Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

Vision Aided Navigation of a Quad-Rotor for Autonomous Wind-Farm Inspection

Durdevic, Petar; Ortiz-Arroyo, Daniel ; Li, Shaobao; Yang, Zhenyu

Published in:
IFAC-PapersOnLine

DOI (link to publication from Publisher):
10.1016/j.ifacol.2019.08.049

Creative Commons License
CC BY-NC-ND 4.0

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Durdevic, P., Ortiz-Arroyo, D., Li, S., & Yang, Z. (2019). Vision Aided Navigation of a Quad-Rotor for Autonomous Wind-Farm Inspection. *IFAC-PapersOnLine*, *52*(8), 61-66. https://doi.org/10.1016/j.ifacol.2019.08.049

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-8 (2019) 61–66

Vision Aided Navigation of a Quad-Rotor for Autonomous Wind-Farm Inspection

Petar Durdevic [*] Daniel Ortiz-Arroyo [*] Shaobao Li [*]
Zhenyu Yang [*]

[*] Aalborg University, Department of Energy Technology, Niels Bohrs Vej 8, Esbjerg, Denmark (e-mail: pdl@et.aau.dk, doa@et.aau.dk).

Abstract: This work presents a vision based navigation system for an autonomous drone that is capable of recognizing and locating wind mills. WindMillNet, a Deep Neural Network created for this purpose was specially trained to recognize wind mills on camera images using transfer learning techniques. The drone powered by WindMillNet scans the horizon to find wind mills, and after perceiving a wind mill, navigates towards it, with the goal of performing its inspection. A hierarchical control system, implemented in the drone provides stability and control of its movements. Our framework was designed using a cyber-physical systems approach using high-level abstractions in modeling, communication, control and computation.

Keywords: Quad-copter,Control,Artificial Intelligence,Vision,Neural Networks

1. INTRODUCTION

In the recent years the application of various drone platforms has attracted a huge interest due to their versatility of use in the industry and in the commercial market, where the quad-copter is one of the most popular types of drones. The industry sees great potential in the application of this technology and has invested up to 127 billion dollars in 2015, according to Raconteur (2016). Most of the current applications of drones rely on human pilots to perform specific tasks. The drones include automatic controllers in hardware and software that control their altitude and position, keeping the pitch and roll angle at 0 degrees and achieving system stability at a specific height, horizontal and vertical positions Bristeau et al. (2011), Hua et al. (2013), Smolyanskiy et al. (2017).

The ultimate goal of this research is to create a drone platform capable of navigating autonomously in offshore or onshore wind farms, to inspect wind mills, looking for potential defects. As of 2016 in the North Sea, 7% of the offshore wind-mill inspections were done by remotely piloted drones, Jamieson (2018). According to Deign (2016), one such inspection costs up to 800 USD per wind mill, where this price includes a pilot and a data analysis technician. However this price does not include the cost of renting a ship to carry the drones, the pilot and crew to reach the wind mill, which according to other sources ranges from 6000-8000 euros. Automating the process of inspection for a full wind mill farm has thus a large economic impact. To address this problem we use a combination of state of the art supervised machine learning and control techniques. Deep neural networks are used for object recognition and control systems for stability and movement control. This combination resembles the way humans use their visual perception capabilities to search for objects within a space, in order to determine the movements needed to reach them. However, the reliable implementation of a visual recognition system for wind mills is one of the most challenging tasks for autonomous inspection. Therefore the focus of this work is in the implementation of a vision based system for navigation and localization of wind mills within an offshore or onshore environment.

Visual-based navigation of unmanned vehicles has been been extensively researched. Some of the approaches that employ neural network techniques are Pomerleau (1989), Pomerleau (1991), Bojarski et al. (2016), Giusti et al. (2016), Smolyanskiy et al. (2017), Loquercio et al. (2018), Drews et al. (2018). In Pomerleau (1991) a video camera captures images of the environment ahead of the autonomous vehicle that are fed into a neural network (NN), with an input layer consisting of an array of 30 x 32 neurons, with 5 hidden layers fully connected to a output layer consisting of 30 neurons, which represent the vehicles direction, left right and straight ahead. A similar approach was reported in an earlier work by Pomerleau (1989), with the addition of a range finder and a road intensity feedback unit to the input layer. NVIDIA's Car in Bojarski et al. (2016) uses three cameras located on the car's dashboard to collect training data for three different directions, where the trained system computes the correct steering direction in order for the vehicle to travel in the middle of a lane. The approach described in Giusti et al. (2016) is based on a deep neural network (DNN) used for visual perception of forest trails, where a monocular image is the input and the output consists of three values which are the probabilities of the three classes [Turn left, Go straight, Turn right]. The training data-set was collected via three head-mounted cameras. The solution was implemented on two quad-rotor platforms and the results were mixed, as a poor camera quality was reported to cause issues in high contrasted images. The performance, however was good in well lit

⋆ This work was sponsored by the Department of Energy Technology at Aalborg University Denmark. In addition, we would like to acknowledge the support extended by Aerodyne AtSite.

conditions. The system was not equipped with obstacle detection which resulted in frequent crashes. The work in Smolyanskiy et al. (2017) is based on trail following using a DNN called TrailNet, which determined the orientation [facing left, facing center, facing right] and lateral offset [shifted left, centered, shifted right] within the terrain, computing a pose which is used for control of the drone. The system is coupled with a visual odometry component for obstacle avoidance.

In Loquercio et al. (2018) a similar approach was taken where a shared residual Convolution Neural Network (ConvNet) named DroNet was fed with a 200 x 200 pixel gray-scale image. The output of this network is a steering angle and the probability of collision, which is then used as a feedback parameter to determine the drone's forward velocity. If the probability of collision is high the forward velocity will be zero, otherwise the drone will fly at maximum speed. More recently in Drews et al. (2018) deep learning-based road detection was combined with particle filters and MPC for control and navigation of an RC off-road four wheeled vehicle on a dirt track.

In all of these works the terrain was well defined, with either a road, a forest trail or a track. However, as is mentioned in Giusti et al. (2016) forest and dessert roads represent a more challenging perception problem than paved roads as their appearance changes and their shape is not constrained as is the case with the city landscape, which in most cases follows regulatory codes. In our case the perception task is even more challenging, since our aim is to find a wind mill and navigate towards it in spaces where there are no landmarks available that can be used for navigation. The use of GPS may partially help in this regard but its relatively low precision restricts its use in this environment.

State of the Art in Deep Neural Networks Deep Neural Networks (DNNs) are one of the most successful machine learning techniques we have today. DNNs achieved remarkable results in solving one of the key problems in artificial intelligence and computer vision: the problem of object recognition in images. A DNN consists of a deep stack of multiple neuron layers, where each layer is capable of learning a partial representation of the features of an object. The architecture of modern DNNs employs a combination of ConvNets that filter out simple object features, pooling layers that merge semantically these features to recognize more complex patterns, normalization layers to avoid the problem of vanishing gradients and fully interconnected layers that provide weight distribution. DNNs also commonly use Rectified Linear Units (ReLUs) and regularization techniques such as dropout together with the stochastic gradient descent and back propagation algorithm for training. Finally, a softmax function calculates the probability that an image belongs to a predefined category of objects.

DNNs with 16 to 30 layers are commonly used in some of the leading image recognition models such as AlexNet, described in Krizhevsky et al. (2012). However, as we add more layers, DNNs have the problem of degradation in accuracy. Deep Residual Networks like ResNet were proposed in He et al. (2016) to avoid this problem. More recently the combination of a residual network with more traditional architectures such as the Inception network has shown even better results in Szegedy et al. (2016), achieving shorter training times. Finally, the Squeeze and Excitation network in Hu et al. (2017) combining fully connected, residual and inception networks has achieved the highest performance in the ILSVRC 2017 image classification contest, with only 2.5% misclassification rate.

One of the main drawbacks of DNNs is that they require millions of labeled images for training. The use of these big data-sets produces long training times, even if DNNs are trained in high performance graphics processing units (GPU). To address this problem, transfer learning techniques were proposed in Razavian et al. (2014). In transfer learning the initial layers of a pre-trained DNN model are kept unchanged but the last layers are substituted to adapt the network to a new image classification task. The goal is to reduce the number of training examples needed by keeping the weights associated to the basic pre-learned patterns contained in most objects such as simple edges and contours. The network is then trained again with a new data-set, to learn the more complex features of new objects using new layers and weights. The modified DNN is trained again by either, fixing the weights in the first layers or using a low learning rate. This is done with the goal of preventing the network to “forget” the pre-trained weights when high learning rates are used. Additionally, in transfer learning, it is also common to use data augmentation. In data augmentation, artificial transformations such as translations, scaling and rotation of images are performed in memory. This is done for two reasons, firstly to increase the size of the data-set used for training and testing, and secondly to avoid over-fitting the network.

2. PROBLEM FORMULATION

The autonomous navigation of a quad-copter or any other robot introduces several challenges. Firstly, commercial drones such as DJI Matrice 600, DJI (2018), have limited capabilities in processing and power consumption. Secondly, the quad-copter must first perceive its surroundings, find a target and then react to it in an *intelligent* manner. Object recognition can be implemented with specialized DNNs, but due to the complexity of its structure, DNNs have high computational costs. To address this issue, training and inference of DNNs are commonly performed on GPUs containing thousands of CUDA cores. Recent advancements in dedicated low power hardware for DNNs, such NVIDIA Jetson TX2, Xavier AGX cards or the 'Movidius Neural Compute Stick', Intel (2019), may be also used to perform inferences locally in the drone.

Our current prototype employs a distributed architecture consisting of a Ground Control Station (GCS) that performs the heavy computations required for training and inference on a high performance GPU. A client in the drone sends images from the camera and receives a probability value indicating if the image contains a wind mill or not. This approach is flexible but has the disadvantage of reducing the level of autonomy and requires a high bandwidth wireless connection to send the images from the camera installed on the drone and to receive commands from the GCS. Streamlining this process will be attend to in future work.

Wind Mill Inspection Our proposed approach for wind mill inspection can be briefly described as follows: 1) The quad-rotor takes off at a random location 2) The perception algorithm based on DNN searches for the wind mill; 3) When the windmill is found the quad-rotor flies towards the desired trajectory.

However, the ultimate goal in wind mill inspection is that the drone could recognize a landmark within the windmill (such as the center of the mill) to make this the initial condition and apply a localization algorithm such as simultaneous localization and mapping (SLAM). SLAM could then be used to scan the whole windmill for faults by constantly calculating its current position within the wind mill space.

3. PERCEPTION-BASED NAVIGATION

This section first describes the DNN and how it was trained, followed by a detailed explanation of the vision based navigation technique.

3.1 Wind Mill Recognition Network

Training of DNNs requires the use of large data-sets to allow them to accurately recognize multiple objects. For instance, AlexNet is able to recognize 1000 different objects, but was trained with millions of images. To avoid this issue, we used transfer learning, adapting AlexNet to make it capable of recognizing wind mills. We called our DNN model, WindMillNet. Figure 1 shows partially the architecture of WindMillNet. For a detailed explanation of all layers in AlexNet see Krizhevsky et al. (2012). WindMillNet has 22 layers including a combination of ConvNets, ReLUs, dropout, polling, fully connected and softMax functions layers.

The last 3 layers of AlexNet were replaced to adapt them to the new wind mill classification task. These layers; are a fully connected network layer, a softmax layer and a classification layer. The fully connected layer, multiples the pre-trained weights associated to the basic features learned by previous layers and adds an offset. The softmax layer applies the softmax function to its inputs to produce a value in the range $[0, 1]$ that represents the probability $P(l) \in [0, 1]$ that an image corresponds to a specific object. Finally, the classification layer calculates the cross entropy loss, a step normally performed in multi-classification problems where the multiple classes are mutually exclusive. To train WindMillNet we used a small

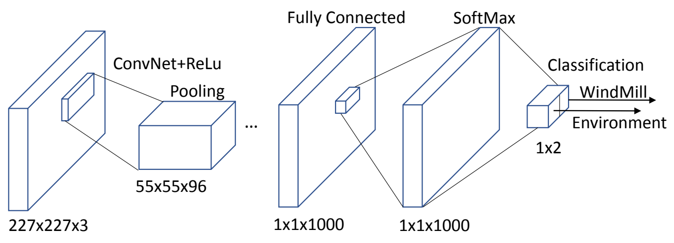

Fig. 1. Architecture of WindMillNet

data-set of 50 wind mills and 50 images representing our testing environment that included different images of our experimental setup room. For data augmentation we used synthetic reflexions and translations to get a total of 300 images; 70% of these images were used for training and 30% for validation. The first layers of the network were trained with stochastic gradient decent using a batch size of 10 and a low learning rate of 0.0001. The last layers were trained with a fast weight learning and bias rate of 20 so that they could learn the new features of wind mills faster.

Figure 2 shows the training progress of the network in 20 epochs with 9 iterations per epoch. WindMillNet reached an accuracy of 96.46% after being trained on a GPU with 1152 CUDA cores.

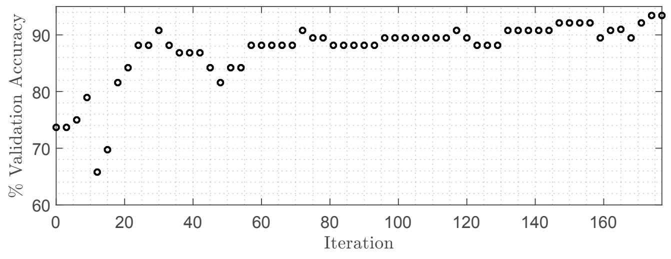

Fig. 2. Accuracy progress in validation

3.2 Vision based navigation using DNN

After taking off, the drone is set to scan the horizon for a windmill by rotating in the yaw direction, during which images are captured by the camera and sent to the GCS. These images are fed to WindMillNet, that classifies them and calculates in real-time the probability $P(l)$ that a wind mill appears in an image. The vision based navigation is a type of event based control, a detailed explanation follows.

Since the drone keeps moving while searching for a wind mill, $P(l)$ will change over time. $P(l)$ will be maximum when the whole wind mill appears in the field of vision of the drone's camera. When that probability is above a threshold value, i.e. $P(l) > threshold$, the drone stops scanning, and the event that moves the drone towards the windmill is triggered. The forward movement event is stopped when the drone reaches a safe distance from the wind mill.

We did not consider the use of optical zooming to increase this probability. In this scenario the effect of the initial distance from the drone to the wind mill on WindMillNet's accuracy should be taken into account. If the wind mill is initially placed too far away from the drone, it will have a relatively small size on the camera that could confuse WindMillNet to mistakenly classify it as part of the environment. To reduce this effect, once the initial scanning of the environment is performed and no windmill has been detected (i.e. if $P(l) < threshold$), our vision system splits the image into equally sized smaller segments and calculates in parallel using four copies of WindMillNet, the probability that each segment may contain a wind mill, with individual probabilities, $P(l)_{ij}$, where i and j are the row and column of the segmented image. This is shown in Figure 3. We call this a **Compounded Eye** vision system, which draws its inspiration from a fly's eye, where each segment is analogous to a fly's lens Srinivasan (2011). This strategy increases the probability of detecting a wind mill in cases where its initial relative distance or position from the drone is too far away.

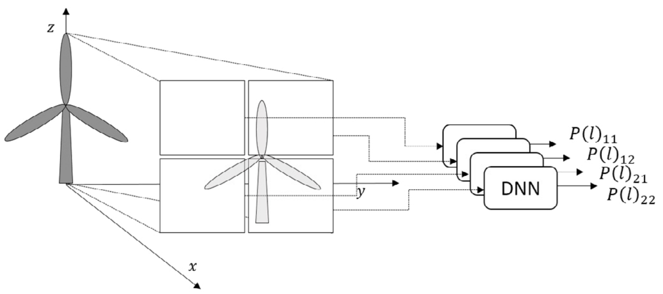

Fig. 3. Parallel DNNs probability calculation for each segment

A similar method is applied to recognize objects in other image segmentation methods. In some of these methods a sliding window is placed on an image and shifted continuously. The window's content is analyzed by a DNN looking for objects similar to the ones used when training it with manually labeled images. Other approaches start with a small window that is increased incrementally as the probability for finding a known labeled object increases.

4. DRONE DYNAMICS AND NAVIGATION CONTROL

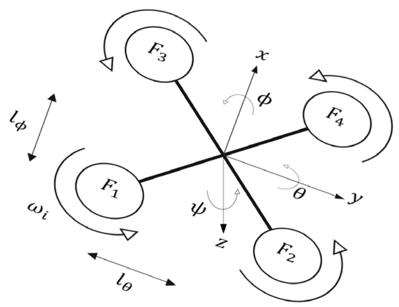

Fig. 4. Kinetic variables and coordinates describing the drone's dynamic.

In this section, a series of PID controllers are designed aiming at stabilizing the quad-rotor and enable its motion towards the wind mill. The quad-rotor is a 6-DOF underactuated system, whose dynamics are modeled based on the works of Dikmen et al. (2009), Bolandi et al. (2013), Choi and Ahn (2015) as follows:

$$m\ddot{\xi} = FRe_3 - mge_3, \tag{1}$$

$$I\ddot{\eta} = \tau - C(\eta, \dot{\eta})\dot{\eta}. \tag{2}$$

where $\xi = [x, y, z]^T$ are the translational positions of the quad-rotor in the inertial frame, $\eta = [\phi, \theta, \psi]^T$ are the Euler angles, m is the mass of the quad-rotor, R is the rotation matrix from body-fixed frame of the quad-rotor to the inertial frame, $C(\eta, \dot{\eta})$ is the *Coriolis matrix*, F is the translational force in the inertial frame, $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^T$ are the control torques in roll, pitch and yaw, I is the inertial matrix and $e_3 = [0, 0, 1]^T$. The coordinates of the quad-rotor are shown as in figure 4.

The control input of each motor of the quad-rotor is a PWM pulse from 0 to 1. Therefore, the control forces (F, τ) need to be further transformed into the PWM values of motors, which is denoted by $[u_1, u_2, u_3, u_4]$. According to the balance of motor moments, the linear mapping between control forces and PWM values of motors can be given by

$$\begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} F_f & F_f & F_f & F_f \\ -F_f l_\phi & -F_f l_\phi & K_f l_\phi & F_f l_\phi \\ F_f l_\theta & -F_f l_\theta & K_f l_\theta & -F_f l_\theta \\ T & -T & -T & T \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \tag{3}$$

where F_f is the maximum force of each motor with PWM at 1, l_ϕ is the distance between motors F1 and F2 to the center of the quad-rotor along forward/backward direction as shown in figure 4, l_θ is the distance between motors F1 and F3 to the center of the quad-rotor along sideways direction, and T is the motor torque constant. It is noted that the mapping matrix in equation 3 between the control forces and motor moments is invertible and thus $[u_1, u_2, u_3, u_4]^T$ can be uniquely determined by the control force $[F, \tau_\phi, \tau_\theta, \tau_\psi]^T$.

Since the *Coriolis matrix* is highly nonlinear and thus difficult to determine accurately, we propose a series of PID controllers using inner-outer loop control strategy, where the outer loop is for translational motion control and the inner loop is for attitude control. The structure of the control system of the quad-rotor is shown in figure 5. When the quad-rotor recognizes the wind mill, it generates the desired position, (x_d, y_d, z_d) and yaw angle ψ_d, that keeps a safe distance from the wind mill. Three PID controllers are designed for the *outer loop* to track the desired position, where the PID controller in z direction generates the translational force for altitude motion, while the other two PID controllers in x and y directions generate the desired pitch θ_d and roll ϕ_d angles, respectively. Then the desired Euler angles $(\phi_d, \theta_d, \psi_d)$ are further sent to the inner loop to generate the control torques in roll, pitch and yaw by three more PID controllers. The parameters of PID controllers are experimentally tuned.

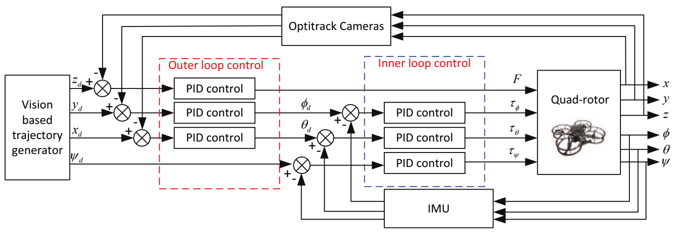

Fig. 5. Structure of navigation control system.

5. IMPLEMENTATION

The implementation was done on Quanser's Autonomous Vehicles Research Studio (AVRS) Quanser (2018). This setup consists of a quad-rotor drone based on the Intel Areo platform, providing a Linux based operating system running on a quad core Intel atom chip and Intel RealSense R200 Camera Intel (2018). The RealSense consists of 3 cameras, a RGB camera and stereoscopic IR to produce depth. For this work we only take advantage of the RGB camera, which allows us to capture images with a resolution up to 1920x1080 pixels at 30 fps. In this work Intel's RealSense image processing library was not used.

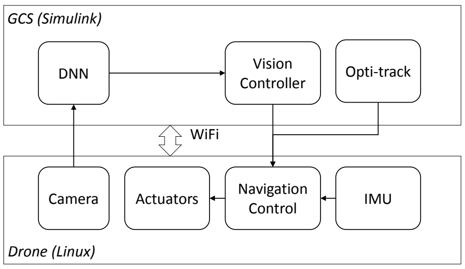

Fig. 6. Block diagram of GCS and drone's main components

The platform communicates with the GCS via wireless communication using the TCP/IP protocol, through a hi-speed router. The GCS is a windows desktop computer running Matlab/Simulink and Quanser's Quarc Software, Quanser (2018). A block diagram of the implementation is shown in figure 6. For our experiments the video is captured in RGB, VGA resolution and sent to the GCS. The images that are received are rescaled on the GCS, since the WindMillNet takes an image of size 227x227x3 pixels. The GCS is equipped with a CUDA capable GPU, the Nvidia TI-1050 GPU. WindMillNet is generated using Matlab GPU coder which generates optimized CUDA code, and is executed as a Simulink S-function. In addition, the system is equipped with six Opti-Track flex 13 cameras, which allow for precise tracking of the drone's movement (calibration reports a precision below 0.5 mm).

6. RESULTS

The results section consists of two parts, the first part tests the vision based navigation algorithms and the second investigates the compounded eye algorithm.

6.1 Image Detection and Localization

To evaluate the implemented control strategy, a small experiment was performed, where the drone was set to scan for a windmill printed on a sheet of paper attached to the wall. The experiment was filmed from the ground and drone's perspective. Three images are shown in figure 7, which show how the drone is scanning for the wind mill *(image 1)*, and when it finds it *(image 2)*, it moves towards it *(image 3)*. The positions of the drone have been

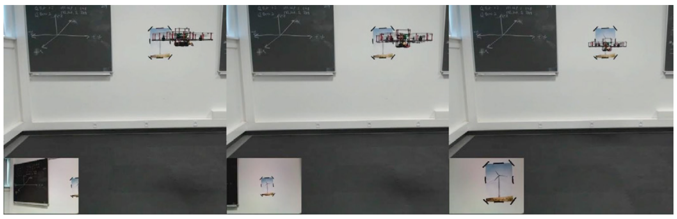

Fig. 7. Snap Shots of the drone scanning the horizon for the windmill, three images show how the drone is scanning and navigating towards the photo of a wind mill

plotted in figures 8 and 9. From figure 8 it is clear how the drones yaw movement occurs for π rad, until the windmill is detected. After this, the drone moves in the x direction towards the wind mill. In this experiment, the drone was set to move to a safe distance from the windmill, 1 m, and then stop.

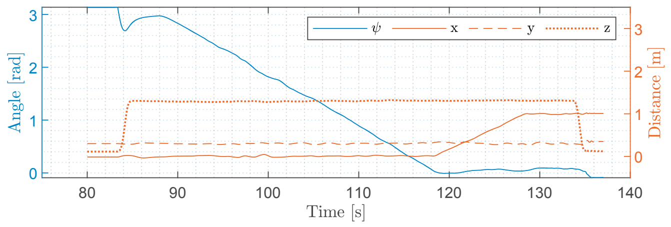

Fig. 8. Position of the drone during the experiment

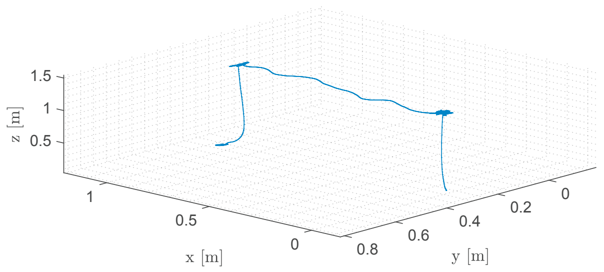

Fig. 9. Position of the drone in 3D, during the experiment

6.2 Segmented Image

The following experiment evaluates our Compounded eye algorithm. Each image from the camera was segmented into 4 parts. Each part was fed into an instance of WindMillNet and the probability that the segment may contain a wind mill was calculated in parallel. This is shown in Figure 3. We performed an experiment moving the drone around the wind mill image circularly in a counterclockwise fashion. Figure 10 shows how the probability for each segment changes, being maximum when the windmill is appearing sequentially in the quadrants $[1,1],[1,2],[2,1],[2,2]$ with probabilities $P(l)_{11}$, $P(l)_{12}$, $P(l)_{21}$, $P(l)_{22}$ respectively. The corresponding images from the camera are shown in figure 11.

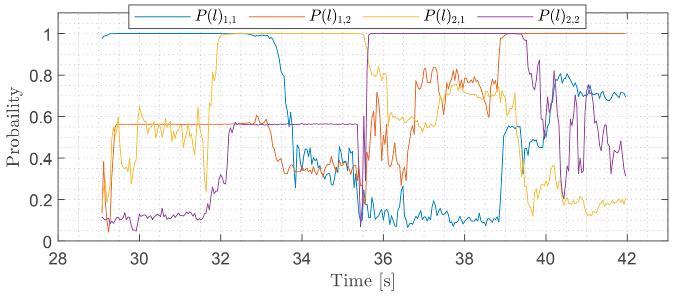

Fig. 10. Probability of segments in the segmented image using the compounded eye, experiment is performed in real-time with video feed of 30 fps

7. CONCLUSIONS

In this work we have used transfer learning to create an image recognition network called WindMillNet that

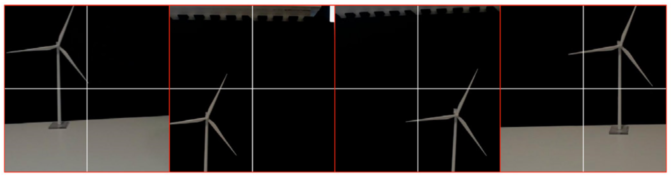
Fig. 11. Four images from the camera feed, showing the view for time 30s 34s 37s 41s, with respect to figure 10

detects wind mills. The output of WindMillNet was used as a feeding parameter to our event based controller for a quad-rotor drone, which aided in finding a wind mill. WindMillNet analyzes images from a live video-stream while the drone is flying. Our experiments showed that we were able to successfully detect and fly toward a model of a wind mill. This method was extended to include what we call the compounded eye method, where an image is segmented and each segment in processed in parallel, in order to increase the likelihood of locating a windmill within an image.

In our future work we will use the 2D spacial information obtained from the compound eye to control drone's movements. This will allow us to navigate autonomously to a wind mill, independent of the initial position of the drone. We will also work on changing the architecture of WindMillNet and the way it is trained to improve its accuracy.

REFERENCES

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316.*

Bolandi, H., Rezaei, M., Mohsenipour, R., Nemati, H., and Smailzadeh, S.M. (2013). Attitude control of a quadrotor with optimized pid controller. *Intelligent Control and Automation*, 4(03), 335.

Bristeau, P.J., Callou, F., Vissiere, D., Petit, N., et al. (2011). The navigation and control technology inside the ar. drone micro uav. In *18th IFAC world congress*, volume 18, 1477–1484. Milano Italy.

Choi, Y.C. and Ahn, H.S. (2015). Nonlinear control of quadrotor for point tracking: Actual implementation and experimental tests. *IEEE/ASME transactions on mechatronics*, 20(3), 1179–1192.

Deign, J. (2016). Fully automated drones could double wind turbine inspection rates. URL http://newenergyupdate.com/wind-energy-update/fully-automated-drones-could-double-wind-turbine-inspection-rates.

Dikmen, İ.C., Arisoy, A., and Temeltas, H. (2009). Attitude control of a quadrotor. In *Recent Advances in Space Technologies, 2009. RAST'09. 4th International Conference on*, 722–727. IEEE.

DJI (2018). MATRICE 600PR. URL https://www.dji.com/dk/matrice600-pro.

Drews, P., Williams, G., Goldfain, B., Theodorou, E.A., and Rehg, J.M. (2018). Vision-based high speed driving with a deep dynamic observer. *arXiv preprint arXiv:1812.02071.*

Giusti, A., Guzzi, J., Ciresan, D.C., He, F.L., Rodríguez, J.P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., et al. (2016). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2), 661–667.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hu, J., Shen, L., and Sun, G. (2017). Squeeze-and-excitation networks. *CoRR*, abs/1709.01507. URL http://arxiv.org/abs/1709.01507.

Hua, M.D., Hamel, T., Morin, P., and Samson, C. (2013). Introduction to feedback control of underactuated vtolvehicles: A review of basic control design ideas and principles. *IEEE Control Systems*, 33(1), 61–75.

Intel (2018). Intel Aero Ready to Fly Drone. URL https://click.intel.com/intel-aero-ready-to-fly-drone-2479.html.

Intel (2019). Movidius neural compute stick. URL https://www.movidius.com/.

Jamieson, P. (2018). *Innovation in wind turbine design.* John Wiley & Sons.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Loquercio, A., Maqueda, A.I., del Blanco, C.R., and Scaramuzza, D. (2018). Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2), 1088–1095.

Pomerleau, D.A. (1989). Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, 305–313.

Pomerleau, D.A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1), 88–97.

Quanser (2018). Autonomous Vehicles Research Studio. URL https://www.quanser.com/products/autonomous-vehicles-research-studio.

Raconteur (2016). The commercial potential of drones in 5 charts. URL https://www.raconteur.net/technology/the-commercial-potential-of-drones-in-5-charts. Online; accessed 12 July 2018.

Razavian, A.S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, 512–519. IEEE Computer Society, Washington, DC, USA. doi:10.1109/CVPRW.2014.131. URL http://dx.doi.org/10.1109/CVPRW.2014.131.

Smolyanskiy, N., Kamenev, A., Smith, J., and Birchfield, S. (2017). Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness. *arXiv preprint arXiv:1705.02550.*

Srinivasan, M.V. (2011). Visual control of navigation in insects and its relevance for robotics. *Current opinion in neurobiology*, 21(4), 535–543.

Szegedy, C., Ioffe, S., and Vanhoucke, V. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261. URL http://arxiv.org/abs/1602.07261.