A bio-motivated vision system and artificial neural network for autonomous UAV obstacle avoidance

Máté Pethő
Department of Physiology and Neurobiology
Eötvös Loránd University
Budapest, Hungary
mate.petho@ttk.elte.hu

Ádám Nagy
Faculty of Information Technology and Bionics
Pázmány Péter Catholic University
Budapest, Hungary
nagy.adam@itk.ppke.hu

Tamás Zsedrovits
Faculty of Information Technology and Bionics
Pázmány Péter Catholic University
Budapest, Hungary
zsedrovits.tamas@itk.ppke.hu

***Abstract*—Unmanned aerial vehicles (UAVs) becoming more and more common. They show excellent potential for multiple types of autonomous work, although they must achieve these tasks safely. For flight-safety, it must be assured that the UAV will avoid collision with any objects in its flight path during autonomous operations. Computer vision and artificial neural networks have shown to be effective in many applications. However, biological vision systems and the brain areas responsible for visual processing may hold solutions capable of acquiring information effectively. We are proposing a novel system, which performs visual cue extraction with algorithms based on the structure and functionality of the retina and the visual cortex of the mammalian visual system, and a convolutional neural network processing data to detect a predefined obstacle using the onboard camera of the UAV. We also examined the effect of preprocessing on calculation time and recognition effectiveness.**

***Index Terms*—UAV, bio-motivated, convolutional neural networks, computer vision, obstacle avoidance**

I. Introduction

Unmanned aerial vehicles (UAVs) are becoming more and more common, and they show excellent potential in many fields, such as aerial imaging, inspection tasks, and other remote sensing applications [1]. The market of UAVs might grow to US$ 51.86 billion by 2025 from US$ 11.45 billion in 2016, which shows their great economic potential as well [2]. Autonomous UAVs could achieve self-sufficient task completion, but to do so, they would need precise and fast obstacle-avoidance algorithms, so they could operate without endangering themselves or their surroundings.

Many previous works implemented biologically motivated solutions for various problems considering tasks for UAVs [3], [4]. An excellent example of that is the work of Nitin et. al. They used the strategy of bees to move a small UAV through a previously unknown gap on a wall using its camera and on-board processing capabilities. The UAV moved in a "zig-zag" fashion in front of the gap. Data was acquired by optical flow processing, and a deep-learning-based algorithm achieved gap detection. After these processes, the UAV was able to maneuver through the gap with high velocity [3].

The mammalian vision system is one of the most precise sensor-system, which makes animals capable of navigating during complex tasks in various environments (for example, hunting or fleeing from a predator). The first stop of visual processing is the retina, which is a multi-layered sensor system.

The retina is capable of converting photons into action potential coded information, which is the method of information flow throughout the nervous system. From the photoreceptors, rods are susceptible to luminance, while the three different cone types (L-, M-, S-cones) are receptive to red, green, and blue wavelengths of the incoming light, respectively [5]. The retinal ganglion cells operate with concentric receptive fields with varying sizes. The receptive fields can be divided into ON and OFF regions; the interplay of these regions extract visual cues for further processing [5]. Information flows in parallel pathways from the retina towards the brain on the optic nerve, containing information from contrast [6], movement direction , edge information [7], and others. Thus, information reaching the primary visual cortex (V1) already contains multiple types of information. In the primary visual cortex, detailed edge-information is acquired from the field of vision by the cortical columns [8]. Further processing (color processing, motion- and depth perception, cognitive mapping) is achieved in the higher-order visual cortices (V2, V3, V4, and others) [9].

Artificial neural networks are suited for classification problems, such as obstacle recognition in the flight path. Complex networks can process multiple information modalities, thus increasing the probability of giving the right answer for a problem. To that end, we can increase the number of hidden layers slightly, but the best solution is to build a hierarchical network. These modular artificial neural networks have strongly separated architectures. Each network will compute its domain [10]. Applications of deep convolutional networks in computer vision are becoming more and more prevalent [11]. Convolutional networks are powerful tools for classification and recognition problems, as they can be taught by fewer images than a conventional neural network. A good example of that is the U-net by Ronneberger et al. This network was created to solve biomedical segmentation applications [12] efficiently, we used it as a base for our network.

In this paper, a biologically-motivated vision system is proposed, which mimicks the processes of the retina and visual cortex of mammals to acquire visual cues (contrast and edge information) and also contains a neural network based on U-Net [12] to process the created data. Examination of the effect of image preprocessing on the learning effectiveness of the neural network was performed subsequently. Tests were also conducted to examine the proposed system's applicability for real-time processing and autonomous flight operation with various data sets, modalities, and scaling factors.

This research has been partially supported by the European Union, co-financed by the European Social Fund through the grant EFOP-3.6.3-VEKOP-16-2017-00002. This work has received funding from Pázmány Péter Catholic University KAP RD and equipment grants (KAP18-51005-1.1-ITK, KAP18-53019-3.6-ITK, and KAP18-53019-3.6-ITK). This research has been partially funded by the Hungarian National Research, Development and Innovation Office Postdoctoral Researcher funding PD 128699. This reseatch was partially supported by the ÚNKP-20-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

II. Proposed system

To achieve obstacle recognition and avoidance, three main tasks must be performed: 1) visual cue extraction, 2) obstacle recognition after learning, and 3) navigation based on the recognition. To model the processes of biological systems in the feature extraction, algorithms were implemented modeling the functions and structure of the retina and primary visual cortex. To achieve learning and autonomous obstacle detection, a data set was created and annotated marking a pre-defined obstacle, which was used to train a neural network.

A. Bio-motivated image preprocessing

Existing computer vision algorithms, as well as biological visual systems, extract visual cues such as contrast, edge, and movement information. In the proposed implementation, two main feature types (contrast and edge information) were extracted using algorithms modeling the processes in the retina and the primary visual cortex of mammals (example outputs are in Fig. 1). The images were also down-scaled to filter it and enhance processing speed.

Contrast can provide information about more significant differences in the input image. The basis of our contrast detecting algorithm is a two-dimensional Gaussian function:

$$\Phi(x,y) = \frac{1}{2\pi\sigma^2} e^{-(\frac{1}{2\pi\sigma^2}(x^2+y^2))} \tag{1}$$

where σdenotes the width of the bell-shaped surface and is the only free variable.

The difference of Gaussians (DoG) function models the signal processing of retinal ganglion cells [13]. Retinal ganglion cells have a central and surrounding receptive field. The interaction of these two regions provides information on contrast, movement, and other features, even before the information reaches higher-level processing brain regions (such as the primary visual cortex) [5]. As in the retina, multiple color channel comparison were performed to gain the contrast information, such as red-green and yellow-blue discrimination, using the following equations:

$$\begin{aligned} RG_{input} &= \frac{R+G}{\sqrt{2}} \\ YB_{input} &= \frac{R+G-2B}{\sqrt{6}} \end{aligned} \tag{2}$$

where R is red, G is green, B is blue color channels of the input picture.

Blue color channel and grayscale information were also processed using the DoG function, as in the case of blue midget cells and the rod pathway in the retina [5]. The process of calculating the intensity based on the input of the three cones was performed by averaging all three color channels. To imitate the process, the created DoG function was convolved on the preprocessed images (channel selection, discrimination).

Edge information is also an essential visual cue in the case of obstacle detection. To acquire this modality, Gabor functions were used as follows:

$$\begin{aligned} g_c(x,y) &:= \cos\omega_x x + \omega_y y e^{-\frac{x^2+y^2}{2\sigma^2}} \\ g_s(x,y) &:= \sin\omega_x x + \omega_y y e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{3}$$

where σrepresents the width of the receptive field, and it was set to be 5, 7, 9, 14, or 20, respectively, to represent the size-dependent sensitivity of ganglion cells in the primary visual cortex. ω_x/ω_y gives the preferred orientation of the receptive field, which was chosen to be $\{-45, 0, 45, 90\}$ thus covering the principal directions. Gabor function was implemented on the output of the three-cone intensity process, which used the average of all the color channels convolved with the DoG kernel. The resulting output resembles the information gained in the primary visual cortex, where cortical columns gather information from edges with a different direction separately [8]. An example of the various preprocessed images is shown in Fig. 1.

Fig. 1: Examples of the output images after processing: the six different contrast images [from up to down and left to right: rod image processing (grayscale image), amacrine image processing (grayscale image), blue cone image processing (blue color channel-based), red-green discrimination (red and green channel-based), yellow-blue discrimination (red, green and blue channel-based) and all cone discrimination (mean of all color channels)] and two examples from the 20 separate images containing edge-information.

As a result of the previously described processes, the system can generate a maximum of 26 outputs (6 containing contrast information, 20 containing edge-information). From these modalities, multiple variations were used for training to see the effect of preprocessing on obstacle detection. The main tests were conducted using 12 modalities from which 6 contained contrast information, while 6 contained edge information as a preliminary test showed this to be the best setup.

The images were also downscaled as it may enhance the detection of more abundant features, and it may also greatly decrease the calculation time as the original calculation time would make real-time processing practically impossible. This process also shows similarities to the mammalian visual system, where information is channeled toward the brain on a decreasing number of parallel channels, while this number dramatically increases after reaching the central nervous system, which can be crudely modeled by the neural network.

B. Obstacle recognition using neural network

The next goal was to teach the UAV to avoid a predefined obstacle autonomously (display panel, an example in Fig. 3). The effect on the learning efficiency and calculation speed of multiple preprocessing methods were examined, such as the previously presented biologically motivated visual processing algorithms and downscaling.

A previously proposed network structure called U-net [12] was used the basis of the neural network. This convolutional neural network was created to solve segmentation tasks on biomedical images. U-net contains a contracting (down-sampling) and an expansive part. In the contracting part, 3x3 convolutions are applied, ending in the rectified linear unit (ReLU). It is finished in a 2x2 max pooling operation. The expanding part contains feature map upsampling followed by 2x2 convolution and two 3x3 convolutions. It ends in a final convolution to get the desired output shape. All levels from the

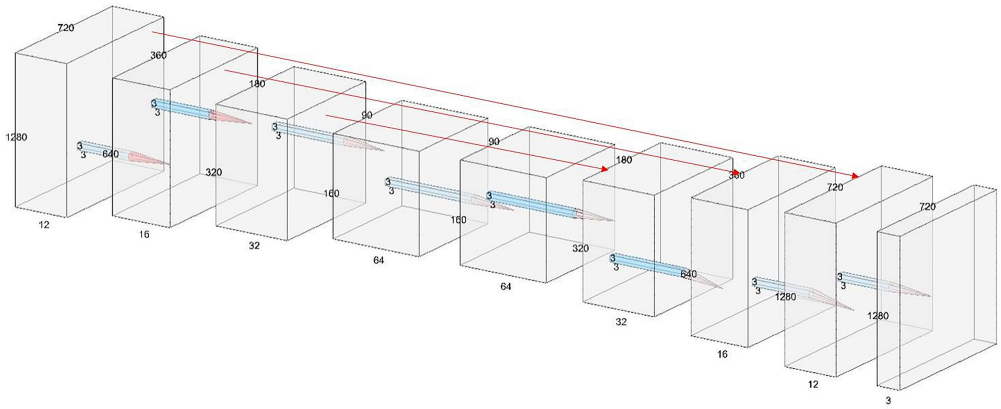

Fig. 2: The structure of the network built to train with pre-processed images. In the case of training without pre-processing the 3 dimensions of the input is 3. Network structure is based on U-net [12]. The dimensions of the layers are denoted on the figure. Blue columns denote the convolution between layers (with its dimension), while red arrows show the concatenation between layers.

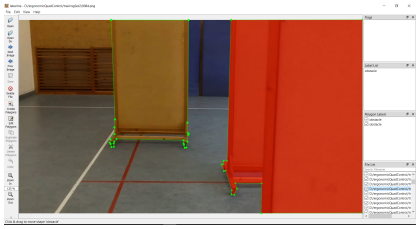

Fig. 3: Example of an image annotated using LabelMe. Display panels were marked separately.

down-sampling part are cropped and concatenated to its equivalent on the expanding part. The detailed layer structure is shown in Fig. 2.

The neural network was built using Tensorflow [14]. New raw data was created and annotated for training and testing purposes. To this end, on-flight videos were captured using a Tello Edu UAV [15]. The recordings were created at two separate sessions. Thus luminance is slightly different in the test area. These videos were later segmented into frames. LabelMe [16] was used to annotate the display panels on the images (Fig. 3). As the strength of U-net lies in the fact that it needs fewer images than other convolutional networks, we annotated a moderate number of images (1534).

To examine the effect of the data set itself on the training, 4 separate training and test sets were created from the 1534 images, from which one training set was constructed using images from only one recording (training set 1/test set 1), two was established randomly from all the annotated images (training set 2/test set 2, training set 3/test set 3), while one training set was constructed manually to make it as diverse as possible using all the available images (training set 4/test set 4). All training set contained 800 images. The remaining of the 1534 images were used as a test set.

The network was trained using various setups: 1) 3000 iterations with unscaled, preprocessed data, 2) 3000 iterations with unscaled, original data, 3) 3000 iterations with 10x downscaled, preprocessed data, 4) 5000 iterations with 10x downscaled, preprocessed data, 5) 10000 iterations with 10x downscaled, preprocessed data, 6) 3000 iterations with 10x downscaled, original data. From these, setup 3-6 were tested extensively, as downscaling was deemed necessary for real-time application. The learning rate was set to be 0.001.

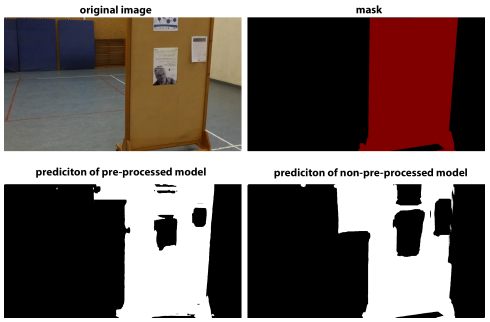

Fig. 4: Examples for the prediction made by the trained neural networks in the case of original unscaled images. Both had smaller efficiency than the downscaled version, pre-processing enhanced its efficiency.

Fig. 5: Example for path search during flight. Green dot denotes pathway, which is considered free.

C. *Navigation based on the neural network model output*

To demonstrate the feasibility of the different obstacle detection variants tests were conducted using a very simple situation. Hence, for the navigation greedy algorithm was used with a simplified version of visual servoing to control the drone. This strategy is not suitable in most real world applications, just served as a basic test case.

As a result of the prediction of the U-net, a map is created where the class of all pixels is calculated. In these segmented frames, pixels that belong to the obstacle are given as logical 1, and pixels belong to the free path are given as logical 0. The task is to find the way through the free path.

First of all the centroids and area of the patches representing the free path are calculated. A greedy algorithm was used to select the way to follow based on the patch area. It means that the one which is chosen to be followed is always the one with the biggest area. As an example, Fig. 5 shows the output of the prediction and centroids

of the two patches. The one with the green dot had the larger area, so that was followed by the aircraft.

The desired behaviour of the system was to move the centroid of this patch to the middle of the image. A constant velocity in the forward direction was applied that is a constant pitch was set. The altitude and yaw angle was also kept constant. The controlled variable was the roll angle of the quadrotor, based on the distance of the centroid from the middle pixel, in the x axis of the image. It means that the quadcopter is flying with a constant velocity forward and it moves left or right based on the location of the centroid in the given frame.

III. Testing data

After multiple models were trained using the four separate training sets, they were tested using the test sets. The network was generally capable of learning to recognize the display panels. To evaluate the models, the number of false positive and negative pixels were calculated first. From these metrics, the false positive and negative percentages were determined to provide the rate of error. Precision and recall were also calculated, which show the fraction of relevant instances among the retrieved instances and the fraction of the total amount of relevant instances that were retrieved, respectively. The precision is calculated as follows:

$$P = \frac{T_p}{T_p + F_p} \tag{4}$$

where T_p is the number of true positive pixels, and F_p is the number of false positive pixels. The recall is calculated as follows:

$$R = \frac{T_p}{T_p + F_n} \tag{5}$$

where F_n is the number of false negative pixels.

Another important factor is processing time which was also measured. The measurements were run on a laptop PC consist of an Intel© Core™ i5-8300H processor, an NVIDIA© GeForce GTX 1050 Ti 4GB, 32GB DDR4 @2666MHz, and 256GB SSD.

The measured statistical values for the whole test set are presented in the form of M±SD, where M is the mean value and SD is the standard deviation.

A. Trained model performance without downscaling

In the case of the models trained without downscaling, we only created two models for preprocessing and without preprocessing. Without downscaling, calculation time would be too long to use the system for real-time obstacle recognition. In the case of preprocessing, 3.98±2.96% of the pixels were false positive, while 1.20±0.61% of the pixels were false negative. Precision was 69.30±29.54%, and recall was 94.11±5.70%. In the case of training without preprocessing, 9.40±9.33% of the pixels were false positive, while 3.27±2.70% of the pixels were false negative. Precision was 50.15±33.65%, and the recall was 82.03±13.98% (Tab. I).

B. Trained model performance with downscaling

Most models were created using 10x downscaling. Tab. I summarizes the mean precision and recall values from each test. Fig. 7 and 8 show the percentage of false positive and negative pixels and the precision and recall of the trained models.

Our results show that preprocessing and downscaling both increased efficiency. When images were used without further preprocessing (model 1), an average of 2.22±1.20% false positive and 0.35±0.12% of false negative pixels were achieved on the test set. Average precision was 85.08±7.26%, while recall was 98.10±0.35%

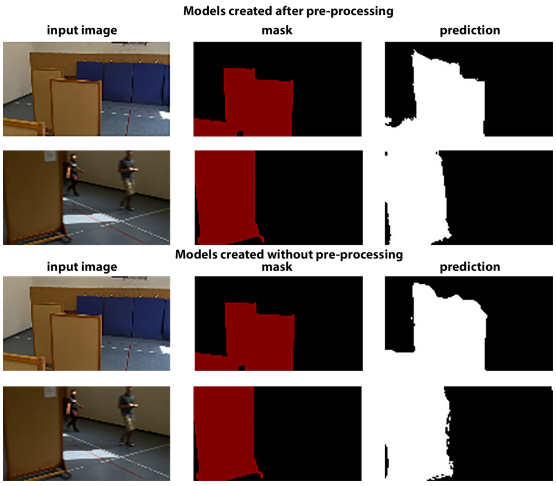

Fig. 6: Examples for the prediction made by the trained neural networks in the case of downscaled images. The first two row shows examples for a model trained after pre-processing the images, while the lower two rows shows examples for a model trained without further preprocessing.

between all data sets. Also, the output prediction contains less noise, at is can be seen in Fig. 6.

The training with 3000 iterations on preprocessed images (model 2) resulted in an average of 2.19±0.72% false positive and an average of 0.74±0.49% false negative pixels through all test sets. The average precision was 82.72±6.35%, while the average recall was 95.81±3.22%. The training with 5000 iterations on preprocessed images (model 3) resulted in 1.71±0.40% false positive pixels and 0.70±0.55% false negative pixels. Average precision, in this case, was 88.00±4.86%, while recall was 96.09±4.85%. Finally, in case of the training on preprocessed images with 10000 iterations (model 4), tests produced 1.62±0.79% false positive pixels and 0.61±0.47% false negative pixels. Average precision was 87.60±4.41%, while recall was 96.70±2.78% in this case.

The best models were acquired using training set 4 during training, in which case the training set was constructed manually. In this case, model 1 resulted in an average of 1.07±0.05% false positive and 0.19±0.09% false positive pixels. The average precision was 87.95±7.57%, while the average recall was 98.43±0.53%. Model 2 resulted in 1.62±0.60% false positive and 0.30±0.11% false negative pixels based on testing. The average precision was 87.69±7.56%, and the average recall was 97.53±0.48% in this case. In the case of model 3, 1.32±0.23% of the pixels were false positive, and 0.20±0.08% percent of the pixels were false negative. Precision was 94.05±1.16%, and the recall was 98.84±0.17%. In the case of model 4 1.30±0.26% of the pixels were false positive, 0.13±0.05% of the pixels were false negative. The average precision was 89.47±6.53%, while the average recall was 98.84±0.17%.

Our overall best model was created from one of the random training sets, with downscaling, preprocessing, and 10000 iterations during training. In the case of the model, 0.75±1.18% of the pixels were false positive, while 0.77±2.24% of the pixels were false negative on the test images. Precision was 97.66±5.02%, and recall was 97.18±9.74%.

However, it must be considered that, in some cases, the solution of the model was better than the provided one. For example, in the case of some images, papers were pinned on the display panels. While the model could solve that, it is not part of the display panel, for

		training set 1	training set 2	training set 3	training set 4
model 1	precision	80.31± 10.39%	78.12± 7.68%	93.95± 1.82%	87.95± 7.57%
	recall	97.61± 1.78%	98.15± 0.58%	98.22± 0.73%	98.43± 0.53%
model 2	precision	81.05± 5.63%	74.44± 6.39%	87.69± 12.78%	87.70± 7.56%
	recall	90.99± 3.22%	97.21± 0.79%	97.52± 0.75%	97.53± 0.48%
model 3	precision	86.02± 3.44%	82.65± 10.15%	89.30± 3.54%	94.05± 1.16%
	recall	91.52± 2.58%	96.95± 0.75%	97.85± 0.65%	98.06± 0.69%
model 4	precision	82.39± 1.11%	85.95± 5.20%	92.56± 4.32%	89.47± 6.53%
	recall	92.62± 2.85%	97.67± 0.52%	97.70± 0.45%	98.83± 0.17%

TABLE I: Summary of the precision and recall values for the individual tests on the scaled images. Model 1 depicts the model resulting from the training with downscaled images without further preprocessing (3000 iterations), while method 2-4 depict the model resulting from the training sessions with preprocessing (3000, 5000 and 10000 iterations, respectively) on each test sets. Training set 1 was constructed using images from only one recording, the training set 2 and 3 were established randomly from all the annotated images, while training set 4 was constructed manually to make it as diverse as possible using all the available images.

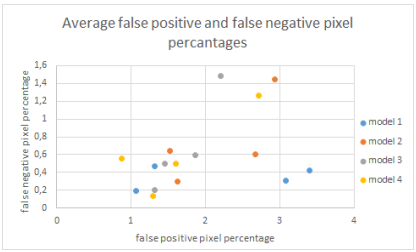

Fig. 7: Scatter plot of the false positive and negative pixel percentages. The X-axis denotes the false positive percentage, while the y-axis denotes the false negative percentage. Test sets are denoted by color code, and test types are denoted by shape.

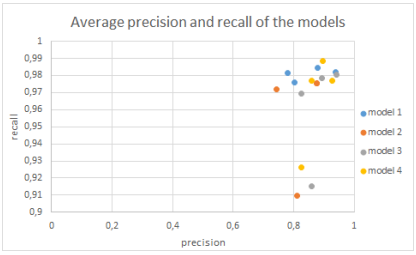

Fig. 8: Scatter plot of precision and recall. The X-axis denotes the precision, while the y-axis denotes the recall. Test sets are denoted by color code, and test types are denoted by shape.

convenience, it was part of the obstacle class annotated in LabelMe (Fig. 3). Although these index-numbers are significant to determine efficiency before experimental testing, based on these data, the best models could be tested during autonomous flight.

C. Calculation time of the processes

Calculation time was also determined for the different processes to examine the possibility of real-time application. Two intervals were calculated: 1) calculation time of the preprocessing, 2) calculation time of the neural network.

In the case of the images without downscaling, calculation times were too large for real-time application, as preprocessing took 3523.1±28.3 ms, while the calculation time of the neural network was 306.1±7.9 ms, resulting in an average of 3831.1±20.4 ms run time for one iteration. With downscaling, this could be decreased significantly.

With the downscaled images preprocessing occurred in an average of 65.7±5.9 ms, while the calculation time of the neural network was 7.7±0.2 ms. The mean total calculation time is 73.5±5.9 ms,

Fig. 9: Test track consisting of three display panels. The UAV had to avoid the front one, then pass through the two remaining panels.

which means that the system is capable of operating at 13.62 Hz even with preprocessing executed. Without preprocessing, execution time before the neural networks shows a mean calculation time of 36.7±2.6 ms. This results in a mean total calculation time of 44.5±2.9 ms, which means that the system is capable of operating at 22.47 Hz. Thus, although preprocessing prolongs calculation time, it is still in an interval which is sufficient for real-time application.

D. Experimental results

Finally, tests were conducted to examine the applicability of the proposed system. To that end, a test track was created consisting of three display panels. Some of the best networks were chosen to provide input for a Tello UAV to navigate through the track. Fig. 9 shows the test track and the desired pathway of the UAV. To communicate with the Tello drone the tello_driver node [17] was used in ROS Melodic Morenia [18] on a laptop PC consists of an Intel© Core™ i7-8750HQ, an NVIDIA™ GeForce© GTX 1050 Ti 4 GB GDDR5, 8GB DDR4 @2666MHz, and 256GB SSD running Ubuntu 18.04. The video stream was processed on the laptop and the commands were sent back to the drone based on the calculated centroid of the free pathway, as it is described in Sec. II-C.

Four downscaled models were tested (from which one was trained using only the image as input, while three was trained after further preprocessing). All models contained enough information so that the UAV could pass through the track during testing.

IV. Discussion

The tests from the different models shows that 1) the consistency of the training set largely ameliorate efficiency, 2) downscaling in itself enhanced effectiveness in case of recognition problems, where large structure have to be classified, and similar structure are present in the background (wooden wall), 3) the other proposed preprocessing

methods may also enhance effectiveness, but the effect is smaller in the case of downscaling than in the case of original image size.

As we could see (Fig. 4), the non-scaled model – although it can identify the panels – has a much larger percent of false positive pixels, which in turn decreases precision by a large margin. The recall was still good, as false negative pixels were still few. In this case, preprocessing was necessary, as a model trained with preprocessed images resulted in much better recognition. Non-pre processed images could not distinguish between the display panel and similar structures, and it resulted in the lowest precision as well as recall from all the models (50% and 82%, respectively).

Downscaling increased precision significantly, and as calculation time made a real-time application possible in only the case of downscaled models, we examined those further. As was anticipated, the effectiveness of the trained model depended highly on the composition of the training image set. Having images from two different days (training set 2, and training set 3) with different illumination improved efficiency in itself. If, instead of randomly assigned images, the training set was manually constructed (training set 4), the overall efficiency of all trained models become even better. Although, the best overall model originated from a randomized training set.

The proposed preprocessing methods were intended to increase the number of modalities for the neural network. In the case of the original image size, their effect was more prominent. In the case of the downscaled images, the effect of preprocessing was less prevalent. Although, as calculation time was sufficient even with preprocessing, it is worthwhile to use those methods as well. Increasing the iteration time through training also improved efficiency. Although, 10000 iterations not always improved efficiency even further, which suggests that around that iteration number overfitting may appear.

Based on the results, to achieve the fastest calculation time possible, it is better to use the original image as input after only downscaling. Although, as we could see, calculation time with preprocessing is around 73.5 ms, which makes it possible to have a data processing speed of 13.62 Hz. Based on the experimental results, it is sufficient to navigate the UAV autonomously, using moderate flight speed. Limitations would appear only if the flight speed would be increased, but a more sophisticated navigation system may counter those limitations.

Although the system was capable of navigating through the test track, its limitations must be considered as well. The moderate number of test images makes it more susceptible to current conditions. Changes in illumination may influence effectiveness, although a more significant number of images or augmentation of gain may counter this problem. Also, this relatively small number of images would make it hard for the system to operate in an environment, which is very different from the one where the videos were recorded. Although display panels could be identified in such an environment, more false positive obstacles could appear.

Future work may include increasing the number of images in the training and test sets obstacle types, in which case preprocessing may become more critical as small features may have to be recognized. Also, navigation can be improved as well, to have a more sophisticated response system for the processed images.

V. Conclusion

In this study, a system was created which is capable of recognizing and avoiding a predefined obstacle. The study shows the effect of preprocessing (feature extraction and downscaling) of the images using biologically motivated approaches on the learning capability. An annotated dataset was created, which can be used for autonomous obstacle recognition and avoidance training in the future. Based on our results, the created models shows good efficiency in recognizing obstacles, as the best model achieved a precision of 97.66±5.02% and a recall of 97.18±9.74%. This data further shows the potential in biologically motivated algorithms. Further classification in the dataset may lead to more complex task completions in the future (recognition of walls, ceiling, and other objects).

References

[1] K. Máthé and L. Buşoniu, "Vision and Control for UAVs: A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection," *Sensors*, vol. 15, no. 7, pp. 14 887–14 916, 6 2015.

[2] The Insight Partners, "Unmanned Aerial Vehicle (UAV) Market to 2025 - Global Analysis and Forecasts by Component by Type and Application," Tech. Rep., 2018.

[3] N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermuller, and Y. Aloimonos, "GapFlyt: Active Vision Based Minimalist Structure-Less Gap Detection For Quadrotor Flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2799–2806, 10 2018.

[4] J. R. Stowers, "Biologically Inspired Visual Control of Flying Robots," Ph.D. dissertation, University of Canterbury, 2012.

[5] H. Wässle, "Parallel processing in the mammalian retina," *Nature Reviews Neuroscience*, vol. 5, no. 10, pp. 747–757, 10 2004.

[6] C. Enroth-Cugell and J. G. Robson, "The contrast sensitivity of retinal ganglion cells of the cat." *The Journal of physiology*, vol. 187, no. 3, pp. 517–52, 12 1966.

[7] T. A. Münch, R. A. da Silveira, S. Siegert, T. J. Viney, G. B. Awatramani, and B. Roska, "Approach sensitivity in the retina processed by a multifunctional neural circuit," *Nature Neuroscience*, vol. 12, no. 10, pp. 1308–1316, 10 2009.

[8] T. S. Lee, D. Mumford, R. Romero, and V. A. Lamme, "The role of the primary visual cortex in higher level vision," *Vision Research*, vol. 38, no. 15-16, pp. 2429–2454, 8 1998.

[9] K. Grill-Spector and R. Malach, "The human visual cortex," *Annual Review of Neuroscience*, vol. 27, no. 1, pp. 649–677, 7 2004.

[10] E. J. W. Boers, H. Kuiper, B. L. M. Happel, and I. G. Sprinkhuizen-Kuyper, "Biological Metaphors In Designing Modular Artificial Neural Networks," in *ICANN '93*. Springer London, 1993, pp. 780–780.

[11] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 2018.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 5 2015, pp. 234–241.

[13] R. A. Young, "The Gaussian derivative model for spatial vision: I. Retinal mechanisms," *Spatial Vision*, vol. 2, no. 4, pp. 273–293, 1987.

[14] M. Abadi, "TensorFlow: learning functions at scale," in *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming - ICFP 2016*. ACM Press, 2016, pp. 1–1.

[15] "Tello Official Website-Shenzhen Ryze Technology Co.,Ltd." [Online]. Available: https://www.ryzerobotics.com/tello-edu

[16] K. Wada, "GitHub - wkentaro/labelme: Image Polygonal Annotation with Python (polygon, rectangle, circle, line, point and image-level flag annotation)."

[17] "tello_driver - ROS Wiki." [Online]. Available: https://wiki.ros.org/tello_driver

[18] "melodic - ROS Wiki." [Online]. Available: http://wiki.ros.org/melodic