Altitude Prediction Over Water Surface Using CNN Trained With Synthetic Images

1st João Vitor da Silva Neto
LASER - UFPB
Universidade Federal da Paraíba
João Pessoa, Brazil
vitor.joao@academico.ifpb.edu.br

2nd Tiago Pereira do Nascimento
LASER - UFPB
Universidade Federal da Paraíba
João Pessoa, Brazil
tiagopn@ci.ufpb.br

3rd Leonardo Vidal Batista
Departamento de Sistemas de Computação
Universidade Federal da Paraíba
João Pessoa, Brazil
leonardo@ci.ufpb.br

***Abstract*—The need for redundant odometry in robotic systems benefits from the recent possibilities of fast and efficient real-time image processing using classifications and regression models. The present work proposes an approach using Convolutional Neural Networks (CNN) to detect the distance between the point of capture of an image and a water surface below the capture. Our CNN was trained initially with synthetic images generated by Blender, and evaluated by real images at heights between 50 and 100 cm. Preliminary tests demonstrated an ability of the system to recognize the changing of height in a continuous shot, varying only the vertical position, even with some noise from elements not present in the training test database, such as dirt from leaves and stones at the bottom of the water.**

***Index Terms*—UAV, water, machine learning, height estimation, synthetic images, odometry**

I. Introduction

With the growing range of possibilities provided by real-time image processing technologies, is natural to find a trend in almost all sensoring technologies: the application of machine learning techniques with traditional sensor captures, assisting, complementing, and even replacing classical numerical methods [1]. The challenges encountered for very specific applications, such as the one dealt with in this present work, demand the survey of methods that are initially unorthodox, often with ramifications in other areas. Even aided by new calculation approaches, sensors application often requires a deeper understanding on the behavior between the hardware and the target, making the hardware choice more specific, and prone to the subject. Otherwise, the equipment may underperform, or even be unsuitable for reliable measurements, regardless of the processing methods.

Currently, the height odometry of aerial vehicles over water lacks accurate measurements, which can lead to the unfeasibility of off-shore applications, the risk of material loss and the high cost of combining multisensor approaches to mitigate the low reliability of data acquisition.

This work aims to present a new approach, based on the premise that it is possible to identify the distance between the vehicle and the water through image captures, using a camera pointed directly at the surface, and processing these images through a convolutional neural network.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Due to the risk of equipment loss, the work began with the rendering of a realistic water surface, and the simulation of the position of the image capture point in order to compose a image database.

After this, the images were processed to add distortions commonly found when capturing images from quadrotors, simulating motion blur. Also, at this stage, a colorspace conversion was carried out to keep the system as light as possible and immune to color differences present in water from different locations.

Once the previous stage was completed, the machine was designed and trained based on an already consolidated neural network model (VGG16), with specific changes for this application.

Finally, real images were captured in a river with a large volume and calm waters for testing, with distances measured between 50 and 100cm at the markings of a pier in the municipality of Cabedelo-PB. These images were used to test the application, presented at the end of the results section.

II. Background and Related Work

Exploring the state of the art of alternatives of height sensoring, new approaches based on the use of images show great potential for identifying the distances between vehicle and surface as seen in artificial intelligence techniques both using neural networks [2] [3], or traditional algorithm-based and hardware-based options with numerical enhancements and optimizations in target detection and tracking operations [4], obstacle avoidance [5] [6], navigation [7], among others [8]. For this purpose, the use of real-time operation technologies such as MobileNet can be considered, offering adequate performance with low latency and low power consumption, so desired in autonomous mobile robotics applications [9].

Currently, the available technologies for altitude estimation for unmanned aerial vehicles are mainly based on barometers, ultrasound, and Light Detection and Ranging (LiDAR) [2]. Such methods have particularities with advantages and disadvantages particular to each technology, making it unfeasible to use them in all and any situations, requiring considerations in applying them according to the terrain, weather conditions, altitude of intended operation, currently lacking a complete method capable of meeting the demands of a generalist way

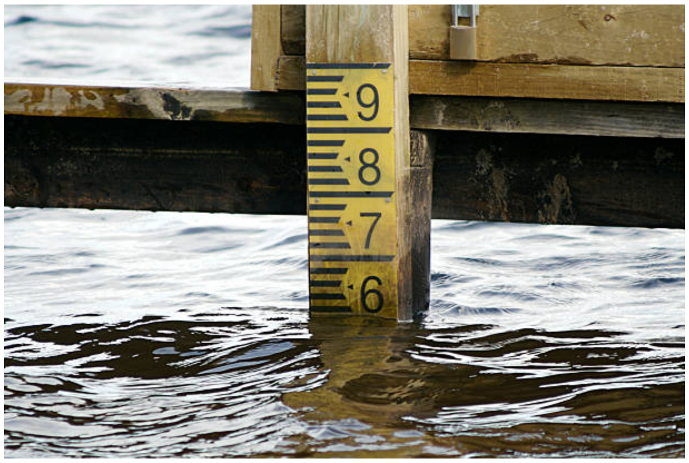

Fig. 1. Example of river level marker gauge. Source: iStock 94928861

[10]. Even using modern sensor fusion filters, some scenarios are still infeasible for autonomous operation, such as highly reflective surfaces found in dunes, snow, and water [11].

The research field of applications with sparse image banks relies on the aid of synthetic images, or modified with elements of real or synthetically generated cut-outs, in order to simulate the desired situations [12] [13] [14] [15]. Initially the process can use much time that could be oriented to experiments, but this allow a safer use of field equipment, especially in proof-of-concept and testing stages, by not requiring the work tools to be subjected to hazardous conditions, or even speed up processes before having them in hand. At the same time, these methods are able to show how accurate the results of predictions and calculations based on the sets manipulated with software tools can be. Moreover, human capital is also saved, dispensing with operators in unhealthy environments,such as wind energy plants, like in [15]. Bringing the discussion to the present scenario, UAV operators would need Operators need a mobile test rig with a camera, computer and secure location to capture images with surface measurements like the one in Fig. 1.

III. Datasets Construction and Training

A. *Artificial Dataset Synthesis*

An image bank was created in Blender, using a photorealistic model based on the application of a time-varying three-dimensional noise generation algorithm to model the waves of a surface with large water volume and low waves activity, such as large lakes, rivers or marine gulfs [16]. In order to help generate the patterns with simpler parameterization, the *Real Water - Waters Shader* plugin was used, with the following maximum turbidity settings and no transparency. The initial result is visible in Fig. 3 a). At the end of parametrization, images were rendered at different heights, contemplating distances of 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 240, 280, 320, 360, 400, 480, 560, 640, 720, 800 centimeters, with 9,000 images each. Using this bank, pre-processing was performed aided by OpenCV in three steps:

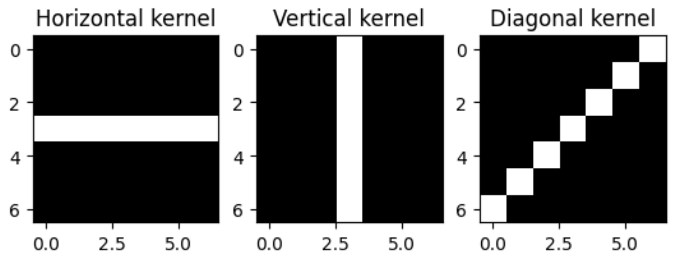

Fig. 2. Kernels used in convolution filter for the linear motion blur

- Motion blur, to simulate moving 90° (front), 45° (diagonal) captures and rotation on its own axis (YAW movement) [17] [18])
- Gaussian blur, with distortions found in camera focus and moisture on lens
- Isolation of the value channel in the HSV color system, to decrease the variability inserted by cloud shadows and water color differences found in real captures, and increase the system's immunity to color variations.

To apply the motion blur effect, the traditional vertical, horizontal and diagonal line kernel models were used, according to the 2, using the method described in Formula 1.

$$H(x,y) = \sum_{i=0}^{M_i-1} \sum_{j=0}^{M_j-1} I(x+i-a_i, y+j-a_j)K(i,j) \quad (1)$$

All kernel filters are 7x7 and the blur intensity was done uniformly, given the nature of the simulation, categorized in the dataframe by direction as follows the sample Table I.

TABLE I
Distribution of files for each height set

Pre-processing	**Percentage**	**N° of files**
Non blurred	50.00%	4500
Vertical motion		
- No gaussian blur	8.33%	750
- Gaussian blur (random radius 1 to 6)	8.33%	750
Horizontal motion		
- No gaussian blur	8.33%	750
- Gaussian blur (random radius 1 to 6)	8.33%	750
Diagonal motion		
- No gaussian blur	8.33%	750
- Gaussian blur (random radius 1 to 6)	8.33%	750
Total	100%	9000

In addition to this distortion, half of each set of heights processed by motion blur was subjected to Gaussian blur, representing possible blurring effects due to lack of focus or humidity.

For this step, the average of the colors on the surface of a real engulfment was extracted by Gaussian distribution, with calibrated images of a sea shore. Such synthetic images were manipulated with distortions common to a regular operation, to simulate the desired scenario, with the help of the OpenCV library, with techniques adjusted to the present case. The process flow diagram is depicted in Fig. 4.

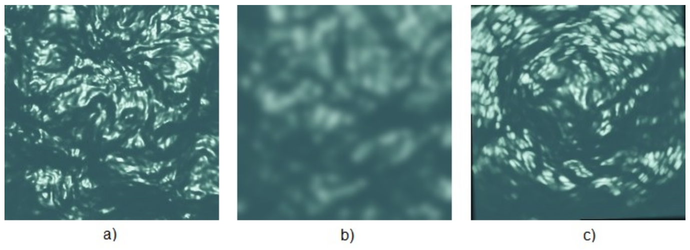

Fig. 3. Synthetic RGB images at 100cm distance. a) Standard; b) Image with Gaussian blur; c) Image with motion blur in the YAW axis

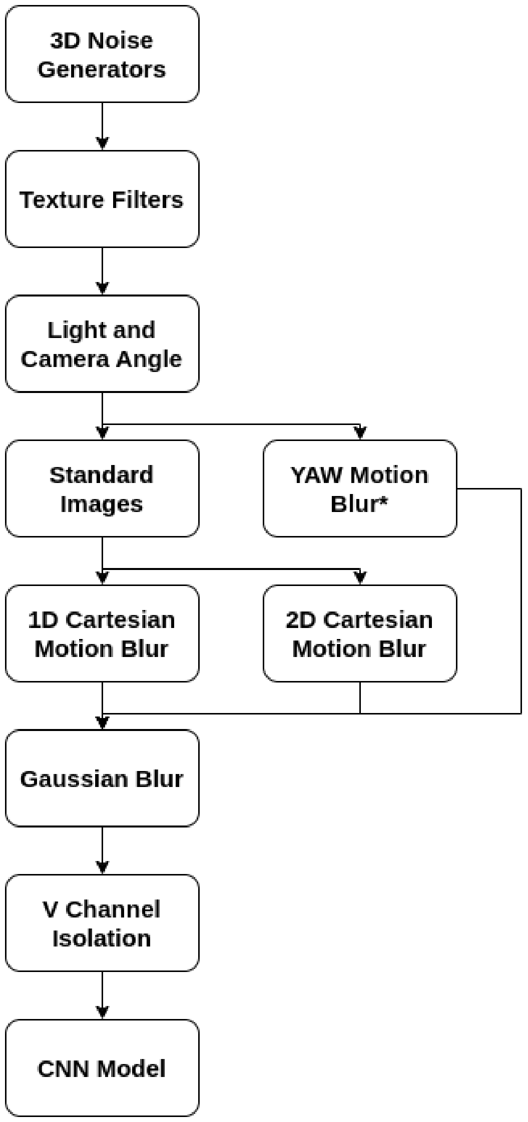

Fig. 4. Preprocessing Flowchart

In the end, 189,000 images were generated, post-processed into equal parts distributions with different deformity intensity degrees, individually documented in a database containing stepping by distance, motion blur direction, motion blur intensity and Gaussian blur intensity. This method was preferred to have control of the type of distortion at each step of the process, and to allow isolation of features for point experiments, rather than using the available and less customizable effects of the data augmentation functions in TensorFlow, and also because of the relatively large set of images, decreasing the possible 'cold posterior effects' [19]. In Fig. 5 are illustrated examples of images post processed by this script, with flow diagrammed in Fig 4.

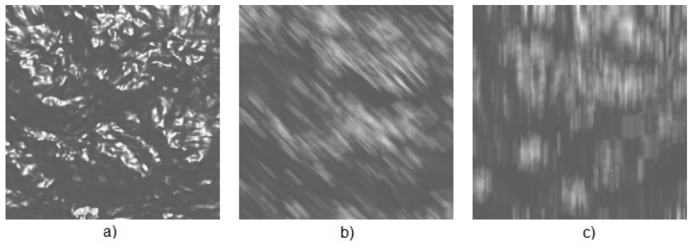

Fig. 5. Synthetic images with V channel isolation at 100cm distance. a) Standard; b) Image with motion blur in X and Y axis; c) Image with motion blur in Y axis + gaussian blur

B. Real Dataset Capturing

To capture the real images, we used a RealSense D455 camera on a 3-meter extension pole, and the videos were recorded in 680x480 pixels resolution and 30 frames per second in .avi format through Microsoft Video 1 codec. The capture sites were in the city of Cabedelo, on the shore of the Paraiba River at two points: one on a rock breakwater, and the other on the pier to Lucena and Restinga Island 6. The height at mean tide was of 3.3m, with depth markings on the foundation pillars structure.

The videos were separated frame by frame, and pre-processed the same steps as in Fig. 4 from the 'Gaussian Blur' block onwards for data augmentation purposes. After separating the images containing leaves, gross positional errors, rocks and other structures, two banks were composed. On the first bank, we have 56,531 images, containing captures at 50, 70, 100, 150, 230, 280 and 330cm.The second bank was taken at the breakwater, without any flat and elevated structure to fit the capture equipment, therefore with less freedom from proper framing of the water surface. 6,409 images at the heights of 50 and 70cm was acquired at this spot, with samples in Fig. 7.

1) Capturing Equipment: An Intel RealSense^{TM} D455 camera (Intel Corp.; Santa Clara, CA, USA) with two 720p resolution infrared sensors and one 1 MP RGB sensor was used. All sensors are of the global shutter type. The computer used in the field has a four-core Intel Core^{TM} i5-10300H processor at 2.5 GHz, and a maximum clock of 4.5 GHz. 24GB DDR4@2333 MHz RAM and NVIDIA GTX 1650 4 GB GDDR6 GPU. Images were captured in 30fps video in AVI extension and GPU-based H264 codec, without brightness compensation.

To extend the camera to the desired distances and heights, a 1" diameter aluminum rod with a length of 3 meters was made, as shown in Figure 8, and inside the rod a shielded unbuffered

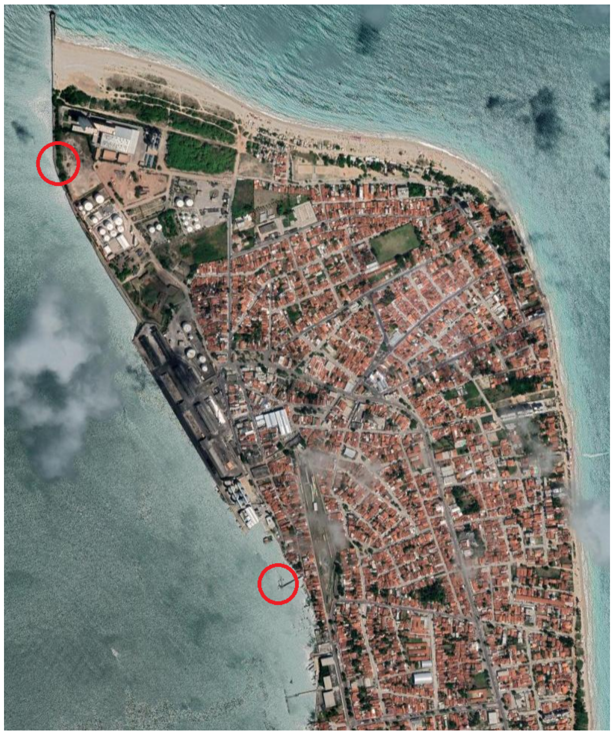

Fig. 6. Capture points marked in red in the city of Cabedelo - PB. Source: Airbus, Maxar Technologies, Google Maps, 2024

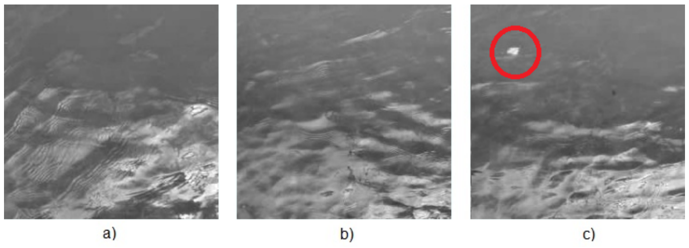

Fig. 7. Samples at the breakwater spot. a) 50cm; b) 70cm; c) 70cm with dirt and leaves

USB 3.1 extension cable was passed, limiting the bandwidth and requiring the resolution to be decreased to 680x480p.

C. CNN Model and Training

1) Training Platform and Synthetic Bench Generation: The machine used as the initial training and rendering platform for the artificial images features an AMD Ryzen^{TM} 1600 six-core processor overclocked to 3.8GHz, a NVIDIA RTX 2060 with GPU top frequency of 1695 MHz, and 6GB GDDR6 memory (NVIDIA Corp.; Santa Clara, CA, USA), dual channel DDR4@2800 MHz 2x8GB RAM (Micron Technology Inc.; Boise City, ID, USA). Ubuntu 18.04 LTS operating system with NVIDIA 530 drivers. In this configuration, the machine converged early with a batch size of 64, so the batch size was adjusted to 512 in a Google Colab instance with an Nvidia T4 GPU.

The final model, used at this stage for the trainings used batches size = 512, with maximum duration set for 100 epochs (patience = 5), with initial learning rate of 0.01 in ADAGRAD. Architecture described in Table II. The software used was running on Tensorflow 2.15.0., with the image pre-processing done with OpenCV 3.4.14 in C++ compiled by gcc-10.

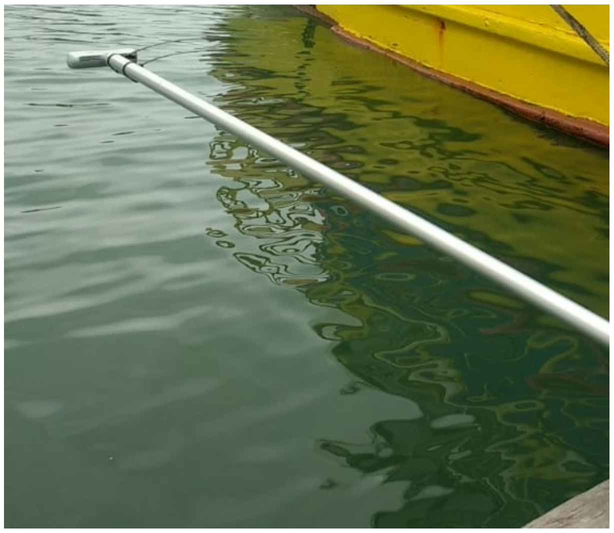

Fig. 8. Camera attached to the pole to capture images on the dikes

2) CNN Architecture: The network in scope was built based on ReLU activators in all the convolutional hidden layers, converging to dense layers at the output, according to the Table II. Using this architecture was a decision based on experience with VGG16, whose model is widely tested and used for research and commercial applications. In this particular case, two layers of color processing were removed from each convolutive pair and associated max pooling, since it only deals with images with the V channel, and not the usual RGB. In addition, the number of convolutions was reduced from 5 to 4, as there were no significant gains in prediction performance. These changes were able to reduce the size of the machine file from 168mb to 13.9mb, and total parameters from 14715201 (56.13MB) to 1208321 (4.61MB) which could mean a big leap in compatibility and runtime performance with simpler hardware. The output converges to a (None, 1) shape dense layer vector, responsible to the final linear regression.

The system outputs a linear regressor, which is responsible for returning the numerical values after processing.

3) Optimizers: Training performance was monitored with ADAM optimization algorithm, based on gradient descent [20]. The ADAM optimizer were chosen due to the ease of implementation and best performance than ADAGRAD in the last epochs.

IV. Results and Discussion

A. Synthetic Dataset

The results show promising behavior of the network with the synthetic generated base in early stages, even without normalization of the output. Later, batch normalization was applied with the approach described in [21], and the results

TABLE II
ARCHITECTURE SUMMARY

Layer(type)	Output Shape	Param #
conv2d_39 (Conv2D)	(None, 220, 220, 32)	2432
max_pooling2d_39 (MaxPooling)	(None, 110, 110, 32)	0
conv2d_40 (Conv2D)	(None, 53, 53, 64)	51264
max_pooling2d_40 (MaxPooling)	(None, 26, 26, 64)	0
conv2d_41 (Conv2D)	(None, 11, 11, 128)	204928
max_pooling2d_41 (MaxPooling)	(None, 5, 5, 128)	0
conv2d_42 (Conv2D)	(None, 1, 1, 256)	819456
max_pooling2d_42 (MaxPooling)	(None, 1, 1, 256)	0
flatten_6 (Flatten)	(None, 256)	0
dense_18 (Dense)	(None, 256)	65792
dense_19 (Dense)	(None, 256)	65792
dense_20 (Dense)	(None, 1)	257

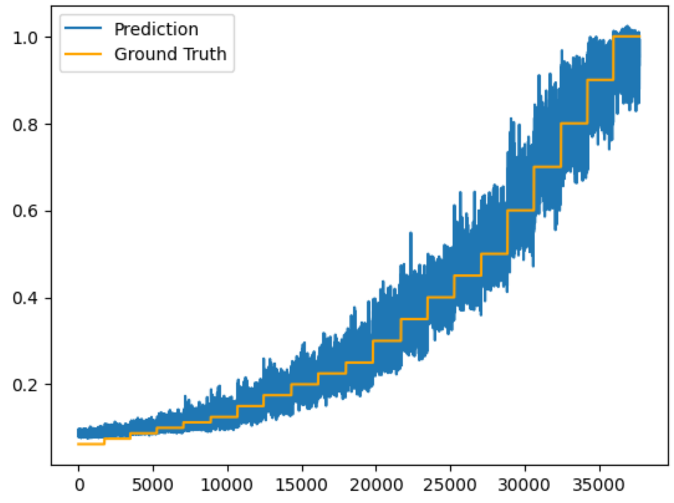

Fig. 9. Normalized result with ADAM optimizer

of training were improved, from RMSE = 0.147 to RMSE = 0.033 in the last epoch. Results shown in Figure 9

B. Real Dataset

The real images were not used to train the network at any time, and the inference result for 6000 real images can be seen in Fig. 10. For the inference test, the model with the best training performance was used, with the batch normalization and ADAM optimizer techniques. In addition, the bias value of the last dense layer (constant b of Equation 2 in dense_20 of Table II) was corrected in the prediction. This value is associated with the weight in index 0 of this layer, being b = 0.103. To find the correction constant c, the output y was manipulated in terms of bias and maximum theoretical height of training images (H_{max} = 800cm). The final formula for correction is in 3.

$$Y = f[\sum_{i=0}^{n}(X \times W) + b] \tag{2}$$

$$c = (Y_{avg} - y_{avg}) * b * H_{max} = 19.88 \tag{3}$$

The results plot are in Fig. 10, and metrics in III.

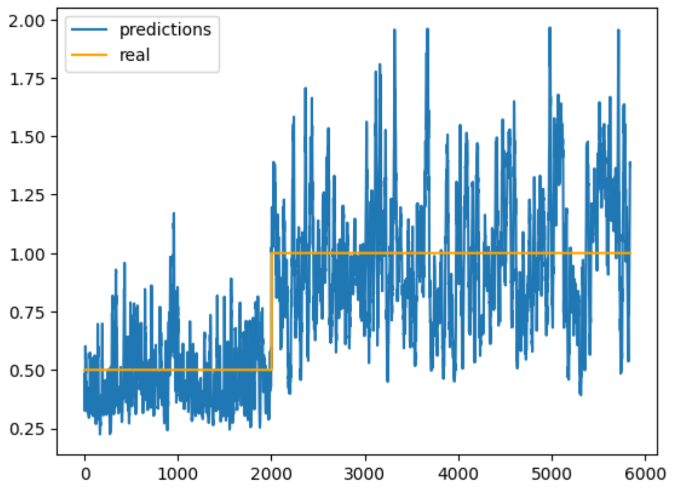

Fig. 10. Inference result with the final model after treatment

TABLE III
INFERENCE RESULTS

Metric	50cm	100cm	Total
mean	45.7cm	98.8cm	-
std	14.4cm	29.4cm	-
RMSE	151cm	294cm	254cm
MAPE	25.4%	24.0%	24.5%

V. CONCLUSION

It was possible to verify, even with training of a synthetic bank, the recognition of different heights in real images with the trained network. However, it was noticed the addition of a bias in the predictions, which increases as there is also an increase in altitude. This phenomenon may be caused by the error in the testing setup itself, and can be repaired by using a fixed pole to suspend the camera. Another cause may be the capture of foreign objects non present in the synthetic images, such as leaves, shadows, and stones.

It is still necessary to embed the solution on a real SoC or hardware platform to benchmark on a UAV, or other fixed frame carrying flight controller such as those based on Pixhawk. In this way it will be possible to recognize the possible reflections of the impact of angle deviations. Another thing to consider is to compare the performance Something to consider is the disturbance of the water surface with the airflow from the propellers of a real drone, with a promising ability to increase the discrepancy between images from different heights, also increasing the ability of the network to infer the different frames.

A future test with rolling shutter cameras should be conducted to verify the impact of these distortions.

ACKNOWLEDGMENT

The study was funded by CAPES, and conducted with cameras from the LASER laboratory, under the guidance of professors Phd. Tiago Pereira do Nascimento and Dr. Leonardo Vidal Batista.

References

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[2] X.-Z. Cui, Q. Feng, S.-Z. Wang, and J.-H. Zhang, "Monocular depth estimation with self-supervised learning for vineyard unmanned agricultural vehicle," *Sensors*, vol. 22, no. 3, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/3/721

[3] A. Joglekar, D. Joshi, R. Khemani, S. Nair, and S. Sahare, "Depth estimation using monocular camera," 2011.

[4] X. Wu, W. Li, D. Hong, R. Tao, and Q. Du, "Deep learning for unmanned aerial vehicle-based object detection and tracking: A survey," *IEEE Geoscience and Remote Sensing Magazine*, vol. 10, no. 1, pp. 91–124, 2022.

[5] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, "Cnn-based single image obstacle avoidance on a quadrotor," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 6369–6374.

[6] K. D. Julian, S. Sharma, J.-B. Jeannin, and M. J. Kochenderfer, "Verifying aircraft collision avoidance neural networks through linear approximations of safe regions," 2019. [Online]. Available: https://arxiv.org/abs/1903.00762

[7] A. Kouris and C.-S. Bouganis, "Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[8] K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, and H. Kappen, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1070–1076, 2017.

[9] X. Hua, X. Wang, T. Rui, F. Shao, and D. Wang, "Light-weight uav object tracking network based on strategy gradient and attention mechanism," *Knowledge-Based Systems*, vol. 224, p. 107071, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121003348

[10] D. Wierzbicki, "Multi-camera imaging system for uav photogrammetry," *Sensors*, vol. 18, no. 8, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2433

[11] M. Gao, C. H. Hugenholtz, T. A. Fox, M. Kucharczyk, T. E. Barchyn, and P. R. Nesbit, "Weather constraints on global drone flyability," *Scientific Reports*, vol. 11, no. 1, Jun. 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-91325-w

[12] Q. xing Zhang, G. hua Lin, Y. ming Zhang, G. Xu, and J. jun Wang, "Wildland forest fire smoke detection based on faster r-cnn using synthetic smoke images," *Procedia Engineering*, vol. 211, pp. 441–446, 2018, 2017 8th International Conference on Fire Science and Fire Protection Engineering (ICFSFPE 2017). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705817362574

[13] M. G. Prasad, S. Chandran, and M. S. Brown, "A motion blur resilient fiducial for quadcopter imaging," in *2015 IEEE Winter Conference on Applications of Computer Vision*, 2015, pp. 254–261.

[14] D. Acharya, K. Khoshelham, and S. Winter, "Bim-posenet: Indoor camera localisation using a 3d indoor model and deep learning from synthetic images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 150, pp. 245–258, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271619300589

[15] Y. Peng, Z. Tang, G. Zhao, G. Cao, and C. Wu, "Motion blur removal for uav-based wind turbine blade images using synthetic datasets," *Remote Sensing*, vol. 14, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/1/87

[16] M. G. Ferrick, "Analysis of river wave types," *Water Resources Research*, vol. 21, no. 2, pp. 209–220, 1985. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/WR021i002p00209

[17] T. Oktay, H. Celik, and I. Turkmen, "Maximizing autonomous performance of fixed-wing unmanned aerial vehicle to reduce motion blur in taken images," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 232, no. 7, pp. 857–868, Mar. 2018. [Online]. Available: https://doi.org/10.1177/0959651818765027

[18] S. Dai and Y. Wu, "Motion from blur," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[19] L. Noci, K. Roth, G. Bachmann, S. Nowozin, and T. Hofmann, "Disentangling the roles of curation, data-augmentation and the prior in the cold posterior effect," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 12 738–12 748. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/6a12d7ebc27cae44623468302c47ad74-Paper.pdf

[20] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[21] T. Ergen, A. Sahiner, B. Ozturkler, J. M. Pauly, M. Mardani, and M. Pilanci, "Demystifying batch normalization in relu networks: Equivalent convex optimization models and implicit regularization," *CoRR*, vol. abs/2103.01499, 2021. [Online]. Available: https://arxiv.org/abs/2103.01499