Supervised Deep Actor Network for Imitation Learning in a Ground-Air UAV-UGVs Coordination Task

Hung The Nguyen*, Matthew Garratt*, Lam Thu Bui†, Hussein Abbass*
*School of Engineering and Information Technology, UNSW-Canberra, University of New South Wales
Canberra, Australia
Email: hung.nguyen@student.adfa.edu.au; m.garratt@adfa.edu.au; h.abbass@adfa.edu.au
†Department of Information Technology, Le Quy Don Technical University Hanoi, Vietnam
Email: lam.bui07@gmail.com

***Abstract*—Ground-Air coordination is a very complex environment for a machine learning algorithm. We focus on the case where an Unmanned Aerial Vehicle (UAV) needs to support a group of Unmanned Ground Vehicles (UGVs). The UAV is required to broadcast an image that contains all UGVs, thus, offering a bird-eye-view on the group as a whole. The source of complexity in this task is twofold. First, coordination needs to occur without communication between the UAV and UGVs. Second, the ability of the UAV to sense the UGVs is coupled with the ability of the UAV to learn how to track laterally the UGVs and adapt its vertical position so that the images of the UGVs are appropriately spaced within the camera field of view.**

In this paper, we propose using the Deep Actor Network component of an Actor-Critic Deep Reinforcement Learning architecture as a supervised learner. The advantage of this approach is that it offers a step towards autonomous learning whereby the full Actor-Critic model can be utilized in the future. Human demonstrations are collected for the deep Actor network to learn from. The system is built using the Gazebo Simulator, Robot Operating System, and the OpenAI Gym. We show that the proposed setup is able to train the UAV to follow the UGVs while maintaining all UGVs within camera range in situations where UGVs are performing complex maneuvers.

***Index Terms*—Deep neural network, Gazebo, Ground-Air Interaction, OpenAI Gym, Human Demonstrations, Learning by Imitation, ROS, UAV, UGV.**

I. Literature review

The problem of coordinating Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) is an active area of research [1]. The UAV is assumed to have a large field of view (FoV) [2] that enables it to support ground UGVs with a continuous feed of real-time situation awareness pictures (RT-SAP). This support enables UGVs to better plan and navigate in the environment and coordinate activities among themselves. The utility of this RT-SAP depends on the FoV of the UAV's camera. For example, a mission objective can be to support the UGV in forward planning; thus, the UAV needs to bring an area that is an order of magnitude (or even more) larger than the area occupied by the UGVs within its FoV. This can be useful in crowd control activities [3], rescue missions in disasters [4], or in surveillance, reconnaissance, and intelligence (SRI) missions in the military [5].

The challenges in the coordination problem between airborne and groundborne vehicles are multifaceted. One aspect of the problem lies in the impact of communication-borne latency on the ability of the vehicles to coordinate actions. The work of Khaleghi et.al. [6] compare different UAV-UGV coordination control systems. Howitt and Richards [7] cover the workload challenges of humans during interaction with the UAV command and control system. These challenges are mainly focusing on teleoperations, while autonomous UAV-UGV systems [8], [9] attempt to focus more on the problem as a distributed artificial intelligence (DAI) and/or multi-agent systems (MASs). Semi-autonomous operations in contested environments have also gained its share in the literature through work of Trentini and Beckman [10].

In this paper, we aim to design an autonomous controller for the UAV to track the UGVs while maintaining all UGVs within the FoV. The task is to ensure that the FoV is neither too small that it excludes some UGVs nor larger than the minimum needed to cover the smallest manifold containing all UGVs. This form of precision coverage is essential in situations where the UAV needs to maintain some form of line-of-sight FoV of the UGVs. While in some situations humans can teleoperate the UAV, human imprecision, latency in communication during teleoperations and even risks of communication-loss in teleoperations urge for designing fully autonomous controllers.

Autonomous controllers come in many types. In this paper, we assume that the controller will merely offer guidance while the low-level controller (control) is handled through the UAV control board. This is not a limitation but a practical way to implement neural-based controllers to maintain low level stability of the platform. Hence, when we refer to autonomous controllers in this paper, we mean in classic control sense autonomous guidance.

The challenge of this problem from machine learning perspective is to design ground-truth that a supervised learner can use to automatically guide the design of the controller. This challenge is even more complex in novel situations that have not been seen before. We address this challenge in two stages.

The first stage is the aim of this paper, where we use human demonstrations data as ground truth for a supervised learner. The second stage is the aim of our future research, where we hope to be able to bootstrap, from human data in simple tasks, autonomous controllers in more complex tasks.

The state space in this problem is uncertain, continuous and very large. Recently, deep learning methods excelled in similar applications related to intelligent mobile systems and to design control policies for robots [11], [12]. Ross et.al. [13] attempted to design autonomous reactive controllers for UAVs in cluttered environments. Their objective was to design a controller to fly while avoiding static objects such as trees in a forest. Other recent attempts looked at Deep Neural Networks (DNN) to design control policies for UAV [14]. The task in this paper is far more complex as the behavior of the UAV is tightly dependent on the behavior of moving targets on the ground that the UAV needs to adapt its behavior to this uncertain and unpredictable behavior. Moreover, the ground UGVs act independently and as a group; which increases the space of uncertainty in this problem space.

In the remainder of this paper, we will first explain the methodology in Section II followed by task and the environment in Section III. The experiments are then presented in Section IV, followed by results in Section V. Conclusions are then drawn in Section VI.

II. The Methodology

Reinforcement learning (RL) has shown a significant success in high-dimensional problems with continuous data [15]. RL agents learn by interacting with the environment. This leads to two challenges. First, they may need to spend a significant amount of time to map the problem state-space. Second, in some problems, this interaction may be costly or unsafe. Third, designing an appropriate reward functions to guide the agent is a non-trivial task.

Supervised learning (SL) learns quickly with many successful stories today on the use of Deep Neural Networks in SL tasks. However, SL agents normally require a significant amount of labelled data for training.

We propose a two stage methodology, whereby an SL agent is used on a reasonable sample of data collected from Humans to seed a model. In the second stage, the model is used to seed an RL agent; so that the agent does not start from scratch. This requires a framework that enables this transfer to happen.

In this paper, we use the Actor-Critic model for the purpose stated above. The Actor can be used as a supervised learner in the first stage. In the second stage, the full framework is adopted as a RL agent. The remainder of this paper focuses on the first stage to design human-competitive models.

The Deep Deterministic Policy Gradients (DDPG) [16] algorithm is designed to solve continuous control problems. In the DDPG architecture, there are two networks: an actor and a critic network. The actor network outputs continuous control policies, and the critic network steers the actor towards improving its control policy.

The Actor Network, a Deep Neural Network in the DDPG actor-critic architecture, is adopted alone as a SL agent. Human demonstrations are collected, where the states of the system are used as inputs to the Actor Network and the human actions are used as the targets. In the next section, this process is explained in details.

III. Environment and Task Description

There are three UGVs: UGV1, UGV2, and UGV3, and a UAV in this UAV-UGVs environment. The aim is for the UAV to maintain all UGVs within FoV without losing any UGV from the FoV or creating a FoV that is much larger than what is needed to accommodate the manifold created by the UGVs. To collect human demonstrations, the system allows a human to teleoperate the UAV from a distance. A pictorial representation of the system is shown in Figure 1.

In all scenarios, UGVs start from their base, where we always assume that the initial location of the UAV is at the center of the UGVs' base. A number of manoeuvre profiles are designed for UGVs. The task for the UAV is decomposed into two objective. In the first objective, the UAV needs to minimize the distance between its own center of mass and the center of mass of the UGVs within its FoV. The second objective is to minimize the difference between the radius of the UAV's camera FoV and the ideal radius needed. We define the ideal radius as the radius of the smallest circle to encapsulate the manifold formed by UGVs' formation. While the UAV has two cameras: a forward-looking camera and a top-down view camera, we only refer to the latter in this paper and don't use the former.

The above environment is built using Gazebo simulator [17], ROS framework [18] and OpenAI Gym [19]. A schematic diagram of the architecture to integrate the three systems together is presented in Figure 2.

Gazebo simulator [17] was used to design the task scenario. The drone simulator package in the simulation was Tum-Simulator [20]. In the paper, this package was simulated for the Parrot AR. Drone 2. The unmanned ground vehicles' simulator package was the Husky simulator [21]. The Husky package simulates a Husky medium size robot mounted with Microsoft Kinect and laser rangefinder sensors.

The control interface agent was programmed in Python to be compatible with OpenAI Gym [19]. When collecting human demonstrations, the control interface agent receives a real video stream from the UAV bottom camera, and simultaneously communicates the human's actions back to Gazebo. The human controls the drone using a Joystick. The human gets to see the image from the UAV top-down camera alone to have a fair setup comparable to the DNN Controller.

The Robot Operating System (ROS) [18] framework was used as an interface between the control interface agent and the Gazebo simulator environment. The actions and states were sent and received through ROS messages. We used ROS Indigo installed on Ubuntu 14.04.

The collected human demonstrations are then used by the DNN training algorithm in a supervised learning mode. Once

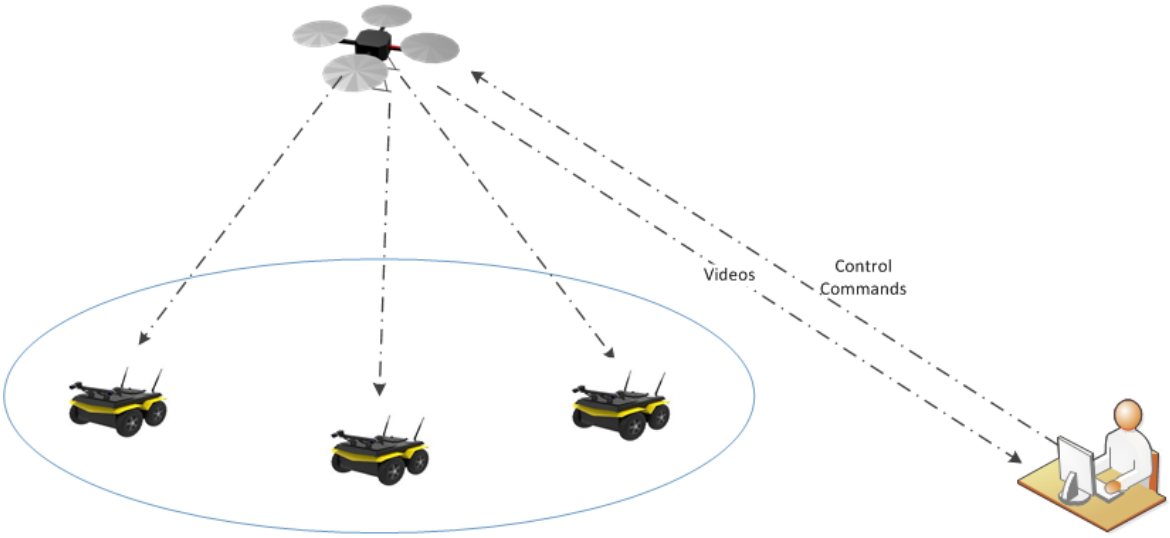


Fig. 1: The model of the UAV and UGVs Coordination Task

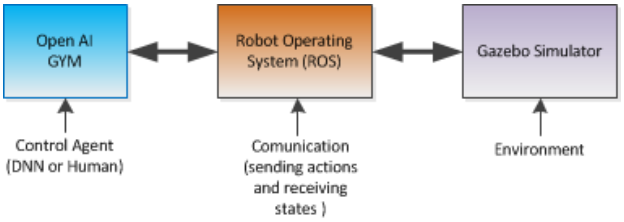


Fig. 2: The simulation environment protocol

training is complete, the same control interface agent is used, but this time with the human being replaced with the trained DNN for testing.

The UGVs' action space consists of 2 continuous real-valued actions representing the linear velocity, V, and the angular velocity (yaw rate), ω. The manoeuvre profile for the UGVs is prescripted according to the scenario being tested.

The UAV actions space consists of 4 continuous real values representing pitch and roll attitude, altitude, and yaw, and are denoted as (p, r, a, y), respectively. The state vector of the environment is a 17-D tuple of the continuous variables presented in Table I.

Meanwhile, (cx, cy) are received from the UAV bottom camera configuration, and $UAVz$, (p, r, a, y), $(V1, \omega1)$, $(V2, \omega2)$, $(V3, \omega3)$ are subscribed from the Gazebo environment. $(UGVx, UGVy)$ and (ra_i, ra_a) are calculated based on pinhole camera model. ra_a is the distance from $(UGVx, UGVy)$ to the furthest UGV position within the bottom image.

The first objective is given in Equation 1, where $||.||$ denotes the second norm. The second objective is given in Equation 2. Both objectives are to be minimized as indicated by the down-facing arrow.

TABLE I: The State Space

State ID	State Name	State Description
1-2	(cx, cy)	Centre of the UAV from bottom camera
3-4	$(UGVx, UGVy)$	UGV Centre of Mass within the UAV image
5	$UAVz$	UAV altitude in Gazebo model
6	ra_i	Ideal UGV radius within image
7	ra_a	Actual UGV radius within image
8-11	(p, r, a, y)	UAV velocity vector
12-17	$(V1, \omega1)$, $(V2, \omega2)$, $(V3, \omega3)$	UGV1, UGV2, and UGV3 velocity vector

$$\downarrow distance\_error = \int_t ||(cx, cy) - (UGVx, UGVy)|| \quad (1)$$

$$\downarrow radius\_error = \int_t (ra_i - ra_a)^2 \quad (2)$$

The DNN architecture used in this paper is given in Figure 3. The network consists of an input layer, two fully-connected hidden layers, and a fully-connected output layer. The states defined in Table I are used as inputs, while the outputs, as discussed above, are p, r, a, y. We use 300 Relu units, tanh for the output layer, the Adam method for optimization, and the Mean Squared Error (MSE) for the loss function. Tensorflow and Keras libaries [22] were used to design the deep neural network. The deep network was trained on an NVIDIA GeForce GTX 1080 GPU.

A. Human Demonstrations Scenarios

We use four maneuvers in this paper: Lateral-Movements-Fixed-Altitude maneuver, Climb-Only maneuver, Descend-Only maneuver, and Lateral-Movements-With-Climb-Descend

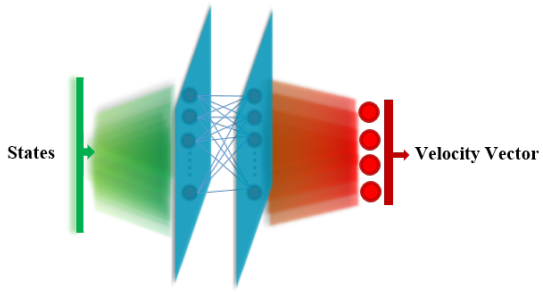


Fig. 3: The Deep neural network structure.

maneuver. The first three maneuvers give us a baseline for simpler manoeuvre, while the fourth maneuver is a more complex manoeuvre. There are 6 basic actions to control the UAV: Move Forward, Move Backward, Turn Left, Turn Right, Climb, and Descend.

In the Lateral-Movements-Fixed-Altitude maneuver, UGVs are allowed to move forward, turn left, and turn right. However, they need to move as a group with synchronized homogeneous action set to maintain their formation intact. This fixes the manifold of the UGV formation over the course of the scenario. In this scenario, the human needs to focus on four basic actions of turning right/left and moving forward/backward.

In the Climb-Only maneuver, three distinct phases are followed. In the first and third phase, UGVs move forward with fixed formation and the same linear velocity. In the second phase, UGV2 and UGV3 separate from UGV1. They separate in opposite directions, which cause the radius required to bring them back to the UAV camera FoV to increase. This necessitates a "climb" action of the UAV. We label the behavior of the UGVs "separation," as the UGVs get spread away from each other similar to the separation/repulsion force in a swarm Boid model.

The Descend-Only maneuver is identical to the Climb-Only maneuver except for the second phase, where UGV2 and UGV3 converge on the position of UGV1, reducing the radius required to bring them back to the UAV camera FoV to increase. This necessitates a "descend" action of the UAV. We label the behavior of the UGVs "cohesion," as the UGVs get attracted to each other similar to the cohesion/attraction force in a swarm Boid model.

In both Climb-Only and Descend-Only maneuvers, the forward movement of the UGVs would mean that the human needs to only control the height of the UAV; increasing it in the first maneuver and descending it in the second.

The last maneuver, Lateral-Movements-With-Climb-Descend, is a combination of the above three. A scenario consists of 4 periods. Four movement patterns of the UGVs are labelled 1 to 4, representing Separation-Left, Cohesion-Right, Separation-Right, and Cohesion-Left manoeuvres, respectively.

Each scenario consists of four periods, where UGVs move according to a manoeuvre drawn randomly from the above four basic manoeuvres. For example, a scenario that follows 1234 will start by by having the UGVs separate while turning left, then when this manoeuvre gets completed, UGVs start converging on each other while turning right, followed by turning right again while separating, and lastly turning left while converging on each other.

In this setup, the human needs to combine actions by moving the UAV in appropriate directions and climbing and/or descending it as necessary to achieve the mission objective

In the rest of the paper, we will use the following short naming style to indicate the four maneuvers: Fixed-Altitude, Climb, Descend, and Combined, respectively.

IV. Experiments

In this section, we aim to evaluate performance between the human subject and our trained DNN in all four maneuvers described above. Furthermore, for each of the four maneuvers, we also investigate two scenarios. In the first scenario, the state space includes ground truth information about the locations of the UGVs for situations where UGVs are allowed to communicate their position to the UAV. In the second scenario, this state information is not available to the UAV; thus only information available on-board of the UAV is used. When assessing these two scenarios, our target is to know whether the extra information obtained from the UGVs offers benefits for training the DNN. Therefore, we have a total of 8 setups as described in Table II.

TABLE II: The Setups in the paper

ID	Name	Meaning
1	S1-Fix-Altitude	Using UAV/UGVs States Space for the Fixed-Altitude maneuver
2	S1-Climb	Using UAV/UGVs States Space for the Climb maneuver
3	S1-Descend	Using UAV/UGVs States Space for the Descend maneuver
4	S1-Combined	Using UAV/UGVs States Space for the Combined maneuver
5	S2-Fix-Altitude	Using UAV States Space for the Fixed-Altitude maneuver
6	S2-Climb	Using UAV States Space for the Climb maneuver
7	S2-Descend	Using UAV States Space for the Descend maneuver
8	S2-Combined	Using UAV States Space for the Combined maneuver

Demonstrations from the human subject were collected for 10 episodes in each maneuver, except for the Fixed-Altitude maneuver in which 5 episodes were performed because the operating path of the UGVs was significantly longer and to balance the labels of the data needed for training DNN. In total, the four data sets had 5296, 4691, 4904, and 5464 instances for the Fixed-Altitude maneuver, Climb maneuver, Descend maneuver, and Combined maneuver, respectively. The DNN is trained for 10,000 epochs with a batch size equal to the number of data instances in each setup.

After training, the deep neural network for each experiment is tested. The testing paths are used, and in each experiment, the agent is tested ten times on randomly generated cases. For the Fixed-Altitude maneuver, Climb maneuver and Descend maneuver, the UGVs movement testing paths are nearly fixed in every episode; however, variations among the maneuvers are caused by uncertainties in the behavioral envelope of the UAV's dynamics.

V. Results and Evaluation

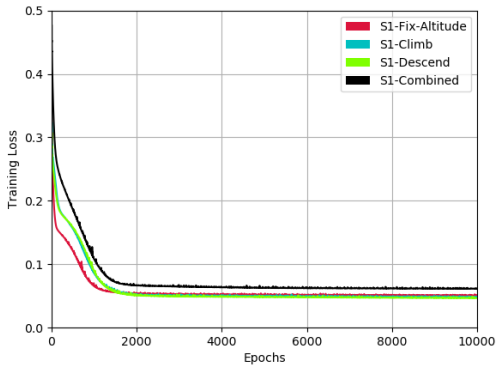


Fig. 4: The Measure of Mean Squared Error over Epochs in S1.

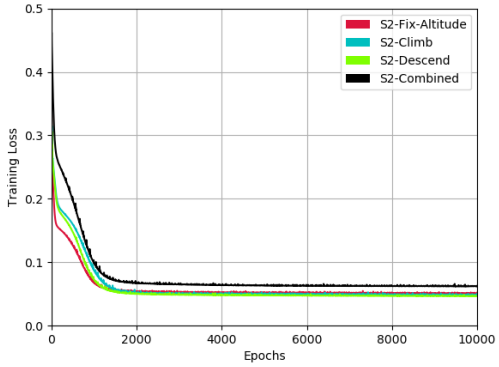


Fig. 5: The Measure of Mean Squared Error over Epochs in S2.

Figures 4 and 5 show the value of Mean Squared Error (MSE) in each of the two scenarios for each maneuver. In both scenarios, the training is very fast (about 40 minutes in real time), the error stopped declining and became stable around 2000 epochs.

To evaluate performance, we calculate the distance between the UAV's Center (cx, cy) and the center of UGVs' mass $(UGVx, UGVy)$, and the difference between the actual radius ra_i and ideal radius ra_a. In Table III, we present the average and standard deviation of these two metrics for each scenario.

The results in Table III are interesting. Our trained DNN in Scenario 2 managed to align the UAV with the UGVs' center of mass with performance equivalent to the human subject in the first three maneuvers, and better in the case of the Combined maneuver although the difference is not statistically significant. The same cannot be said with regard to the radius. In the Climb and Combined maneuvers, regarding to the radius error, the DNN seems to overshot the height of the UAV, pushing it to a higher altitude than needed.

Scenario 1 had inferior performance than Scenario 2 in terms of adjusting its altitude. This is surprising because in Scenario 1, the state space includes exact information on the velocity of the individual UGVs and the ideal radius. The DNN caused the UAV to overshot in its altitude in all maneuvers requiring altitude adjustments, while in the first maneuver, the DNN figured out the easy solution of not adjusting the UAV height. The DNN in this scenario was worse than Scenario 2 in terms of alignment with UGVs' center of mass in all maneuvers except the Climb maneuver.

It is worth mentioning that despite the variations discussed above, in both Scenario, the DNN always maintained the UGVs within the range of the camera in all maneuvers and all test cases. To better understand the phenotypic differences between the human performance and DNN, we visualize the behaviour of the UAV when it is under human control and compare it with the behaviour when it is under DNN control in each scenario. For space limitations, we restrict the visualization to those presented in Figures 6 to 13.

Some general observations can be made on the figures. First, Figure 6 shows a smoother track by the DNN when UGVs make sharp turns when compared to the human. When tracking UGVs laterally (Figure 9), once more DNN seems to track in a smoother manner, while the human seems to be attempting to track optimal on the cost of generating consistent steering of the vehicle. Such a behavior consumes more energy.

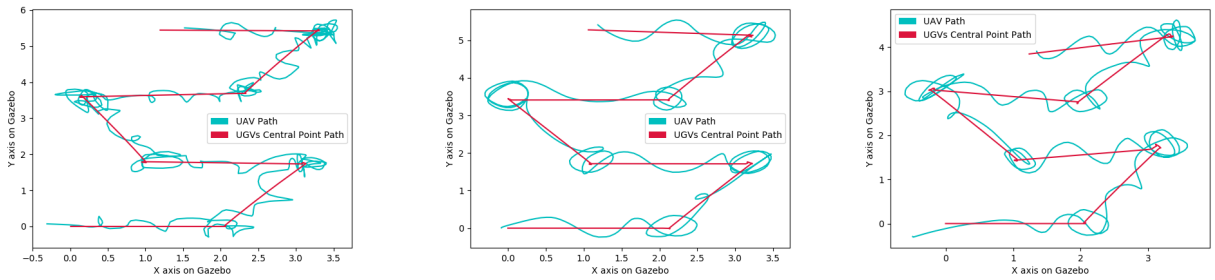
VI. Conclusion and Future Work

In the paper, we used the Deep Actor Network as a supervised learner in a ground-air interaction context. An unmanned aerial vehicle (UAV) attempts to maintain a mobile group of unmanned ground vehicles (UGVs) within its camera range. We used a human to generate the training set and tested two scenarios. In the first scenario, the state space for the Deep Neural Network (DNN) included information on UGVs and the ideal target radius that needs to covered, as well as information from the UAV's camera. In the second scenario, the state for DNN excluded UGV information and only included information drawn from the UAV's Camera. In each scenario, we tested four maneuvers: three simple ones where the UAV needs to either track UGVs by moving forward, climbing and descending. In the fourth maneuver, the UGVs move in more complex maneuvers where all three forms of behaviour (lateral tracking, climbing and descending) are used.

It was interesting to see that the DNN trained very fast, with only 2000 epochs, it reached its maximum performance. The extra information obtained from the UGVs did not offer advantages for DNN. In fact, DNN was able to learn better

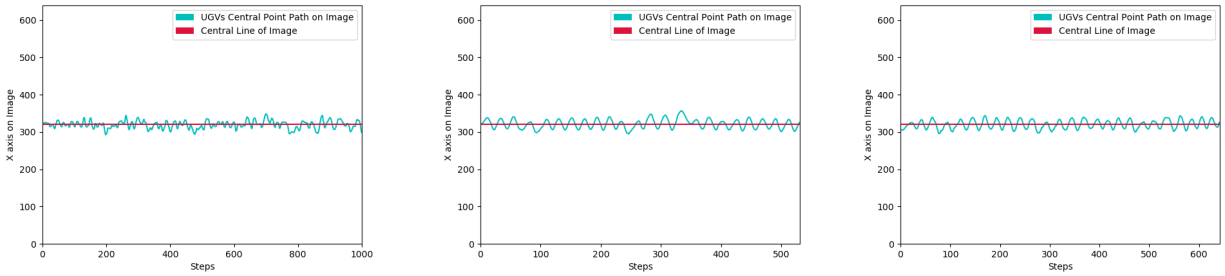
TABLE III: Average and Standard Deviations of Errors in All Testing Experiments

Experiment ID	Human		Scenario 1		Scenario 2	
	Distance Errors MSE $\mu \pm \sigma$	Radius Errors MSE $\mu \pm \sigma$	Distance Errors MSE $\mu \pm \sigma$	Radius Errors MSE $\mu \pm \sigma$	Distance Errors MSE $\mu \pm \sigma$	Radius Errors MSE $\mu \pm \sigma$
Fix-Altitude	15.6 $\pm$ 9.1	3.4 $\pm$ 3.3	16.7 $\pm$ 0.7	1 $\pm$ 0	15.6 $\pm$ 6.7	2.6 $\pm$ 1.6
Climb	14.6 $\pm$ 9.5	1.8 $\pm$ 1.3	14.9 $\pm$ 8.9	19.6 $\pm$ 8.7	17.7 $\pm$ 8.2	19.1 $\pm$ 8.2
Descend	15.9 $\pm$ 8.1	4.8 $\pm$ 4.6	24.4 $\pm$ 33.6	27.7 $\pm$ 19.7	16.7 $\pm$ 9.2	8.7 $\pm$ 6.8
Combined	14.2 $\pm$ 9	12.9 $\pm$ 21.9	15.7 $\pm$ 9.8	23.4 $\pm$ 12.9	12.6 $\pm$ 8.1	26.2 $\pm$ 19.1



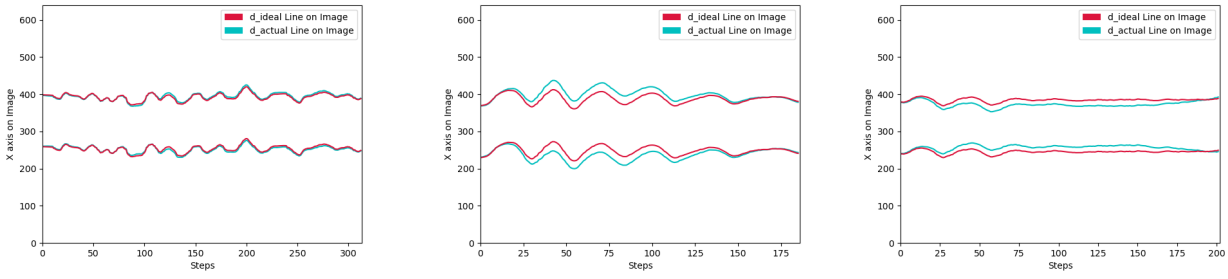
(a) Human Control (b) S1-Fix-Altitude (c) S2-Fix-Altitude

Fig. 6: The UGVs' Center and UAV Trajectories on Gazebo Model in Lateral-Movements-Fixed-Altitude maneuver



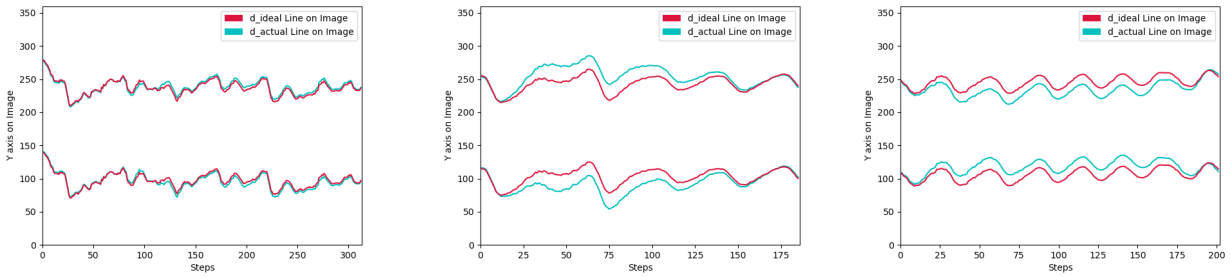
(a) Human Control (b) S1-Fix-Altitude (c) S2-Fix-Altitude

Fig. 7: The UGVx Trajectories in Lateral-Movements-Fixed-Altitude maneuver



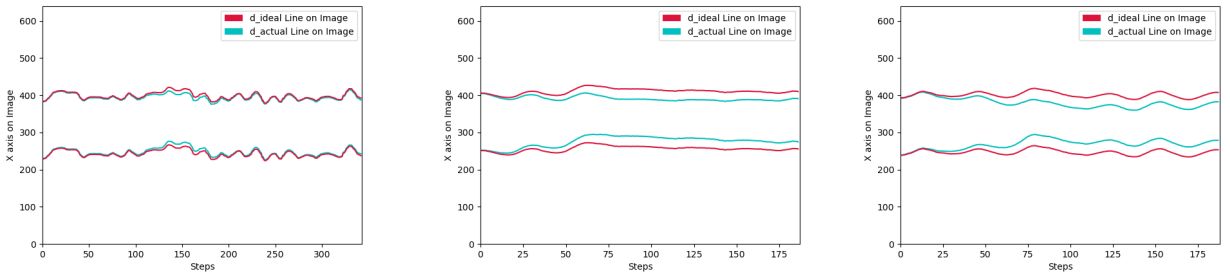
(a) Human Control (b) S1-Climb (c) S2-Climb

Fig. 8: The Ideal and Actual UGVs Circle Trajectories on Horizontical Image in Climb-Only maneuver



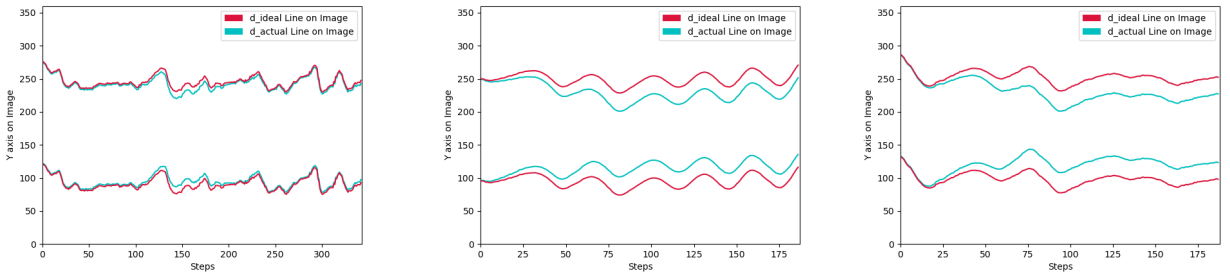
(a) Human Control (b) S1-Climb (c) S2-Climb

Fig. 9: The Ideal and Actual UGVs Circle Trajectories on Vertical Image in Climb-Only maneuver



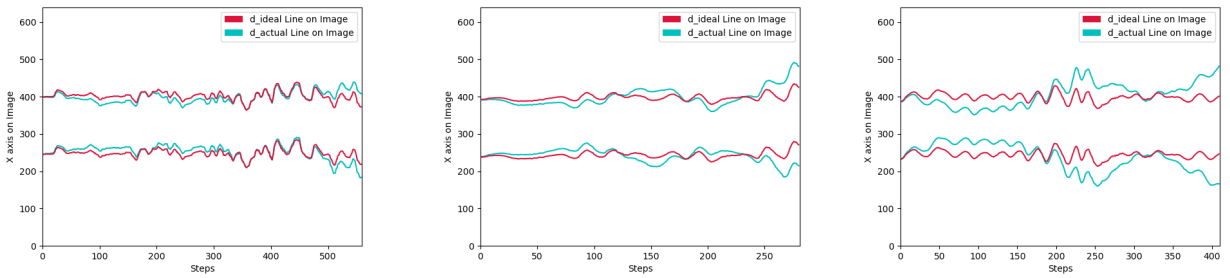
(a) Human Control (b) S1-Descend (c) S2-Descend

Fig. 10: The Ideal and Actual UGVs Circle Trajectories on Horizontical Image in Descend-Only maneuver



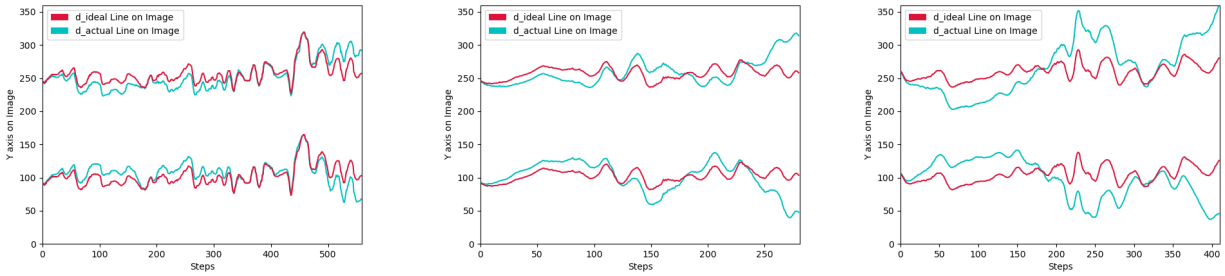
(a) Human Control (b) S1-Descend (c) S2-Descend

Fig. 11: The Ideal and Actual UGVs Circle Trajectories on Vertical Image in Descend-Only maneuver



(a) Human Control (b) S1-Combined (c) S2-Combined

Fig. 12: The Ideal and Actual UGVs Circle Trajectories on Horizontical Image in Lateral-Movements-With-Climb-Descend maneuver



(a) Human Control (b) S1-Combined (c) S2-Combined

Fig. 13: The Ideal and Actual UGVs Circle Trajectories on Vertical Image in Lateral-Movements-With-Climb-Descend maneuver

in the second scenario where only local information from the UAV's camera were used. Moreover, Our trained DNN produced behaviors generalizing performance equivalent to the human on other similar scenarios that were not seen during the training phase.

In our future work, we will use the trained neural network to seed the deep deterministic policy gradients algorithm [16] which uses an Actor-Critic deep reinforcement learning.

ACKNOWLEDGEMENT

The authors are indebted to Dr. David Aha for valuable comments on the manuscript.

REFERENCES

[1] J. Chen, X. Zhang, B. Xin, and H. Fang, "Coordination between unmanned aerial and ground vehicles: A taxonomy and optimization perspective," *IEEE Transactions on Cybernetics*, vol. 46, no. 4, pp. 959–972, April 2016.

[2] R. Hudjakov and M. Tamre, "Aerial imagery terrain classification for long-range autonomous navigation," in *Optomechatronic Technologies, 2009. ISOT 2009. International Symposium on*. IEEE, 2009, pp. 88–91.

[3] A. M. Khaleghi, D. Xu, S. Minaeian, M. Li, Y. Yuan, J. Liu, Y.-J. Son, C. Vo, and J.-M. Lien, "A DDDAMS-based UAV and UGV team formation approach for surveillance and crowd control," in *Proceedings of the 2014 Winter Simulation Conference*. IEEE Press, 2014, pp. 2907–2918.

[4] I. Kruijff-Korbayová, F. Colas, M. Gianni, F. Pirri, J. Greeff, K. Hindriks, M. Neerincx, P. Ögren, T. Svoboda, and R. Worst, "TRADR project: Long-term human-robot teaming for robot assisted disaster response," *KI-Künstliche Intelligenz*, vol. 29, no. 2, pp. 193–201, 2015.

[5] J. Y. Chen, "UAV-guided navigation for ground robot tele-operation in a military reconnaissance environment," *Ergonomics*, vol. 53, no. 8, pp. 940–950, 2010.

[6] A. M. Khaleghi, D. Xu, S. Minaeian, M. Li, Y. Yuan, J. Liu, Y.-J. Son, C. Vo, A. Mousavian, and J.-M. Lien, "A comparative study of control architectures in UAV/UGV-based surveillance system," in *IIE Annual Conference. Proceedings*. Institute of Industrial and Systems Engineers (IISE), 2014, p. 3455.

[7] S. Howitt and D. Richards, "The human machine interface for airborne control of UAVs," 2003.

[8] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, J. Kim, and S. Sastry, "Pursuit-evasion games with unmanned ground and aerial vehicles," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 2948–2955.

[9] M. Liu, C. Amato, E. P. Anesta, J. D. Griffith, and J. P. How, "Learning for decentralized control of multiagent systems in large, partially-observable stochastic environments." in *AAAI*, 2016, pp. 2523–2529.

[10] M. Trentini and B. Beckman, "Semi-autonomous UAV/UGV for dismounted urban operations," in *Proceedings of*, 2010, pp. 76 921C–76 921C.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[12] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2759–2764.

[13] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive UAV control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1765–1772.

[14] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 528–535.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[17] N. Koenig and A. Howard, "Gazebo-3D multiple robot simulator with dynamics," 2006.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[20] H. Huang and J. Sturm, "Tum simulator," *Ros package at http://wiki.ros. org/tum_simulator*, 2014.

[21] ClearpathRobotics., "ROS husky robot," *Ros package at http://wiki.ros.org/Robots/ Husky*, 2017.

[22] F. Chollet *et al.*, "Keras: Deep learning library for theano and tensorflow," *URL: https://keras. io/k*, 2015.