ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/324095604

A deep learning approach towards autonomous flight in forest environments

Conference Paper · February 2018

DOI: 10.1109/CONIELECOMP.2018.8327189

CITATIONS
19

READS
707

4 authors, including:

Leticia Oyuki Rojas Pérez
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
17 PUBLICATIONS **79** CITATIONS
SEE PROFILE

Jose Martinez-Carranza
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
127 PUBLICATIONS **554** CITATIONS
SEE PROFILE

Israel Cruz Vega
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
52 PUBLICATIONS **328** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Chaos Prediction and Control View project

Project RedICA: Red temática CONACYT en Inteligencia Computacional Aplicada View project

All content following this page was uploaded by Israel Cruz Vega on 14 November 2018.

The user has requested enhancement of the downloaded file.

A Deep Learning Approach Towards Autonomous Flight in Forest Environments

Sinahi Dionisio-Ortega, L. Oyuki Rojas-Perez, Jose Martinez-Carranza and Israel Cruz-Vega

***Abstract*—Nowadays unmanned aerial vehicles (UAV's) are increasingly considered in several engineering and safety applications to explore unknown environments. Forest environments are among these environments of interest where autonomous exploration and navigation is desirable with the caveat that GPS may not be accessible for localization of the drone. Motivated by the latter, in this paper we present a methodology for autonomous navigation of UAV in forest environments, paying particular attention to competences desired in autonomous navigation, namely detection and evasion of trees. Obstacle detection is performed using a deep neural network approach trained with a limited database but that exploits transfered learning from a well-known network architecture called Alexnet. In addition to the detection of a tree, a direction of avoidance is also recognized, this is, evasion to the left or to the right is obtained with the same learning scheme. Our approached was assessed in simulation by using the simulator Gazebo and also in a real outdoors scenario populated with trees.**

I. Introduction

There exist several scenarios where exploration or navigation of the scene is required but with hazardous conditions for humans access. In addition, the terrain may not have the conditions for wheeled or legged robots to move easily. For these cases, the use of UAV's (commonly known as drones) has become a great alternative [1].

In this work, we are interested in those environments where a drone is capable of flying autonomously. In particular, in those environments where GPS may not be reliable or not accessible at all. Such scenario calls for a solution where the drone makes use of onboard sensors as the only mean to observe the environment and make decisions for the flight.

An important competence in autonomous flight is that of sense-and-avoid, proven to be vital for the drone not to collide with obstacles present in the environment. In particular, we are interested in developing such capability for a drone that flies autonomously in a forest environment.

For the above, we assume that GPS is unavailable and that the drone is only equipped with an onboard camera, an Inertial Measurement Unit (IMU) and altimeter for knowing the drone height w.r.t. the ground.

To addressed the problem stated before, we propose an approach based on a deep learning scheme. Constructing a Convolutional Neural Network, typical of deep learning works, involves creating a large training dataset which can take a considerable amount of time. Moreover, the training of the architecture may also take a long time plus considerable processing capabilities such as the use of expensive dedicated hardware namely, GPU processors or even clusters of GPU's.

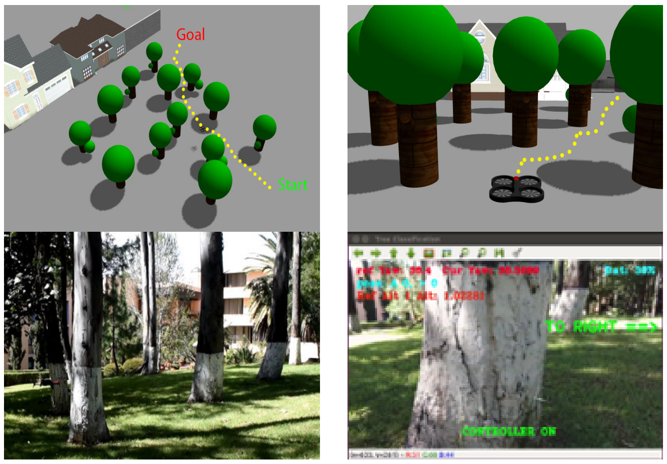


Fig. 1: We have developed a system based on deep learning for obstacle detection and avoidance in a forest environment where the trees are recognized as obstacles during autonomous flight. We carried out evaluations in simulation, images in first row, and in a real environment, second row.

A popular methodology for deep learning has become the use of an already trained architecture, whose learning is transferred to a smaller neural networked tailored to provide classification for the desired classes. In this spirit, we have used the AlexNet network which classifies the images captured by the drone's onboard camera. The output vector produced by AlexNet is passed to a customized set of final layers for the classification of three classes: (1) Free-space (2) Obstacle-Close (3) Obstacle-Very-Close.

Therefore, our deep learning approach is coupled to a state machine system that reads the output of the network and translates it into flight commands to be sent to the drone for its flight across the forest environment, thus enabling it to detect a tree when the drone gets close to an obstacle, which is used to slow down the flight speed. When the tree is deemed to be *Very-Close* then the system starts an evasion maneuver where the same network is used to assess which side, left or right, the evasion should be led. Our approach was tested in the well known simulated Gazebo. We also conducted initial tests in a real outdoor environment populated with trees.

To describe our approach, the rest of the paper is organized as follows. Section II describes the related work. Section III provides the theoretical background necessary for the development of the simulation and control tasks. Section IV shows the design of the experimental part and the results obtained to finally, present in section V the conclusion and future interests related to this work.

II. Related work

Among the specialized control algorithms developed for UAV's, autonomous control is currently a subject of great importance like the Sense-and-Avoid System (SA). In this system, the UAV's makes use of sensors onboard the machine to gather information, and a decision mechanism tries to produce an appropriate response to the incoming data, which finds a feasible path that finally, the flight controller uses to drive the vehicle to the avoidance maneuver.
For instance, some terrestrial robots have been highlighted in this area. In [2], it is proposed a robot track compiling the knowledge of an expert avoiding collisions with different obstacles. The prediction of the avoidance directions is performed by a previously trained convolutional neural network (CNN) with the human expertise. Following the use of expert knowledge and the facilities of fuzzy logic to represent this information in the form of rules, Olivares et al. [3], propose an approach with fuzzy logic where a UAV receives the data of an image processor to detect and trace the objectives in the environment. The evasion strategies result from an optimizing control process using cross-entropy. The proposed work in [4] presents the use of a CNN to estimate the depth of an image. Followed by the design of a control algorithm based on arbitrary behavior. The vehicle moves away from the presented obstacles by analyzing the created depth map. In [5], the images are classified employing deep learning for flights in a forest path and centered on the perception of the environment. The deep learning architecture presents three output classes: turn right, go straight and turn left, allowing the quadrotor to have an appropriate autonomous flight and avoiding collisions.

In general, through a revision of literature it is noted that the computational intelligence algorithms improve the control strategies with exceptional properties like learning, tolerance to imprecision and uncertainty, and adaptability to changes in the environment, among others. Many of the works mentioned so far, handle deep learning architectures as the primary tool for image processing, forming the base to synthesize control policies of avoidance. Although there are other approaches to novel technologies like the use of radars, acoustic sensors, laser scanners or light detection and ranging (LiDAR), it is necessary consider the high power consumption and the fact that they are not often found in micro/small size UAV's.

III. System Framework

In this section we describe the deep learning approach employed in this work. We also describe the evasion algorithm based on the output of the deep learning network.

A. Deep Neural Networks and AlexNet topology

The neural networks gave way to an even higher and complex study called deep learning; this branch uses as main tool deep neural networks (DNN) that are primarily neural networks with several hidden layers [6]. The DNNs have high-level learning capabilities and are used to a great extent for the classification of images with particular characteristics. An essential tool of DNNs is the use of convolutional layers, which have the capacity to generate a complex abstraction of the input data.

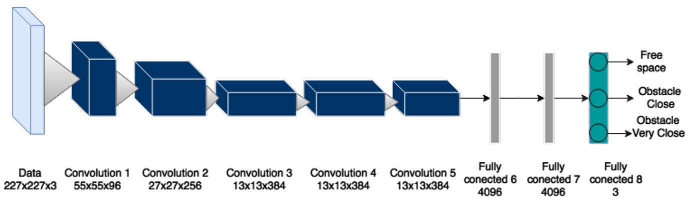


Fig. 2: Deep neural network modified based on AlexNet, with three classes at the output.

There are many popular topologies for deep neural networks; mainly this work uses a topology called AlexNet. AlexNet was initially occupied for the ImageNet Large-Scale Visual Recognition Challenge 2010 (ILSVRC-2010) classifying 1.2 million high-quality images in 1000 different classes [7], that in our case the structure of the last layer was changed to obtain only three classes. The structure of the modified network is shown in Figure 2, using the adjustment of the measures proposed by Caffe-framework to 227x227 pixels for the input images.

Caffe is a popular framework employed in deep learning [8], which allows the training of these network structures with images and is one of the best alternatives to other efficient non-free software such as Matlab [9]. Widely recommended, Caffe is easy for executable applications and additionally to be open source, it has pre-trained models for fine-tuning popular network topologies and which in turn allows execution in c++, python, and Matlab. The programming pertinent to the project is realized in the language c++ since regarding speed exceeds the other two languages allowed by the framework Caffe, as well being a language compatible with ROS libraries. The models trained by Caffe are executed utilizing OpenCV that is a free library for artificial vision.

B. Autonomous Flight and Evasion algorithm

The evasion algorithm decides a possible direction to avoid the obstacle. For the latter, the previously trained AlexNet network output is used, see Figure 2. To carry out the autonomous flight, we designed a state machine based on the interpretation of the network, see Figure 3.

To describe the logic behind this state machine, it must be assumed that during all the flight mission images are obtained from the onboard camera in a frame-to-frame basis and that every single image is passed to the network in order to obtain one of the three possible frame classifications. The classification will output a number according to the class:(0) Free-Space; (1) Obstacle-Close; (2) Obstacle-Very-Close.

Thus, at the outset, the vehicle is commanded to take off (Start state in the state machine) and once a desired height and heading is achieved then a classification value is read from the network. For the sake of simplicity, for this work we also assumed that free space will followed immediately after taking off, hence the drone will fly forwards with a given *fast* speed, reading at all times the classification values returned by the network. If the classification value changes to 1 then the it means that *the tree is close*, hence the drone is commanded by the state machine to reduce its speed to a *slow* speed, thus continuing its flight forward but at slower pace.

If the classification value changes to 2 it means *the tree is very close* and hence an evasion maneuver has to be performed. For this work, our evasion maneuver consist of a first step, which is to enter into hovering for some time and later deciding onto which side to fly, either left or right.

For the above decision to be made, for every time the classification turned out to be 1 (tree is close), a second classification is carried out with the same image, except that the image is evenly split in two images vertically. Thus, the right half of the image is sent to the network whose output is interpreted as follows: if the returned classification value is 1 it means the tree is likely to be located to the right of the drone, therefore the drone has to fly to the left; if the returned classification value is 0 it means the opposite hence the drone has to fly to the right.

The decision on what side to fly described above is stored in a flag named *evasion-side* with values 0 for having to fly to the left and 1 for having to fly to the right. However, this flag is used until the classification value from the whole image returns the value 2 (tree is very close), and for which the second classification is turned off. In this scenario, the value to be sent to the state machine is assigned as follows: if *evasion-side* is 0 then the value to be read by the state machine is 2, meaning that the drone has to be commanded to fly to the left; in contrast, if *evasion-side* is 1 then the value to be read by the state machine is 3, meaning the drone has to fly to the right.

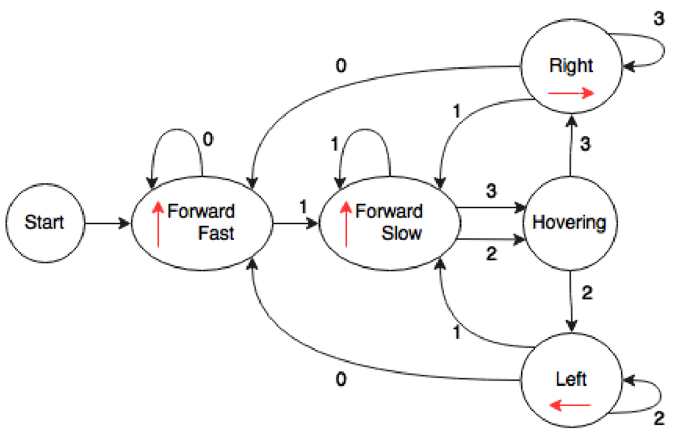
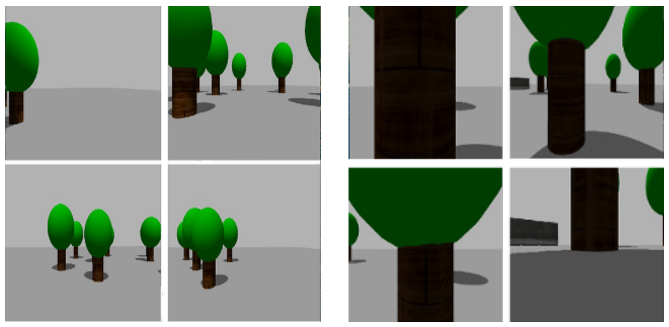


Fig. 3: State Machine for Autonomous Flight and Tree Avoidance based on the output of our customized AlexNet network. The numbers in the edges correspond to the interpretation of the output returned by the network: (0) Free-Space; (1) Obstacle-Close; (2) Obstacle-Very-Close/Evade to the Left; (3) Obstacle-Very-Close/Evade to the Right.


(a) Samples of the images used for ”free space” class.

(b) samples of the images used for ”obstacle” class.

Fig. 4: Example of captured images for deep neural network training.

IV. Experimentation and Results

In this section we present our experimental framework to assess our approach. For the latter, we evaluated the whole framework using a simulated environment. This included training of the our customized AlexNet network with synthetic camera images supplied by the simulator, and execution of the autonomous flight commanded by the state machine described in section III-B. After validating the proper functioning of our approach, we carried out experiments in a real environment with real trees without modification of our approach.

A. Database generation

Training was carried out for simulated and real images. In both cases, the database consists of 9500 images for each class, Figure 4 shows training images for the simulation experiments and the corresponding for the real environment experiments.

As mentioned before, we carried out a fine-tuning of the AlexNet architecture, thus by changing the last layer to output the three classes mentioned before. A total of 450000 iterations were performed during training, using a computer with a NVIDIA GeForce GTX 970M GPU card, which took 36 hours approximately.

B. Structure of the control

In order to promote the speed in the execution of commands, the algorithm runs completely in the C++ language. Both actions, the reading, and processing of the trained network were carried out using the OpenCV libraries with an additional module for the application of deep neural networks. A processing package was created with the ROS programming structure for the control of speeds and directions of the simulated vehicle as well as the acquisition of images needed for the proposed algorithm. These velocity conditions are set in a range of -1 to 1 for the simulated air vehicle in directions in a 3D plane, such as x, y, and z. That is, forward, left and up, can take values in the range from 0 to 1; for the back, right and down directions can take values in the range from 0

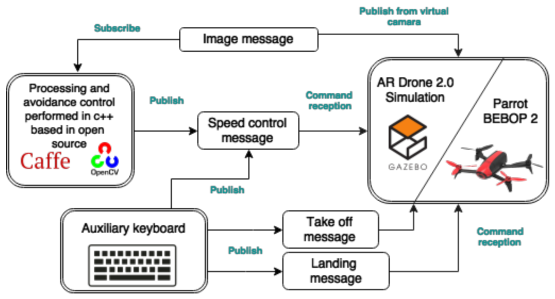


Fig. 5: Structure of the control, with an auxiliary keyboard to position, take off and land the vehicle before the activation of the control program.

to -1. To have an average velocity, an overall value of 0.6 was chosen for forwarding movement and sideways movement in evasion. Also has an auxiliary keyboard to take off, land and position the drone before the start of the control program. The structure of the control is shown in figure 5.

C. Simulation Experiments

In this work, we used a simulation platform called Gazebo, installed as part of the Robotic Operating System (ROS) created with a Berkeley Software Distribution (BSD) license with the open source trend. This software provides the capacity to simulate robots models [10], such as the Parrot AR Drone 2.0 used in this work and shown in Figure 6. ROS is the system that allows the interaction of diverse robotic platforms through the use of nodes, where users can subscribe, read and publish specific commands, besides it has tools and libraries for the development or execution of code [11].

The Gazebo simulator together with ROS enables the control of UAV with flight in a seamlessly fashion by providing a control interface identical to that use with real vehicles. The simulator also supplies images seen on the front of the drone as if it were a real camera; this provides excellent support since the proposed control requires the acquisition of images for processing and control action.

Training the deep learning network with the Gazebo simulator requires an experimental design in a simulated forest constituted by some trees in a natural-like position. From this model, we extracted images through the views offered by the simulated front vehicle. Images for the training as described before, have a size of 638 x 356 pixels at the proposed distances and after that, re-sized to 227 x 227 pixels, allowable for the AlexNet topology proposed by Caffe.

For the simulation experiments, we tested with different simulated scenarios populated with different trees and located at different positions, see Figure 7. In all these different scenarios, our approach proved to be effective in terms of detection and evasion of a tree that became an obstacle. The vehicle was taken off in many random positions avoiding to create the same trajectory in all the tests and to acquire the image of the trees in different views, even from a location very close to the trees.

Fig. 6: MAV's used in this work. The image to the left is the Model Parrot AR Drone 2.0 for Gazebo simulator and the image to the right is the Parrot Bebop 2.

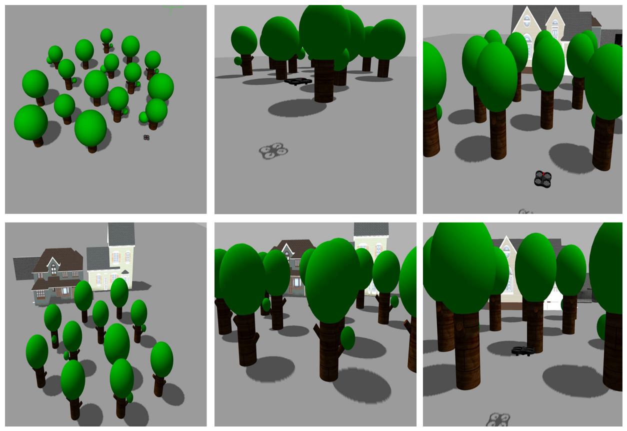


Fig. 7: Different scenarios for simulations with random tree positions.

In total, 100 flight simulations were carried out with three evaluation criteria. The first one where the vehicle has flights with complete evasion and without errors. The second consists of small frictions with the tree but without colliding or impacting following its trajectory correctly. The last one criterion where the vehicle could not classify accurately and had collisions. This simulation perspective seeks to offer the limited vehicle tools concerning the training performed by subjecting it to a more complex environment, to analyze if a more sophisticated real application is feasible. The graph of the results of the simulations performed is shown in the Figure 8.

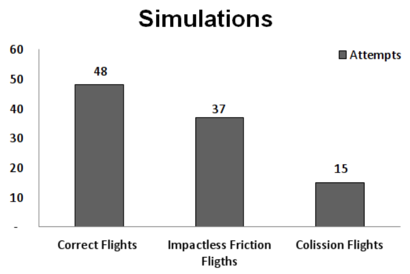
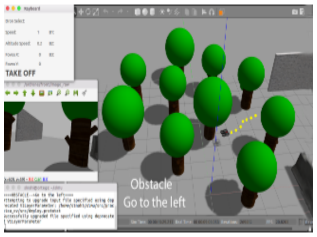
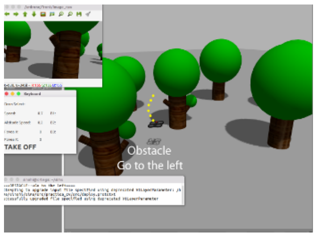


Fig. 8: Number of simulations obtained with correct flights, flights with frictions without impacts and collision flights.

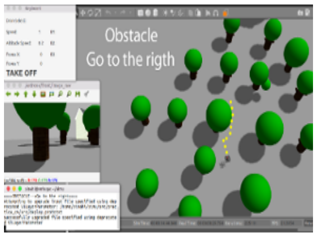
In the simulations, 48 correct flights were obtained, 37 with frictions without impacts. It is clear that this type of flights was considered partially correct because the control



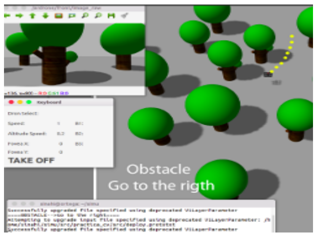
(a) Obstacle detection and left evasion in simulation.



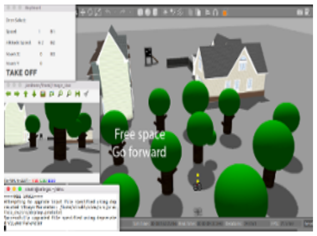
(b) Evasion to the right in the simulation.



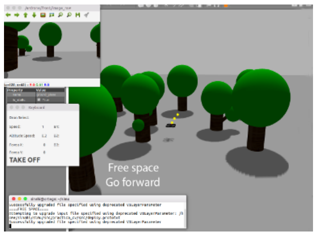
(c) Obstacle detection and right evasion in simulation.



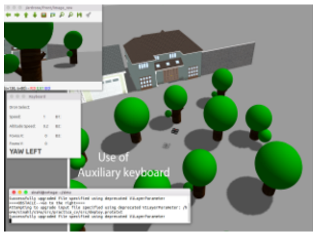
(d) Evasion to the right in the simulation.



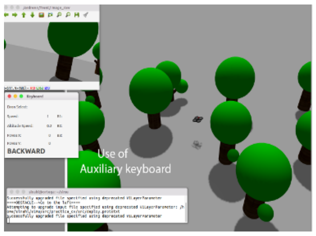
(e) Detection of free space and forward movement.



(f) Detection of free space and forward movement.



(g) Position adjustment in yaw with keyboard.



(h) Position adjustment with keyboard.

Fig. 9: Simulations obtained applying the proposed control and detection through deep learning.

determined the moment and the direction of evasion, as was proposed; those protuberances created in the design of the tree's environment cause the drone to encounter situations not contemplated that propitiate a small friction but not a collision. The 15 collision flights where the vehicle does not detect the obstacle are caused by models not contained in the database of the training as well as the exaggerated proximity to the obstruction. We observed that among the funds seen by the vehicle there were houses, walls or objects, even the classification was not affected, depending more than anything on how close was the drone of the tree.

Being a wholly simulated environment (see the figure 9), the proposed algorithm obtains efficient evasion maneuvers in general. The control strategy, implemented through the programming language in c ++ was high-speed, although it is recommended to use a computer with good processing power, that's because although the controller is fast, the simulation is affected by a high number of resources needed for its use. Also, we want to emphasize that the use of open source software does not fail with the purpose entrusted, having an effectiveness able to be compared with other computer tools of payment.

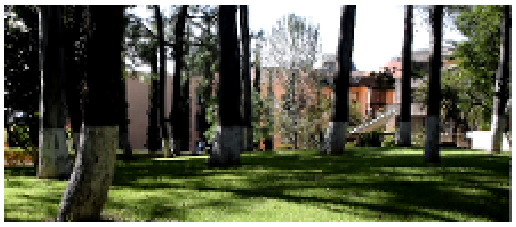


Fig. 10: Area designated for experiments in a real environment.

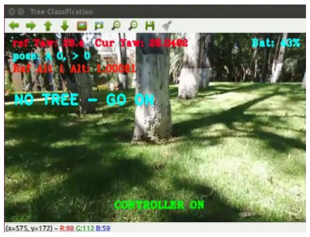
D. *Real Environment Experiments*

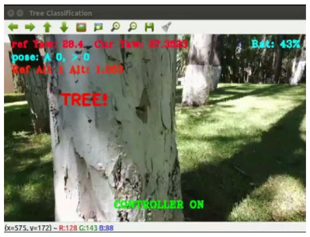
A set of experiments was carried out in a real environment shown in Figure 10. The distance between a tree and a tree is approximately of two meters. This allowed us to increase the speed of the MAV and test from different start positions.

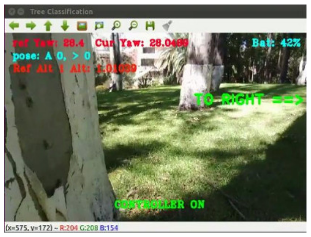
For these experiments, we used the Robotics Operating System (ROS) in its Kinetic Kame version. As an aerial platform we used a Parrot drone model Bebop 2, see Figure 6, for which the communication was established via the ROS package Bebop-Autonomy. For this work, we down-sampled the images acquired from the drone's onboard camera to have a 227×277 resolution at 30 fps. Communication link in between the drone and the Ground Control Station is established via WiFi.

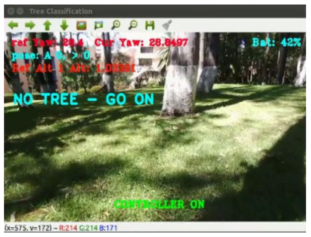
We implemented two proportional controllers to keep the drone's yaw and altitude fixed to a desired value. The state of the drone's yaw and altitude was read from Bebop's odometry. The image, obtained from the drones onboard camera, is processed by the deep learning network, which is explained in section III-B. Based on this classification the controller decides when to reduce or increment the speed, in what direction and when to stop the drone using the state machine shown in Figure 3. Figure 11 shows our front-end interface and a external view of the drone during its autonomous flight in the forest environment.

We carried out 10 flights in a real environment to check the effectiveness of the deep learning network in an uncontrolled environment, with a success rate of **100 %**. During the flights four cases were repeated: (1) four times the drone avoided two trees by the right side and then went forward; (2) Two times the drone avoided one tree by the right side, went forward, avoided one tree by left side and then went forward; (3) Two times the drone avoided one tree by the left side, went forward







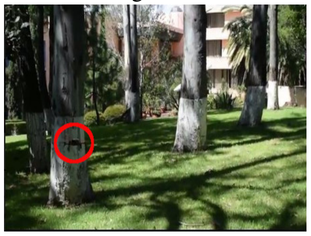


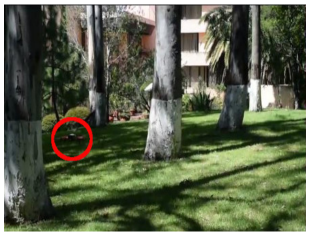
(a) Free space and fast forward movement.

(b) Obstacle detection and slow forward movement.

(c) Very close obstacle and evasion to the right.

(d) Free space and fast forward movement.





(e) External view of free space.

(f) External view of Obstacle detection.

(g) External view of evasion to the right.

(h) External view of free space.

Fig. 11: Examples of our proposed deep learning network in a real environment.

around 4 meters, avoided one tree by the right side and then went forward; (4) Two times the drone avoid three trees by the right side and then went forward. In all flights, the drone successfully increased, reduced the speed an avoided the tree according to the classification.

V. Conclusion

In this work, we have presented a sense-and-avoid system for autonomous flight of a quadcopter in forest environments. Our approch is based on a pre-trained deep neural network known as AlexNet. The goal has been to enable the drone to recognize obstacles, in the shape of trees, and to execute an avoidance maneuver right after the obstacle has been detected.

The tree/obstacle recognition is achieved via the classification of the images acquired from the drone's onboard camera in a frame-to-frame basis. The classification output is read by a state machine that commands the autonomous flight according to three possible values: free-space; obstacle-close; and obstacle-very-close. The last two classes are combined in a decision procedure in order to decide what side the drone should fly for the evasion maneuver.

Our approach was evaluated in simulation using the Gazebo simulator available in ROS. We also tested in a real environment where our system ran in real time, this is, at the onboard camera frame rate. In both cases, simulation and real experiments, our approach was successful in recognizing the trees when these became obstacles, and in performing the avoidance maneuver.

Future work involves performing experiments in more forest environments and to couple it within a more complex autonomous flight system.

Acknowledgment

This work has been partially funded by the Royal Society through the Newton Advanced Fellowship with reference NA140454.

References

[1] M. Hassanalian and A. Abdelkefi, "Classifications, applications, and design challenges of drones: A review," *Progress in Aerospace Sciences*, 2017.

[2] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, 2006, pp. 739–746.

[3] M. A. Olivares-Mendez, L. Mejias, P. Campoy, and I. Mellado-Bataller, "Cross-entropy optimization for scaling factors of a fuzzy controller: a see-and-avoid approach for unmanned aerial systems," *Journal of Intelligent & Robotic Systems*, 2013.

[4] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, "Cnn-based single image obstacle avoidance on a quadrotor," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017.

[5] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, 2016.

[6] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey."

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[9] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 689–692.

[10] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004.

[11] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

View publication stats