Future Depth as Value Signal for Learning Collision Avoidance

Klaas Kelchtermans[1] and Tinne Tuytelaars[1]

Abstract— **The constant miniaturization of robots reduces the array of available sensors. This raises the need for robust algorithms that allow robots to navigate without collision based solely on monocular camera input. Towards this goal, we propose a new learning-based method for the task of obstacle avoidance. We propose a neural network policy for monocular collision avoidance with self-supervision that does not require actual collisions during training. To this end, we demonstrate that a neural network is capable of evaluating an input image and action by predicting the expected depth in the future. In this sense, the future depth can be seen as an action-value-signal. In comparison with our baseline model that is based on predicting collision probabilities, we show that using depth requires less data and leads to more stable training without need for actual collisions. The latter can be especially useful if the autonomous robot is more fragile and not capable to deal with collisions (e.g. aerial robots). The proposed method is evaluated thoroughly in simulation in a ROS-Gazebo-Tensorflow framework and will be made available on publication[2].**

I. INTRODUCTION

Collision avoidance is one of the core tasks of autonomous navigation besides road following and destination pursuing. Smaller robots solely equipped by a light-weight camera and a small GPU are capable of performing more and more complex tasks. General collision avoidance remains however challenging. Methods based on tracking keypoints and keeping a map combined with path-planning have gained impressive results [1]. However, these methods are unreliable in case of blurred images, abrupt motions or lack of features to track. In order to build an algorithm that can deal with new situations and that can adjust its features in a data-driven fashion without having to tweak many parameters, we look at learning algorithms. These systems have the benefit of learning and adapting from their mistakes which makes them more suitable for dynamic environments[2].

Deep neural networks (DNN) have succeeded at increasingly more complex tasks in computer vision and reinforcement learning [3], [4], [5], [6]. Convolutional neural networks (CNN) can handle high dimensional input data, like monocular RGB images, thanks to the parameters shared spatially. Earlier attempts to use CNNs to predict control relied on imitation learning in order to learn models to imitate an expert (human) that collected the data [7]. Other work succeeded at training a CNN to learn a quadcoptor to follow forest trails based on a big dataset collected with a straight-, left- and right-looking camera . The CNN was trained with supervised learning [8] on this set of labeled data. In [9], an iterative procedure (DAGGER) was demonstrated, using a dataset that was partially collected by a human and partially by the policy being trained, resulting in an aggregated dataset. Both methods required a big amount of annotated data which is impractical in most applications. Moreover, using a demonstration flight to represent collision avoidance might not be the best setup as there can be many equally valid routes.

Alternatively, one can decouple feature extraction and control. For instance, in Michel's car [10] a CNN is used to estimate depth upon which a separate reinforcement learning algorithm is applied. Similarly, in [11], estimated depths are used as input for a behavior arbitration algorithm in order to extract a policy for indoor navigation. The behavior arbitration algorithm however needs a number of parameter tweaking steps that differ for each environment and robot.

Recent work by Kahn et al. [12] comes closest to this work. They succeed at making a deep recurrent neural network (RNN) perform monocular autonomous navigation through a lengthy corridor. They introduce a reinforcement learning method, called generative computation graph, that is learned with a self-supervised reward signal. The reward is negative for each collision. This collision is detected by a sudden change of inertia coming from the IMU. As the labeling happens without human intervention we refer to this as self-supervision. The neural network evaluates a number of series of future actions by estimating the probability of a collision within the next horizon of frames.

The main drawback of this system however is the need for collisions. For many robots, this is impossible or at least very expensive, especially if you want to apply it to aerial robots. In this work, we therefore explore an alternative system that uses depth as a self-supervised reward. This avoids the requirement for numerous collisions and might actually learn in a more stable way than predicting sparse collision probabilities. If the goal is to avoid any close obstacle, a depth scan seems to be very indicative of which direction is to be preferred. Following this reasoning one could think of the depth scan as an objective that should be maximized. To simplify the setup in this exploratory phase of the project, we decided to work with a LiDAR as depth sensor rather than an estimated depth map. This suffices as a proof-of-concept. At a later stage this LiDAR could be substituted with a depth estimation network taking monocular camera as input. Although this might seem to be a big assumption, [10] and [11] already indicated the potential of estimated depth maps as well as most recent work [13].

The research question we investigate in this work is whether depth seen at future time-steps can be used as a value-function for evaluating the current state-action-pair for

[1]The authors are with KU Leuven, ESAT-PSI, imec, Belgium. firstname.lastname@esat.kuleuven.be

[2]Project page: kkelchte.github.io/depth-q-net

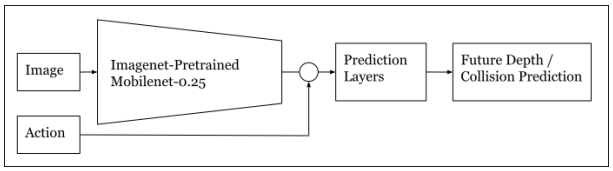

Fig. 1. Shared architecture of collision and depth prediction network.

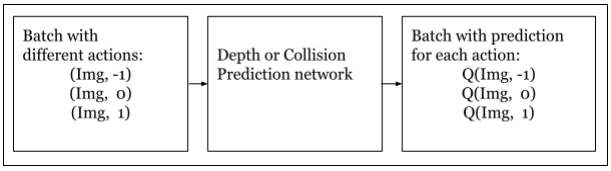

Fig. 2. Evaluation of a batch over different actions in one forward pass.

the application of collision avoidance. In order to demonstrate potential benefit we compare our method with a baseline model that uses real collisions as in [12]. Besides overcoming the need for collisions, we demonstrate how predicting action-dependent future depth leads to more stable learning behavior requiring much less training data.

In the remainder of the paper we first describe the general background (Section II). Section III describes our method in more detail as well as our baseline model. Section IV shows the experimental setup as well as the results. In the appendix the reader can find more details on the training procedure.

II. BACKGROUND

The annotation as well as most of the formulas are based on [2] although simplified for readability. In reinforcement learning, an agent interacts with its environment by performing actions. The agent tries to find a policy π that maps current state s to an action a at time t in order to maximize the expected accumulated discounted reward G, named the return. This is estimated by the state-action value function or Q-value:

$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi,T}[\sum_{k=0}^{H} \gamma^k R_{t+k+1} | s_t, a_t]$

in which γ represents the discount factor and R the expected reward. The value-function depends on the one hand on the environment bringing the agent to the next state according to a transition model, $T(s_{t+1}|s_t, a_t)$, and on the other hand the policy $\pi(a_t|s_t)$ picking the action from that next state.

Our algorithm is model-free in the sense that it does not try to model the transition function T explicitly. By definition, the value-function Q should be the sum over infinitely many time steps. However, it is often preferred to work with a finite horizon H. A specific set of algorithms are called myopic, which means that they only care about the immediate reward, $Q^\pi(s_t, a_t) = \mathbb{E}_{\pi,T}[R_{t+1}|s_t, a_t]$. This corresponds to a horizon H of 1.

In deep reinforcement learning (DRL), the value-function is approximated by a neural network. The neural network is then trained on experiences the agent has collected by interacting with its environment: (s_t, a_t, r_t, s_{t+1}). Sampling batches of experiences in order to train the algorithm is called experience replay. The value-function can then be updated using the bellman-equation in a temporal difference setting:

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$

In this situation $Q(s_{t+1}, a_{t+1})$ is referred to as the bootstrap. If the experience is collected by the policy being trained, the bootstrap can be estimated for the same policy. In this case the algorithm is called on-policy. In some algorithms the data can be collected by a different policy in which case they are referred to as off-policy. This is very beneficial as it allows collected data to be reused for training multiple agents overcoming excessive amounts of experience collection. Our proposed method can be trained off-policy.

III. METHOD

In this section we first explore the feasibility of using depth as a reward or a value-signal in a collision-avoidance setting. Later we explain the architecture of our Depth-Q-net. In the end, we explain our baseline model based on collision prediction inspired by the work of Kahn [12].

Feasibility of Depth as Value- or Reward-signal

It appears that when using depth as a reward or return function, both lead to a surprising contradiction:

Let's first look at a situation where the difference in depth is given as a **reward**. This means that an increase in depth corresponds to a positive reward. However, mathematically this leads to a value-function, V_t, that is similar to the negative of the depth:

$V_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$

$V_t = D_{t+1} - D_t + \gamma(D_{t+2} - D_{t+1}) + \gamma^2(D_{t+3} - D_{t+2}) + ...$

$V_t = -D_t + (1-\gamma)D_{t+1} + \gamma(1-\gamma)D_{t+2} + ...$

$V_t \approx -D_t$

Note that we made abstraction of the relation of the value-function with a policy. Previous derivation only holds in case of navigating straight in the direction of the camera. From a general policy iteration point of view, this would mean that a state with closer objects and lower depth corresponds to a higher value function so it is preferred over a state with objects further away. This is of course not what we want.

Alternatively, we could assume the absolute depth as a **value-signal**. This means according to the Bellman equation that a decrease in depth, for instance by navigating towards an object, corresponds to a positive reward. For now, we leave out the discount factor γ, in order to avoid clutter in the equation.

$D_t^{'} = r_t + D_{t+1}^{'} \Rightarrow r_t = D_t^{'} - D_{t+1}^{'}$

This should not come as a surprise as the depth is then seen as the amount of +1 rewards per traveled distance in that direction up until the moment of collision. If the agent learns to focus on the short term reward, the agent will be drawn towards collisions rather than the other way around.

Obstacle avoidance is a reactive behavior. This means that the optimal control should be predictable given solely current view of the robot. This holds in cases with little drift and large enough field-of-view. Using a myopic agent simplifies the use of depth as a reward signal in reinforcement learning

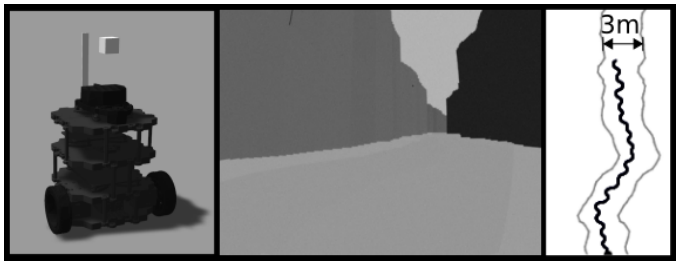

Fig. 3. Canyon example. Left: Turtlebot, middle: RGB input, right: top-down view of successful trajectory.

significantly. In this case the agent only cares about the immediate reward, picking actions that make the maximum reward most likely. In this setting the value-function corresponds to the immediate reward, in our case the depth map at the future time step. This does also resolve the contradiction explained above.

$Q(I_t, a_t) = r_t = D_{t+1}$

In this case the model learns to predict the depth seen at the next frame D_{t+1} given current frame I_t and proposed action a_t. In this paper we provide the future depth with a LiDAR at the next time step. However, the depth map could also be provided by a CNN depth-estimator as the label is allowed to be noisy. In that case the future depth network can be learned from the depth prediction on the next frame and the learning becomes fully monocular.

Depth-Q-Net

In figure 1, you can see the architecture of the network. A mobilenet-0.25 [14] convolutional neural network extracts Imagenet-pretrained [15] features from the current view of the robot. The action is concatenated to the extracted feature before it is fed to a fully-connected prediction layer of 4096 nodes and 1 fully-connected output layer of 26 nodes, corresponding to 26 depth bins from the LiDAR smoothed over 4deg from a forward field-of-view of 104deg.

The prediction layers are trained on batches of experiences in a supervised fashion with an absolute loss.

The policy is extracted by evaluating the future depth predictor on the current frame concatenated with different actions in one forward pass (see figure 2). The policy selects the action for which the predicted future depth scan has the maximum minimum depth:

$\pi(I_t) = argmax_{a_t}(min(D_{t+1}(I_t, a_t)))$

This means that the policy only focuses on the closest obstacle and takes the action that makes this closest obstacle as far as possible. This allows us to simplify the training of the future depth prediction in the sense that we only care about the closest depth. Therefore we clip the depth in our experiments at 2m.

Baseline: Coll-Q-Net

In order to explore the feasibility of using depth rather than collisions as self-supervised reward signal, we implement a similar baseline model named Coll-Q-Net. The DNN models a value-function that approximates the probability of a collision within the next H frames given current input frame and an action, similar to [12].

$Q(I_t, a_t) = \sum_{k=1}^{H} P(collision_{t+k}|I_t, a_t)$

In our experiments the horizon H is taken as 5 time steps.

The prediction layers of figure 1 contain one fully-connected layer of 4096 nodes and one fully-connected layer of 25 nodes. This results in approximately the same amount of parameters as the Depth-q-net. The fully-connected output layer has one node which contains a sigmoid activation function in order to map the output within the range of $[0, 1]$ to represent a probability. The prediction layers are trained with a cross-entropy loss. Experiments indicated the cross-entropy loss to be more stable than the mean-squared error or absolute loss.

The policy is extracted in a similar fashion, evaluating the action-value-function for a batch of actions as visible in figure 2. The action with the lowest probability of collision in the near future is selected:

$\pi(I_t) = argmin_{a_t} Q(I_t, a_t)$

Training

The prediction layers of the Depth-Q-Net and Coll-Q-Net are trained in an off-policy manner. This is beneficial as a single dataset can be collected from which multiple models can be trained in parallel, avoiding the need for data collection during training.

All experiences (I_t, a_t, y_t, d_{t+1}) are saved in a dataset and used to train the models in a supervised fashion. At each collision, the simulated environment is restarted and the last H frames get a future collision label $y_t = 1$, similar to the previous work of Kahn et al. [12]. The Depth-Q-Net does not require any collisions, therefore the last H frames of each run are discarded when training Depth-Q-Net. The data is collected by an exploration policy that randomly picks a continuous action (yaw-turn) in the range [-1:1] according to an Ornstein-Uhlenbeck process [16] to impose temporal correlation.

Although it did not have a big influence, it appeared that keeping the feature extracting part of the Mobile-0.25 net fixed was beneficial for the robustness against overfitting.

IV. EXPERIMENTS

We first discuss the simulated environment after which the results follow.

Environment

The experiments are done on a small turtlebot burger in a simulated canyon made with ROS [17] and Gazebo [18]. The simulated canyons are generated randomly during data collection. You can see an example of such a canyon in figure 3. The camera is set up on the same axis as the laser scan in order to ensure that the field-of-views between the LiDAR and the camera are aligned. The camera works at 10fps and provides frames of size 410x308 covering a field-of-view of 104deg.

The goal of the policy is to navigate the turtlebot through unseen canyons while it is driving at a fixed speed (0.5m/s)

TABLE I

ONLINE PERFORMANCE IN SIMULATED CANYON

Data	Average Distance(std)		Success rate(std)		Imitation loss(std)		Cross-Entropy(std)	Abs Diff(std)
#runs	CQN	DQN	CQN	DQN	CQN	DQN	CQN	DQN
50	1.07 (1.72E-05)	**1.68 (0.113)**	0.0 (0.0)	0.0 (0.0)	**1.03 (9.54E-03)**	2.42 (9.32E-02)	0.57 (0.128)	0.08 (3.91E-03)
100	1.04 (1.28E-04)	**4.91 (0.458)**	0.0 (0.0)	**0.7 (0.5)**	**1.22 (1.13E-02)**	1.50 (1.23E-02)	0.75 (3.09E-02)	0.08 (2.33E-03)
200	2.69 (1.28)	**12.19 (1.03)**	0.0 (0.0)	**8.0 (1.4)**	1.80 (0.307)	**0.97 (3.82E-02)**	0.85 (0.269)	0.06 (2.91E-03)
500	6.72 (0.937)	**21.96 (0.247)**	2.7 (1.2)	**20.0 (0.0)**	1.46 (6.59E-02)	**0.60 (3.54E-02)**	0.36 (9.37E-02)	0.05 (8.67E-04)
700	8.32 (1.31)	**22.21 (0.106)**	4.7 (2.1)	**20.0 (0.0)**	1.25 (6.15E-02)	**0.56 (2.61E-02)**	0.26 (2.19E-02)	0.05 (2.43E-03)
900	14.14 (3.04)	**22.30 (0.153)**	11.3 (3.8)	**20.0 (0.0)**	1.04 (0.107)	**0.55 (3.66E-02)**	0.18 (4.27E-02)	0.04 (1.19E-03)

by steering with the yaw velocity $[-1, 1]$ for a collision free distance of 15m. At test time, we only use 3 quantization levels for possible actions: -1, 0, 1. The networks are tested in 20 canyons unseen in the training data. The only difference in training Depth-Q-Net and a Coll-Q-Net is the learning-rate (0.1 and 0.01) and the selection of the loss (absolute and cross-entropy).

More detailed information on the training datasets and hyperparameters can be found in table II and the appendix.

Results

We want to investigate the benefit of using depth as a self-supervised reward signal over collisions. Besides the benefit of avoiding the need for real collisions, we are curious if the reward signal leads to more robust training for instance in the setting of having less training data.

Figure 5 shows the convergence of the absolute loss for the Depth-Q-Net and the cross-entropy loss for the Coll-Q-Net on both training and validation data for networks trained on different sizes of data.

Experience showed how not only the validation loss but especially the variance of the validation loss over a batch demonstrates potential overfitting. For the Depth-Q-Net this is visible for a dataset of 100 runs or less while for the Coll-Q-Net this is already visible at a dataset of 200 runs.

Table I gives details on the performance of the different networks. The standard deviation is calculated on 3 networks initialized with different seeding. The performance of each network is evaluated in 20 canyons. This results in an average collision free distance as well as a success rate (number of times the distance was more than 15m). Because this work solely focuses on the task of collision avoidance we do not evaluate on the time spend to travel a certain distance. The canyon however is made small enough to avoid spinning on a local spot.

The imitation loss is calculated as the MSE between the action picked by the network and the action provided by a heuristic based on the behavior arbitration algorithm taking depth as input[11]. The value is added as it gives a better measure on how the performance appears similar to an expert.

Although the reward signal is of a higher dimension and the networks are trained on less data (leaving out the collision), our method, Depth-Q-Net outperforms the baseline, Coll-Q-Net significantly on all different evaluation strategies when trained on a large enough dataset (500 runs).

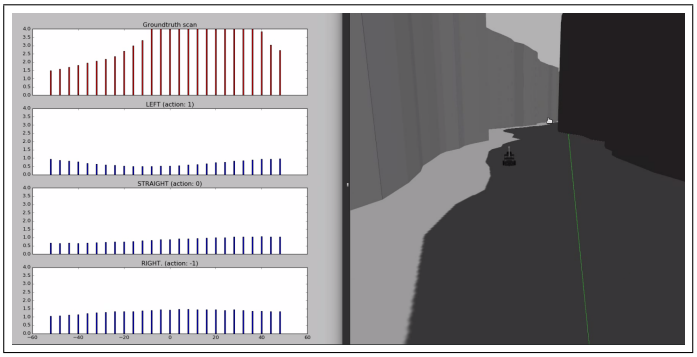

Fig. 4. Depth-Q-Net qualitative result. Left in red: ground truth scan, left in blue: future depth for left turn(up), straight(middle) and right turn(down). Right: position of robot in canyon.

In case there is not enough data, the Depth-Q-Net could still manage to pass through the canyon a number of times when trained solely on 200 or 100 runs. This is not the case for the baseline Coll-Q-Net. Although the imitation loss is less bad for the Coll-Q-Net when trained on 50 or 100 runs, the network could not drive much more than 1m without a collision.

Figure 4 gives a snapshot of the evaluation of a Depth-Q-Net in the canyon. On the left you can see in red the ground-truth lazer scan. For each action, the future depth is predicted by the network. As the Turtlebot is slightly heading to the left wall, there is an increase in overall depth for turning to the right and a decrease for turning to the left. Depth-Q-Net is only trained to predict values up until 2m while the actual depth is at 4m. Although this is far from accurate in absolute values, it is good enough to extract a working policy from.

V. CONCLUSION

In this work we explore the feasibility of using depth as a self-supervised value/reward-signal instead of collisions detected by an IMU for learning collision avoidance (as in [12]). We demonstrate that the use of depth not only leads to better performance (less collisions), it also requires less data. This is because the labels of collisions are more sparse, giving relevant feedback only at the end of each run while the Depth-Q-Net can be trained on any consecutive pair of images.

In future work the depth could be provided with a monocular depth-estimation network, in which case it would not require any additional sensors besides the RGB camera. The

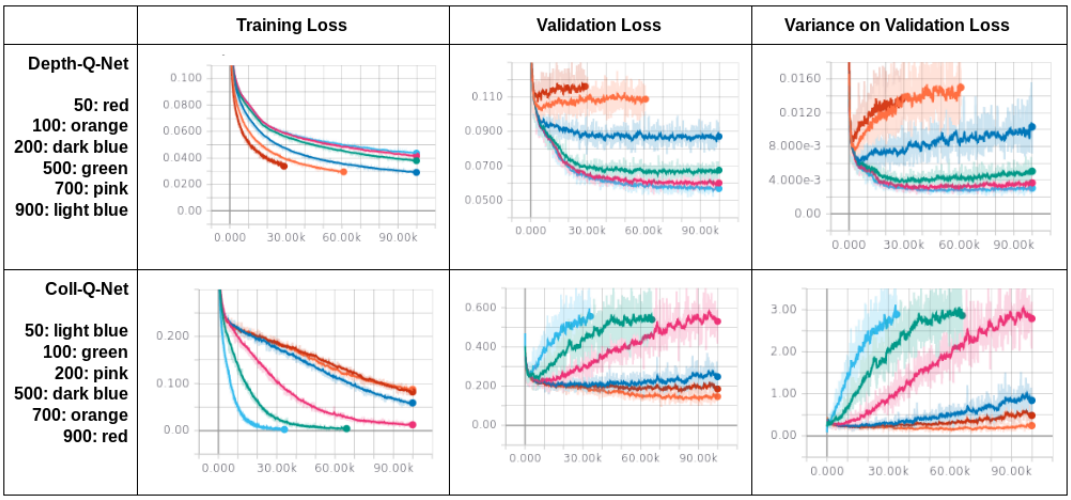

Fig. 5. Convergence difference for different networks trained on different sizes of datasets.

TABLE II
DETAILS DIFFERENT SIZES OF DATASETS

# runs	# samples	# without collision
50	2214	1858
100	4258	3945
200	8854	7797
500	23361	20517
700	33303	29152
900	42424	38032

Depth-Q-Net will learn to predict the estimated depth at the next step given current RGB image and proposed action.

On the other hand, a real-world experiment could even further demonstrate the benefit of training a Depth-Q-Net over a Coll-Q-Net as acquiring data in the real world is much more difficult especially if it requires collisions. Unfortunately lack of time did not allow us to include these experiments although the simulated results already demonstrate clearly the feasibility of our method.

Although depth-value-signals are probably not the full solution for monocular collision avoidance, it can be interesting to add in the form of intrinsic motivation for general autonomous robots provided with a camera.

In the current work we only look at the next frame. With a recurrent neural network more aggressive behaviors might be learned where planning over multiple time steps is required.

ACKNOWLEDGEMENTS

This work was supported by the CAMETRON research project of the KU Leuven (GOA) and the Omnidrone research project (FWO-SBO).

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, pp. 1147–1163, 10 2015.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2017.

[3] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive Mapping and Planning for Visual Navigation," *CVPR*, 2017.

[4] 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.

[5] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," *NIPS*, 6 2014.

[6] 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*, 2 2016.

[7] S. Daftry, J. A. Bagnell, and M. Hebert, "Learning Transferable Policies for Monocular Reactive MAV Control," *International Symposium on Experimental Robotics (ISER)*, 8 2016.

[8] A. Giusti, J. Guzzi, D. C. Ciresan, 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.

[9] 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.

[10] J. Michels, A. Saxena, and A. Y. Ng, "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning," *ICML*, no. 22, 2005.

[11] 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.

[12] G. Kahn, A. Villaflor, 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.

[13] M. A. Anwar and A. Raychowdhury, "NAVREN-RL: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images," 7 2018.

[14] 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.

[15] 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.

[16] G. E. Uhlenbeck and L. S. Ornstein, "On the Theory of the Brownian Motion," *Physical Review*, vol. 36, pp. 823–841, 9 1930.

[17] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *ICRA*, 2009.

[18] Open Source Robotics Foundation, "Gazebo," 2018.

VI. Appendix

Training deep neural networks can be tedious. In order to reproduce the results we share our hyperparameters. They can also be found in the code itself on the project page: kkelchte.github.io/depth-q-net.

- Dropout of 0.5 after the output of the Mobilenet-0.25.
- Batch size of 64.
- Weight decay of 4e-5.
- Adadelta optimizer.
- Xavier initialization.
- Ended after 1000 epochs ($\approx$ 3h) .

Coll-Q-Net had to be trained with a lower learning rate (0.01) than Depth-Q-Net (0.1). As mentioned in the paper, Coll-Q-Net trained best with a cross-entropy loss and Depth-Q-Net with an absolute difference.