A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots

Alessandro Giusti[1], Jérôme Guzzi[1], Dan C. Cireşan[1], Fang-Lin He[1], Juan P. Rodríguez[1]
Flavio Fontana[2], Matthias Faessler[2], Christian Forster[2]
Jürgen Schmidhuber[1], Gianni Di Caro[1], Davide Scaramuzza[2], Luca M. Gambardella[1]

***Abstract*—We study the problem of perceiving forest or mountain trails from a single monocular image acquired from the viewpoint of a robot traveling on the trail itself. Previous literature focused on trail segmentation, and used low-level features such as image saliency or appearance contrast; we propose a different approach based on a Deep Neural Network used as a supervised image classifier. By operating on the whole image at once, our system outputs the main direction of the trail compared to the viewing direction. Qualitative and quantitative results computed on a large real-world dataset (which we provide for download) show that our approach outperforms alternatives, and yields an accuracy comparable to the accuracy of humans that are tested on the same image classification task. Preliminary results on using this information for quadrotor control in unseen trails are reported. To the best of our knowledge, this is the first paper that describes an approach to perceive forest trials which is demonstrated on a quadrotor micro aerial vehicle.**

***Index Terms*—Visual-Based Navigation; Aerial Robotics; Machine Learning; Deep Learning**

Videos and Dataset

A narrated video summary of this paper, additional figures, videos and the full training/testing datasets are available at http://bit.ly/perceivingtrails.

I. Introduction

AUTONOMOUSLY following a man-made trail (such as those normally traversed by hikers or mountain-bikers) is a challenging and mostly unsolved task for robotics. Solving such problem is important for many applications, including wilderness mapping [1] and search and rescue; moreover, following a trail would be the most efficient and safest way for a ground robot to travel medium and long distances in a forested environment: by their own nature, trails avoid excessive slopes and impassable ground (e.g. due to excessive vegetation or wetlands). Many robot types, including wheeled, tracked and legged vehicles [2], are capable of locomotion along real-world trails. Moreover, Micro Aerial Vehicles (MAVs) flying under the tree canopy [3], [4] are a compelling and realistic option made possible by recent technological advances (such as collision-resilience [5]).

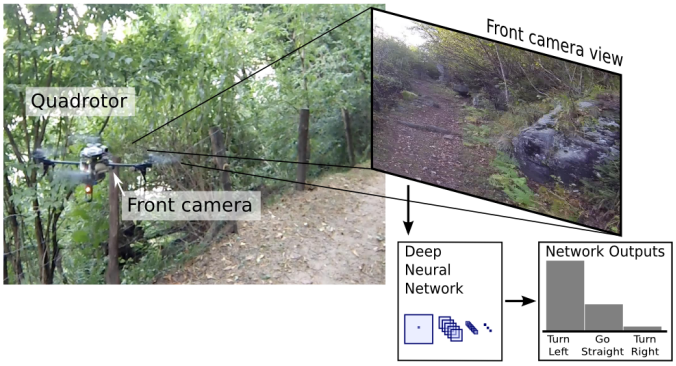

Fig. 1: Our quadrotor acquires the trail images from a forward-looking camera; a Deep Neural Network classifies the images to determine which action will keep the robot on the trail.

In order to successfully follow a forest trail, a robot has to *perceive* where the trail is, then *react* in order to stay on the trail. In this paper, we will describe a machine-learning approach to the visual perception of forest trails and show preliminary results on an autonomous quadrotor. We consider as input a monocular image from a forward-looking camera.

Perceiving real-world trails in these conditions is an extremely difficult and interesting pattern recognition problem (see Figure 2), which is often challenging even for humans (e.g., losing a trail is a common experience among casual hikers). Computer Vision and Robotics literature mainly focused on paved road [6], [7], [8], [9] and forest/desert road perception [10].

The latter is a significantly more difficult problem than the former, because unpaved roads are normally much less structured than paved ones: their appearance is very variable and often boundaries are not well defined. Compared to roads, the perception of trails poses an even harder challenge, because their surface appearance can change very frequently, their shape and width is not as constrained, and they often seamlessly blend with the surrounding area (e.g. grass).

Several previous works [11], [12] dealing with trail perception solved a *segmentation* problem, i.e., aimed at determining which areas of the input image correspond to the image of the trail. In order to solve this task, one needs to explicitly define *which visual features characterize a trail*. Rasmussen et al.

Manuscript received: August, 31, 2015; Accepted December, 18, 2015.
This paper was recommended for publication by Editor Jana Kosecka upon evaluation of the Associate Editor and Reviewers' comments. This research was supported by the Swiss National Science Foundation (SNSF) through: the National Centre of Competence in Research (NCCR) Robotics (www.nccr-robotics.ch); and the Supervised Deep / Recurrent Nets grant (project code 140399)

[1] Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, Switzerland.
[2] Robotics and Perception Group (RPG), University of Zurich, Switzerland
Digital Object Identifier (DOI): see top of this page.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2015.2509024, IEEE Robotics and Automation Letters

Fig. 2: Three images from our dataset. Given an image, we aim to determine the approximate direction the trail is heading with respect to the viewing direction (respectively, left, right and straight ahead).

[11] relied on *appearance contrast*, whereas Santana et al. adopted image conspicuity [12]; both features are conceptually similar of image saliency [13]. For every pixel of an image, saliency quantifies how much such pixel visually "stands out" from the rest; for example, the pixels belonging to a small colored object on an uniform background will be characterized by an high saliency with respect to the saliency computed for the background pixels. If we assume that the trail image exhibits some sort of marked visual difference with respect to its surroundings, then saliency will be high for trail pixels, and low elsewhere. This information, which by itself is expected to be very noisy, was aggregated [14] with a number of simple (symmetric, triangular shape [11]) or complex (spatial-temporal integration based on virtual ants [12]) geometry priors in order to infer the trail position and direction in the image, thus producing a rough segmentation of the trail. A similar segmentation system using data from stereo omnidirectional cameras and a laser rangefinder was also implemented [15] to steer a wheeled robot along a trail.

In this paper, we follow a different approach and cast the trail perception problem as an *image classification* task: we estimate the approximate direction the trail with respect to the direction of view by adopting a supervised machine learning approach based on Deep Neural Networks (DNNs), a state-of-the-art *deep learning* technique that operates directly on the image's raw pixel values (Section III-B). DNNs have recently emerged as a powerful tool for various computer vision tasks (e.g. object classification [16], [17], biomedical image segmentation [18]), often outperforming other techniques. One of the advantages of DNNs over common alternatives for supervised image classification is *generality*: in fact, features are learned directly from data, and do not have to be chosen or designed by the algorithm developers for the specific problem on which they are applied.

Machine learning techniques have been used for a long time [19], [20] to map visual inputs to actions. When the goal is obstacle avoidance, several works [21], [22], [23] obtained good results with simple biologically-inspired controllers based on optical flow features. More recently, deep learning techniques have also been adopted by Sermanet, Hadsell et al. [24], [25] for autonomous navigation of a robot in various nonstructured environments; in these works, the terrain visible in front of the robot is classified for traversability, which provides high-level information for obstacle-free path planning. Ross et al. [3] used imitation learning [26], [27] to steer a quadrotor to avoid trees in an outdoor environment; the controller is previously trained by manually piloting the robot for a short time. In our case, the visual perception task is harder (Fig. 2) since real-world trails have much more appearance variability than trees at close range. This requires a more powerful classifier to be trained with a significantly larger training dataset, which would be impractical to acquire by manually piloting a robot: therefore, we acquire the dataset offline, by means of a simple but efficient expedient introduced in Section III-A.

A. Contributions

Our main contributions are:

- a trail perception technique based on Deep Neural Networks (Section III-B) which bypasses the challenging problem of determining the characteristic features of a trail;
- a large dataset, provided for download, efficiently acquired on real-world hiking trails (Section III-A), that was used to train and test the Deep Neural Network;
- a quantitative comparison of our approach with state-of-the-art methods and humans on an unseen testing set (Section IV).
- The system implementation and demonstration (see video attachment) on a quadrotor following a previously-unseen trail.

II. Problem Formulation

Consider a generic scene with a single trail in a wilderness setting. Our input is an image acquired by a camera situated above the trail. In the following, we assume the viewpoint height is similar to the average height of a person (approximately 1.7m), because it is high enough to provide a good view over the surrounding ground, but still a realistic sensor position for medium-sized all-terrain ground robots; moreover, we can expect that this height is mostly free of obstacles on trails in forested areas, and as such, a reasonable choice for MAVs.

Let $\vec{v}$ be the direction of the camera's optical axis; we assume that $\vec{v}$ lies on the horizontal plane. Furthermore, let $\vec{t}$ be the dominant direction of the trail: we define $\vec{t}$ as the (horizontal) direction towards which a hiker would start walking if standing at the position of the robot, with the goal of remaining on the trail (see Figure 3).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2015.2509024, IEEE Robotics and Automation Letters

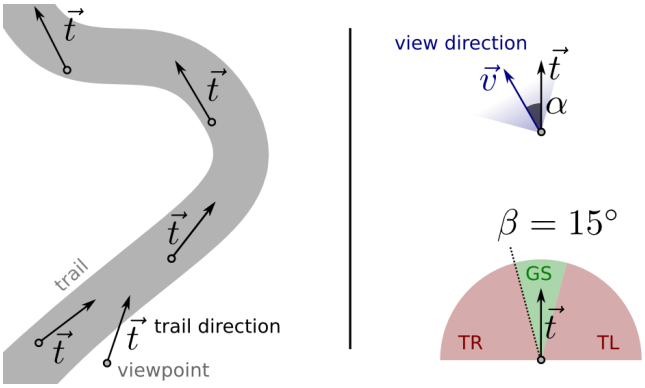

Fig. 3: Left: given a point, $\vec{t}$ is the direction a hiker would walk in order to follow the trail. Right: illustration of $\vec{v}$, α, β (see text).

Let α be the signed angle between $\vec{v}$ and $\vec{t}$: we consider three classes, which correspond to three different actions that the (human or robotic) carrier of the camera should implement in order to remain on the trail, assuming that the camera is looking at the direction of motion.

Turn Left (TL) if $-90° < \alpha < -\beta$; i.e., the trail is heading towards the left part of the image.

Go Straight (GS) if $-\beta \leq \alpha < +\beta$; i.e., the trail is heading straight ahead, at least in the close range.

Turn Right (TR) if $+\beta \leq \alpha < +90°$; i.e., the trail is heading towards the right part of the image.

Given the input image, our goal is to classify it in one of these three classes. In the following, we consider $\beta = 15°$.

Note that, in case the absolute value of α is large, the trail may entirely lie outside of the camera field of view; e.g., this happens if the robot is looking in a perpendicular direction with respect to the trail. In that case, the image only allows us to infer that the true class is *not* Go Straight (GS).

III. Visual Perception of Forest Trails

We solve the problem as a supervised machine learning task, which is extremely challenging because of the wide appearance variability of the trail and its surroundings: perceptions are heavily affected by lighting conditions, vegetation types, altitude, local topography, and many other factors. We deal with such challenges by gathering a large and representative labeled dataset, covering a large variety of trails and a long distance on each.

A. Dataset

To acquire such a dataset, we equip a hiker with three head-mounted cameras: one pointing 30° to the left, one pointing straight ahead, and one pointing 30° to the right; the fields of view of the three cameras partially overlap and cover approximately 180 degrees (see Figure 3). The hiker then swiftly walks a long trail, by taking care of always looking straight along its direction of motion. The dataset is composed by the images acquired by the three cameras.

Each image is labeled, i.e. it is associated to its ground truth class. Because of the definition of our classes, all images acquired by the central camera are of class GS: in fact, they were acquired while the hiker was walking along the trail, and looking straight ahead (i.e., $\alpha \approx 0°$) in the direction of motion. Conversely, the right looking camera acquires instances for the TL class, with $\alpha \approx 30°$; and the left-looking camera acquires instances of the TR class ($\alpha \approx -30°$).

The dataset[1] is currently composed by 8 hours of 1920×1080 30fps video acquired using three GoPro Hero3 Silver cameras in the configuration outlined above, and covers approximately 7 kilometres of hiking trails acquired at altitudes ranging from 300m to 1,200m, different times of the day and weather. Exposure, dynamic range and white balance are automatically controlled by the cameras. To avoid long exposure times, which would yield to motion-blur, all sequences are acquired during daytime, excluding twilight. Many different trail types and surroundings are represented, ranging from sloped narrow alpine paths to wider forest roads. Acquisitions are normally uninterrupted unless for technical reasons or to avoid long sections on paved roads; this ensures that the dataset is representative not only of ideal, "clean" trails but also of frequent challenging or ambiguous spots often observed in the real world. Synchronized GPS and compass information has been recorded for most sequences, but is unused at the moment.

The dataset has been split in disjoint *training* (17,119 frames) and *testing* (7,355 frames) sets. The split was defined by carefully avoiding that the same trail section appears in both the training and testing set. The three classes are evenly represented within each set.

B. Deep Neural Networks for Trail perception

We use a DNN [17] as an image classifier, and adopt the network architecture detailed in Figure 5, that has been shown to perform well when applied to a large amount of image classification problems [17]; in particular, we consider a matrix of $3 \times 101 \times 101$ neurons as the input layer, followed by a number of hidden layers and three output neurons.

The input image is first anisotropically resized to a size of 101×101 pixels; the resulting $3 \times 101 \times 101$ RGB values are directly mapped to the neurons in the input layer. For a given input, the DNN outputs three values, representing the probability that the input is of class TL, GS, TR, respectively.

Training a net: The 17,119 training frames are used as training data. The training set is augmented by synthesizing left/right mirrored versions of each training image. In particular, a mirrored training image of class TR (TL) yields a new training sample for class TL (TR); a mirrored GS training sample yields another GS training sample. Additionally, mild affine distortions ($\pm 10\%$ translation, $\pm 15°$ rotation, $\pm 10\%$ scaling) are applied to training images to further increase the number of samples. The DNN is trained using backpropagation for 90 epochs, which requires about 3 days on a workstation equipped with an Nvidia GTX 580 GPU. The learning rate is initially set to 0.005, then scaled by a factor of 0.95 per epoch.

[1]The whole dataset is available as supplementary material [28]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2015.2509024, IEEE Robotics and Automation Letters

Fig. 4: *Left:* stylized top view of the acquisition setup; *Right:* our hiker during an acquisition, equipped with the three head-mounted cameras.

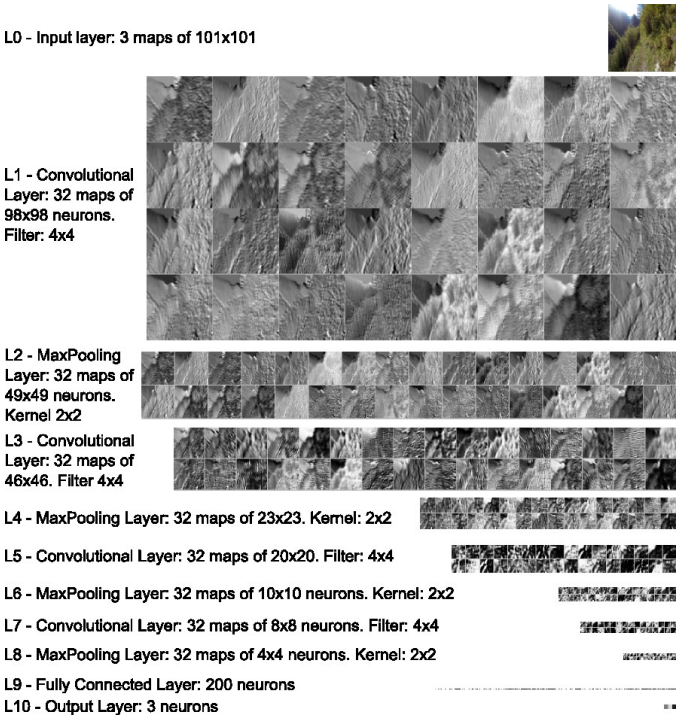

Fig. 5: Architecture for the DNN [17] used in our system, and representation of the maps in each layer

DNN architecture: A DNN is a feed-forward connectionist model built out of successive pairs of convolutional and max-pooling layers, followed by several fully connected layers (the architecture adopted in our system is illustrated in Figure 5). Input pixel intensities, rescaled to the range $[-1, 1]$, are passed through this complex, hierarchical feature extractor. The fully connected layers at the end of the network act as a general classifier. The free parameters (weights), initialized with random numbers from an uniform distribution in the range $[-0.05, 0.05]$, are jointly optimized using stochastic gradient descent to minimize the misclassification error over the training set.

Convolutional layers [29] perform 2D convolutions of their input maps with a rectangular filter. When the previous layer contains more than one map, the results of the corresponding convolutions are summed and transformed by a scaled hyperbolic tangent activation function. Higher activations will occur where the filter better matches the content of the map, which can be interpreted as a search for a particular feature. The output of the *max-pooling (MP) layers* [30] is formed by the maximum activations over non-overlapping square regions. MP layers decrease the map size, thus reducing the network complexity. MP layers are fixed, non-trainable layers selecting the winning neurons. Typical DNNs are much wider than previous CNN, with many more connections, weights and non-linearities. A GPU implementation [31] is used to speed up training. The *output layer* is a fully connected layer with one neuron per class (i.e. TL, GS and TR), activated by a softmax [32] function. Each output neuron's activation can be interpreted as the probability of the input image belonging to that class.

IV. Experimental Results

Performance metrics: We use the testing set defined in Section III-A (7355 images) in order to compute performance metrics for different classification techniques.

For the three-class classification problem defined in Section II, we compute the absolute accuracy (i.e. fraction of correctly classified images) and the confusion matrix.

We additionally consider a derived two-class classification problem, on which additional, more robust performance measures can be defined. In the two-class problem, one has to decide whether an input image is of class GS or not (i.e., whether a trail is visible straight ahead, or not). The image is classified as GS if and only if $P(\text{GS}) > T$. We report the accuracy of the binary classifier for $T = 0.5$, and the corresponding precision, recall, and the area under the ROC curve (the last is a robust metric and does not depend on the choice of T).

Comparisons: In the following, we compute the performance of our technique (**DNN**), where $P(\text{TL})$, $P(\text{GS})$ and $P(\text{TR})$ are directly computed by applying the DNN model to the input frame. We compare its performance to three alternatives.

1. **Simple Saliency-based Model.** We compute saliency maps of the input frame using Itti's model [33], as in Santana et al. [12]. This map is computed on the image hue only, which preliminary experiments shown to be the configuration where saliency is most correlated to trail location. The saliency map is discretized to 16×9 blocks, and the average saliency for each block yields a 144-dimensional feature vector. A SVM model with an

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2015.2509024, IEEE Robotics and Automation Letters

TABLE I: Results for the three-class problem.

	DNN	Saliency	[12]	Human1	Human2
Accuracy	85.2%	52.3%	36.5%	86.5%	82.0%

TABLE II: Results for the two-class problem.

	DNN	Saliency	[12]	Human1	Human2
Accuracy	95.0%	73.6%	57.9%	91.0%	88.0%
Precision	95.3%	60.9%	39.8%	79.7%	84.0%
Recall	88.7%	46.6%	64.6%	95.1%	81.6%
AUC	98.7%	75.9%	–	–	–

RBF kernel is learned from the training set to map this feature vector to $P(\text{TL})$, $P(\text{GS})$ and $P(\text{TR})$.

2. **The method by Santana et al..** The algorithm in [12] is applied to the frames extracted from our videos (50 iterations per frame) and its output trail soft segmentation is sampled at each of the testing frames. In order to map a class to a segmentation, we follow the quantitative evaluation in [12]: a single representative point for the trail is computed as the centroid of the largest connected component in the binarized trail probability map; the threshold is computed as $0.85 \cdot M$, where M is the maximum value of the probability map. Then, we classify an image as TR (respectively, TL) if the x coordinate of the point is larger than $(0.5+k) \cdot W$ (respectively, smaller than $(0.5-k) \cdot W$), where W is the image width; else, the image is classified as SC. k is chosen in order to optimize the accuracy of the resulting classifier.
3. **Two human observers,** each of which is asked to classify 200 randomly sampled images from the testing set in one of the three classes.

Tables I and II report quantitative results for the three- and two-class problems, respectively. We observe that DNN methods perform comparably to humans, meaning that they manage to extract much of the information available in the inputs. The Simple Saliency Model and [12] perform poorly on this data, which is expected as image saliency is not correlated to the trail location in most of our dataset.

Failure Cases and Qualitative Results

Figure 6 reports success and failure cases on the testing set. We observe that instances which are easy for our system are also trivial for human observers, whereas instances that our system failed (third and fourth row) are in fact difficult and ambiguous cases also for humans.

We also implemented an additional qualitative test using videos acquired from a Samsung Galaxy SIII cellphone on a sloped forest trail in a different region than those in which our dataset was acquired. These videos have a much lower field of view than the dataset videos (about 60° vs 120°), are highly compressed, and frequently exhibit under/over exposed areas due to the limited dynamic range of the sensor. The viewing direction rotates frequently in such a way that the trail is not always in the center of the frame, and is often not visible at all. Figure 7 reports three representative frames from one of these videos (available as supplementary material [28]), overlayed with the outputs of our system.

Control of a Robot for Autonomous Trail Following

It is interesting to study how well a robot navigates in the real world when using exclusively the information provided by our vision system as input. In order to investigate this, we implemented a simple reactive controller which translates our system's outputs to control signals as follows. *Yaw* (i.e. steering) is proportional to $P(\text{TR}) - P(\text{TL})$; a positive value steers the robot to the right, and a negative value steers the robot to the left. *Speed* is proportional to $P(\text{GS})$.

We tested such controller on two platforms. 1) A Parrot ARDrone controlled by a laptop (Figure 8, left). 2) A standalone quadrotor [34] (Figure 8, center and right) equipped with: a forward-looking MatrixVision mvBlueFox global shutter color camera (752×480 pixels), used for trail detection; a down-looking MatrixVision mvBlueFox global shutter grayscale camera (752×480 pixels), used for feeding a Semi-direct monocular Visual Odometry (SVO) pipeline [35]; an onboard Odroid-U3 system for image processing and control. The Odroid processor runs both our deep neural network and the SVO pipeline simultaneously at more than 15fps; the former component produces high-level velocity commands, whereas the latter is used for feedback position control.

The main problem we observed during our tests in realistic conditions was the much lower image quality acquired by the quadrotors' cameras as compared to the gopro images in the training dataset; this yielded a lower performance of the classifier compared to the testing datasets. This was especially apparent in situations with strong sky-ground contrast, as the dynamic range of the mvBlueFox camera cannot capture well-exposed footage. We also observed that the quadrotor is often unable to negotiate trails if there is not enough free space besides the trail centerline: in fact, if the yaw is roughly correct, the classifier compensates a lateral shift only when the quadrotor is about one meter off the centerline; because the current pipeline does not implement explicit obstacle detection and avoidance, this results in frequent crashes on trails with narrow free space. On wide trails with even lighting conditions, the robot was able to successfully follow the trail for a few hundreds of meters.

V. CONCLUSIONS

We trained a Deep Neural Network for visually perceiving the direction of an hiking trail from a single image. Trained on a large real-world dataset and tested on a disjoint set, the system performs better than alternatives and comparably to humans. By operating on the raw RGB frames, we bypass the need to define characteristic features of trails, which is a very difficult task given the huge variability of their appearance. The system was implemented and demonstrated on a real quadrotor. Preliminary field tests showed promising results. The training and testing datasets are provided for download.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2015.2509024, IEEE Robotics and Automation Letters

(a) GS frames with highest $P(\text{GS})$, i.e. frames where the path is easily found as being straight ahead

(b) not-GS images with lowest $P(\text{GS})$, i.e. frames where the path is easily found as *not* being straight ahead

(c) GS images with lowest $P(\text{GS})$, i.e. failure cases where the path should be straight ahead but was not detected as such

(d) not-GS images with highest $P(\text{GS})$, i.e. failure cases where the path is not straight ahead but was detected as such

Fig. 6: Success and failure cases. More examples are reported in supplementary material [28].

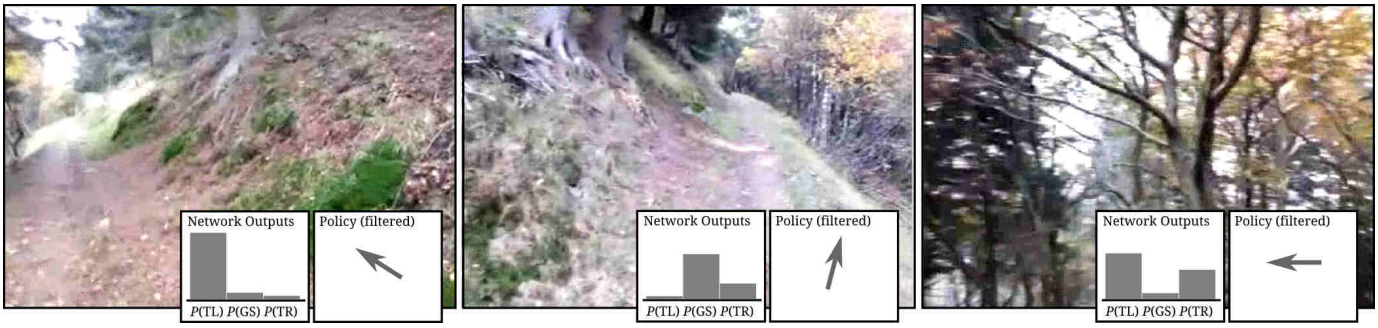

Fig. 7: Three representative frames form the cellphone video robustness test. For each frame, we report the raw network outputs for that frame (bar graph), and the motion policy (arrow) derived from such outputs averaged over the previous 10 frames (see text). The rightmost frame is acquired when looking sideways, so the trail is not visible; the DNN is then rightfully confused among TR and TL, but returns a very small value for $P(\text{GS})$.

.

Fig. 8: Images from preliminary field testing. Left: the Parrot ARDrone controlled by a laptop. Center, right: the standalone quadrotor with SVO and onboard processing [34].

.

REFERENCES

[1] Google, "Maps trekker." www.google.com/maps/about/partners/streetview/trekker.

[2] M. Hutter *et al.*, *StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion*. 2012.

[3] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Proc. ICRA*, 2013.

[4] A. J. Barry, A. Majumdar, and R. Tedrake, "Safety verification of reactive controllers for uav flight in cluttered environments using barrier certificates," in *Proc. ICRA*, 2012.

[5] A. Briod, P. Kornatowski, J.-C. Zufferey, and D. Floreano, "A collision-resilient flying robot," *Journal of Field Robotics*, vol. 31, no. 4, 2014.

[6] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine Vision and Applications*, pp. 1–19, 2012.

[7] E. D. Dickmanns and A. Zapp, "A curvature-based scheme for improving road vehicle guidance by computer vision," in *Mobile Robots, SPIE Proc. Vol. 727, Cambridge, Mass.*, pp. 161–168, 1986.

[8] E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen, "The seeing passenger car 'VaMoRs-P'," in *Proc. Int. Symp. on Intelligent Vehicles '94, Paris*, pp. 68–73, 1994.

[9] E. D. Dickmanns, *Dynamic Vision for Perception and Control of Motion*. Springer, 2007.

[10] S. Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, 2006.

[11] C. Rasmussen, Y. Lu, and M. Kocamaz, "Appearance contrast for fast, robust trail-following," in *Proc. IROS*, pp. 3505–3512, 2009.

[12] P. Santana, L. Correia, R. Mendonça, N. Alves, and J. Barata, "Tracking natural trails with swarm-based visual saliency," *Journal of Field Robotics*, vol. 30, no. 1, pp. 64–86, 2013.

[13] A. Toet, "Computational versus psychophysical bottom-up image saliency: A comparative evaluation study," *IEEE Transactions on PAMI*, vol. 33, no. 11, pp. 2131–2146, 2011.

[14] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *Proc. ECCV*, pp. 581–594, 2006.

[15] C. Rasmussen, Y. Lu, and M. Kocamaz, "A trail-following robot which uses appearance and structural cues," in *Field and Service Robotics*, pp. 265–279, Springer, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, pp. 1106–1114, 2012.

[17] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification," in *Proc. CVPR*, pp. 3642–3649, 2012.

[18] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," in *Proc. NIPS*, pp. 2852–2860, 2012.

[19] J. Schmidhuber and R. Huber, "Learning to generate artificial fovea trajectories for target detection," *International Journal of Neural Systems*, vol. 2, no. 1 & 2, pp. 135–141, 1991.

[20] D. A. Pomerleau, "Neural network perception for mobile robot guidance," 1993.

[21] J. Conroy, G. Gremillion, B. Ranganathan, and J. S. Humbert, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Autonomous robots*, vol. 27, no. 3, 2009.

[22] M. V. Srinivasan, "Visual control of navigation in insects and its relevance for robotics," *Current opinion in neurobiology*, vol. 21, no. 4, pp. 535–543, 2011.

[23] A. Beyeler, J.-C. Zufferey, and D. Floreano, "Vision-based control of near-obstacle flight," *Autonomous robots*, vol. 27, no. 3, 2009.

[24] P. Sermanet *et al.*, "A multirange architecture for collision-free off-road robot navigation," *Journal of Field Robotics*, vol. 26, no. 1, 2009.

[25] R. Hadsell *et al.*, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[26] J. Kober and J. Peters, "Learning motor primitives for robotics," in *Proc. ICRA*, pp. 2112–2118, 2009.

[27] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011.

[28] "Supplementary material." http://bit.ly/perceivingtrails, 2015.

[29] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient BackProp," in *Neural Networks: Tricks of the trade* (G. Orr and M. K., eds.), Springer, 1998.

[30] D. Scherer, A. Müller, and S. Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," in *Proc. International Conference on Artificial Neural Networks*, 2010.

[31] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification," in *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 1237–1242, 2011.

[32] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[33] L. Itti, C. Koch, E. Niebur, *et al.*, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on PAMI*, vol. 20, no. 11, pp. 1254–1259, 1998.

[34] 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 micro aerial vehicle," *Journal of Field Robotics*, 2015.

[35] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. ICRA*, 2014.