DroNet: Learning to Fly by Driving

Antonio Loquercio, Ana I. Maqueda, Carlos R. del-Blanco, and Davide Scaramuzza

***Abstract*—Civilian drones are soon expected to be used in a wide variety of tasks, such as aerial surveillance, delivery, or monitoring of existing architectures. Nevertheless, their deployment in urban environments has so far been limited. Indeed, in unstructured and highly dynamic scenarios, drones face numerous challenges to navigate autonomously in a feasible and safe way. In contrast to traditional "map-localize-plan" methods, this letter explores a data-driven approach to cope with the above challenges. To accomplish this, we propose DroNet: a convolutional neural network that can safely drive a drone through the streets of a city. Designed as a fast eight-layers residual network, DroNet produces two outputs for each single input image: A steering angle to keep the drone navigating while avoiding obstacles, and a collision probability to let the UAV recognize dangerous situations and promptly react to them. The challenge is however to collect enough data in an unstructured outdoor environment such as a city. Clearly, having an expert pilot providing training trajectories is not an option given the large amount of data required and, above all, the risk that it involves for other vehicles or pedestrians moving in the streets. Therefore, we propose to train a UAV from data collected by cars and bicycles, which, already integrated into the urban environment, would not endanger other vehicles and pedestrians. Although trained on city streets from the viewpoint of urban vehicles, the navigation policy learned by DroNet is highly generalizable. Indeed, it allows a UAV to successfully fly at relative high altitudes and even in indoor environments, such as parking lots and corridors. To share our findings with the robotics community, we publicly release all our datasets, code, and trained networks.**

***Index Terms*—Learning from demonstration, deep learning in robotics and automation, aerial systems: perception and autonomy.**

I. Introduction

SAFE and reliable outdoor navigation of autonomous systems, e.g., unmanned aerial vehicles (UAVs), is a challenging open problem in robotics. Being able to successfully navigate while avoiding obstacles is indeed crucial to unlock many robotics applications, *e.g.,* surveillance, construction monitoring, delivery, and emergency response [1]–[3]. A robotic system facing the above tasks should simultaneously solve many challenges in perception, control, and localization. These become particularly difficult when working in urban areas, as the one illustrated in Fig. 1. In those cases, the autonomous agent is not only expected to navigate while avoiding collisions, but also to safely interact with other agents present in the environment, such as pedestrians or cars.

Fig. 1. DroNet is a convolutional neural network, whose purpose is to reliably drive an autonomous drone through the streets of a city. Trained with data collected by cars and bicycles, our system learns from them to follow basic traffic rules, *e.g.*, do not go off the road, and to safely avoid other pedestrians or obstacles. Surprisingly, the policy learned by DroNet is highly generalizable, and even allows to fly a drone in indoor corridors and parking lots.

The traditional approach to tackle this problem is a two step interleaved process consisting of (i) automatic localization in a given map (using GPS, visual and/or range sensors), and (ii) computation of control commands to allow the agent to avoid obstacles while achieving its goal [1], [4]. Even though advanced SLAM algorithms enable localization under a wide range of conditions [5], visual aliasing, dynamic scenes, and strong appearance changes can drive the perception system to unrecoverable errors. Moreover, keeping the perception and control blocks separated not only hinders any possibility of positive feedback between them, but also introduces the challenging problem of inferring control commands from 3D maps.

Recently, new approaches based on deep learning have offered a way to tightly couple perception and control, achieving impressive results in a large set of tasks [6]–[8]. Among them, methods based on reinforcement learning (RL) suffer from significantly high sample complexity, hindering their application to UAVs operating in safety-critical environments. In contrast, supervised-learning methods offer a more viable way to learn

Manuscript received September 10, 2017; accepted January 2, 2018. Date of publication January 23, 2018; date of current version February 1, 2018. This letter was recommended for publication by Associate Editor S. Oh and Editor D. Lee upon evaluation of the reviewers' comments. This work was supported in part by the Swiss National Center of Competence Research Robotics through the Swiss National Science Foundation, in part by the SNSF-ERC starting grant, and in part by the Ministerio de Economía, Industria y Competitividad (AEI/FEDER) of the Spanish Government under Project TEC2016-75981 (IVME). *(Corresponding author: Antonio Loquercio.)*

A. Loquercio and D. Scaramuzza are with the Robotics and Perception Group, Department of Informatics, and Department of Neuroinformatics, University of Zurich and ETH Zurich, Zurich 8092, Switzerland (e-mail: loquercio@ifi.uzh.ch; davide.scaramuzza@ieee.org).

A. I. Maqueda and C. R. del-Blanco are with the Grupo de Tratamiento de Imágenes, Information Processing and Telecommunications Center and ETSI Telecomunicación, Universidad Politécnica de Madrid, Madrid 28040, Spain (e-mail: amn@gti.ssr.upm.es; cda@gti.ssr.upm.es).

Digital Object Identifier 10.1109/LRA.2018.2795643

effective flying policies [6], [9], [10], but they still leave the issue of collecting enough expert trajectories to imitate. Additionally, as pointed out by [10], collision trajectories avoided by expert human pilots are actually necessary to let the robotic platform learn how to behave in dangerous situations.

Contributions: Clearly, a UAV successfully navigating through the streets should be able to follow the roadway as well as promptly react to dangerous situations exactly as any other ground vehicle would do. Therefore, we herein propose to use data collected from ground vehicles which are already integrated in environments as aforementioned. Overall, this work makes the following contributions:

- We propose a residual convolutional architecture which, by predicting the steering angle and the collision probability, can perform a safe flight of a quadrotor in urban environments. To train it, we employ an outdoor dataset recorded from cars and bicycles.
- We collect a custom dataset of outdoor collision sequences to let a UAV predict potentially dangerous situations.
- Trading off performance for processing time, we show that our design represents a good fit for navigation-related tasks. Indeed, it enables real-time processing of the video stream recorded by a UAV's camera.
- Through an extensive evaluation, we show that our system can be applied to new application spaces without any initial knowledge about them. Indeed, with neither a map of the environment nor retraining or fine-tuning, our method generalizes to scenarios completely unseen at training time including indoor corridors, parking lots, and high altitudes.

Even though our system achieves remarkable results, we do not aim to replace traditional "map-localize-plan" approaches for drone navigation, but rather investigate whether a similar task could be done with a single shallow neural network. Indeed, we believe that learning-based and traditional approaches will one day complement each other.

II. Related Work

A wide variety of techniques for drone navigation and obstacle avoidance can be found in the literature. At high level, these methods differ depending on the kind of sensory input and processing employed to control the flying platform.

A UAV operating outdoor is usually provided with GPS, range, and visual sensors to estimate the system state, infer the presence of obstacles, and perform path planning [1], [4]. Nevertheless, such works are still prone to fail in urban environments where the presence of high rise buildings, and dynamic obstacles can result in significant undetected errors in the system state estimate. The prevalent approach in such scenarios is SLAM, where the robot simultaneously builds a map of the environment and self-localizes in it [5]. On the other hand, while an explicit 3D reconstruction of the environment can be good for global localization and navigation, it is not entirely clear how to infer control commands for a safe and reliable flight from it.

Recently, there has been an increasing research effort in directly learning control policies from raw sensory data using Deep Neural Networks. These methodologies can be divided into two main categories: (i) methods based on reinforcement learning (RL) [7], [11] and (ii) methods based on supervised learning [6], [9], [10], [12], [13].

While RL-based algorithms have been successful in learning generalizing policies [7], [8], they usually require a large amount of robot experience which is costly and dangerous to acquire in real safety-critical systems. In contrast, supervised learning offers a more viable way to train control policies, but clearly depends upon the provided expert signal to imitate. This supervision may come from a human expert [6], hard-coded trajectories [10], or model predictive control [12]. However, when working in the streets of a city, it can be both tedious and dangerous to collect a large set of expert trajectories, or evaluate partially trained policies [6]. Additionally, the domain-shift between expert and agent might hinder generalization capabilities of supervised learning methods. Indeed, previous work in [9], [13] trained a UAV from video collected by a mountain hiker but did not show the learned policy to generalize to scenarios unseen at training time. Another promising approach has been use simulations to get training data for reinforcement or imitation learning tasks, while testing the learned policy in the real world [11], [14], [15]. Clearly, this approach suffers from the domain shift between simulation and reality and might require some real-world data to be able to generalize [11]. To our knowledge, current simulators still fail to model the large amount of variability present in an urban scenario and are therefore not fully acceptable for our task. Additionally, even though some pioneering work has been done in [14], it is still not entirely clear how to make policies learned in simulation generalize into the real world.

To overcome the above-mentioned limitations, we propose to train a neural network policy by imitating expert behaviour which is generated from wheeled manned vehicles only. Even though there is a significant body of literature on the task of steering angle prediction for ground vehicles [16], [17], our goal is not to propose yet another method for steering angle prediction, but rather to prove that we can deploy this expertise also on flying platforms. The result is a single shallow network that processes all visual information concurrently, and directly produces control commands for a flying drone. The coupling between perception and control, learned end-to-end, provides several advantages, such as a simpler and lightweight system and high generalization abilities. Additionally, our data collection proposal does not require any state estimate or even an expert drone pilot, while it exposes pedestrians, other vehicles, and the drone itself to no danger.

III. Methodology

Our learning approach aims at reactively predicting a steering angle and a probability of collision from the drone on-board forward-looking camera. These are later converted into control flying commands which enable a UAV to safely navigate while avoiding obstacles.

Since we aim to reduce the bare image processing time, we advocate a single convolutional neural network (CNN) with a relatively small size. The resulting network, which we call DroNet,

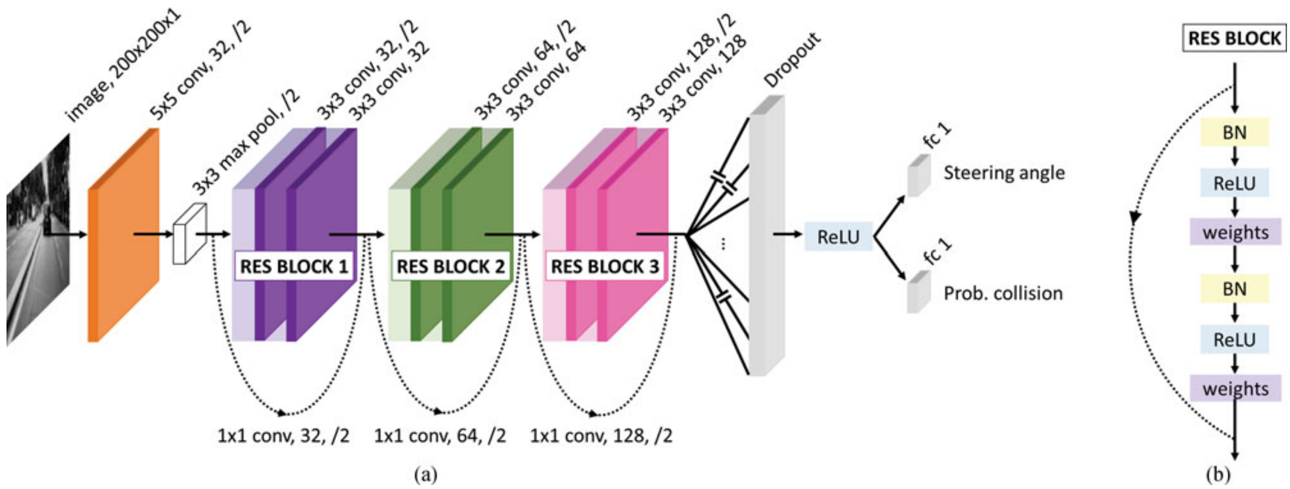


Fig. 2. (a) DroNet is a forked Convolutional Neural Network that predicts, from a single 200×200 frame in gray-scale, a steering angle and a collision probability. The shared part of the architecture consists of a ResNet-8 with 3 residual blocks (b) followed by dropout and ReLU non-linearity. Afterwards, the network branches into 2 separated fully-connected layers, one to carry out steering prediction, and the other one to infer collision probability. In the notation above, we indicate for each convolution first the kernel's size, then the number of filters, and eventually the stride if it is different from 1.

is shown in Fig. 2(a). The architecture is partially shared by the two tasks to reduce the network's complexity and processing time, but is then separated into two branches at the very end. Steering prediction is a regression problem, while collision prediction is addressed as a binary classification problem. Due to their different nature and output range, we propose to separate the network's last fully-connected layer.

During the training procedure, we use only imagery recorded by manned vehicles. Steering angles are learned from images captured from a car, while probability of collision, from a bicycle.

A. Learning Approach

The part of the network that is shared by the two tasks consists of a ResNet-8 architecture followed by a dropout of 0.5 and a ReLU non-linearity. The residual blocks of the ResNet, proposed by He *et al.* [18], are shown in Fig. 2(b). Dotted lines represent skip connections defined as 1×1 convolutional shortcuts to allow the input and output of the residual blocks to be added. Even though an advantage of ResNets is to tackle the vanishing/exploding gradient problems in very deep networks, its success lies in its learning principle. Indeed, the residual scheme has been primarily introduced to address the degradation problem generated by difficulties in networks' optimization [18]. Therefore, since residual architectures are known to help generalization on both shallow and deep networks [18], we adapted this design choice to increase model performance. After the last ReLU layer, tasks stop sharing parameters, and the architecture splits into two different fully-connected layers. The first one outputs the steering angle, and the second one a collision probability. Strictly speaking the latter is not a Bayesian probability but an index quantifying the network uncertainty in prediction. Slightly abusing the notation, we still refer to it as "probability".

We use mean-squared error (MSE) and binary cross-entropy (BCE) to train the steering and collision predictions, respectively. Although the network architecture proves to be appropriate to minimize complexity and processing time, a naive joint optimization poses serious convergence problems due to the very different gradients' magnitudes that each loss produces. More specifically, imposing no weighting between the two losses during training results in convergence to a very poor solution. This can be explained by difference of gradients' magnitudes in the classification and regression task at the initial stages of training, which can be problematic for optimization [19]. Indeed, the gradients from the regression task are initially much larger, since the MSE gradients' norms is proportional to the absolute steering error. Therefore, we give more and more weight to the classification loss in later stages of training. Once losses' magnitudes are comparable, the optimizer will try to find a good solution for both at the same time. For the aforementioned reasons, imposing no or constant loss weight between the two losses would likely result in sub-optimal performance or require much longer optimization times. This can be seen as a particular form of curriculum learning [19]. In detail, the weight coefficient corresponding to BCE is defined in (1), while the one for MSE is always 1. For our experiments, we set $decay = \frac{1}{10}$, and $epoch_0 = 10$.

$$L_{tot} = L_{MSE} + \max(0, 1 - \exp^{-decay(epoch - epoch_0)}) L_{BCE} \tag{1}$$

The Adam optimizer [20] is used with a starting learning rate of 0.001 and an exponential per-step decay equal to 10^{-5}. We also employ hard negative mining for the optimization to focus on those samples which are the most difficult to learn. In particular, we select the k samples with the highest loss in each epoch, and compute the total loss according to (1). We define k so that it decreases over time.

(a) (b)

Fig. 3. (a) Udacity images used to learn steering angles. (b) Collected images to learn probability of collision. The green box contains no-collision frames, and the red one collision frames.

B. Datasets

To learn steering angles from images, we use one of the publicly available datasets from Udacity's project [21]. This dataset contains over 70,000 images of car driving distributed over 6 experiments, 5 for training and 1 for testing. Every experiment stores time-stamped images from 3 cameras (left, central, right), IMU, GPS data, gear, brake, throttle, steering angles and speed. For our experiment, we only use images from the forward-looking camera (Fig. 3(a)) and their associated steering angles.

To our knowledge, there are no public datasets that associate images with collision probability according to the distance to the obstacles. Therefore, we collect our own collision data by mounting a GoPro camera on the handlebars of a bicycle. We drive along different areas of a city, trying to diversify the types of obstacles (vehicles, pedestrians, vegetation, under-construction sites) and the appearance of the environment (Fig. 3(b)). This way, the drone is able to generalize under different scenarios. We start recording when we are far away from an obstacle and stop when we are very close to it. In total, we collect around 32,000 images distributed over 137 sequences for a diverse set of obstacles. We manually annotate the sequences, so that frames far away from collision are labeled as 0 (no collision), and frames very close to the obstacle are labeled as 1 (collision), as can be seen in Fig. 3(b). Collision frames are the types of data that cannot be easily obtained by a drone but are necessary to build a safe and robust system.

C. Drone Control

The outputs of DroNet are used to command the UAV to move on a plane with forward velocity v_k and steering angle θ_k. More specifically, we use the probability of collision p_t provided by the network to modulate the forward velocity: the vehicle is commanded to go at maximal speed $V_{\max}$ when the probability of collision is null, and to stop whenever it is close to 1. We use a low-pass filtered version of the modulated forward velocity v_k to provide the controller with smooth, continuous inputs ($0 \leq \alpha \leq 1$):

$$v_k = (1-\alpha)v_{k-1} + \alpha(1-p_t)V_{\max}, \tag{2}$$

Similarly, we map the predicted scaled steering s_k into a rotation around the body z-axis (yaw angle θ), corresponding to the axis orthogonal to the propellers' plane. Concretely, we convert s_k from a $[-1, 1]$ range into a desired yaw angle θ_k in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and low-pass filter it:

$$\theta_k = (1-\beta)\theta_{k-1} + \beta\,\frac{\pi}{2}\,s_k \tag{3}$$

In all our experiments we set $\alpha = 0.7$ and $\beta = 0.5$, while $V_{\max}$ was changed according to the testing environment. The above constants have been selected empirically trading off smoothness for reactiveness of the drone's flight. As a result, we obtain a reactive navigation policy that can reliably control a drone from a single forward-looking camera. An interesting aspect of our approach is that we can produce a collision probability from a single image without any information about the platform's speed. Indeed, we conjecture the network to make decision on the base of the distance to observed objects in the field of view. Convolutional networks are in fact well known to be successful on the task of monocular depth estimation [15]. An interesting question that we would like to answer in future work is how this approach compares to an LSTM [22] based solution, making decisions over a temporal horizon.

IV. Experimental Results

In this section, we show quantitative and qualitative results of our proposed methodology. First, we evaluate the accuracy of DroNet with a set of performance metrics. Then, we discuss its control capabilities comparing it against a set of navigation baselines.

A. Hardware Specification

We performed our experiments on a Parrot Bebop 2.0 drone. Designed as an outdoor hobby platform, it has a basic and rather inaccurate, visual odometry system that allows the user to provide only high-level commands, such as body-frame velocities, to control the platform. Velocity commands are produced by our network running on an Intel Core i7 2.6 GHz CPU that receives images at 30 Hz from the drone through Wi-Fi.

B. Regression and Classification Results

We first evaluate the regression performance of our model employing the testing sequence from the Udacity dataset [21].

TABLE I
QUANTITATIVE RESULTS ON REGRESSION AND CLASSIFICATION TASK: EVA AND RMSE ARE COMPUTED ON THE STEERING REGRESSION TASK, WHILE AVG. ACCURACY AND F-1 SCORE ARE EVALUATED ON THE COLLISION PREDICTION TASK

Model	EVA	RMSE	Avg. accuracy	F-1 score	Num. Layers	Num. parameters	Processing time [fps]
Random baseline	-1.0 ± 0.022	0.3 ± 0.001	$50.0 \pm 0.1\%$	0.3 ± 0.01	–	–	–
Constant baseline	0	0.2129	75.6%	0.00	–	–	–
Giusti *et al.* [9]	0.672	0.125	91.2%	0.823	6	5.8×10^4	23
ResNet-50 [18]	0.795	0.097	96.6%	0.921	50	2.6×10^7	7
VGG-16 [23]	0.712	0.119	92.7%	0.847	16	7.5×10^6	12
DroNet (Ours)	0.737	0.109	95.4%	0.901	8	3.2×10^5	20

Our model compares favorably against the considered baselines. Despite being relatively lightweight in terms of number of parameters, DroNet maintains a very good performance on both tasks. We additionally report the on-line processing time in frames per second (fps), achieved when receiving images at 30 Hz from the UAV.

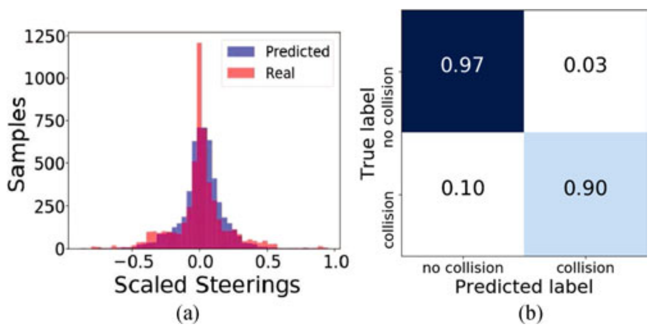


Fig. 4. Model performance: (a) Probability Density Function (PDF) of actual vs. predicted steerings of the Udacity dataset testing sequence. (b) Confusion matrix on the collision classification evaluated on testing images of the collected dataset. Numbers in this matrix indicate the percentage of samples falling in each category.

To quantify the performance on steering prediction, we use two metrics: root-mean-squared error (RMSE) and explained variance ratio (EVA)[1] To asses the performance on collision prediction, we use average classification accuracy and the F-1 score[2]

Table I compares DroNet against a set of other architectures from the literature [9], [18], [23]. Additionally, we use as weak baselines a constant estimator, which always predicts 0 as steering angle and "no collision", and a random one. From these results we can observe that our design, even though 80 times smaller than the best architecture, maintains a considerable prediction performance while achieving real-time operation (20 frames per second). Furthermore, the positive comparison against the VGG-16 architecture indicates the advantages in terms of generalization due to the residual learning scheme, as discussed in Section III-A. Our design succeeds at finding a good trade-off between performance and processing time as shown in Table I and Fig. 4. Indeed, in order to enable a drone to promptly react to unexpected events or dangerous situations, it is necessary to reduce the network's latency as much as possible.

[1]Explained Variance is a metric used to quantify the quality of a regressor, and is defined as $EVA = \frac{\mathrm{Var}[y_{true} - y_{pred}]}{\mathrm{Var}[y_{true}]}$.

[2]F-1 score is a metric used to quantify the quality of a classifier. It is defined as F-1 $= 2\frac{precision \times recall}{precision + recall}$.

C. Quantitative Results on DroNet's Control Capabilities

We tested our DroNet system by autonomously navigating in a number of different urban trails including straight paths and sharp curves. Moreover, to test the generalization capabilities of the learned policy, we also performed experiments in indoor environments. An illustration of the testing environments can be found in Figs. 5 and 6. We compare our approach against two baselines:

a) *Straight Line Policy:* Trivial baseline consisting in following a straight path in open-loop. This baseline is expected to be very weak, given that we always tested in environments with curves.

b) *Minimize Probability of Collision Policy:* Strong baseline consisting in going toward the direction minimizing the collision probability. For this approach, we implemented the algorithm proposed in [10], which was shown to have very good control capabilities in indoor environments. We employ the same architecture as in DroNet along with our collected dataset in order to estimate the collision probability.

As metric we use the average distance travelling before stopping or colliding. Results from Table II indicate that DroNet is able to drive a UAV the longest on almost all the selected testing scenarios. The main strengths of the policy learned by DroNet are twofold: (i) the platform smoothly follows the road lane while avoiding static obstacles; (ii) the drone is never driven into a collision, even in presence of dynamic obstacles, like pedestrians or bicycles, occasionally occluding its path. Another interesting feature of our method is that DroNet usually drives the vehicle to a random direction in open spaces and at intersections. In contrast, the baseline policy of minimizing the probability of collision was very often confused by intersections and open spaces, which resulted in a shaky uncontrolled behaviour. This explains the usually large gaps in performance between our selected methodology and the considered baselines.

Interestingly, the policy learned by DroNet generalizes well to scenarios visually different from the training ones, as shown in Table II. First, we noticed only a very little drop in performance when the vehicle was flying at relatively high altitude (5 m). Even though the drone's viewpoint was different from a ground vehicle's one (usually at 1.5 m), the curve could be successfully completed as long as the path was in the field of view of the camera. More surprisingly was the generalization of our method to indoor environments such as a corridor or a parking lot. In

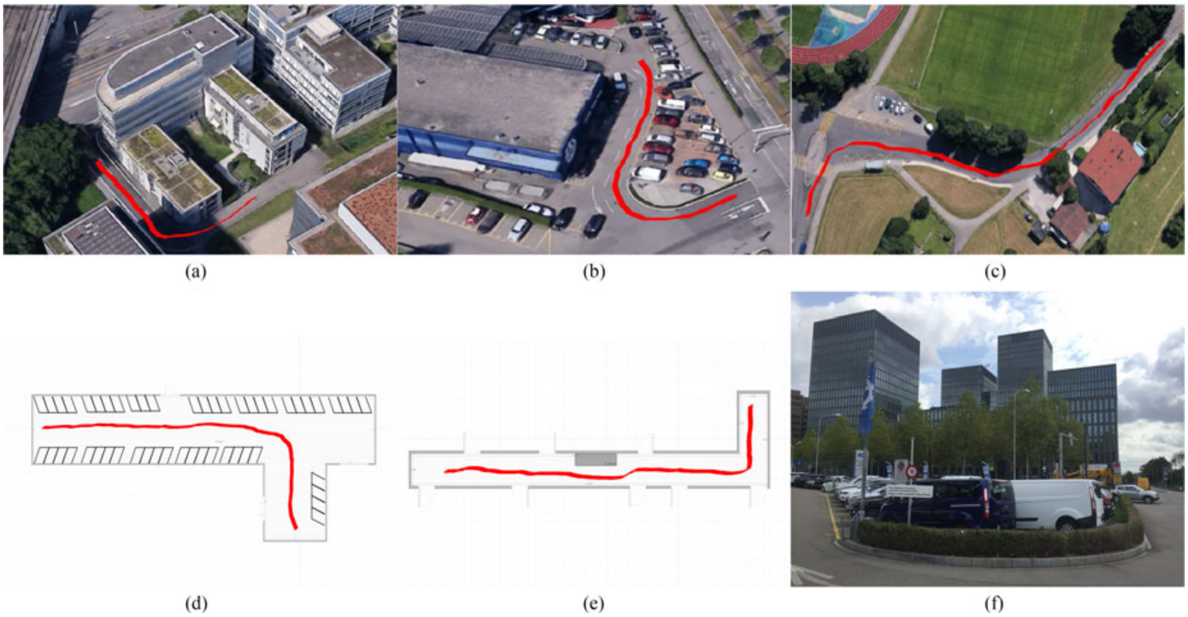


Fig. 5. Testing environments: (a) Outdoor 1 is a 90° curve with a dead end. This scenario is also tested with the drone flying at high altitude (5 m), as shown in Fig. 6. (b) Outdoor 2 is a sharp 160° curve followed by a 30 m straight path. A closer view of this environment can be seen in (f). (c) Outdoor 3 is a series of 2 curves, each of approximately 60°, with straight paths in between. Moreover, we also tested DroNet on scenarios visually different from the training ones, such as (d) an indoor parking lot, and (e) an indoor corridor. (a) Outdoor 1. (b) Outdoor 2. (c) Outdoor 3. (d) Indoor Parking Lot. (e) Indoor Corridor. (f) Closer view of Outdoor 2.

Fig. 6. High altitude Outdoor 1: In order to test the ability of DroNet to generalize at high altitude, we made the drone fly at 5 m altitude in the testing environment Outdoor 1. Table II indicates that our policy is able to cope with the large difference between the viewpoint of a camera mounted on a car (1.5 m) and the one of the UAV.

these scenarios, the drone was still able to avoid static obstacles, follow paths, and stop in case of dynamic obstacles occluding its way. Nonetheless, we experienced some domain-shift problems. In indoor environments, we experienced some drifts at intersections which were sometimes too narrow to be smoothly performed by our algorithm. In contrast, as we expected, the baseline policy of [10], specifically designed to work in narrow indoor spaces, outperformed our method. Still, we believe that it is very surprising that a UAV trained on outdoor streets can actually perform well even in indoor corridors.

D. Qualitative Results

In Fig. 8 and, more extensively in the supplementary video, it is possible to observe the behaviour of DroNet in some of the considered testing environments. Unlike previous work [9], our approach always produced a safe and smooth flight. In particular, the drone always reacted promptly to dangerous situations, *e.g.,* sudden occlusions by bikers or pedestrians in front of it.

To better understand our flying policy, we employed the technique outlined in [24]. Fig. 7 shows which part of an image is the most important for DroNet to generate a steering decision. Intuitively, the network mainly concentrates on the “line-like” patterns present in a frame, which roughly indicate the steering direction. Indeed, the strong coupling between perception and control renders perception mainly sensitive to the features important for control. This explains why DroNet generalizes so well to many different indoor and outdoor scenes that contain “line–like” features. Conversely, we expect our approach to fail in environments missing those kind of features. This was for example the case for an experiment we performed in a forest, where no evident path was visible. However, placed in a forest surrounding with a clearly visible path, the drone behaved better.

Furthermore, the importance of our proposed methodology is supported by the difficulties encountered while carrying out outdoor city experiments. If we want a drone to learn to fly in a city, it is crucial to take advantage of cars, bicycles or other manned vehicles. As these are already integrated in the urban streets, they allow to collect enough valid training data safely and efficiently.

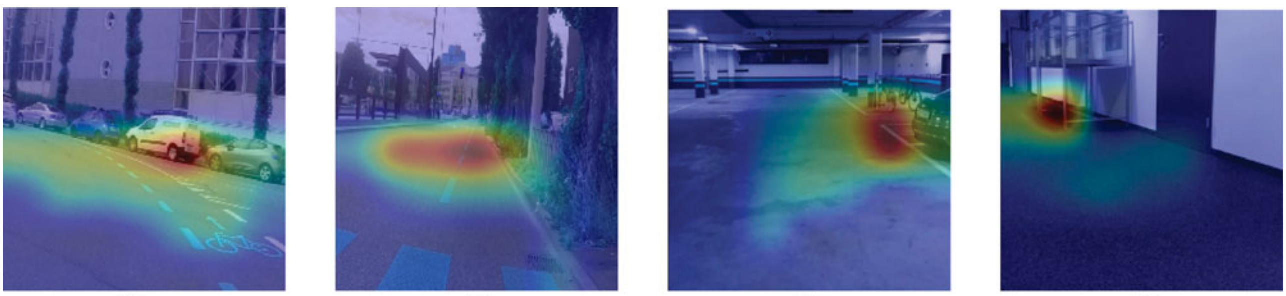
V. Discussion

Our methodology comes with the advantages and limitations inherent to both traditional and learning-based approaches. The advantages are that, using our simple learning and control scheme, we allow a drone to safely explore previously unseen

TABLE II
AVERAGE TRAVELLED DISTANCE BEFORE STOPPING: WE SHOW HERE NAVIGATION RESULTS USING THREE DIFFERENT POLICIES ON A SEVERAL ENVIRONMENTS

	Urban Environment			Generalization Environments		
Policy	Outdoor 1	Outdoor 2	Outdoor 3	High Altitude Outdoor 1	Corridor	Garage
Straight	23 m	20 m	28 m	23 m	5 m	18 m
Gandhi *et al.* [10]	38 m	42 m	75 m	18 m	**31 m**	23 m
DroNet (Ours)	**52 m**	**68 m**	**245 m**	**45 m**	27 m	**50 m**

Recall that [10] uses only collision probabilities, while DroNet uses also predicted steering angles, too. *High Altitude Outdoor 1* consists of the same path as *Outdoor 1*, but flying at 5 m altitude, as shown in Fig. 6.



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

Fig. 7. Activation maps: Spatial support regions for steering regression in city streets, on (a) a left curve and (b) a straight path. Moreover we show activations on (c) an indoor parking lot, and (d) an indoor corridor. We can observe that the network concentrates its attention to "line-like" patterns, which approximately indicate the steering direction.

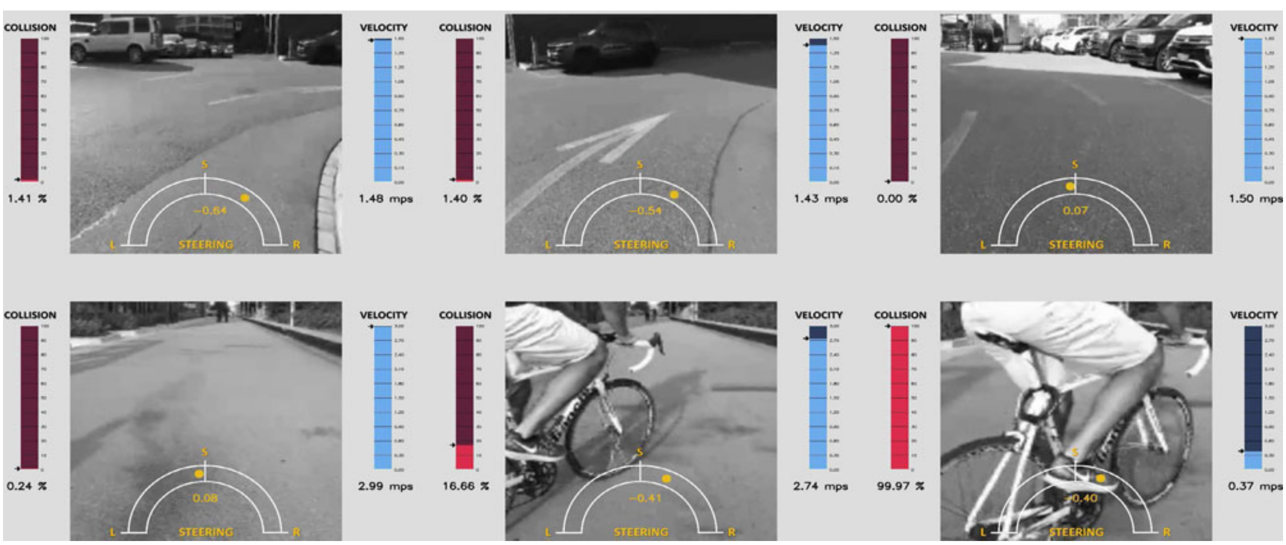


Fig. 8. DroNet predictions: The above figures show predicted steering and probability of collision evaluated over several experiments. Despite the diverse scenarios and obstacles types, DroNet predictions always follow common sense and enable safe and reliable navigation.

scenes while requiring no previous knowledge about them. More specifically, in contrast to traditional approaches, there is no need to be given a map of the environment, or build it online, pre-define collision-free waypoints and localize within this map. An advantage with respect to other CNN-based controllers [6], [9], [11]–[13] is, that we can leverage the large body of literature present on steering angle estimation [16], [17] on both the data and the algorithmic point of view. As shown in the experiments, this gives our method high generalization capabilities. Indeed, the flying policy we provide can reliably fly in non-trivial unseen scenarios without requiring any re-training or fine-tuning, as it is generally required by CNN-based approaches [11]. Additionally, the very simple and optimized network architecture can make our approach applicable to resource constrained platforms.

The limitations are primarily that the agile dynamics of drones is not fully exploited, and that it is not directly possible to explicitly give the robot a goal to be reached, as it is common in other CNN-based controllers [9], [13], [25]. There are several ways to cope with the aforementioned limitations. To exploit the drone agility, one could generate 3D collision-free trajectories, as *e.g.,* in [25], when high probability of collision is predicted. To generalize to goal-driven tasks, one could either provide the network with a rough estimate of the distance to the goal [26], or, if a coarse 2D map of the environment is available, exploit recent learning-based approaches developed for ground robots [27]. Moreover, to make our system more robust, one could produce a measure of uncertainty, as in [28]. In such a way, the system could switch to a safety mode whenever needed.

VI. Conclusion

In this letter, we proposed DroNet: a convolutional neural network that can safely drive a drone in the streets of a city. Since collecting data with a UAV in such an uncontrolled environment is a laborious and dangerous task, our model learns to navigate by imitating cars and bicycles, which already follow the traffic rules. Designed to trade off performance for processing time, DroNet simultaneously predicts the collision probability and the desired steering angle, enabling a UAV to promptly react to unforeseen events and obstacles. We showed through extensive evaluations that a drone can learn to fly in cities by imitating manned vehicles. Moreover, we demonstrated interesting generalization abilities in a wide variety of scenarios. Indeed, it could be complementary to traditional "map-localize-plan" approaches in navigation-related tasks, *e.g.,* search and rescue, and aerial delivery. For this reason, we release our code and datasets to share our findings with the robotics community.

Acknowledgment

For supplementary video see: https://youtu.be/ow7aw9H4BcA. The project's code, datasets, and trained models are available at: http://rpg.ifi.uzh.ch/dronet.html

References

[1] S. Scherer *et al.*, "River mapping from a flying robot: State estimation, river detection, and obstacle mapping," *Auton. Robots*, vol. 33, no. 1/2, pp. 189–214, 2012.

[2] N. Michael *et al.*, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *J. Field Robot.*, vol. 29, no. 5, pp. 832–841, 2012.

[3] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV," *J. Field Robot.*, vol. 33, no. 4, pp. 431–450, 2016.

[4] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *Proc. IEEE Int. Conf Robot. Autom.*, 2014, pp. 4974–4981.

[5] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization." in *Proc. Robot. Sci. Syst. XI*, Jul. 2015.

[6] S. Ross *et al.*, "Learning monocular reactive UAV control in cluttered natural environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 1765–1772.

[7] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," arXiv:1509.02971, 2015.

[8] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[9] A. Giusti *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 661–667, Jul. 2016.

[10] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Sep. 2017.

[11] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017.

[12] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "Plato: Policy learning using adaptive trajectory optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017.

[13] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2017, pp. 4241–4247.

[14] F. Sadeghi and S. Levine, "(cad)2rl: Real single-image flight without a single real image," in *Proc. Robot. Sci. Syst. XIII*, Jul. 2017.

[15] M. Mancini, G. Costante, P. Valigi, T. A. Ciarfuglia, J. Delmerico, and D. Scaramuzza, "Towards domain independence for learning-based monocular depth estimation," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1778–1785, Jul. 2017.

[16] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3530–3538.

[17] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," in *IEEE Int. Conf. Comput. Vision (ICCV)*, Oct. 2017.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[19] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Int. Conf. Learn. Representations*, 2015.

[21] Udacity, "An Open Source Self-Driving Car," 2016, [Online]. Available: https://www.udacity.com/self-driving-car

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556, 2014.

[24] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Oct. 2017.

[25] S. Yang, S. Konam, C. Ma, S. Rosenthal, M. Veloso, and S. Scherer, "Obstacle avoidance through deep networks based intermediate perception," arXiv:1704.08759, 2017.

[26] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1527–1533.

[27] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, "Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017.

[28] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," in *Proc. Robot., Sci. Syst.*, 2017.