

DoShiCo challenge: Domain Shift in Control prediction

Klaas Kelchtermans* and Tinne Tuytelaars*

Abstract—Training deep neural network policies end-to-end for real-world applications so far requires big demonstration datasets in the real world or big sets consisting of a large variety of realistic and closely related 3D CAD models. These real or virtual data should, moreover, have very similar characteristics to the conditions expected at test time. These stringent requirements and the time consuming data collection processes that they entail, are currently the most important impediment that keeps deep reinforcement learning from being deployed in real-world applications. Therefore, in this work we advocate an alternative approach, where instead of avoiding any domain shift by carefully selecting the training data, the goal is to learn a policy that can cope with it. To this end, we propose the DoShiCo challenge: to train a model in very basic synthetic environments, far from realistic, in a way that it can be applied in more realistic environments as well as take the control decisions on real-world data. In particular, we focus on the task of collision avoidance for drones. We created a set of simulated environments that can be used as benchmark and implemented a baseline method, exploiting depth prediction as an auxiliary task to help overcome the domain shift. Even though the policy is trained in very basic environments, it can learn to fly without collisions in a very different realistic simulated environment. Of course several benchmarks for reinforcement learning already exist - but they never include a large domain shift. On the other hand, several benchmarks in computer vision focus on the domain shift, but they take the form of a static datasets instead of simulated environments. In this work we claim that it is crucial to take the two challenges together in one benchmark.

I. INTRODUCTION

Reinforcement Learning (RL) is gaining more and more interest due to the strong representational power of deep neural networks (DNNs) [1], [2], [3], [4], [5]. Where RL used to be a field of low dimensional discrete state-spaces to learn policies succeeding at basic games like tic tac toe [6], DNN policies have shown to perform more and more complex tasks on high dimensional [4] and even partially observable state-spaces like first-person views [7] and autonomous driving [8]. It is however remarkable that the number of success stories in the form of real-world applications remains low (but see [9], [10] for notable exceptions). In this work we want to take an important step towards the use of deep reinforcement learning (DRL) algorithms in real-world applications by explaining current impediments as well as defining a novel benchmark to stimulate research in this direction.

Using RL algorithms to train DNN policies in the real world is highly impractical due to several reasons. The first

and most obvious one is the fact that most algorithms are to be trained on-policy as they are based on trial-and-error. For tasks like autonomous navigation, this means that a suboptimal policy will be steering the robot or drone, leading to possibly fatal crashes and rather unhappy researchers.

A second reason is the large amount of training experiences that is required. Even if your algorithm could train from an offline dataset, it would require numerous hours of demonstrating the same task over and over again. In [11], it is shown that collecting a dataset of 11,500 crashes of a drone allows training a policy to avoid obstacles, though it is clear that this strategy is not an option for many applications.

Moreover, there are other reasons, like the exploitation of parallelism, the possibility to reset the robot to a certain state and facilitating the reproduction of results, that make training DNN policies most feasible in simulated environments.

This, however, introduces a novel problem when testing the policy in the real world. Namely, the simulation and the real world will never look or act exactly alike. This is referred to as a *domain shift* between the input at training and test time. There have been several attempts to deal with this issue, though none of them seems to head in a promising direction: researchers have tried to make the simulated environment photo-realistic [12], or to introduce such a large variance over simulated environments in order to generalize to the real world [9], or lastly to augment the training data with domain shift techniques so it looks more and more like the real world [13]. These techniques seem to come with a lot of overhead if we want to apply them to more specific tasks, e.g. autonomous surveillance of a particular building site or autonomous inspection of a windmill. Instead of putting effort in data collection or in building simulated environments for each specific task, i.e. trying to minimize the discrepancy between train and test conditions, it is worth exploring alternative strategies, that can effectively cope with this domain shift.

In computer vision, several benchmarks exist that focus on the domain shift for different tasks like image segmentation and depth prediction [14], [15], [16]. We argue however that for the setting of control prediction or policy training, it is crucial to evaluate and preferably train in an online fashion. This is due to the non-i.i.d. (independent and identically distributed) nature of sequential decision processes. If a mistake is made at a certain point in time, a compound error over time will lead to very different outcomes. In that sense, evaluating solely on a static dataset can never fully represent the on-policy performance. In this work we propose a challenge in the form of train and test environments instead

*The authors are with KU Leuven, ESAT-PSI, imec, Belgium. first-name.lastname@esat.kuleuven.be

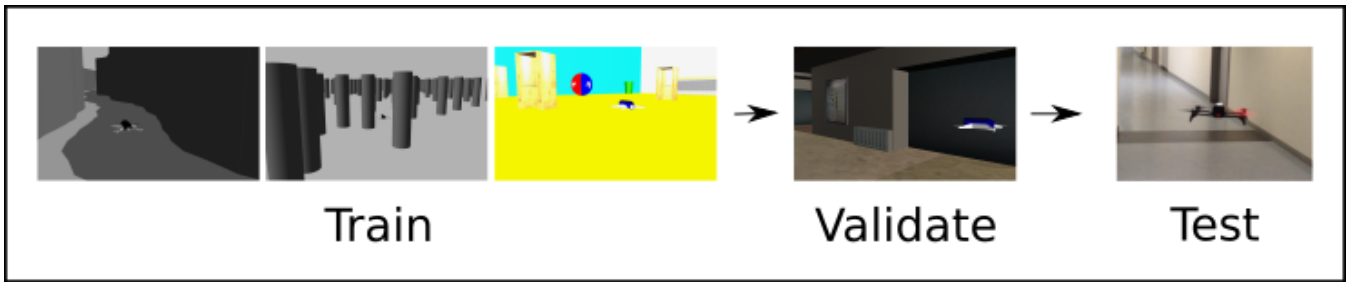


Fig. 1. DoShiCo Challenge: train a neural control network end-to-end on three sets of very basic simulated environments (train in "Canyon", "Forest" and "Sandbox"), so that it can fly in a more realistic environment (validate in "ESAT") as well as take control decisions on real-world data (test on the "Almost-Collision dataset")

of static data.

On the other hand within the RL community, benchmarks exist that compare different DRL training algorithms like THOR [17], VIZDOOM [18], OpenAI [19], CARLA [20], TORCS [21], Udacity [8], etc. However, they never incorporate a large domain shift. To the knowledge of the authors, there does not exist a benchmark that combines the two challenges: solving a Partially-Observable Markov Decision Process (POMDP) by training a DNN policy end-to-end, together with the domain adaptation to the real-world. By introducing this challenge of training DNN policies end-to-end, we believe we can boost this research field, taking DRL algorithms to the next step towards real-world applications.

This requires policies that can generalize to previously unseen conditions, perform well over a wide action range while at the same time being insensitive to irrelevant differences, all learned in an end-to-end fashion. In the case of obstacle avoidance, this could be realized by exploiting visual cues such as relative pose and depth and by learning invariance to color or texture. Note that the question whether this policy is trained in a supervised fashion or in a reinforced fashion remains open, i.e. the benchmark can be used for both.

Building on the above observations, we make two contributions. First, we propose a domain shift challenge for control prediction (DoShiCo) that can serve as a benchmark for comparing different training strategies. Given three sets of very basic simulated environments, instances of which can be generated randomly during training, the goal is to train a DNN policy end-to-end for the task of collision avoidance so that it can generalize to previously unseen and more realistic conditions, as captured by our synthetic yet more realistic validation environment and real-world test data (see Figure 1). To avoid the issue with online control in a real-world setting, a classification task using a dataset of 'almost collisions' is provided as a proxy for flying in the real world.

Second, we evaluate a two baselines in this setting, building on a MobileNet [22] pretrained on ImageNet [23]. In an attempt to train the control end-to-end without losing the robustness to varying imaging conditions and without overfitting to the basic simulated environments, we demonstrate the benefit of an auxiliary task [24]. The use of auxiliary tasks helps the extracted features to focus on the information that is relevant for the task which makes the learning less prone to

fitting toward irrelevant features only visible in the training environments. In particular, we demonstrate how auxiliary depth prediction can reduce the impact of the domain shift and we show that our model succeeds in flying in the more realistic simulated validation world although it was trained solely on a mix of very basic environments.

In order to use DoShiCo as a benchmark, we integrated the full setting of ROS, Gazebo and Tensorflow in a Docker and Singularity image. All the code, 3D environments and the best trained checkpoints of the model are publicly available to ensure reproducibility of the results¹.

The remainder of this paper is organized as follows. First, in section II, we start with a short background, defining monocular obstacle avoidance in an RL setting. Second, in section III, we describe the related work. Next, we give the details of the DoShiCo challenge (Section IV) and details on our baseline models (Section V). In Section VI, the experimental results are discussed. Section VII concludes the paper together with a discussion.

II. BACKGROUND

In the general RL setting, an agent interacts with the environment according to a policy, π , which maps the currently observed state s_t to an action $a_t = \pi(s_t)$. The environment brings the agent to a next state s_{t+1} according to the dynamics of the environment $T(s_{t+1}|s_t, a_t)$ with a corresponding reward r_{t+1} . The goal is then to find good parameters for this policy, maximizing the cumulative reward in the longterm [25].

Most of the RL algorithms are based on the Markov property. In this case, the information provided in the state should be enough to select an optimal action. Such problems are referred to as Markov Decision Processes (MDPs). In most applications, however, this assumption does not hold. In those cases, the process is seen as an MDP but the agents observation is not the full state. These problems are referred to as partially observable MDPs (POMDP).

As mentioned in the intro, in state-of-the-art RL methods, states have shifted from low dimensional game specific variables to high dimensional raw images, and further to partially observable states for example in first-person views.

¹ [kkelchte.github.io/doshico](https://github.com/kkelchte/doshico)

In the DoShiCo challenge the observed state is represented by the current view of a drone. The action is the applied yaw turn while flying at a fixed speed.

III. RELATED WORK

Learning to control

When it comes to autonomous navigation directly from high dimensional camera input, a first family of solutions relies on geometric techniques for simultaneously mapping and localizing the agent (SLAM). Besides the extra computational power, these algorithms often suffer from a lack of features to track [26]. In a similar spirit yet more robust, Gupta et al. [27] have recently demonstrated how a joint neural architecture, called a CMP (Cognitive Mapping and Planning), can effectively learn to map and plan jointly trained in an end-to-end fashion.

More related to our work, yet avoiding the end-to-end complexity and bypassing the domain shift issue, there are some works that rely on depth estimation as an intermediate step. Michels et al. [1] predict depth and then train a controller in simulation with reinforcement learning. In [28] depth is estimated from single images using a CNN, as in [29], and then used to avoid obstacles with a behavior arbitration algorithm [30]. For the task of obstacle avoidance this might be a valid solution. There are however different and more complex tasks for which depth estimation does not provide enough information. In those settings end-to-end training is required in order to extract higher level features.

Moreover, Levine et al. [3] demonstrated that training end-to-end results in more stable and efficient learning. Pomerleau [31] successfully trained a single layer network end-to-end with apprenticeship learning for the task of road following. Lecun et al. [32] trained a CNN to predict the steering angle of a small car based on stereo input from an offline dataset. In more recent work, Ross et al. [33] train an SVM (Support Vector Machine) iteratively with imitation learning in a forest. Giusti et al. [34] train a deep neural network end-to-end for following trails in a forest from a large offline dataset gathered manually.

These are all examples demonstrating the promising path of end-to-end image-based control prediction networks.

Dealing with the virtual-real domain shift

All control models previously mentioned are trained in an environment that is close to the test environment. However, as indicated above, for many real-world applications this is unfeasible. There have been some preliminary attempts to cope for instance with different weather conditions [35], [20]. In the recent work [36] of Li et al., they succeed at generalizing over different game parameters in a Cart-Pole game (cart mass and pole length). The step from simulation to the real world entails however a much bigger domain shift. In computer vision, several methods for domain adaptation have been proposed (see [37] for a recent survey), but mostly in rather artificial setups and, to the best of our knowledge, never in the context of control prediction networks.

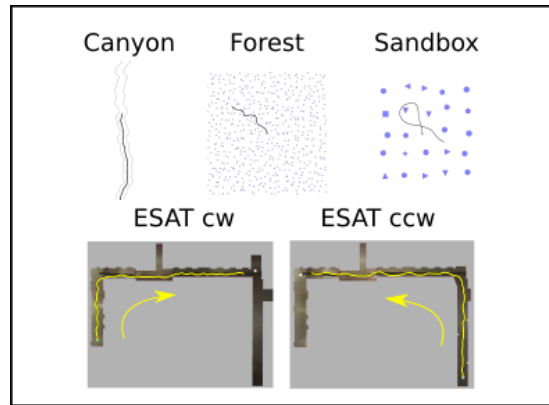


Fig. 2. Top-down view of the training (top) and validation (bottom) environments with an example trajectory.

Finally, the idea of using auxiliary tasks for our baseline model stems from Multi-Task Learning (MTL). In computer vision, MTL has been demonstrated to improve the performance of one task by sharing the network with other tasks – for instance object detection together with classification and segmentation [38]. Mirowski et al. [39] demonstrate the use of depth prediction as an auxiliary task in order to learn a deep neural agent to navigate and localize itself in a simulated maze. They demonstrate a positive impact on data efficiency and performance, however they did not research the domain shift.

IV. THE DOShiCo CHALLENGE

The DoShiCo challenge contains of training a DNN policy end-to-end for the task of obstacle avoidance with drones. Based on the high-dimensional RGB input from the current view of the drone, the policy needs to predict the action, in this case the applied yaw turn while flying at a fixed speed. The policy is trained in very basic simulated environments, chosen specifically to train features relevant for obstacle avoidance. After this, the policy is evaluated *online* in a very different and more realistic simulated environment and *offline* on a small dataset of real-world videos.

The DNN agent should be initialized from scratch or with imagenet-pretrained weights. Pretraining the neural network with intermediate representations, like depth maps, separates the two challenges and is therefore not allowed. The policy should be trained end-to-end. The challenge does not restrict the training time or the architecture of the network.

A Mix of Basic Training Environments: Canyons, Forests and Sandboxes

Snapshots of the type of environments are shown in Figure 1. Top-down views can be seen in Figure 2. One type of training environments is a canyon that bends randomly, inspired by [40]. The goal is to fly more than 45m through the canyon at a constant speed of 1.3m/s without colliding with the walls. From the same work, we copied the idea of a forest with cylinders placed on random spots in which the goal is to cross 45m without collision. The third environment

is called the Sandbox. It is a box of 20x20m with walls in varying colors and a number of different objects spread around. We picked 13 basic objects with different shapes found on the Gazebo model server [41]. In the sandbox the agent needs to get further than 7m from the starting position without collision. In each environment the agent is spawned at location (0, 0). The environments are made on the fly.

The three training environments intuitively help to learn a different type of behavior. In the canyon, the agent learns to focus on perspective lines relevant for wall- and corridor-following. In the forests, the agent learns to avoid moving vertical lines relevant for general obstacle avoidance. The sandboxes ensure a healthy invariance towards a variety of shapes and visual features.

More realistic validation environment: ESAT

The network is evaluated *online* in the ESAT environment. We build a lookalike model of our corridor at the electrical engineering department ESAT. It is important to test online in order to know how well the agent has solved the POMDP. The policy is tested through the ESAT corridor in 2 directions. Successful example flights are depicted in the top-down view of Figure 2.

Real-world test data: Almost-Collision dataset

Collision avoidance on a real drone performed by different policies is hard to compare correctly. Real-world experiments are influenced by many external factors such as battery state, propeller state, on board electronics, etc.. A drone might crash in one test, deteriorating all consecutive flights.

Alternatively using an imitation loss from a demonstration flight by a pilot is biased towards the specific flying behavior of the pilot which is unfair for comparing one policy to another.

In order to compare collision avoidance in a fair and quantitative way on real-world data, we made a small dataset containing sequences of images of situations in which only one control is suitable: straight, left or right turn. We make sure that collision is very nearby in all trajectories, without actually crashing. This conveniently resulted in the Almost-Collision dataset.

We recorded data on seven different locations that differ a lot in visible features. Snapshots are shown in Figure 3. The trajectories are tagged with different visible cues. These cues are a type of feature specific to this type of collision: perspective lines, vertical lines and strange shapes.

The image sequences are around 3 to 5 seconds at 20fps. They are labeled with the control required to avoid collision. Over the seven locations a total of 25 trajectories are collected with an equal amount of left and right target controls with the exception of one trajectory with straight as target control. The total size is around 1600 frames. For classification, the predicted angular velocity in yaw is discretized with thresholds ± 0.3 for left, straight and right.

Evaluation

In order to evaluate a model on the DoShiCo challenge it is important to see first how well the model has solved

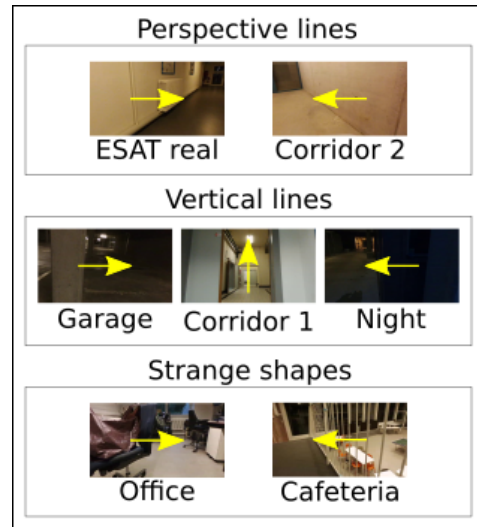


Fig. 3. Snapshots from the seven different locations of the Almost-Collision dataset. The yellow arrow indicates the target direction. Different trajectories have different visual cues to indicate the approaching collision.

the POMDP in the three training environments online. The second evaluation is also online in the ESAT environment for which it needs to cope with a primal domain shift. This second criteria is the most important one as it combines the difficulty of a domain shift together with the on-policy POMDP setting. The final criteria is an indication for the use of this agent in a real-world obstacle avoidance scenario.

During our experiments we encountered very high variance in the online results of policies trained with different seeding. This made it hard to compare different setups as well as hyperparameters. Jaderberg et al. [24] compare performances of the top 3 policies picked from 50 policies trained with different hyperparameters. We found that plotting the performance expressed as a percentage of the ranked population allows a proper comparison.

V. BASELINE MODEL

Here, we describe our training strategy as well as the architectures of the models used in our experiments. Note that the DoShiCo challenge does not provide any restrictions on how the policy is trained.

Training strategy

Our baseline model is trained in a straightforward imitation learning way, namely behavioral cloning. This means that the data is collected by flying a number of times through the simulated training environments. This was automated with a heuristic based on groundtruth depth provided by the simulated environment. The policy is trained on this data in a supervised fashion, minimizing the difference between the estimated control and the control applied by the heuristic. The heuristic collects data of 100 flights in each of the three types of training environments.

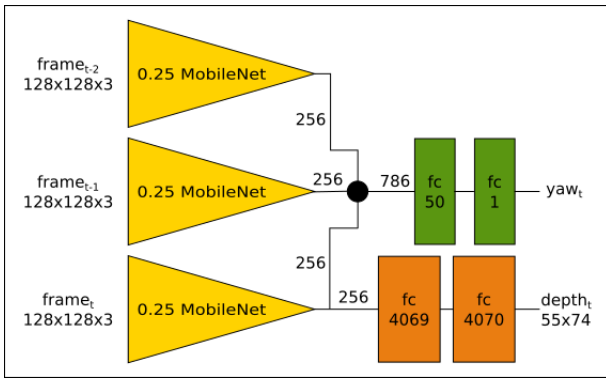


Fig. 4. The architecture of the policy. The figure is best seen in color. The yellow parts are 0.25-MobileNets that share weights over three consecutive frames. The circle represents the concatenation of the extracted features. The green part is the control prediction that consists of two fully connected layers. The orange part represents the auxiliary depth prediction layers.

TABLE I
ONLINE PERFORMANCE IN SIMULATION

	Average distance [m]					
	TOP 5		TOP 3		TOP 1	
	NAUX	AUXD	NAUX	AUXD	NAUX	AUXD
Canyon	43.96	38.41	42.33	43.08	38.05	41.79
Forest	45.99	50.24	42.90	48.48	48.67	51.35
Sandbox	7.03	8.62	6.98	9.11	9.22	8.09
ESAT	47.03	57.63	50.08	61.66	60.25	71.69

Architectures: NAUX & AUXD

The architecture of the baseline model is shown in Figure 4. The base network is called NAUX (for "No Auxiliary task") and contains a feature extracting part (yellow) and two fully-connected layers for control (green). In order to give the neural network a sense of time, the network takes three consecutive frames as input. Each frame is fed to a feature extracting part with shared weights, with the architecture of mobilenet-0.25 [22]. The weights of the mobilenet-0.25 are initialized from a model pretrained on Imagenet [23]. The features are concatenated (black circle) and fed to the control prediction part (green). The control part has a fully connected hidden layer with 50 nodes with ReLu activation and an output layer with no activation function. The control output is a continuous value of the angular velocity in yaw.

The network with auxiliary depth prediction is further referred to as AUXD. It is build on top of NAUX. The extracted features of the last frame are fed to two fully connected layers that predict a depth frame of 55x74, based on the work of Eigen et al. [29] (see Figure 4).

VI. EXPERIMENTAL RESULTS

The performance is measured as the average collision-free distance traveled by the policy online over 10 runs in the ESAT environment. The population over 50 policies is visible in Figure 5. Please note that the evaluation on the more realistic ESAT environment entails a primal domain

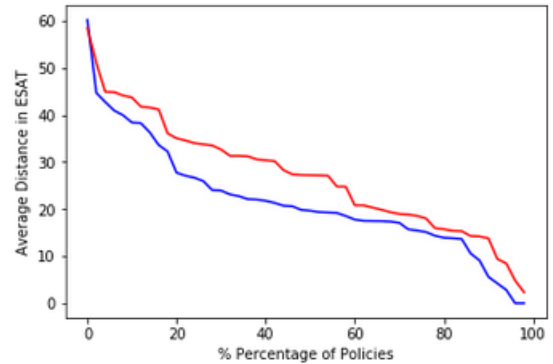


Fig. 5. The variance of the on-policy performance as distance [m] traveled in the ESAT validation environment over the percentage of the population of policies. The red and blue lines correspond to the NAUX and AUXD architectures.

shift which augments the variance over the different trained policies.

The blue line is the ranked performance of the NAUX networks trained without auxiliary depth. The red line defines the AUXD networks trained with depth prediction. It is clearly visible how the use of auxiliary depth improves the general online performance on the validation environment.

Table I shows the performance averaged over the top 5, top 3 and best policies. The policies are selected based on their average distance traveled online in the ESAT environment. The first three rows are the online performances of both NAUX and AUXD policies tested in new generated environments similar to the training environment. The performances are expressed as average collision-free distance.

Evaluating the performance on environments similar to the training environments is the common practice in training deep neural control. It is clear how, both with and without auxiliary depth, the policies can already succeed a large number of times. In other words, it has learned to succeed at avoiding collisions in the basic environments. Training on longer and more data with possibly intermediate on-policy iterations or larger networks could improve these numbers further but that is not the goal of this work.

As Figure 5 implied, the auxiliary depth has a consistent positive impact on the validation performance. This positive impact is less present in environments similar to the training environment which confirms the believe that the auxiliary task helps to regularize over a domain shift.

The step to the real world comes with an extra domain shift. The best policies are quantitatively evaluated on the Almost-Collision dataset. The results are shown in table II. The top rows show the results per location and with an overall average taken with equal weight for each location. The bottom rows show the results per type of visual cue with an overall average taken with equal weight for each visual cue. The best accuracy between NAUX and AUXD is put in bold if the difference is significant (greater than 5%). The improvement is less distinct with auxiliary depth on real world data. Although the average top 5 and top 3

TABLE II
ACCURACIES ON THE ALMOST-COLLISION DATASET [%]
NA \sim NAUX AND AD \sim AUXD

	TOP 5		TOP 3		TOP 1	
	NA	AD	NA	AD	NA	AD
ESAT real	35	64	28	80	27	73
Corridor 1	40	57	38	61	40	60
Corridor 2	53	39	51	45	49	21
Office	76	99	78	100	100	100
Cafeteria	29	25	30	26	42	34
Garage	50	62	45	56	46	58
Night	77	63	72	71	76	59
Avg. Loc.	51	58	49	63	54	58
Strange	46	54	45	57	52	60
Perspective	46	49	43	53	46	44
Vertical	70	72	68	76	66	73
Avg. Cue	54	58	52	62	55	59

performances indicate a positive trend towards the use of auxiliary depth.

VII. DISCUSSION AND CONCLUSION

The ability of neural networks to be trained in simulation and still perform robustly in the real world is a major challenge for applying deep neural control in real-world applications. A lack of benchmarks that combines the online evaluation environment with a domain shift, makes it hard to compare different methods.

You could argue that the general task of obstacle avoidance is already solved once you can predict proper depth maps over different domains as intermediate features. This is true, however the goal of DoShiCo is not to solve monocular obstacle avoidance. The goal is to find ways of training DRL policies in such way that they generalize over different domains. In many real-world applications it might not be convenient to find informative intermediate representations that can be trained separately.

As a baseline we propose a model that successfully uses auxiliary depth prediction learned in a behavioral cloning fashion. Our model succeeds at taking the dummy domain shift from the basic mixed environments to the ESAT validation environment. Though the performance on real-world data still has to be improved.

As many questions remain unanswered: 1. Is it best to train a model with imitation learning or fully reinforced when looking at such a large domain shift? 2. How can different architectures help to generalize over different domains while still allowing end-to-end training? 3. What is the influence of a discrete versus continuous action-space on the generalization ability? Further study will have to demystify the training of image-based DNN policies. By proposing the DoShiCo challenge, we want to boost this research field, taking RL a step further towards the real-world.

ACKNOWLEDGEMENTS

This work was supported by the CAMETRON research project of the KU Leuven (GOA).

REFERENCES

- [1] J. Michels, A. Saxena, and A. Y. Ng, "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning," *ICML*, no. 22, 2005.
- [2] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," *NIPS*, 2006.
- [3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, pp. 1–40, 2016.
- [4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *NIPS Deep Learning Workshop*, 2013.
- [5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *ICML*, 2016.
- [6] D. Michie, "Experiments on the mechanisation of game learning. 1. characterization of the model and its parameters," *Computer Journal*, vol. 1, p. 232263, 1963.
- [7] D. Wierstra, A. Forster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients," *Logic Journal of IGPL*, vol. 18, pp. 620–634, 10 2010.
- [8] Udacity, "Open Source Self-Driving Car," 2018.
- [9] F. Sadeghi and S. Levine, "(CAD)²RL: Real Single-Image Flight without a Single Real Image," *Robotics: Science and Systems(RSS)*, 2017.
- [10] G. Kahn, A. Villafior, B. Ding, P. Abbeel, and S. Levine, "Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation," *International Conference on Robotics and Applications(ICRA)*, 2018.
- [11] D. Gandhi, L. Pinto, and A. Gupta, "Learning to Fly by Crashing," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3948–3955, 2017.
- [12] S. R. Richter, V. Vineet, S. Roth, V. Koltun, and T. Darmstadt, "Playing for Data: Ground Truth from Computer Games," *ECCV*, no. 14, 2016.
- [13] J. Yoo, Y. Hong, and S. Yoon, "Autonomous UAV Navigation with Domain Adaptation," *CORR*, 2017.
- [14] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," *CVPR*, 2012.
- [15] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," 4 2016.
- [16] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "VisDA: The Visual Domain Adaptation Challenge," 2017.
- [17] M. Roozbeh and K. Eric, "THOR Challenge," 2017.
- [18] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, "ViZDoom: A Doom-based AI research platform for visual reinforcement learning," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, IEEE, 9 2016.
- [19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 6 2016.
- [20] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," 11 2017.
- [21] B. Wymann, C. Dimitrakakis, A. Sumner, E. Espi, and C. Guionneau, "TORCS: The open racing car simulator," 2015.
- [22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *Arxiv 1704.04861*, 2017.
- [23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 12 2015.
- [24] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, K. Kavukcuoglu, and D. London, "Reinforcement Learning with Unsupervised Auxiliary Tasks," *ICLR*, 2017.
- [25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2017.
- [26] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *arXiv:1607.02565*, 7 2016.
- [27] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive Mapping and Planning for Visual Navigation," *CVPR*, 2017.

- [28] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. V. Eycken, "CNN-based Single Image Obstacle Avoidance on a Quadrotor," *ICRA*, 2017.
- [29] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," *NIPS*, 6 2014.
- [30] P. Althaus and H. I. Christensen, "Behaviour Coordination for Navigation in Office Environments," *Intelligent Robots and Systems IEEE/RSJ*, 2002.
- [31] D. A. Pomerleau, "Rapidly Adapting Artificial Neural Networks for Autonomous Navigation," *NIPS*, 1991.
- [32] Y. Lecun, U. Muller, J. Ben, and E. Cosatto, "Off-Road Obstacle Avoidance through End-to-End Learning," *NIPS*, 2005.
- [33] S. Ross, N. Melik-Barkhudarov, S. K. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and H. Martial, "Learning Monocular Reactive UAV Control in Cluttered Natural Environments," *ICRA*, pp. 1765–1772, 2013.
- [34] A. Giusti, J. Guzzi, D. C. Cirean, F.-L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, pp. 661–667, 7 2016.
- [35] S. Daftry, J. A. Bagnell, and M. Hebert, "Learning Transferable Policies for Monocular Reactive MAV Control," *International Symposium on Experimental Robotics (ISER)*, 8 2016.
- [36] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Learning to Generalize: Meta-Learning for Domain Generalization," *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [37] G. Csurka, "Domain Adaptation for Visual Applications: A Comprehensive Survey," in *Domain Adaptation in Computer Vision Applications* (Gabriela Csurka, ed.), Springer, 2017.
- [38] R. Caruana, T. Mitchell, H. Simon, and D. Pomerleau, "Multitask Learning Rich Caruana 23 September 1997," *CMU-CS-97-203. School of Computer Science. Carnegie Mellon University. Pittsburgh, PA 15213*, 1997.
- [39] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, and D. London, "Learning To Navigate In Complex Environments," *ICLR*, 2017.
- [40] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "PLATO: Policy Learning using Adaptive Trajectory Optimization," *ICRA*, 2017.
- [41] Open Source Robotics Foundation, "Gazebo," 2018.