Intelligent Embedded and Real-Time ANN-based Motor Control for Multi-Rotor Unmanned Aircraft Systems

George Michael*, Nectarios Efstathiou*, Kyriacos Mantis*, Theocharis Theocharides*† and Danilo Pau‡
*Department of Electrical and Computer Engineering, University of Cyprus
†KIOS Research and Innovation Center of Excellence
‡Advanced System Technology, STMicroelectronics, Agrate Brianza

***Abstract*—Constant technological advancements in commercial multirotor unmanned aerial vehicles (drones) resulted in their deployment in more and more applications, ranging from entertainment to disaster management and many more domains. However, in contrast to their powerful and diverse entrance into our lifestyle and society, they do not yet provide sufficient intrinsic fail-safe mechanisms to prevent accidents that may occur due to technical problems or unforeseen flight incidents such as turbulent winds, inexperienced pilots, and so on. Therefore, in the current study, we propose the use of an integrated intelligent motor controller, which is trained to recognize incidents directly from the on-board sensors (barometer, gyroscope, compass and accelerometer) and react in real-time, adjusting the drone's motors. The goal is to provide a small, intelligent, low-power, real-time, built-in controller for multirotor UAVs that will be able to understand a dangerous scenario right before it happens, start taking counter measures to keep the drone safe, and provide the pilot with a bigger reaction-time window. We propose the use of an artificial neural network, implemented in a lightweight embedded processing board, that is able to recognize and react in real-time to various turbulent situations. Experimental results suggest that our controller is able to respond properly and timely to wind changes (turbulence) allowing the drone to maintain its expected state and path.**

I. Introduction

The constant technological advancements in commercial multi-rotor unmanned aerial vehicles (UAVs a.k.a drones) and concurrently the reduced costs, have created a huge diverse application domain and a greater impact in everyday life, not only in entertainment but also regarding multiple commercial applications in several fields, both in terms of criticality (i.e. surveillance, monitoring, control, disaster management) and in terms of overhaul business models in advertising, land surveying, and many more. It is widely accepted that multi-rotor UAVs can provide a fast and efficient solution. However, in contrast to their powerful and diverse entrance into the market, we believe that they do not provide sufficient failsafe mechanisms to prevent accidents that may occur due to technical problems or unforeseen real world situations such as air turbulence, pilot inexperience, loss of line-of-sight, and many more reasons. As such, they are still treated with caution, as evident by national laws that limit their usage and require extensive training, extremely high insurance costs, and several other restrictions, so that they can fly legally. Some countries treat them as manned aircraft, limiting their application spectrum and taking away a potentially life-saving tool especially when used in disaster management. [1][2][3][4]

In fact, their adoption and acceptability in more applications lies within making them more reliable and trust-worthy, so that these situations can be completely avoided if possible. A prime example of such situation is the crash of a camera-equipped drone during the 2nd slalom run of the Alpine Skiing World Cup. A drone came down crashing to the ground right next to the skier Marcel Hirscher at Madonna di Campiglio in 2015. From the footage, it appears that the UAV was completely out of control and the pilot was not able to redirect it to a safer location before it crashed or get it to the ground while minimizing or eliminating possible damage. [5][6]

This paper therefore, presents an initial approach towards an intelligent lightweight, low power, built-in controller for multirotor UAVs, based on artificial neural networks, which will be able to monitor and possibly predict a dangerous scenario as fast as possible, even at its early conception stages, and(autonomously) initiate counter measures to keep the drone as stable as possible, and providing the pilot with a larger reaction-time window. Part of the scope of this work is to utilize sensors that are already available on most commercial UAVs and do not rely on external sensing such as GPS, enabling the drone to be completely self-reliant. Furthermore, while relatively accurate, a GPS-guided flight is not as accurate as it should be for more applications, especially considering that drones may fly indoors or in remote and potentially dangerous terrains such as canyons, etc. Our implementation should be suitable for new environments and different locations as it was never trained for specific conditions only.

Another contribution of this work is the holistic and experimentally verifiable approach that we took in designing the intelligent controller, both during training and evaluation of its performance. Data collection was done via a drone with a flight controller's sensors being logged. The drone was exposed to several scenarios under different weather conditions, using both remote control induced conditions and an actual fishing line to upset the drone. At the same time the UAV remote controller (RC) instructions were also monitored

978-1-5386-2880-5/17/$31.00

to provide the system with knowledge of what the pilot was initially intending to achieve. To record and analyse a greater spectrum of tests, we used a fishing line to pull the drone into different directions and rotate it to simulate scenarios we did not meet during our flight time. The flight data was used to develop and train an Artificial Neural Network (ANN) which is used to identify an unwanted situation, and thus intelligently provide the on-board motor controller with the necessary signals (pitch, roll, yaw and throttle) to keep the drone stable as fast as possible. The trained ANN was then implemented on an embedded lightweight platform, with the flight controller's sensors, which was reattached on a drone for real-time evaluation. Our initial results show that for induced situations, as well as real-life situations (i.e. actual turbulent air) the ANN can achieve accurate predictions in excess of 80% of either naturally or artificially induced incidents and scenarios. However, there were scenarios, especially when we had high winds, where the system was not able to cope with the force of nature, highlighting that further work is necessary, especially regarding the crucial stage of data collection. However, to the best of our knowledge, this is the first attempt at training and evaluating such a controller with real-life scenarios and data. We therefore hope that our work will stimulate further research in this field.

Consequently, this paper is organised as follows. Some background on the operational parameters of multi-rotor UAVs and their influence on our design approach is given in Section II, along with some related work on intelligent drone controllers. Section III presents the design and development of the ANN based controller, which is experimentally verified by the procedure and setup detailed in Section IV. The results are presented in Section V nd the paper concludes giving some future research directives in Section VI.

II. Background and Related Work

Commercial multi-rotor UAVs rely on the use of at least 4 propellers, arranged either in a traditional cross or in an X-cross, with the blades rotating opposite each other. The motors that control the blades traditionally operate in pairs in a simple linear mode (i.e. increasing or decreasing speed depending on where the pilot wishes to steer the drone). The propellers work in unison to steer the drone and provide the basic control functions/flight states: pitch, roll, yaw and throttle. Figure 1 shows how the rotors rotate on quadcopters, where rotors 1 and 3 rotate clockwise and rotors 2 and 4 rotate counterclockwise.

- A quadcopter hovers by adjusting its altitude in a constant level. The hover is done by applying equal thrust to all rotors. This technique is used to downlift and uplift a quadcopter by adjusting smaller and bigger thrusts equally to all rotors.
- aw is the self-turn of the quadcopter either left or right, by applying more thrust to the rotors rotating in one direction. In the third configuration in Figure 1 you can see how this adjustment is done on the rotors.
- Pitch and Roll is the movement of the quadcopter either forward or backward, and either left or right respectively.

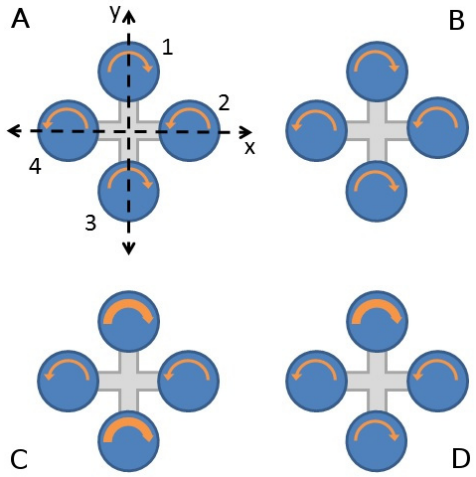

Fig. 1: A. Motor rotation on quatrocopters.
B. Motor power for hover.
C. Motor power for yaw.
D. Motor power for pitch and roll.

Pitch and Roll are done by applying more thrust to one rotor and less thrust to its diametrically opposite rotor. In the last configuration of 1 you can see how this adjustment is done on the rotors.

However, these UAV systems were dynamically unstable, thus basic control algorithms were applied, reacting to specific feedback and stabilising the drone. In fact, various control methods were used to solve the stabilization issue on UAVs; some use a PID control [7], others an H-infinity control [8], and more recently predictive control [9]. The most common solution for this problem is the use of PID control [10] which can be combined with a PD controller [11], usually preferred due to its simplicity and effectiveness. The PID controllers operate on the flight states: roll, pitch, yaw, altitude and position, to perform their corrections. The problem with PID controllers is that they require tuning and setting of Proportional, Integral and Derivative multipliers, which is a time-consuming and laborious process. If the user does not achieve a good tuning, this method provides limited stability.

In general, PID controllers have trouble operating under conditions that are far from the optimal hover point, and thus may not be suitable for lightweight commercial drones that are typically used today; in fact, these popular micro aerial vehicles (less than ten kilograms) can get heavily affected by disturbances due to intense wind gusts, and many more unpredicted reasons [12]. Some solutions with good results [13][14][15] suggest the use of neural networks that create a non-linear mapping from inputs to outputs and can capture the UAV flight dynamics by creating a robust controller. One such solution is an adaptive neural network controller based on CMAC (Cerebellar Model Articulation Controller) [14], which has good results even during considerable wind disturbance, but this algorithm is considered as a memory hog, thus unsuitable for a lightweight embedded system such

as our targeted drones. Some work has also been done on self-tuning adaptive control using neural networks [15]. The combination of genetic algorithms with neural networks is another approach, creating neuro-evolutionary algorithms that rank each controller based on a cost function [16]. Higher performing cost functions are mutated, with some probability, in order to search for optimal solution. However, such a solution is even harder to be implemented on a lightweight embedded processing platform, as it takes a very long time for the generations to reach the optimal solution and thus will not be able to operate in real-time.

Hence, our aim is to create a light-weight predictive controller based on artificial neural networks, which will be capable of turning in real-time results, fit on a battery-powered embedded processing platform, and run using on-board sensing data. The challenge goes beyond the training and evaluation stage though, given that the ANN has to be optimized to fit within such an embedded platform.

III. ANN DESIGN AND DEVELOPMENT

Motivated from the promising results of ANN-based controllers and constrained by the physical dimensions, power consumption and real-time computation, we proposed the use of an ANN-based intelligent motor controller. The ANN needs to run on a low-power embedded processor that is capable of being part of the drone payload. The proposed ANN receives discretely sampled sensory data from on-board sensors in a time-series format and performs pattern identification. The objective of the ANN is to be trained to identify when any detected drone motion is indeed what the pilot intends to perform, or if it is part of an unwanted situation, such as turbulence, pilot error etc. Thus, the training of the ANN is crucial, and data collection is perhaps the greatest challenge in this quest. Moreover, the ANN footprint must also be as minimal as possible, both in terms of memory and program code and in terms of resources (neurons, synapses, weights), so that it can be implemented on lightweight hardware that can be included on a drone's standard equipment. The following steps highlight the development of our proposed ANN.

A. Training

Perhaps the most important decision lies in determining the data, in terms of its format, amount, and timing (availability, etc.) required to train the targeted ANN. Part of this work lies within the data collection process. Therefore, our first step was to perform several measurements during real flights, and collect real sensor data, to be able to train the ANN as realistically as possible. Most commercial UAVs are equipped with four basic intrinsic sensors: the barometer, the gyroscope, the compass and the accelerometer. These sensors provide readings relevant to motion, direction of motion, velocity during motion, and of course altitude. In contrast to GPS sensors that rely on readings from geopositioning satellites, none of these sensors require external communication to work, thus they are ideal for facilitating our targeted autonomy level. When fused, these sensor data provide enough information for the ANN to detect and recognize an incident compared to the expected normal drone behaviour. When used in conjunction with the input of the UAV RC, (i.e. to ensure that the ANN receives the intentions of the pilot as part of the supervised training process) it can then be trained to identify intended vs. non-intended motion/trajectory, and respond to adjust the control signals (pitch, roll, yaw and throttle). We briefly describe the sensory data and their importance next:

- The barometer (or pressure altimeter) is used to measure altitude through atmospheric pressure. This information is useful to understand sudden wind gusts that might push the drone high in the air or detect gradual inclination of the drone's altitude during its flight.
- The gyroscope (or gyro) allows us to measure rotational motion. Using its 3-axis output we can automatically perceive the full rotational movements of the drone and handle titling caused by air. Tilting can cause the drone to accelerate fast in a direction and get it out of control. It is imperative to limit such behaviour as fast as possible to avoid these situations. An uncontrollable yaw (the drone spinning around itself) results in rapid loss of control and usually violent fall.
- The solid state compass (or magnetometer) identifies changes to the orientation of the drone and helps restore the drone to its original intended direction. Monitoring this device allows the ANN to understand minor changes to the orientation of the drone that might not be easily perceived by the pilot, but collectively with the other motions it can completely change the drone's trajectory path.
- The accelerometer measures acceleration; an important sensor that enables the ANN to know the rate of change of the velocity of the drone in the 3 axes, thus allowing the ANN to detect changes that otherwise may not be detected by the compass or the gyroscope. This scenario can happen when air pushes the drone in the same direction as its intended motion trajectory, thus the gyroscope will not detect rotational movement and the compass will not detect any orientation changes.

The next step is determining the collection of this data. As the objective is to maintain real-time response, it was decided to collect the data via discrete sampling of the sensor readings, which would then be fed to the on-board ANN in time-series format. Thus, the ANN would constantly receive data from the sensors during a sampling window with n intervals with dt being the series interval, and would output any adjustment decisions to the drone in terms of adjusting the pitch, yaw, roll and throttle signals at the end of every sampling window. The timing duration of the sampling window is determined by the performance of the neural network, whereas the sampling interval within the window is determined by the ability of the host board to receive the sensory data. These two parameters therefore are important in the context of deciding any optimization requirements to speed up the ANN at the cost of reduced accuracy, something we discuss later in the

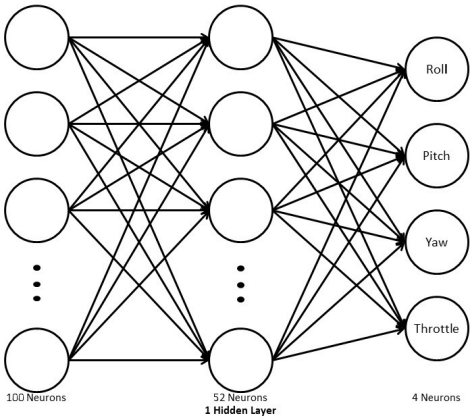

Fig. 2: A graphic representation of our MLP ANN

trade-offs.

We used Theano [17] to process the data and we trained the network using a back-propagation supervised training algorithm. The data was classified as "intended trajectory" vs. "not intended trajectory", with the necessary correct adjustment outcome of the ANN as the acceptable output. The collected flight data was therefore analysed along with the drone trajectory and its flight data information, and was fused together in tuples. In addition to the sensory data, for training purposes, for every interval the readings of the barometer, gyro, compass and accelerometer were fused into a tuple; each tuple was then used to form a time-series data pattern, where the expected motion trajectory was described with the received tuples. In the event when the received tuples described an unintended trajectory (as we used real-life data collection), the correct trajectory types were fused with whatever number of tuples within that interval corresponded to the intended motion, and the unintended trajectory was labelled as such, along with the "breaking" point. In this way, the training data consisted of all types of information; n tuples which described motion within each dt window, and the expected (or unexpected) data that differentiated between what was anticipated and what was caused unintentionally. Moreover, this enabled us to identify the mistake in the drone's trajectory and exactly classify the suggested action that had to be taken by the controller.

Each motion trajectory then was subsequently fed into the ANN and was labelled as intended or not-intended (depending on the monitored RC signal), and the ANN was also trained to identify the tuple sequence that caused an unintended situation as described above. Thus, the ANN was then configured to output the expected corrective action in the form of roll, pitch, yaw and throttle.

B. ANN Design Trade-offs

Our initial experimentations suggested that an appropriate number of sampling (tuple) intervals would be set at n = 20. Based on our targeted embedded platform, the expected ANN processing time per sampling window was set at 2 seconds, which would provide a sufficient reaction time to the network prior to the pilot taking their own corrective actions, yet it would not be prohibitive for implementation purposes on an average embedded processor. We chose a multi-layer perceptron as our ANN with 10*20 nodes (i.e. the number of sensory data per tuple per n) on the input layer and 4 nodes on the output layer (pitch, roll, yaw and throttle). Our training resulted to 102 hidden nodes, which was a trade-off achieved when we varied the number of hidden layer neurons and saw that the accuracy did not increase further. Our goal was to achieve 95% accuracy during training while minimizing the risks of over-fitting our network. Given our targeted performance window and the fact that the ANN would be deployed in a constrained embedded processor with limited processing and memory capabilities powered by a battery, we decided to keep the number of hidden layers and hidden nodes as low as possible.

C. Optimizations

Embedded devices usually have limited resources, so in order to achieve real-time performance, our objective was to have both a fast and a power-efficient ANN. For this reason we performed certain optimizations to reduce execution time and lower the energy consumption, while also minimizing the memory footprint of the ANN. Minimizing the memory footprint enabled us to increase the range of neuron instances that could be created in parallel. We briefly list these optimizations next:

- To boost performance, we loaded the weights directly into code as an array to avoid linking memory-intensive libraries that allow us to facilitate operations over the host file system.
- We used a circular buffer to store the data used during the processing to avoid the usage of dynamic memory allocation which is very expensive (in terms of allocation and release).
- The targeted compiler (GCC) was configured to perform several speed optimizations to increase performance while risking an increase in the code footprint, making the system more efficient and power friendly, as it would produce the same results faster. This was performed over several iterations to result in a pareto-optimal solution.
- Raw data was normalized and tupled only once every timing window (i.e. every dt), and then data was stored in its new form to be reused over its 20 cycle lifespan.
- We introduced an orientation filter which uses a quaternion representation, in order to fuse accelerometer, magnetometer and gyroscope data into a representation of the drone in a three-dimensional space [18]. This important step allowed us to reduce the input sensory data per tuple from 10 down to 5. Although we understand that this change may affect the accuracy of our ANN, the benefits are extremely important, as the number of input neurons was reduced from 200 down to 100, the number of necessary hidden nodes to 52 and the number of total weights to less than half of the original. The impact is also reflected in the amount of several thousand less

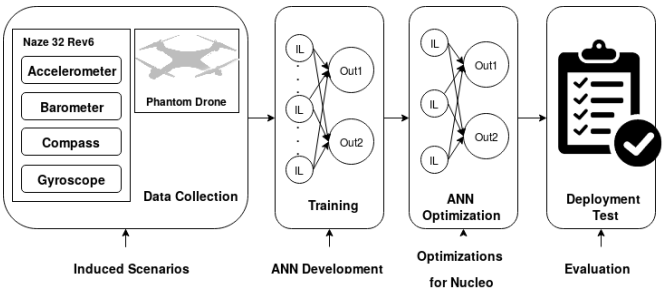

Fig. 3: A graphic representation of the workflow we followed to perform our experiment

multiplications and additions. Note worthily, this change also positively affects the overall memory of the system as we dropped the memory consumption as well, which is another important advantage in our targeted application.

D. Performance Evaluation Strategy

Our aim was to evaluate the ANN using real-life data and measure the ANN's performance using four metrics; accuracy, power consumption, code/memory footprint, and performance (delay). Thus, we obtained evaluation data as described in Section IV-B.

IV. Experimental setup and approach

As previously mentioned, our evaluation strategy involved real-world testing. Therefore, we partitioned the experimental setup in two phases. The first phase involved the data collection part and the training of the neural network, whereas the second phase involved the deployment of the neural network on a drone and the evaluation under real-life scenarios.

We used the following hardware for our experimentation mounted on a UAV:

1) The Raspberry Pi 3 was used for data logging and for controlling the whole system.
2) The STM32 Nucleo-144 electronic prototyping platform by STMicroelectronics hosted the brains of our neural network.
3) A USB power bank was used for power supply and for various power consumption measurements.
4) The Naze 32 Rev6 Full flight controller unit was used as we could not access the commercial FC on the drone.
5) The DJI Phantom 2 is a ready to fly, multifunctional quad-rotor system.

A. Phase 1 - Data Collection and Training

The setup for the collection of the training data included the integration between a single board computer (SBC) and the flight controller (FC), and the deployment of the two on the drone to capture real flight data. During this stage, we used a Raspberry Pi 3 to serve as the single board computer and the Naze32 as the flight controller, onboard the DJI Phantom 2 quadcopter.

The integration of the SBC and the FC was done over a custom built interface implementing serial communication over USB. The communication protocol used was MSP (Multiwii Serial Protocol [19]), which is a plain text communication method. The MSP protocol is used for exchanging data from the flight controller to any other device that supports it. The Pi was controlled over WiFi via SSH, giving us access to the whole system while the drone was in the air and it facilitated the filtering and sorting of flight data on the go. All flights were recorded on video, allowing us to review them later on and understand data that appeared to be strange while we were trying to provide the ground truth information for the test vectors.

The objective was to collect as many flight data under real conditions. We also collected data while the drone was on the ground with the rotors off and we kept changing its orientation and its angle. This allowed us to read any sensory data associated with jitter and noise and thus calibrate the NAZE32 sensors. We also performed part of these measurements while the rotors were spinning at different speeds with no propellers on, to identify additional noise caused by motors as well. Besides that, we recorded some basic movements such as roll, pitch and yaw along with hovering and vertical take offs and landings in a closed space, again to help us better understand what we see and how it translates to meaningful information.

Following the recording of the basic movements, we made recordings outside to observe drifting and tilting caused by air. Finally we collected the information that the flight controller provided while the drone was being pulled and rotated by a fishing line. The fishing line was attached to the drone and was pulled by a person on the ground, to simulate as many unwanted incidents as possible, and collect as many associated data as possible. In order to eliminate noise, these experiments were run multiple times in an identical fashion, and the training dataset was constructed. The training procedure is outlined in Section III-A.

B. Phase 2 - Deployment and Real World Testing

The neural network was implemented on an STM32 Nucleo-144 that met our criteria, as it was set for high speed, low footprint, small size, and low power consumption. Thus, it was attached to the drone, connected to the NAZE32 controller, and used during the evaluation phase.

Under this configuration, we powered on the Raspberry Pi through the USB power bank, the Nucleo through the Pi and the Naze through the Nucleo board.

The Nucleo and subsequently our ANN was loaded using C and the Eclipse IDE. The STM32CubeF7 API and framework was used to access the Nucleo peripherals. We compiled the code using the GDB ARM cross compiler and used the OPENOCD framework for on-chip debugging.

Once the ANN was loaded, we started the evaluation. First, we removed the propellers of the drone to prevent accidents. Next, we hung the drone from the ceiling and by using fishing line we were inducing movements while reading the outputs of our neural network via the Raspberry Pi. Once the in-lab testing was concluded, we then proceeded to evaluate the drone in real-world scenarios, flying it outside.

V. Experimental Results

We followed the experimental approach described in Section IV to evaluate the performance of our ANN. First, we compared the accuracy of the ANN when running induced motion trajectories that were also part of the training set. This gave us extremely good accuracy (well over 90%). The next step was to evaluate the accuracy of the ANN in terms of forecasting specific trajectories and taking corrective action when receiving completely unknown data. This was achieved by flying the drone freely, monitoring the flight controller and the pilot's remote controller data, and observing the drone's motion through video analysis at a later stage. Upon each flight completion, we measured the correct and erroneous decisions made by our ANN. Further on, we measured the percentage of false positives (i.e. when the pilot was intending to perform a specific action, and the ANN mistook it for an error and produced the wrong motor controller output).

Additionally to our evaluation in terms of accuracy, we provide results corresponding to the performance of the ANN in terms of sampling window (0.35 seconds), power consumption (200 microwatts) per sampling window computation, and the memory (23 KB) and code (1.7 MB) footprint.

While the accuracy remains relatively low (83%), the majority of the mispredictions were the result of high wind gusts and flying over hills (which always causes turbulent air). These incidents can be hardly emulated in a controlled environment, thus further research is necessary to improve the training set. As this is to the best of our knowledge a first attempt at building such a prototype, we cannot compare it to existing work. However, a lot remains to be done. The amount of real-world scenarios remains infinite, thus it is imperative for researchers to continue building upon this work to improve the accuracy and the ANN itself. Nonetheless, we have shown that a light-weight ANN, built on a low-power and portable embedded processing platform such as the STM Nucleo board, can be a promising initial step towards a trust-worthy drone system, necessary for modern societal needs and the emerging vast application domain. Especially in comparison with CMAC/PID, our implementation is lightweight and it could be used on more powerful platforms like the Raspberry Pi with no additional risks.

VI. Conclusion

There is great potential in using ANNs as intelligent flight controllers. Their ability to take decisions even with noisy or unknown data makes them a good candidate for the position. Unlike any deterministic system based on a fixed algorithm, they need a lot of training before they start being productive. Their training needs to be consistent and it must be similar to the real world scenarios the ANN is supposed to be part of.

References

[1] J. S. Duncan, "Framing the future of aviation," European Parliament - European Data Protection Supervisor, http://ec.europa.eu/transport/sites/transport/files/modes/air/news/doc/2015-03-06-drones/2015-03-06-riga-declaration-drones.pdf, Declaration, March 2015, riga declaration on remotely piloted aircraft (drones).

[2] J. Foster, "Report on safe use of remotely piloted aircraft systems (rpas), commonly known as unmanned aerial vehicles (uavs), in the field of civil aviation (2014/2243 (ini))," European Parliament - Committee on Transport and Tourism, http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+REPORT+A8-2015-0261+0+DOC+PDF+V0//EN, Report A8-0261/2015, September 2015, result of final vote (45+ vs 1-).

[3] F. A. Administration, "Unmanned Aircraft Systems (UAS) Regulations & Policies," https://www.faa.gov/uas/resources/uas_regulations_policy/, [Online; Last accessed 11-May-2017].

[4] J. S. Duncan, "Small unmanned aircraft systems (suas)," U.S. Department of Transportation Federal Aviation Administration, https://www.faa.gov/uas/media/AC_107-2_AFS-1_Signed.pdf, Advisory Circular 107-2, June 2016, initiated by: AFS-800.

[5] I. S. Federation, "Drone incident in Madonna di Campiglio," http://www.fis-ski.com/alpine-skiing/news-multimedia/news/article=drone-incident-madonna-campiglio.html, December 2015, [Online; Last accessed 11-May-2017].

[6] ——, "Drone incident in Madonna di Campiglio," http://www.fis-ski.com/alpine-skiing/news-multimedia/news/article=infront-sports-media-follow-statement-drone-accident.html, December 2015, [Online; Last accessed 11-May-2017].

[7] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2451–2456 vol.3.

[8] J. Gadewadikar, F. Lewis, K. Subbarao, and B. M. Chen, "Attitude control system design for unmanned aerial vehicles using h-infinity and loop-shaping methods," in *2007 IEEE International Conference on Control and Automation*, May 2007, pp. 1174–1179.

[9] Y. Kang and J. K. Hedrick, "Linear tracking for a fixed-wing uav using nonlinear model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1202–1210, Sept 2009.

[10] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor uav," in *2007 European Control Conference (ECC)*, July 2007, pp. 4001–4008.

[11] B. Erginer and E. Altug, "Modeling and pd control of a quadrotor vtol vehicle," in *2007 IEEE Intelligent Vehicles Symposium*, June 2007, pp. 894–899.

[12] J. F. Shepherd, III and K. Tumer, "Robust neuro-control for a micro quadrotor," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '10. New York, NY, USA: ACM, 2010, pp. 1131–1138. [Online]. Available: http://doi.acm.org/10.1145/1830483.1830693

[13] T. Madani and A. Benallegue, "Adaptive control via backstepping technique and neural networks of a quadrotor helicopter," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 6513 – 6518, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016399906

[14] C. Nicol, C. J. B. Macnab, and A. Ramirez-Serrano, "Robust neural network control of a quadrotor helicopter," in *2008 Canadian Conference on Electrical and Computer Engineering*, May 2008, pp. 001 233–001 238.

[15] F. C. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 44–48, April 1990.

[16] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ec02

[17] http://deeplearning.net, "Theano," https://github.com/Theano/, 2009, [Online; Last accessed 11-May-2017].

[18] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1–7.

[19] Multiwii, "Multiwii Serial Protocol," http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol, 2013, [Online; Last accessed 11-May-2017].