Enhancement in Novel State Representation Learning for Intelligent Navigation Systems

Geng LiangYu   
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384909@u.nus.edu

Han Dongchou Francis  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singaporee0385045@u.nus.edu

Tan Chin Gee  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384927@u.nus.edu

*Abstract*—This work evaluated the improvement gained from refining a novel algorithm used to address the simulation-to-real transfer (sim-real) problem in autonomous first-person view (FPV) drone navigation. A Residual Network (ResNet) model replaced the original feature encoder of the Cross-Modal Variational Auto-Encoder (CMVAE) algorithm used for solving the sim-real problem. Feature encoders with the ResNet architecture demonstrated better navigational performance compared with the original feature encoder. The system increases the benefits of using simulated environments for the development of intelligent vehicular navigation systems such as (1) savings on cost and time, (2) ease of collecting a large amount of annotated training data, and (3) flexibility in changing the training environment on the fly. The industry potential of the CMVAE approach for rapid development of intelligent robotics systems is apparent.

# INTRODUCTION

*A. Simulation-to-Real Transfer Problem*

The training of autonomous navigation system, as it is practiced now, relied heavily on data intensive methods such as deep learning and reinforcement learning [1] [2]. The data collection challenges of these methods are well-understood, and these problems can generally be addressed by using simulated environments [3] [4]. Collecting data using a simulator is a feasible way of collecting large quantity of data that otherwise would have been more difficult if collected manually. Simulated data, as compared with real-world data, are cheaper and easier to collect which makes it an ideal candidate for data collection.

However, the use of simulated data also poses unique challenges for the development of autonomous navigation system. The drawback of models trained on simulated data is arguably the decay of performance when models are deployed in the real world. This is known as the reality gap or simulation-to-real transfer (sim-real) problem [5] [6].

The crux of the sim-real problem is that a simulated environment, despite the best effort to make it as realistic to the real-world as it can be with current technology, is still not an actual copy of the real-world [7]. There are perceivable visual differences between simulated and real environments, and the complexity of forces and dynamics acting on objects in the real-world is simplified in simulation. Hence, simulated environment cannot fully model all aspects of reality. All these differences reduce the performance of simulation models deployed in the real-world.

Prior works had tried addressing the sim-real problem by concentrating on specific characteristics of the issue. These works typically fall under the approach of (1) creating better simulators that minimizes the differences between simulation and reality, or (2) making simulation models more robust to the effect of the sim-real problem. The first approach is driven by the idea that as the simulation becomes more accurate, so will the transition of simulation models to reality becomes smoother. The second approach abandons the idea of a fully accurate simulation, in favor of methods that aim to generalize simulation models to real-world deployment. The typical methods under this approach are domain randomization and domain adaption.

*B. Simulators improvement*

A focus on improving the fidelity of simulated environments has led to more visually realistic simulators, such as the Microsoft Aerial Informatics and Robotics Simulation (AirSim) [8] and CARLA [9]. This approach of developing ever more accurate simulators will continue unabated in the foreseeable future, bolstered by increasing computing power and advancement in hardware technology. Despite these expected advancements in simulators, it is likely that simulation models will still fail to properly generalize to the real-world due to the slightest of discrepancy between these two environments [10].

*C. Domain Randomization*

Domain randomization [11] [12] and domain adaptation [13] [14] [15] are methods that seek to reduce the impact of performance decay that is triggered by transferring simulation models to the real-world. Domain randomization takes place during the simulation training of models, where features that are non-relevant to the task, such as lighting condition and the relative position of elements, are changed randomly. Models are therefore exposed to a multitude of different environments where important aspects of the environment are kept unchanged. Hence, the sim-real problem is minimized in these models because they had learnt to ignore non-informative elements for the real-world task, which may confuse it, through randomization. The obvious shortcoming of the domain randomization approach is the greater efforts and costs required to train models in many different simulated environments.

*D. Domain Adaption*

Some proposed solutions viewed the incorporation of labeled real-world data as an answer to the sim-real problem. Domain adaption focuses on fine-tuning trained simulation models to bridge the reality gap, so that models can work well in the real-world. Labeled data collected from the real-world is often required when constructing the bridge that links simulated models to the real-world. There is ongoing effort to minimize the size of labeled data, and therefore preserve the benefits conferred by the training of models in a simulation. An example solution is the use of Generative Adversarial Network (GAN) [16] in combination with 93 labeled and 186 unlabeled images to successfully deploy simulation-trained visuomotor policies in the real-world [17]. Despite the efforts of these works, the reliance on labeled data does negates some of the benefits gained from using a simulator for model training. A more satisfactory approach that uses only simulation data is needed.

*E. Representation Learning Using Cross-Modal Variational Auto-Encoder*

A solution that eliminates the need for labeled real-world data uses Cross-Modal Variational Auto-Encoder (CMVAE) [18] for representational learning solely within a simulation environment [19]. The CMVAE feature encoder architecture differed from regular autoencoders by possessing only one encoder but having one decoder for each data modalities. Two or more data modalities are simultaneously trained, instead of only one data input typically seen in regular autoencoders. This work focused on autonomous first-person view (FPV) drone navigation, which is similar in nature to the task of drone racing that only rely on images from a single RGB camera for navigation.

The CMVAE’s approach takes the two different modalities of the (1) video-stream of the drone’s FPV camera and (2) poses of racing gates in the simulated environments, and learnt a representation of the drone’s state that has much fewer dimensions than the raw inputs of both modalities. These state representations can reconstruct the original inputs approximately despite its smaller size. This is termed as the concept of *autoencoding* in unsupervised representation learning [20].

These state representations learnt by the CMVAE’s encoder capture dense informatic latent variables that are most important for decision-making in autonomous navigation of drone through gates with unknown locations. A navigation policy trained with these latent variables retains high real-world performance and addresses the sim-real problem, because it has learnt to focus on relevant features of the real-world to guide its navigation and minimizes the influences of trivial differences between its training and deployment environments on its performance. This effect was attributed to the prevention of overfitting through the regularization brought about by forcing CMVAE to learn the latent variables for multiple data modalities.

After obtaining the state representation, the next step taken was imitation learning for training the navigation policy. Behavior cloning (BC) [21], which is a simplistic imitation learning algorithm, was used to learn the optimal path through the gates that was generated by a minimum jerk trajectory planner [22]. This completes the entire training, and the drone is now ready for real-world deployment.

# RELATED WORKS

*A. Drone Racing*

Drone racing is a sport in which human pilots fly their drone at high speed through racing gates, guided only by live video-stream from the drone’s FPV camera [23]. Pilots take these visual images and translate them into a series of visuomotor commands to navigate the racecourse at high speed.

There are numerous difficult technical challenges in developing autonomous racing drones, and of which this work will address the challenge of optimal real-world navigation of simulation model. Related works that tackled key technical challenges of autonomous drone navigation are presented.

*B. Vision-Based Drone Navigation*

Vision-based autonomous drone navigation is characterized by its primary use of visual sensors such as monocular cameras and stereo cameras [24]. Vision-based drone, like all mobile robots, has a typical architecture that comprises perception systems to detect surroundings, localization and mapping systems to know its states and the environment, path planning systems to chart the route, and motor control systems to move the robot [25]. Due to visual sensor’s ability to offer user a tremendous amount of information, it is not a simple task to extract features that provide useful positioning information to a drone.

Because of limitations to the payload of the drone, it can be impractical for some small drones to be outfitted with multiple sensors. Therefore, a more general approach of enhancing the drone’s environmental perception ability is needed, as opposed to obtaining better pose estimation by multiple sensor data fusion [26]. Monocular cameras are especially suited for applications where compactness and minimum weight are critical. In addition to that, lower price and flexible deployment make them a good option for drones.

*C. Computer Vision Algorithms*

The Residual Network (ResNet) is a state-of-the-art computer vision algorithm [27]. It has been widely incorporated in numerous works since it emerged as the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015 in image classification, detection, and localization [28] [29] [30]. ResNet solves the vanishing gradient problem through adding shortcut connection between layers, and thus this architecture permits the construction of very deep networks that helps to increase accuracy in image recognition tasks.

A ResNet with 18 layers architecture was utilized to build a visual classifier of forest as seen by a drone [29]. This classifier must decide the direction to move the drone while it is perceiving the forest in front of it. The resultant set of instructions serves to navigate the drone through a collision-free path in the forest.

*D. Representation Learning of State for Navigation Policy*

State representation learning focuses on extracting abstract and compact features from data in environment where the action of an agent affects said environment [31]. The resultant features need to be encoded with essential information required for the task, and to disregard irrelevant data. Training on a compact set of informationally rich features permits more efficient learning of navigation policies, compared with doing so using raw sensory data [32].

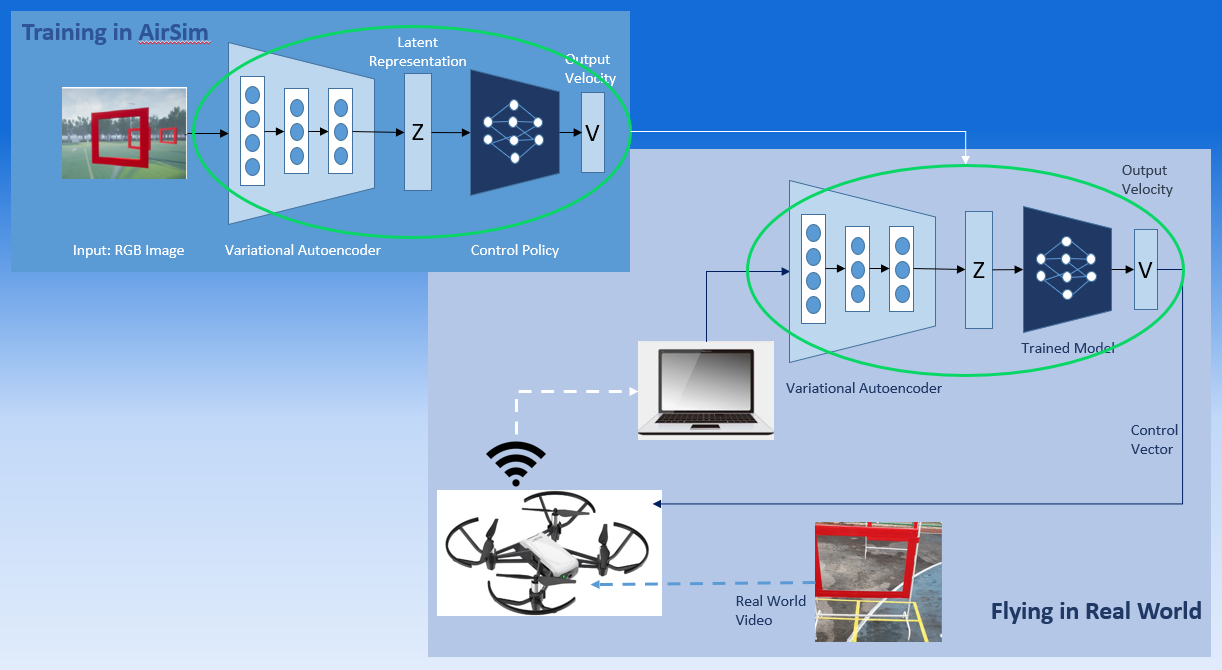
Adversarial networks were used for the learning of state representations [33]. The GAN-based state representation learning framework is an alternative approach to VAE-based ones. There is a perception in the literature of GAN-based framework being less suitable for state representation learning because they were slower and harder to train compared with their VAE counterparts [34] [35] [36] [37]. Other issues concerning the GAN-based approach were training instability and the requirement for having a priori knowledge of the data [38]. VAE-based framework, however, was more effective at extracting useful representations of data [39].

# SOLUTION

*A. Re-designing the Feature Encoder Architecture*

It is obvious that the use of CMVAE in training an autonomous drone navigation system does not require localization and an explicit map of the environment to navigate, and this approach also does not require any real-world training of the simulation model [19]. Examining the CMVAE architecture, this work came to the decision that the depth of the network had not been fully exploited. A modification of network depth can bring improved navigational performance. Due to its higher utility to the task of autonomous navigation model development, this work has chosen the CMVAE’s approach and examined the extent to which its feature encoder can be refined to yield higher performance during real-world drone deployment. The original BC model were also modified because it was relevant for this work’s aim of improving representational learning in autonomous FPV drone navigation.

The architecture of the entire system is display in Figure 1. A laptop is connected to the drone through wireless communication WIFI. The drone used its camera to capture video-feed of the racing gates in its field-of-view (FOV). Image inputs are fed into the VAE-based feature encoder. The resultant outputs are then fed to the trained navigation model in the laptop to generate a series of output velocity to autonomously navigate the drone through multiple red gates.

Figure 1. System Architecture

The goal of this work was to enhance the process of learning a latent state representation from different data modalities, as described by [19] in their two-steps approach of training a FPV drone to autonomously navigate a set of racing gates. Using a latent state representation to train navigation policy reduces the impact caused by sim-real transfer. The feature encoder is the key module responsible for generating the latent state representation, and thus this work focused on improving the performance of this module. Said enhancements were implemented by re-designing the feature encoder with state-of-the arts computer vision algorithm ResNet [27].

The re-designed feature encoder architecture and the original architecture are displayed in Table 1 and 2. A full pre-activation residual block was added to the architecture of the original feature encoder.

The differences in the number of parameters between the modified and original BC models are displayed in Table 3 and 4. Modifications made to the model were the addition of a dense layer and placement of dropouts to prevent overfitting.

*B. Microsoft Aerial Informatics & Robotics Simulation*

Training and evaluation of the re-designed architecture was conducted in the open-source AirSim environment, which is a simulator for experimenting with algorithms for autonomous vehicles such as drones and cars. The simulated environment uses the Unreal Engine built of AirSim to create a controllable training and testing platform, that is not only highly realistic in terms of the physics and visuals of the real-world, and also drastically reduces the costs and time involved in the collection of massive amount of training data required by deep learning and reinforcement learning. Keyboard shortcuts enable weather conditions such as fog and rain to be changed easily in the simulated environment.

AirSim is equipped with a generic drone model that when combined with its pre-loaded RGB camera quicken the pace of experimentation right from the start. Crashing the drone

repeatedly in AirSim does not incurs costs, unlike that of training with real drone. Images, data streams and vehicular control in AirSim are communicated programmatically through its Application Programming Interfaces (APIs).

Table 1. ResNet architecture

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Param # |
| Max\_pooling2d | Multiple | 0 |
| Batch\_normalization | Multiple | 128 |
| Batch\_normalization\_1 | Multiple | 128 |
| Batch\_normalization\_2 | Multiple | 128 |
| Batch\_normalization\_3 | Multiple | 256 |
| Batch\_normalization\_4 | Multiple | 256 |
| Batch\_normalization\_5 | Multiple | 512 |
| Batch\_normalization\_6 | Multiple | 512 |
| Batch\_normalization\_7 | Multiple | 1024 |
| Conv2d | Multiple | 2432 |
| Conv2d\_1 | Multiple | 9248 |
| Conv2d\_2 | Multiple | 9248 |
| Conv2d\_3 | Multiple | 1056 |
| Conv2d\_4 | Multiple | 18496 |
| Conv2d\_5 | Multiple | 36928 |
| Conv2d\_6 | Multiple | 2112 |
| Conv2d\_7 | Multiple | 73856 |
| Conv2d\_8 | Multiple | 147584 |
| Conv2d\_9 | Multiple | 8320 |
| Conv2d\_10 | Multiple | 295168 |
| Conv2d\_11 | Multiple | 590080 |
| Conv2d\_12 | Multiple | 33024 |
| Dense | Multiple | 16448 |
| Dense\_1 | Multiple | 2080 |
| Dense\_2 | Multiple | 660 |
| Total params: 1,249,684 | | |
| Trainable params: 1,248,212 | | |
| Non-trainable params: 1,472 | | |

Table 2. Original architecture

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Param # |
| Max\_pooling2d | Multiple | 0 |
| Batch\_normalization | Multiple | 128 |
| Batch\_normalization\_1 | Multiple | 128 |
| Batch\_normalization\_2 | Multiple | 128 |
| Batch\_normalization\_3 | Multiple | 256 |
| Batch\_normalization\_4 | Multiple | 256 |
| Batch\_normalization\_5 | Multiple | 512 |
| Conv2d | Multiple | 2432 |
| Conv2d\_1 | Multiple | 9248 |
| Conv2d\_2 | Multiple | 9248 |
| Conv2d\_3 | Multiple | 1056 |
| Conv2d\_4 | Multiple | 18496 |
| Conv2d\_5 | Multiple | 36928 |
| Conv2d\_6 | Multiple | 2112 |
| Conv2d\_7 | Multiple | 73856 |
| Conv2d\_8 | Multiple | 147584 |
| Conv2d\_9 | Multiple | 8320 |
| Dense | Multiple | 32832 |
| Dense\_1 | Multiple | 2080 |
| Dense\_2 | Multiple | 660 |
| Total params: 346,260 | | |
| Trainable params: 345,556 | | |
| Non-trainable params: 704 | | |

Table 3. Modified BC model

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Param # |
| Dense | Multiple | 2816 |
| Dropout | Multiple | 0 |
| Dense\_1 | Multiple | 16448 |
| Dropout\_1 | Multiple | 0 |
| Dense\_2 | Multiple | 1040 |
| Dropout\_2 | Multiple | 0 |
| Dense\_3 | Multiple | 68 |
| Dropout\_3 | Multiple | 0 |
| Total params: 20,372 | | |
| Trainable params: 20,372 | | |
| Non-trainable params: 0 | | |

Table 4. Original BC model

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Param # |
| Dense | Multiple | 2816 |
| Dense\_1 | Multiple | 4112 |
| Dense\_2 | Multiple | 68 |
| Total params: 6,996 | | |
| Trainable params: 6,996 | | |
| Non-trainable params: 0 | | |

*C. Tello Drone*

A budget-friendly quadrocopter drone, Tello by Ryze Tech Inc., served as the autonomous vehicular platform for testing the trained model in the real environment (Figure 2). This commercially available drone was chosen for its front mounted RGB camera that matched the configuration of the simulated drone and its ease of programming in Python. The drone’s in-built camera has an 82.6-degree FOV and can capture five-megapixel images and 720-pixel resolution videos at 30 frames per second [40]. The light-weigh of the drone helps it to survive multiple crashes.

The physical drone can be controlled by sending software developer kit (SDK) commands via a UDP connection to the flight controller of the drone. Set commands attempt to set new sub-parameter values such as the drone’s current speed. Using set commands, a user can remotely control the drone via four channels. The Tello SDK provides both, linear movement commands (takeoff, land, up, down, forward, backward, left, right) as well as angular movement commands (cw, ccw). Executing the cw command causes the drone to rotate clockwise and executing the ccw command causes the drone to rotate counterclockwise by a given degree [41].

Figure 2. Tello Drone



# METHOD

*A. State Representation Learning*

The CMVAE architecture forms the foundation of the feature encoder, and also enable it to generate the latent state representation from a combination of RGB images and data on the pose of racing gates relative to the frame of the drone. Data on the pose of racing gates was provided by the AirSim environment. Feature encoder samples latent state representation from a normal distribution.

Training procedure followed the same steps outlined in [18]. Dataset was comprised of 50K 64 x 64 images that were paired with their respective data of relative gate poses seen in these images. Training and validation were allocated 80% and 20% of the dataset. Training losses recorded included (i) MSE loss between actual and reconstructed images, and (ii) MSE loss for gate pose reconstruction.

*B. Imitation Learning*

Imitation learning proceeded exactly as described in [19]. A minimum jerk trajectory planner in combination with a pure-pursuit path tracking controller generated an expert trajectory. Then velocity commands that match the expert trajectory were paired with their corresponding FPV images to create the training dataset for imitation learning.

This work used the imitation learning technique, BC to train the navigation policy [21]. Training of the control network proceeded with trained weights of the feature encoder kept unchanged. A total of two navigation policies were trained for different feature encoder architectures: (i) original and (ii) ResNet. Feature encoder weights were not updated during the training of the navigation policy.

# RESULTS

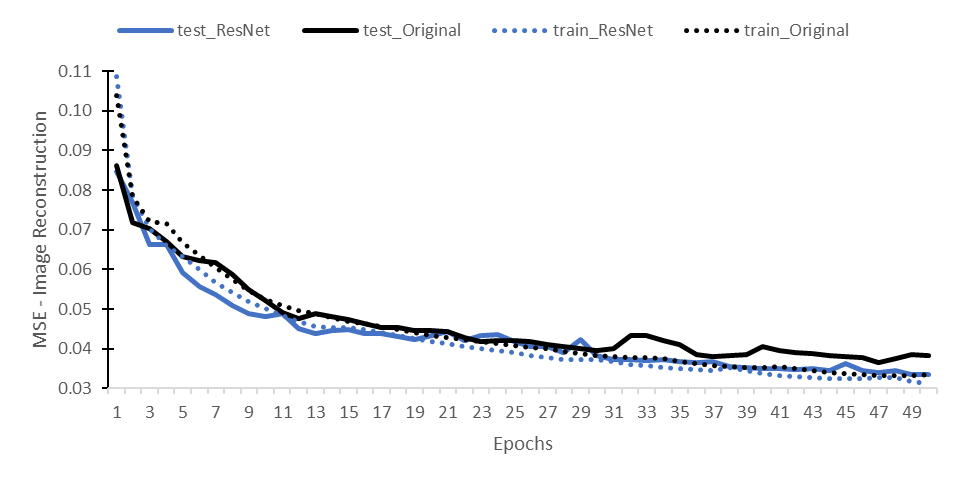
*A. Training & Test Loss*

Training loss as represented by the dashed lines reduced with each epoch. Solid lines display the testing loss. The ResNet and original architectures are shown in blue and black colored lines respectively for image reconstruction, gate pose and navigation policy (Figure 3, 4 & 5).

The test loss of ResNet architecture was lower than that of the original architecture for both image reconstruction and gate pose. Comparing between ResNet and the original architecture, the losses were 0.033 versus 0.038 for image reconstruction and 0.048 versus 0.053 for gate pose.

The modified BC model managed to achieve a test loss of 0.012 as compared with 0.026 for the original BC model. Additionally, they were fewer gate collisions in the assessments of drone trained with the modified BC model.

Figure 3. Training and test loss of image reconstruction



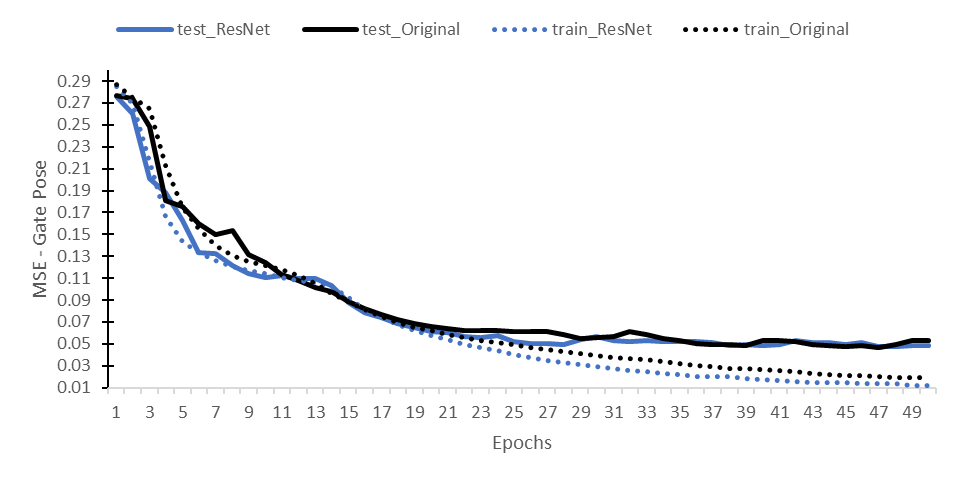
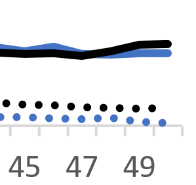
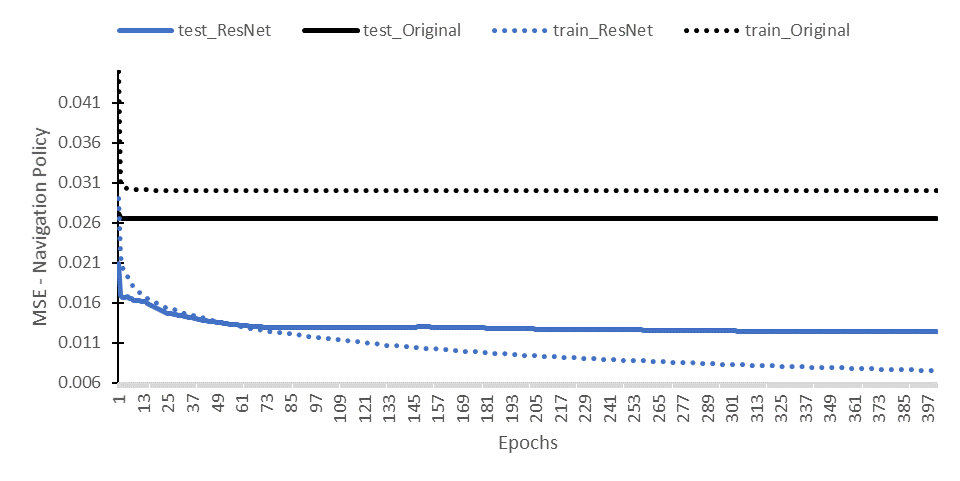
Figure 4. Training and test loss of gate pose

Figure 5. Training and test loss of navigation policy (BC)



*B. Navigation Performance in Simulated Environments*

Assessment of the drone’s navigation performance in the AirSim took place in the map of an open soccer field. Red colored squares positioned over the entire field served as gates for the drone to fly through (Figure 6).

Simulated flight assessments were conducted on navigation policies obtained with the ResNet and original feature encoders. The assessment put each navigation policy through a randomized 50m circular track of eight gates for 10 trials, and under two weather conditions: sunny and foggy (Figure 7 & 8).

The sunny weather condition was identical to the weather in the images used for training the feature encoders and the navigation policies. A count of gates that were successfully navigated serves as the performance metric. Each trial ends when the drone misses a gate or collided with a gate. Trials were also ended when the drone successfully flied through 30 consecutive gates.

Figure 6. AirSim Simulated Environment

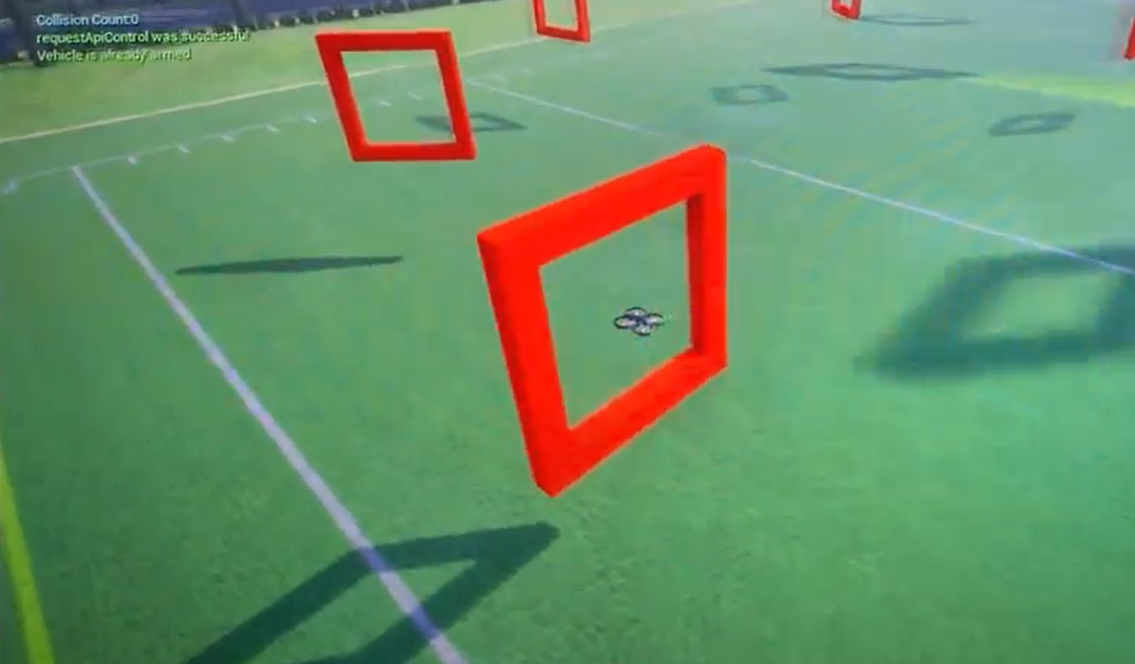


Figure 7. Sunny Environment



Figure 8. Foggy Environment



Table 5 shows the average number of successfully navigated gates in the assessment for different feature encoder architectures. The ResNet architecture performed better than the original regardless of the weather conditions. Thus, the encoder with the ResNet architecture demonstrated less degradation of performance compared with the original feature encoder. When the weather is sunny and thus have the same condition as during training, both architectures unsurprisingly showed higher navigation performance compared with the foggy condition.

Table 5. Mean number of gates passed in 10 trials

|  |  |  |
| --- | --- | --- |
| Weather Conditions | ResNet | Original |
| Sunny | 23.1 | 19.3 |
| Foggy | 9.3 | 7.7 |

*C. Real-World Flight with Non-Customized Drone*

A physical drone was used to test the transfer of simulation-trained navigation policy to real-world deployment. A Tello drone served as the platform that implemented the navigation policy in real-world setting. The drone flew from a spot that was two meters in front of the first gate and navigated itself through a series of three red gates positioned in a basketball court. Gate’s dimensions were 60 x 90 centimeters. Each gate was placed approximately 0.5 meter to the left of the gate in front of it. The distance between each gate was 2.2 meters (Figure 9).

Figure 9. Real-World Environment



Several factors affected the navigation assessment of the drone. The Tello drone used in this project is smaller, lighter, has less technologically sophisticated camera, and was not outfitted with on-board computing capability compared to the custom-built drone mentioned by [19]. The drone weights 80 grams and has a small body size of 15.2 x 15.2 x 3.3 centimeters. Windy conditions easily blew the light drone off course. The drone’s in-built camera has an 82.6-degree FOV. This narrow FOV made it challenging for the drone to perceive gates at the peripheral of its vision. To counter this hardware issue, the gates were placed closer together for the assessment.

A few steps were taken to adapt the less technologically powerful Tello drone for real-world flight. A set of intermediary APIs was created so that the set of commands coming from the laptop can communicate with the Tello drone APIs for navigation. Adjustments were made to reduce the frequency of camera image updates and the frequency of issuing commands to the drone. Scaling factors were also implemented to adjust the sensitivity of the drone’s navigational control.

Despite the lower technical capabilities of the drone, it has managed to successfully navigate the gates using only video-feeds from its front-mounted camera in combination with the navigation policy.

# DISCUSSION & CONCLUSION

*A. Improved Navigational Performance via Deeper Network*

This work aims to correct a weakness in the novel CMVAE approach used to reduce the sim-real problem [19]. The solution was to increase the network depth of the original CMVAE architecture used for the feature encoder. Drone with feature encoder modified with the re-designed ResNet architecture demonstrated improved navigational performance, compared with the original feature encoder.

In simulation, the modified drone flew through four gates more than drone with the original architecture. This improved performance was observed in assessment condition that was like the training environment, but also in foggy condition that added noises to the drone’s video-feed.

The results of this work also demonstrated that an off-the-shelf inexpensive drone can successfully implement a state-of-the art autonomous navigation system. These improvements strongly suggested that this work has successfully extract more navigational efficiency from the feature encoder that might have been initially overlooked.

*B. Implications*

This work demonstrated a potential third approach to make simulation models more robust to the effect of the sim-real problem, and that also addresses the shortcomings of existing methods. Getting rid of these shortcomings meant that there are increased benefits of using simulated environments for the development of intelligent vehicular navigation systems such as (1) savings on cost and time, (2) ease of collecting a large amount of annotated training data, and (3) flexibility in changing the training environment on the fly.

Using CMVAE to jointly convert two or more simulated sensory data to highly informatic representation variables with much lower dimensionality is a departure from the typical practices of domain randomization and domain adaptation.

Domain randomization achieves its robustness to real-world performance decay by training model in many different simulated environments. Randomizing trivial aspects of the simulated environment to train model to ignore them is both time-consuming and costly.

In contrast, the CMVAE approach has no need for this type of repeated trainings. The use of multi-modal data helps to regularize weights in the simulation-trained model. Every source of data constrains the parameters and thus minimizing overfitting of the model to the simulated environment.

Domain adaptation espouses the use of models trained in physical environment to fine-tune parameters of simulation-trained models. This is obviously counter-productive by having to spent extra efforts to collect non-simulation data after obtaining the simulation-trained model. This work did not have to collect additional real-world data for the drone to autonomously navigate in the real-world.

*C. Limitations*

Given the tight timeline given for this project, this work opted to focus on a single state-of-the-art computer vision algorithm. Greater navigational performance might have been achieved with the use of other architectures. However, this was a trade-off made to balance between the ambitious goals of re-designing the feature encoder with a comprehensive list of state-of-the-art computer vision algorithm against other competing focus of the project.

*D. Future Studies*

Future studies might choose to address the limited number of computer vision algorithms investigated in this work by incorporating more frameworks in subsequent studies. Other potentially fruitful research directions include examining the application of the CMVAE approach to the development of models for other robotics tasks, merging other sensory modalities such as auditory and tactile for state representation learning, and to study the incorporation of GAN-based model as an alternative framework to the CMVAE.

The autonomous navigation model is a good industrial showcase. Not only did the project make an academic contribution to solving the sim-real problem, this work is also an initial effort in exploring the industrial feasibility of using the CMVAE approach to quickly and cheaply develop models useful for robotics tasks in simulated environment. This project has successful achieved its intended goals.

##### References

1. O. Walker, F. Vanegas, F. Gonzalez, and S. Koenig, “A deep reinforcement learning framework for UAV navigation in indoor environments,” in *IEEE Aerospace Conference Proceedings*, 2019, pp. 1-14.
2. C. Wang, J. Wang, X. Zhang, and X. Zhang, “Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning,” in *Proc. IEEE Global Conference on Signal and Information Processing,* 2017, pp.858-862.
3. K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, “Generalization through simulation: Integrating simulated and real data inot deep reinforcement learning for vision-based autonomous flight,” in *IEEE International Conference on Robotics and Automation*, 2019, pp.6008-6014.
4. C. Sampedro, A. Rodriguez-Ramos, I. Gil, L. Mejias, and P. Campoy, “Image-based visual servoing controller for multirotor aerial robots using deep reinforcement learning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp.979-986.
5. N. Jakobi, P. Husbands, and I. Harvey, “Noise and the reality gap: The use of simulation in evolutionary robotics,” in F. Moran, A. Moreno, J. Merelo, and P. Chacon (Eds.), *Advances in artifical life: Proceedings of the Third European Conference on Artifical Life*. Berlin: Springer-Verlag, pp.704-720, 1995.
6. S. James, A. J. Davision, and E. Johns, “Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task,” in *Conference on Robot Learning*, 2017, pp.334-343.
7. M. Neunert, T. Boaventura, and J. Buchli, “Why off-the-shelf physics simulators fail in evaluating feedback controller performance: A case study for quadrupedal robots,” in *Advances in Cooperative Robotics*, World Scientific, 2017, pp.464-472.
8. S. Shah, D. Dey, C. Lovett, and A. Kapoor. “AirSim: High-fidelity visual and physical simulation for autonomous vehicles,” in *Field and Service Robotics*, Springer, 2018, pp.621-635.
9. A. Dosovitskiy, G. Ros, F. Codvilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Confernce on Robot Learning*, 2017, pp.1-16.
10. K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, “Quantifying generalization in reinforcement learning,” in *Proceedings of the International Conference on Machine Learning*, 2018, pp.1282-1289.
11. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *International Conference on Intelligent Robots and Systems*, 2017, pp.23-30.
12. A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, “Deep drone racing: From simulation to reality with domain randomization,” *arXiv preprint arXiv: 1905.09727*, 2019.
13. G. Csurka. “Domain adaptation for visual applications: A comprehensive survey,” *arXiv preprint arXiv: 1702.05374*, 2017.
14. K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, “Using simulation and domain adaptation to improve efficiency of deep robotic grasping,” *arXiv preprint arXiv: 1709.07857*, 2017.
15. F. Golemo, A. A. Taiga, P. Y. Oudeyer, and A. Courville, “Sim-to-real transfer with neural-augmented robot simulation,” in *Conference on Robot Learning*, 2018, pp.817-828.
16. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editor, *Advances in Neural Information Processing Systems 27*, pp.2672-2680, 2014.
17. F. Zhang, J. Leitner, Z. Ge, M. Milford, P. Corke, “Adversarial discriminative sim-to-real transfer of visuo-motor policies,” in *International Journal of Robot Research,* 2019, pp.1229-1245.
18. A. Spurr, J. Song, S. Park, and O. Hilliges, “Cross-modal deep variational hand pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 89-98.
19. R. Bonatti, R. Madaan, V. Vineet, S. Scherer, and A. Kapoor, “Learning visuomotor policies for aerial navigation using cross-modal representations,” *arXiv preprint arXiv: 1909.06993v2*, 2020.
20. M. Tachannen, O. Bachem, and M. Lucic, “Recent advances in autoencoder-based representation learning,” *arXiv preprint arXiv: 1812.05069*, 2018.
21. T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, “An algorithmic perspective on imitation learning,” in *Foundations and Trends in Robotics*, 2018, pp.1-179.
22. C. Richter, A. Bry, and N. Roy, “Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments,” in *Robotics Research*, Springer, 2016, pp.649-666.
23. E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, “Deep drone racing: Learning agile flight in dynamic environments,” *arXiv preprint arXiv:1806.08548*, 2018.
24. Y. Lu, Z. Xue, G. S. Xia, and L. Zhang, “A survey on vision-based UAV navigation,” *Geo-spatial Information Science*, 2018, pp.21-32.
25. C. Badue, R. Guidolini, R. Vivacqua Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. Paixão, F. Mutz, L. Veronese, T. Oliveira-Santos, and A. F. De Souza, ‘‘Self-driving cars: A survey,’’ *arXiv preprint arXiv:1901.04407*, 2019.
26. S. Shen, Y. Mulgaonkar, N. Micheal, and V. Kumar, “Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV,” in *IEEE International Conference on Robotics and Automation*, 2014, pp.4974-4981.
27. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *arXiv preprint arXiv:1512.03385*, 2015.
28. M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, “State-of-the-art survey on deep learning theory and architectures,” *Electron*, 2019, p.292.
29. L. Chen, W. Wang, and J. Zhu, “Learning transferable UAV for forest visual perception,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, AAAI Press, 2018, pp.4883–4889.
30. L. Yu, C. Luo, X. Yu, X. Jiang, E. Yang, C. Luo, and P. Ren, “Deep learning for vision-based micro aerial vehicle autonomous landing,” in International Journal of Micro Air Vehicles, 2018, pp.171-185.
31. T. Lesort, N. Diaz-Rodriguez, J. -F. Goudou, and D. Filliat., “State representation learning for control: An overview,” *arXiv preprint arXiv: 1802.04181*, 2018.
32. J. Munk, J. Kober, and R. Babuska, “Learning state representation for deep actor-critic control,” in *Proceedings of the 55th Conference on Decision and Control (CDC)*, 2018, pp.4667-4673.
33. X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2016, pp.2172-2180.
34. Z. Lin, K. K. Thekumparampil, G. Fanti, and S. Oh, “InfoGAN-CR and ModelCentrality: Self-supervised model training and selection for disentangling GANs,” *arXiv preprint arXiv: 1906.06034v3*, 2019.
35. Y. Wu, Y. Burda, R. Salakhutdinov, and R. B. Grosse, “On the quantitative analysis of decoder-based generative models,” *arXiv preprint arXiv: 1611.04273*, 2017.
36. T. Q. Chen, X. Li, R. b. Grosse, and D. K. Duvenaud, “Isolating sources of disentanglement in variational autoencoders,” in *Advances in Neural Information Processing Systems*, 2018, pp.2615-2625.
37. H. Kim, and A. Mnih, “Disentangling by factorising,” in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp.2649-2658.
38. I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, “Beta-VAE: Learning basic visual concepts with a constrained variational framework,” in *International Conference o Learning Representations*, 2017.
39. W. Lee, D. Kim, S. Hong, and H. Lee, “High-fidelity synthesis with disentangled representation,” *arXiv preprint arXiv: 2001.04296*, 2020.
40. Shenzhen RYZE Tech Co., Ltd. and Da-Jiang Innovations Science and Technology Co., Ltd. Tello sdk (version 2.0). https://dl-cdn.ryzerobotics. com/downloads/Tello/Tello\%20SDK\%202.0\%20User\%20Guide.pdf, 2018. [Last Accessed: 22. June 2020]. pages 11, 13, 14, 15.
41. ryzerobotics.com, “Tello SDK”, [Online] Available: https://dlcdn.ryzerobotics.com/downloads/tello/20180910/Tello%20SDK%20 Documentation%20EN\_1.3.pdf.