neural network uses a depth image, an elevation map and the robot 3D orientation as inputs to determine a navigation action. The proposed network can be seen in Fig. 2. There are three input branches, one for each input type. The elevation map and the 3D orientation branches are merged together by connecting to a shared convolutional layer (CL). The output of this CL merges with the output of the depth image branch.

Depth image branch: depth images are downsampled from 480×640 to an 84×84 single channel array. Each element in the array represents a specific distance measurement. There are 4 CLs in this branch, the first two have 32 filters and the remaining have 64 filters. Filter sizes decrease gradually from 5×5 to 3×3, Fig. 3a. The output of the last layer is reshaped to a vector and concatenated head-to-tail with the reshaped output vector of the CL which merges the other two branches.

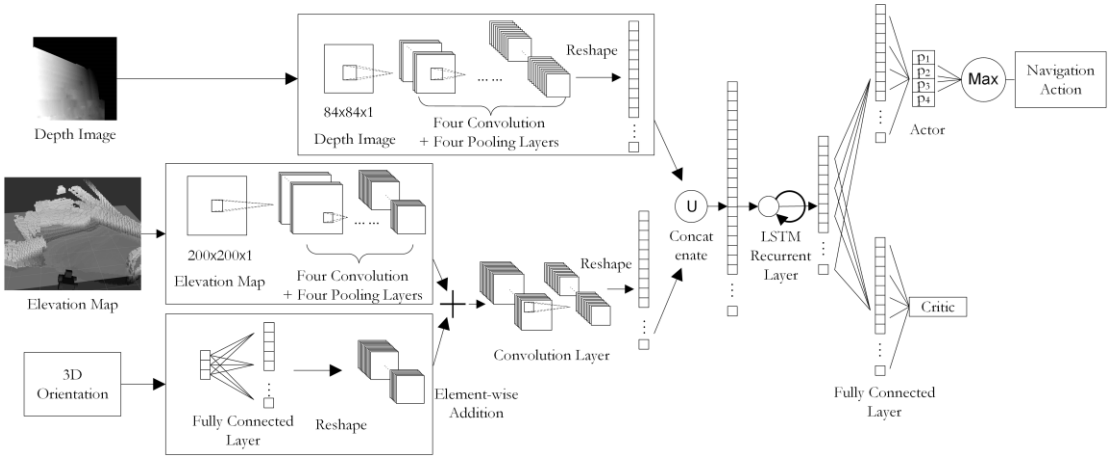
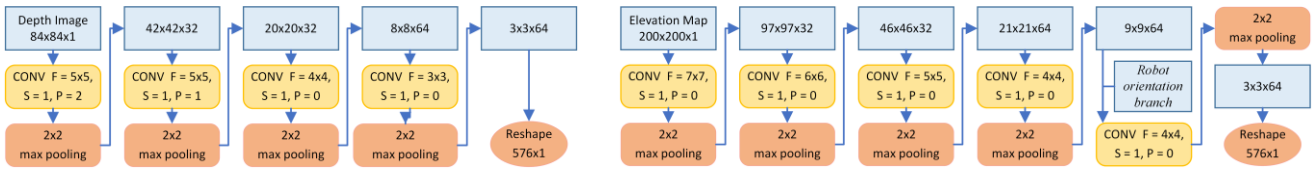


Figure 2. The proposed A3C network architecture. The three rectangular bounding boxes from the top to bottom represent the *Depth image branch, Elevation map branch,* and *Robot orientation branch,* respectively.



(a) (b)

Figure 3. Hidden layer configurations of the CNNs for: (a) the depth image branch, and (b) the elevation map branch and the layer after merging with the robot orientation branch. CONV represents a convolutional layer with filter size F, stride S, and padding P.

Elevation map branch: The elevation map is inputted as a single channel 84×84 array. The first 2 CLs of this branch have 32 filters and the remaining use 64 filters. Each filter produces an activation map as the result of convolutions. Filter sizes decrease gradually from 7×7 to 4×4 to account for the decrease of activation map size, Fig. 3b.

Robot orientation branch: the 3D orientation (α, β, γ) of the robot are input as a vector with 3 elements and processed through a fully connected layer (FCL). The output of this layer is reshaped to a volume of 9×9×64 to match the output shape of the elevation map branch. Using element-wise addition, the two volumes are merged and used as an input to an additional CL layer. The combination of α and β orientations with the elevation maps helps to infer 3D terrain conditions that the robot is traversing. This facilitates the learning of a policy that helps the robot to avoid steep slopes that can potentially roll or flip the robot over. Additionally, the robot heading γ allows the network to learn that when the robot's motion aligns with the direction of the goal target location, it usually approaches the goal quicker and gets higher rewards.

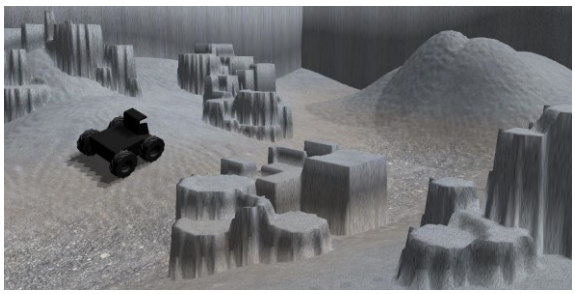


Figure 4. An example of the robot training environment.

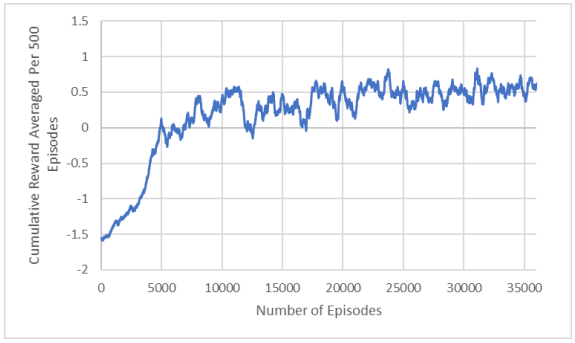


Figure 5. Cumulative Reward Per Episode Averaged Per 500 Episodes.

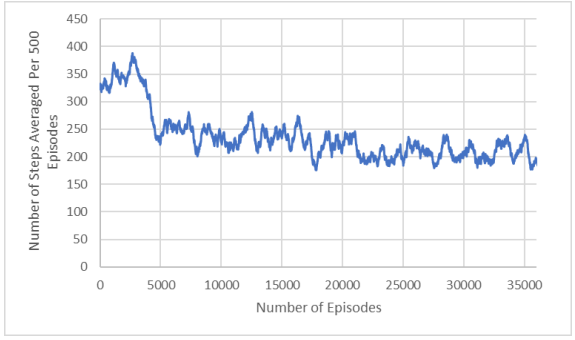


Figure 6. Number of Steps Per Episode Averaged Per 500 Episodes.

In this network, Rectified Linear Units (ReLU) [35] are applied on the output of all CLs. Once the branches are combined, the merged tensor is fed into a Long Short-Term Memory (LSTM) layer. This recurrent layer can improve DRL training by better capturing underlying states of a partially observed environment [36].

After the LSTM layer, the network separates into two output branches, *actor* and *critic*. The *actor* output branch has one FCL that outputs a 4-dimensional vector, each dimension representing one navigation action. The value of each dimension shows how likely a certain action is optimal (i.e., p_1 to p_4). The Softmax [35] function is applied to normalize the values. The action with the highest likelihood of being optimal is chosen as a_t. The *critic* branch also uses an FCL. It outputs a single value that represents the estimation of the value function $V(s_t; \theta_v)$, which is used to improve future policies.

4) Training

In order to train the A3C network for the *DRL Navigation* module, we created a 3D Simulator in Gazebo in the Robot Operating System (ROS). A 3D physics model for the Jaguar 4x4 wheeled mobile platform was created including weight, center of mass, speed and ground resistance to match the physical robot. The robot was equipped with both 3D odometry and a front-facing depth sensor.

We used 2,000 randomly generated unique 100 m^2 environments with percentage of traversability varying from 70-90%. These environments included traversable terrain that was non-flat, which included hills and valleys and irregular-shaped rubble piles, as shown in Fig. 4. The robot's starting location and goal target location within each environment were randomly selected for every episode on the traversable portion of the environment.

For our training, nine agent threads were simultaneously used on an AMD Ryzen Threadripper 1950x CPU. Each agent having its own unique environment. The values of the parameters used in training are as follows: $\zeta = 0.01$; $\Gamma = 0.99$; $\mu = 0.05$; $d_g = 0.5$; $n_m = 600$; and $t_{max} = 60$. The learning rate was set to 0.0001[32].

Fig. 5 and Fig. 6 present the training results for the cumulative rewards per episode and the number of steps per episode, both averaged across 500 episodes. As can be seen in Fig. 5, the network started to converge at 22,000 episodes for an average cumulative reward of 0.5. The number of steps per episode also decreased to an average of 208 steps.

IV. Experiments

We conducted experiments in new environments to validate the performance of our proposed DRL network for robot navigation in environments with unknown rough terrain. These environments varied in size from 100 m^2 to 400 m^2 and with percent traversability ranging from 80-90%. Namely, within our simulator, 600 environments, each with a unique terrain distribution, were generated. The starting location and goal target location for the mobile robot were randomly selected, however, these locations were constrained to the traversable portion of each environment similar to training.

TABLE I. Success Rate for Robot Reaching Goal Target Locations

Environment Size	Traversability		
	90%	85%	80%
100 m^2	84.1%	81.7%	71.3%
400 m^2	85%	83.1%	76.7%

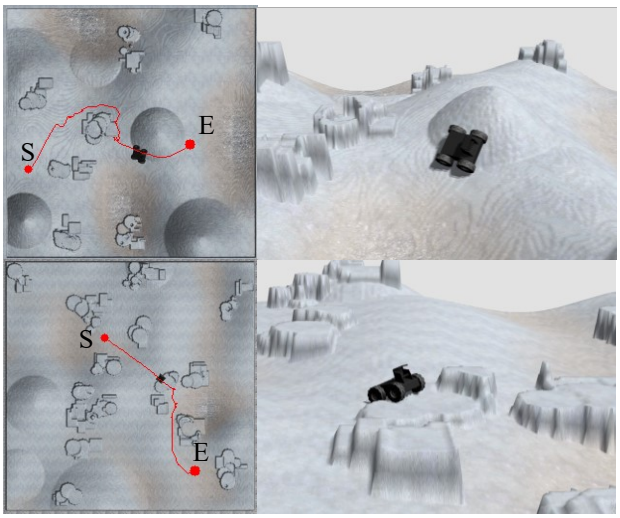


Figure 7. Topological view (left) and side view (right) of the robot traversing a hill and an irregular-shaped rubble pile. The robot navigation path is shown on the topological view. S represents the starting point and E is the end goal target location.

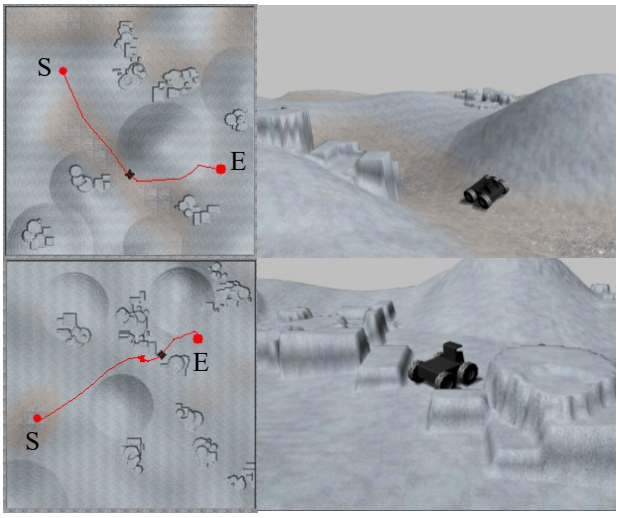


Figure 8. Topological view (left) and side view (right) of the robot navigating narrow pathways. The robot navigation path is shown on the topological view. S represents the starting point and E is the end goal target location.

A. Results

The robot was able to navigate to target goal locations in the majority of the environments. For the training stage we only used environment sizes of $100m^2$, whereas in testing we included the large environment size of $400m^2$ as well. The experimental results can be found in Table I. We obtained high success rates regardless of the size of the environment, showing that our approach is robust to new environments with different terrain distribution and sizes than used in training. As expected, the success rates were higher for the environments with less non-climbable obstacles, i.e., 85-90% traversability. The average computation time for the DRL Navigation module to select an action was 0.074 s.

Robot navigation paths in example environments with unknown rough terrain are shown in Fig. 7 and Fig. 8. In Fig. 7, it can be seen that the robot learns to traverse different terrain (i.e., a hill and an irregular-shaped rubble pile) while avoiding large non-traversable obstacles. Fig. 8 shows how the robot is able to traverse narrow pathways between large non-traversable objects.

From the experiments, we observed if the robot traversed too close to deep valleys with large descending slopes, it could slip down into the valley and not be able to get out (e.g. being stuck). This led to an undesirable termination state.

V. Conclusion

In this paper, we investigate the use of DRL for robot navigation in environments with unknown rough terrain such as the ones found in USAR. Namely, we developed a network based on the A3C architecture which uses depth images, elevation maps, and 3D orientation as inputs to determine optimal robot navigation actions. The network was trained in unique simulated 3D environments which varied in terms of level of traversability. Experiments in new environments ranging in size and traversability were conducted. The results showed that the DRL approach can successfully navigate a robot in an environment towards a target goal location when the rough terrain is unknown. Future work will include implementing and testing the navigation approach on a physical robot in similar environments.

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] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, Mar. 2015.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3310–3317 vol.4, 1994

[4] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 995–1001, 2000.

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 500–505, 1985.

[6] J. Barraquand, B. Langlois, and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. Syst. Man Cybern.*, vol. 22, no. 2, pp. 224–241, Mar. 1992.

[7] M. Wang and J. N. K. Liu, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends," *Robot. Auton. Syst.*, vol. 56, no. 7, pp. 625–643, Jul. 2008.

[8] F. Niroui, B. Sprenger, and G. Nejat, "Robot exploration in unknown cluttered environments when dealing with uncertainty," in *Proceedings of IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pp. 224–229, 2017.

[9] A. V. Savkin and C. Wang, "Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation," *Robot. Auton. Syst.*, vol. 62, no. 10, pp. 1568–1580, Oct. 2014.

[10] A. Chilian and H. Hirschmüller, "Stereo camera based navigation of mobile robots on rough terrain," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4571–4576, 2009.

[11] S. Lacroix and R. Chatila, "Motion and perception strategies for outdoor mobile robot navigation in unknown environments," in *Experimental Robotics IV*, Springer, Berlin, Heidelberg, pp. 538–547, 1997.

[12] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain," *Int J Rob Res*, vol. 29, no. 12, pp. 1565–1592, Oct. 2010.

[13] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick, "Traversability classification using unsupervised on-line visual learning for outdoor robot navigation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pp. 518–525, 2006.

[14] R. Valencia-Murillo, N. Arana-Daniel, C. Lopez-Franco, and A. Alanis, "Rough Terrain Perception Through Geometric Entities for Robot Navigation," in *Proceedings of International Conference on Advances in Computer Science and Engineering*, 2013.

[15] B. Sofman, E. Lin, J. A. Bagnell, J. Cole, N. Vandapel, and A. Stentz, "Improving robot navigation through self-supervised online learning," *J. Field Robot.*, vol. 23, no. 11-12, pp. 1059–1075, Nov. 2006.

[16] M. Happold, M. Ollis, and N. Johnson, "Enhancing Supervised Terrain Classification with Predictive Unsupervised Learning," in *Robotics: Science and Systems*, 2006.

[17] S. Zhou, J. Xi, M. W. McDaniel, T. Nishihata, P. Salesses, and K. Iagnemma, "Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain," *J. Field Robot.*, vol. 29, no. 2, pp. 277–297, Mar. 2012.

[18] C. Wellington, A. Courville, and A. Stentz, "A Generative Model of Terrain for Autonomous Navigation in Vegetation," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1287–1304, Dec. 2006.

[19] 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.

[20] Y. Liu, G. Nejat, and B. Doroodgar, "Learning based semi-autonomous control for robots in urban search and rescue," in *Proceedings of IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–6, 2012.

[21] B. Doroodgar and G. Nejat, "A hierarchical reinforcement learning based control architecture for semi-autonomous rescue robots in cluttered environments," in *Proceedings of IEEE International Conference on Automation Science and Engineering*, pp. 948–953, 2010.

[22] W.-Y. G. Louie and G. Nejat, "A victim identification methodology for rescue robots operating in cluttered USAR environments," *Adv. Robot.*, vol. 27, no. 5, pp. 373–384, Apr. 2013.

[23] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning," *ArXiv13125602 Cs*, Dec. 2013.

[24] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A Brief Survey of Deep Reinforcement Learning," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[25] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep Reinforcement Learning with Successor Features for Navigation across Similar Environments," *ArXiv161205533 Cs*, Dec. 2016.

[26] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation," *ArXiv170910489 Cs*, Sep. 2017.

[27] G. Brunner, O. Richter, Y. Wang, and R. Wattenhofer, "Teaching a Machine to Read Maps with Deep Reinforcement Learning," *ArXiv171107479 Cs Stat*, Nov. 2017.

[28] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31–36, 2017.

[29] J. Vilela, Y. Liu, and G. Nejat, "Semi-autonomous exploration with robot teams in urban search and rescue," in *Proceedings of IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–6, 2013.

[30] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.

[31] P. Fankhauser and M. Hutter, *Robot Operating System (ROS) – The Complete Reference, chapter 5*, vol. 1. Springer, 2016.

[32] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of International Conference on Machine Learning (PMLR),* Feb. 2016.

[33] 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," in *Proceedings of International Conference on Learning Representations*, Feb. 2018.

[34] G. Hinton, "rmsprop: Divide the gradient by a running average of its recent magnitude," presented at the COURSERA: Neural Networks for Machine Learning, 2012.

[35] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.

[36] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," *ArXiv150706527 Cs*, Jul. 2015.