Learning to Race in New Environments

Elia Kaufmann, Davide Scaramuzza

***Abstract*—This paper presents an ablation study of the generalization performance of navigation policies in the context of autonomous drone racing. The approach extends on our previous conference work presented at the conference on robotic learning (CoRL) 2018. While our previous approach performed very well in known environments, the performance dropped significantly when the appearance of the environment changed compared to the one seen at training time. To overcome this challenge, we randomize the appearance of both the background and the gates during data generation in simulation. We show successful knowledge transfer to new simulated environments and identify the most important factors needed for successful transfer. Furthermore, we perform a neural architecture search to identify the network with the optimal accuracy/inference time trade-off.**

Supplementary Material

A video illustrating our experiments can be found here: https://youtu.be/K5EYLjmB9KA. For reference, please see the video of our previous conference work [1]: https://youtu.be/8RILnqPxo1s.

I. Introduction

Drone racing is an emerging sport that recently gained a lot of media attention with races being held on an international level. Human pilots undergo years of training to master the sensorimotor skills involved in racing. Such skills would also be valuable to autonomous systems in applications such as disaster response or structure inspection, where drones must be able to quickly and safely fly through complex dynamic environments [2].

Enabling an autonomous drone to race through such a race track brings up fundamental challenges in robotics in the areas of system modeling, onboard perception, localization and mapping, trajectory generation and optimal control. For this reason, autonomous drone racing has attracted significant interest from the research community, giving rise to multiple autonomous drone racing competitions [3], [4].

One approach to autonomous racing is to fly through the course by tracking a precomputed global trajectory. However, global trajectory tracking requires to know the race-track layout in advance, along with highly accurate state estimation, which current methods are still not able to provide [5]–[7]. Indeed, visual inertial odometry [5], [6] is subject to drift in estimation over time. SLAM methods can reduce drift by relocalizing in a previously-generated, globally-consistent map. However, enforcing global consistency leads to increased computational demands that strain the limits of on-board processing. In addition, regardless of drift, both odometry and SLAM pipelines enable navigation only in a predominantly-static world, where waypoints and collision-free trajectories can be statically defined. Generating and tracking a global trajectory would therefore fail in applications where the path to be followed cannot be defined *a priori*. This is usually the case for professional drone competitions, since gates can be moved from one lap to another.

Fig. 1: By combining a convolutional neural network with trajectory generation and control methods, our vision-based, autonomous quadrotor is able to successfully fly through race tracks with high agility.

Our approach to autonomous drone racing was presented in [1] and does neither need accurate knowledge of the track layout nor does it rely on a prebuilt map to localize or drift-free state estimation. Instead of relying on a precomputed, global trajectory, our approach directly identifies waypoints from a single image using a CNN. These waypoints, expressed in a body-centered frame, are used to compute minimum-jerk trajectory segments [8] that are executed by a low-level controller [9]. That way, our approach combines the reactive properties of the image-based CNN prediction with the agility and precision offered by state-of-the-art controllers. The approach is optimized to run fully onboard on a resource-constrained platform.

In the earlier version [1], the perception system was track specific: it required training data from the target race track. Therefore, significant changes in the track layout, background appearance, or lighting would hurt performance. In order to increase the generalization abilities and robustness of our perception system, we propose to use domain randomization [10]. The idea is to randomize during data collection all the factors to which the system must be invariant, i.e. illumination, viewpoint, gate appearance and background. We show that domain randomization leads to a $3\times$ increase in closed-loop performance relative to our earlier work [1] when evaluated in environments or conditions not seen at training time.

Interestingly, the perception system becomes invariant not only to specific environments and conditions but also to the training domain. We show that after training purely in simulation, the perception system can be deployed on a

physical quadrotor that successfully races in the real world. On real tracks, the policy learned in simulation has comparable performance to one trained with real data, thus alleviating the need for tedious data collection in the physical world.

II. Related Work

A. Drone Racing

Drone racing has seen a significant increase in interest in the robotics community in the recent years. Previous works include approaches based on visual servoing, where a robot is given a set of target locations that are then identified and tracked with hand-crafted detectors [11]–[13]. Such approaches however typically fail in presence of occlusions, partial visibility and motion blur. To overcome these challenges, recent work proposed to use a learning-based approach for identifying the next target [14]. Although working reliably in low-speed regimes, that approach fails when used in a more dynamic scenario due to limited agility. Image-based visual servoing is reliable when the difference between current and reference images is small, which is not always the case when flying at high speed.

Another recent approach to autonomous drone racing proposes to directly learn low level control commands from images [15]. Such an end-to-end policy is agnostic to drift in state estimation but requires a large amount of training data since a platform-stabilizing controller has to be learned. Furthermore, such approaches suffer from the typical problems of end-to-end control: (i) limited generalization to new environments and platforms and (ii) difficulties in deployment to real platforms due to high computational requirements (requires high inference rate).

Our approach features the robustness of learning-based perception while also addressing the generalization and feasibility challenges by using modularization. We take advantage of the perceptual awareness of CNNs to produce navigation commands while benefiting from the high speed and reliability of classic control pipelines.

B. Generalization to New Environments

Learning navigation policies from actual track data has a shortcoming: the learned policy typically has poor generalization performance when exposed to changes in the track appearance, such as changes in the illumination, significant changes of the track layout or changes in the gate appearance. Recent works propose to use domain randomization [10] to improve the robustness of their policies [16]–[21]. Levine et al. [22] train a collision avoidance policy purely in simulation and deploy it on a real platform.

As in aforementioned works, we use domain randomization to increase generalization. Our work applies this technique to drone racing. Specifically, we identify the most important simulation properties to randomize in order to achieve good generalization.

III. Method

Our approach, explained in detail in [1], makes use of two subsystems: perception and control. The perception system uses a Convolutional Neural Network (CNN) to predict a goal direction in local image coordinates, together with a desired navigation speed, from a single image collected by a forward-facing camera. The control system uses the navigation goal produced by the perception system to generate a minimum-jerk trajectory [8] that is tracked by a low-level controller [9].

A. Training Procedure

We train the perception system with imitation learning, using automatically generated globally optimal trajectories as a source of supervision. We refer the reader to [1] for details about the computation of the global trajectory and the generation of training labels. The network is trained by minimizing a weighted MSE loss. In order to collect diverse data, we perform visual scene randomization in the simulated environment, while keeping the approximate track layout fixed. Apart from randomizing visual scene properties, the data collection procedure remains unchanged compared to [1].

We randomize the following visual scene properties: (i) the textures of the background, floor, and gates, (ii) the shape of the gates, and (iii) the lighting in the scene. For (i), we apply distinct random textures to background and floor from a pool of 30 diverse synthetic textures (Figure 2a). The gate textures are drawn from a pool of 10 mainly red/orange textures (Figure 2c). For gate shape randomization (ii), we create 6 gate shapes of roughly the same size as the original gate. Figure 2d illustrates four of the different gate shapes used for data collection. To randomize illumination conditions (iii), we perturb the ambient and emissive light properties of all textures (background, floor, gates). Both properties are drawn separately for background, floor, and gates from uniform distributions with support $[0, 1]$ for the ambient property and $[0, 0.3]$ for the emissive property.

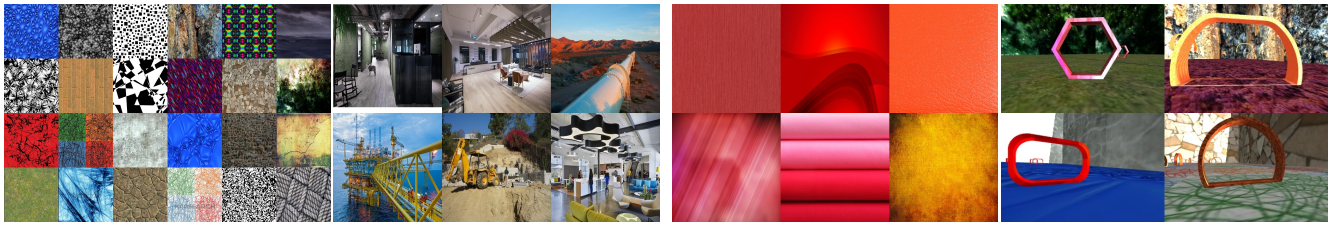
While the textures applied during data collection are synthetic, the textures applied to background and floor at test time represent common indoor and outdoor environments (Figure 2b). For testing we use held-out configurations of gate shape and texture not seen during training.

IV. Experiments

We evaluate the performance of our approach in various simulated environments. We perform an ablation study to identify the most import factors that enable successful transfer to new environments. Furthermore, we perform an architecture search to identify the network with the optimal accuracy/inference time trade-off.

A. Experimental Setup

For all our simulation experiments we use Gazebo as the simulation engine and the RotorS extension [23] for simulating the quadrotor platform. Specifically, we simulate the AscTec Hummingbird multirotor, which is equipped with a forward-looking 300×200 pixels RGB camera. The platform is spawned in a flying space of cubical shape with side length of 70 meters, which contains the experiment-specific race track. The flying space is bounded by background and floor planes whose textures are randomized (see Figure 2).



(a) (b) (c) (d)

Fig. 2: To test the generalization abilities of our approach, we randomize the visual properties of the environment (background, illumination, gate shape, and gate texture). This figure illustrates the random textures and shapes applied both at training (**a**) and test time(**b**). For space reasons, not all examples are shown. In total, we used 30 random backgrounds during training and 10 backgrounds during testing. We generated 6 different shapes of gates and used 5 of them for data generation and one for evaluation. Similarly, we used 10 random gate textures during training and a different one during evaluation. **a)** Random backgrounds used during training data generation. **b)** Random backgrounds used at test time. **c)** Gate textures. **d)** Selection of training examples illustrating the gate shapes and variation in illumination properties.

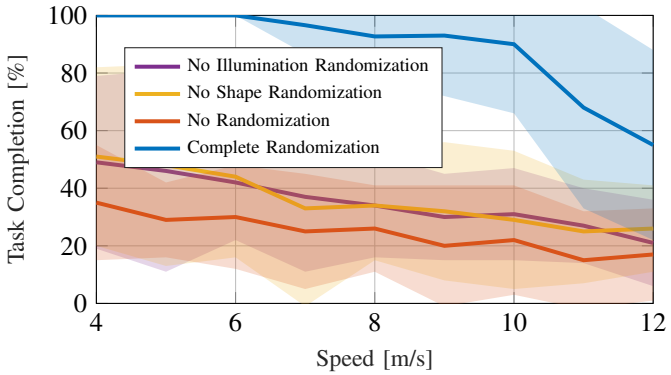


Fig. 3: Generalization tests on different backgrounds after domain randomization. More comprehensive randomization increases the robustness of the learned policy to unseen scenarios at different speeds. Lines denote mean performance, while the shaded areas indicate one standard deviation. Background randomization has not been included in the analysis: without it the policy fails to complete even a single gate pass.

B. Experiments in Simulation

We evaluate the generalization abilities of our approach to environment configurations not seen during training. Specifically, we drastically change the environment background (Fig. 2b) and use gate appearance and illumination conditions held out at training time. We compare policies that are trained with various amount of randomization. Figure 3 shows the result of this evaluation. As expected, if data collection is performed in a single environment, the resulting policy has limited generalization (red line). To make the policy environment-agnostic, we performed domain randomization while keeping the approximate track layout constant (details in Sec. III-A). Clearly, both randomization of gate shape and illumination lead to a policy that is more robust to new scenarios. Furthermore, while randomization of a single property leads to a modest improvement, performing all types of randomization simultaneously is crucial for good transfer. Indeed, the simulated policy needs to be invariant to all of the randomized features in order to generalize well.

Surprisingly, as we show below, the learned policy can not only function reliably in simulation, but is also able to control a quadrotor in the real world. In Section IV-D we present an evaluation of the real world control abilities of this policy trained in simulation, as well as an ablation study to identify which of the randomization factors presented above are the most important for generalization and knowledge transfer.

C. Analysis of Accuracy and Efficiency

The neural network at the core of our perception system constitutes the biggest computational bottleneck of our approach. Given the constraints imposed by our processing unit, we can guarantee real-time performance only with relatively small CNNs. Therefore, we investigated the relationship between the capacity (hence the representational power) of a neural network and its performance on the navigation task. We measure performance in terms of both prediction accuracy on a validation set, and closed-loop control on a simulated platform, using the *task completion rate* (explained in Figure 5) as metric. The capacity of the network is controlled through a multiplicative factor on the number of filters (in convolutional layers) and number of nodes (in fully connected layers). The network with capacity 1.0 corresponds to the DroNet architecture [24].

Figure 4 shows the relationship between the network capacity, its test loss (RMSE) on a validation set, and its inference time on an Intel UpBoard (our onboard processing unit). Given their larger parametrization, deeper architectures have a lower generalization error but largely increase the computational and memory budget required for their execution. Interestingly, a lower generalization loss does not always correspond to a better closed-loop performance. This can be observed in Figure 5, where the network with capacity 1.5 outperforms the one with capacity 2.0 at high speeds. Indeed, as shown in Figure 4, larger networks entail smaller inference rates, which result in a decrease in agility.

In our previous conference paper [1], we used a capacity factor of 1.0, which appears to have a good time-accuracy trade-off. However, in the light of this study, we select a

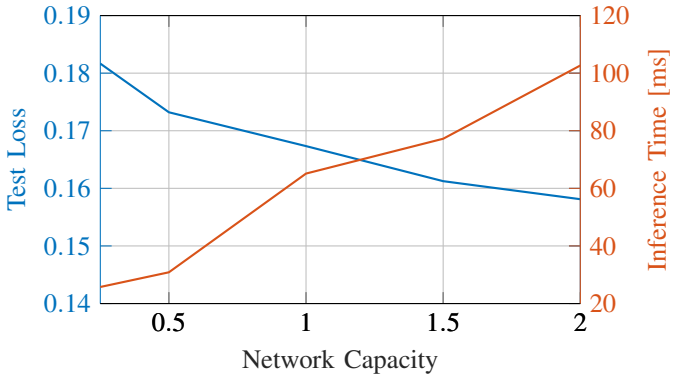


Fig. 4: Test loss and inference time for different network capacity factors. Inference time is measured on the actual platform.

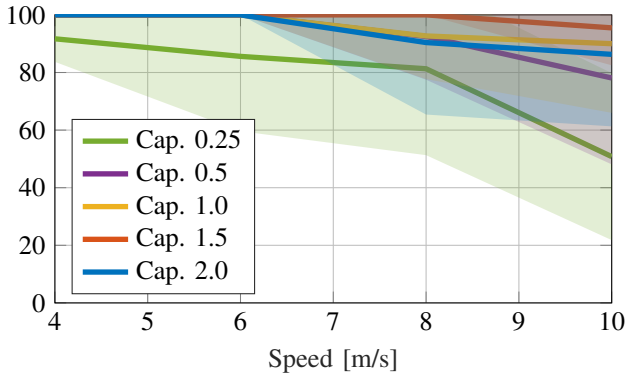


Fig. 5: Comparison of different network capacities on different backgrounds after domain randomization. Performance is stated in terms of *task completion rate*. A task completion rate of 100% is achieved if the drone can complete five consecutive laps without crashing.

capacity factor of 0.5 for all our new experiments to ease the computational burden.

D. Simulation to Real World Transfer

We now attempt direct simulation-to-real transfer of the navigation system. To train the policy in simulation, we use the same process to collect simulated data as in Section III-A. The resulting policy, evaluated in simulation in Figure 3, is then used without any finetuning to fly a real quadrotor. Despite the large appearance differences between the simulated environment (Figure 2d) and the real one, the policy trained in simulation via domain randomization has the ability to control the quadrotor in the real world. Thanks to the abundance of simulated data, this policy can not only be transferred from simulation to the real world, but is also more robust to changes in the environment than the policy trained with data collected on the real track. As can be seen in the supplementaty video, the policy learned in simulation can not only reliably control the platform, but is also robust to drastic differences in illumination and distractors on the track.

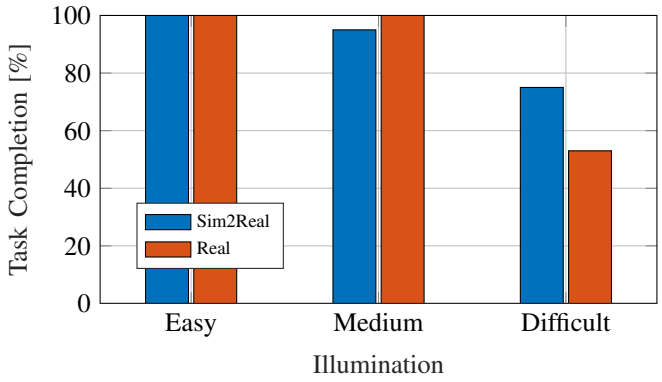


Fig. 6: Performance comparison (measured with task completion rate) of the model trained in simulation and the one trained with real data. With *easy* and *medium* illumination (on which the real model was trained on), the approaches achieve comparable performance. However, with *difficult* illumination the simulated model outperforms the real one, since the latter was never exposed to this degree of illumination changes at training time. The supplementary video illustrates the different illumination conditions.

To quantitatively benchmark the policy learned in simulation, we compare it against a policy that was trained on real data. We use the same metric as explained in Figure 5 for this evaluation. All experiments are repeated 10 times and the results averaged. The results of this evaluation are shown in Figure 6. The data that was used to train the "real" policy was recorded on the same track for two different illumination conditions, *easy* and *medium*. Illumination conditions are varied by changing the number of enabled light sources: 4 for the easy, 2 for the medium, and 1 for the difficult. The supplementary video illustrates the different illumination conditions.

The policy trained in simulation performs on par with the one trained with real data in experiments that have the same illumination conditions as the training data of the real policy. However, when the environment conditions are drastically different (i.e. with very challenging illumination) the policy trained with real data is outperformed by the one trained in simulation.

V. Conclusion

We have presented an ablation study of the generalization performance of our previously presented approach to autonomous drone racing in simulated environments. We analyzed the influence of various degrees of domain randomization on the generalization performance of our navigation policy. Furthermore, we investigated the impact of the network capacity on both the prediction accuracy and the inference rate. We measured the prediction accuracy both in terms of test loss on a held-out test set, as well as in closed loop in a previously unseen, simulated environment. We show that domain randomization leads to a 3× increase in closed-loop performance relative to our earlier work [1] when evaluated in environments or conditions not seen at training time.

REFERENCES

[1] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Conference on Robot Learning (CoRL)*, 2018.

[2] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield *et al.*, "The grand challenges of science robotics," *Science Robotics*, vol. 3, no. 14, p. eaar7650, 2018.

[3] H. Moon, Y. Sun, J. Baltes, and S. J. Kim, "The IROS 2016 competitions," *IEEE Robotics and Automation Magazine*, vol. 24, no. 1, pp. 20–29, 2017.

[4] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. de Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T.-C. Au, and S. J. Kim, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, vol. 1, no. 1, pp. 611–625, 2019.

[5] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Semi-direct visual odometry for monocular and multi-camera systems," *IEEE Transactions on Robotics, Vol. 33, Issue 2, pages 249-265*, 2017.

[6] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[8] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[9] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2018.

[10] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[11] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1116–1127, 2005.

[12] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[13] S. Li, M. Ozo, C. De Wagter, and G. de Croon, "Autonomous drone race: A computationally efficient vision-based navigation and control strategy," *arXiv preprint arXiv:1809.05958*, 2018.

[14] S. Jung, S. Hwang, H. Shin, and D. H. Shim, "Perception, guidance, and navigation for indoor autonomous drone racing using deep learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2539–2544, 2018.

[15] M. Müller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "Teaching UAVs to race: End-to-end regression of agile controls in simulation," in *European Conference on Computer Vision Workshops*, 2018.

[16] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *Internation Conference on Learning Representation (ICLR)*, 2017.

[17] M. Wulfmeier, I. Posner, and P. Abbeel, "Mutual alignment transfer learning," in *Conference on Robot Learning (CoRL)*, 2017.

[18] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[19] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *Conference on Robot Learning (CoRL)*, 2017.

[20] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on Robot Learning (CoRL)*, 2017.

[21] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real viewpoint invariant visual servoing by recurrent control," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[22] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Robotics: Science and Systems*, 2017.

[23] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS–a modular gazebo MAV simulator framework," in *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.

[24] A. Loquercio, A. I. Maqueda, C. R. D. Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.