Underwater Gesture Recognition Using Classical Computer Vision and Deep Learning Techniques

Mygel Andrei M. Martija, Jakov Ivan S. Dumbrique, and Prospero C. Naval, Jr
Department of Computer Science, University of the Philippines, Quezon City, Philippines
Email: mmmartija@up.edu.ph

***Abstract*—Underwater Gesture Recognition is a challenging task since conditions which are normally not an issue in gesture recognition on land must be considered. Such issues include low visibility, low contrast, and unequal spectral propagation. In this work, we explore the underwater gesture recognition problem by taking on the recently released Cognitive Autonomous Diving Buddy Underwater Gestures dataset. The contributions of this paper are as follows: (1) Use traditional computer vision techniques along with classical machine learning to perform gesture recognition on the CADDY dataset; (2) Apply deep learning using a convolutional neural network to solve the same problem; (3) Perform confusion matrix analysis to determine the types of gestures that are relatively difficult to recognize and understand why; (4) Compare the performance of the methods above in terms of accuracy and inference speed. We achieve up to 97.06% accuracy with our CNN. To the best of our knowledge, our work is one of the earliest attempts, if not the first, to apply computer vision and machine learning techniques for gesture recognition on the said dataset. As such, we hope this work will serve as a benchmark for future work on the CADDY dataset.**

***Index Terms*—underwater robot vision, gesture recognition, convolutional neural networks, feature extraction**

I. Introduction

Human-Robot interaction (HRI) can be invaluable in facilitating human tasks in traditionally challenging and potentially harsh environments such as underwater. Underwater robots can help human divers in a myriad of ways such as by providing illumination, fetching tools, going to otherwise difficult-to-reach areas, and monitoring the diver for any signs of distress.

Computer vision can play a central role in this field. Gesture recognition can allow divers to communicate with their accompanying robot. However, gesture recognition is a much more daunting task in the underwater setting than on-land. The environment presents numerous technological challenges for the robot such as sudden illumination changes, unequal spectral propagation (i.e. color content is affected by depth and distance), low contrast, and changes in visibility due to turbidity [1]. Underwater gesture systems should work under different unconstrained water conditions.

Manuscript received September 21, 2019; revised February 6, 2020.

In this paper, we utilize the Cognitive Autonomous Diving Buddy (CADDY) Underwater Gestures dataset to explore the underwater gesture recognition problem. This public dataset was just released early this year. We use 3 different models and evaluate their performance, two of which are based on classical computer vision that rely on hand-engineered features, and the third is a convolutional neural network (CNN). The first non-deep learning model uses Histogram of Gradients (HOG) for feature extraction and the other one is a combination of SIFT and Bag of Visual Words (BOVW). In both models, we use Support Vector Machine (SVM) as the machine learning algorithm for classification. For the CNN, we use a classifier with a ResNet-50 [2] backbone.

We achieve the best classification accuracy with the ResNet-50 architecture at 97.06%.

II. Related Work

Gesture recognition is a problem with a wide array of applications. In [3], HOG was used together with an SVM classifier for gesture recognition in human-vehicle interaction. An automated gesture recognition system may rely on different input types; popular choices include requiring the user to use glove-based wearable devices [4], extraction of hand/skeletal keypoints [5], or simply raw visual data input. In [6], hand skeletal data is used as input. Keypoints in the form of the positions of joints in the hand are fed to a CNN. Other works that utilize deep learning to solve the gesture recognition problem include [7], [8], and [9].

In [10], underwater gesture recognition is tackled. They first addressed the diver-following problem where the underwater robot needs to track and follow the diver it is accompanying. One of their methods relies on a Hidden Markov model to analyze the diver's motion signature in the spatial domain. They then used a CNN-based model for the hand gesture recognition problem. For their CNN, they explored the Faster RCNN [11] and single-shot multi-box detector architectures [12].

III. Methodology

A. Dataset

The CADDY Underwater Gestures dataset was released by [13] in early 2019. There are a total of 16 gestures (*start, end, up, down, backward, here, mosaic, boat, carry, delimiter, photo, one, two, three, four,* and

doi: 10.18178/joig.8.1.9-14

five), all of which are illustrated in Fig. 1. Hence, we have a total of 17 labels (including *none*) for our recognition problem. In total, the dataset consists of 10,322 stereo pair images collected in 8 different underwater scenarios (i.e. sites and water conditions). These scenarios are summarized in Fig. 2. The dataset is broken down into 5,919 true positives (i.e. the diver in the image is executing a gesture) and 4,403 true negatives.

Figure 1. CADDY dataset gestures. From top left to bottom right: start, end, up, down, backward, here, mosaic, boat, carry, delimiter, photo, one, two, three, four, five.

Location	Scenario	Type	Dynamics
Biograd na Moru, Croatia	Biograd-A	Open sea	No current
	Biograd-B	Open sea	No current
	Biograd-C	Open sea	Strong currents, diver non-static
Brodarski Institute, Croatia	Brodarski-A	Indoor pool	No current
	Brodarski-B	Indoor pool	No current
	Brodarski-C	Indoor pool	No current
	Brodarski-D	Indoor pool	No current
Genova Italy	Genova-A	Outdoor pool	Diver non-static

Figure 2. CADDY dataset recording scenarios.

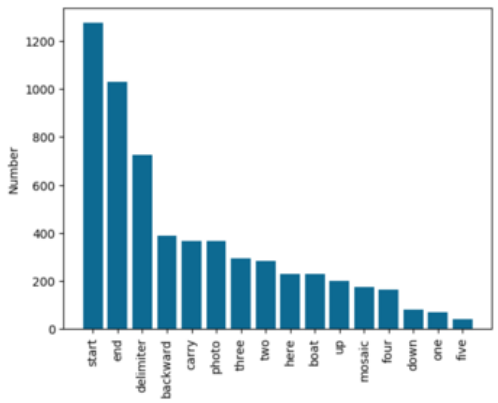


Figure 3. Histogram of CADDY gesture frequencies.

The distribution of the 16 gestures in terms of frequency is illustrated in Fig. 3. The dataset is highly imbalanced and will require data augmentation for better classifier performance. These methods will be discussed in the succeeding sections. The dataset is split into training and test sets with a 70:30 ratio.

B. Classical Computer Vision Approaches

Extracting the pertinent features from an image is an essential step in non-deep learning-based image classification. In this section, we discuss the two feature extraction and representation techniques used in this study. The codes for all the models are available at https://github.com/igygi/underwater-gesture-recognition.

1) Histogram of Gradients

Histogram of Gradients is a feature descriptor that relies on gradient orientation on localized portions of an image. The HOG framework is illustrated in Fig. 4. HOG employs a sliding window approach. For each window that we slide over the image, we obtain 8x8 cells and compute for the gradient of every pixel in the said cell. We then use these gradient values to craft a histogram for each cell. The bins of the histogram represent the gradient directions. Hence, in the unsigned case, which is what we used in our study, the histogram has angles from 0 ° to 160 ° (since 180 ° wraps back to 0 °) for its bins. Each pixel is assigned to a bin (or bins if the angle falls in between two bins) depending on its gradient direction and its contribution to the bin/s is determined by its gradient magnitude.

To mitigate lighting variations, the histograms are then normalized. But instead of normalization across each histogram, we normalize across a concatenation of histograms (e.g. 4 histograms derived from a block of 4 adjacent 8x8 cells in our example in Fig. 4). The output is our feature representation for the portion of the image covered by the block. We then do the same for the rest of the image and concatenate all the feature vectors to finally end up with the feature representation for the entire image.

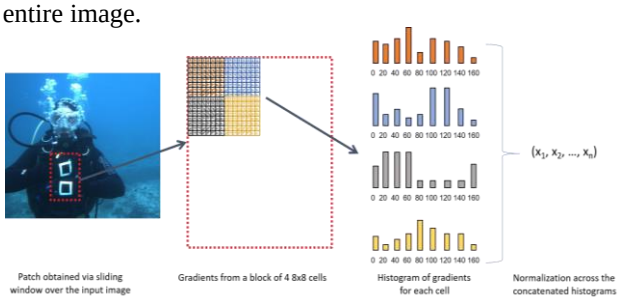


Figure 4. Histogram of gradients framework

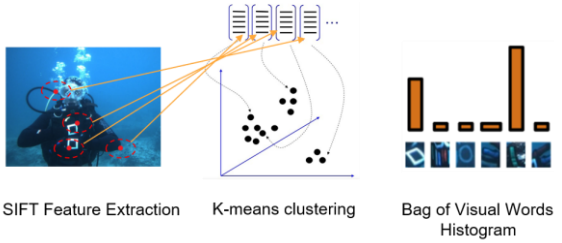


Figure 5. SIFT + Bag of visual words framework.

To train our HOG-based model, we slightly modify the algorithm used in [14] summarized as follows:

Step 1: Extract *positive and negative patches from the training set*

Step 2: Obtain *the HOG for the extracted patches*

Step 3: Train *the classifier using the HOG descriptors as input.*

Step 4: *Test the trained classifier on the negative images by running a sliding window over multiple scales of the image.*

Step 5: Perform *hard negative mining – extract difficult-to-classify patches from* Step 4.

Step 6: *Re*-train *the classifier on the initial set of positive and negative patches plus the hard examples.*

Upon inference, we obtain multiple scales for an input image and once again run a sliding window over every scale. The HOG descriptor for each patch is then fed to the classifier. Since we can potentially have conflicting predictions for a single frame (e.g. the prediction for one window is class *A* while the prediction for another window is class *B*), we take the decision on the window with the highest confidence to be our final prediction for the image. If all the windows were classified as *none*, then the frame is classified under the *none* class. We use a Linear SVM for our classifier. We decided not to perform data augmentation on the HOG training set since the input to the model during training are the region-of-interest (RoI) patches already, instead of the entire image. As will be shown later in the results, HOG performs well even without data augmentation. Appendix A lists the rest of the implementation details used in our HOG approach.

2) SIFT + Bag of visual words

Our second classical-based approach is similar to HOG in that we also construct a frequency histogram of features for an input image. This time, the histogram represents a bag of visual words (BOVW). The bins in the histogram are the centroids obtained from performing k-means clustering on the descriptors computed from the training images. Moreover, instead of HOG, we use the SIFT detector [15] as our feature extractor. This process is illustrated in Fig. 5.

Initial experiments with SIFT+BOVW on the imbalanced dataset produced poor results. We therefore perform upsampling on the minority classes simply by duplicating randomly selected samples from the said classes

C. Convolutional Neural Network

For our last model, we use a deep convolutional neural network. We want to avoid using a CNN architecture that is too deep since a larger model would mean longer training time and more importantly, a longer inference time. Such is infeasible for practical applications of underwater gesture recognition since underwater robots do not have much computing power. Hence, we use a vanilla CNN with a ResNet-50 [6] backbone. ResNet uses 'skip' or 'shortcut' connections to ease training. We attach a fully connected layer at the end of the CNN and the final layer has 17 units, one for each gesture class.

Since deep learning models require a lot of training data to achieve superior performance, we perform data augmentation on the CADDY dataset. The following techniques were employed to upsample the minority classes:

- Rotation: angle is randomly chosen between -10 ° and 10 °.
- Translation: shift is randomly chosen between -10 to 10 pixels in the vertical and horizontal directions.

We use the Adam optimizer [16] with a 0.0005 learning rate. The CNN was trained on an Nvidia RTX 2070 machine for 50 epochs.

IV. Experimental Results and Analysis

A. Recognition Accuracy

We first present and compare the performance of the SIFT+BOVW and HOG models by looking at their confusion matrices. BOVW (Fig. 6, left) finds *mosaic* and *carry* to be the easiest to classify among the gestures with 88% and 77% recall, respectively. Nine gestures (*start, end, up, down, boat, delimiter, one, two,* and *four)* meanwhile have less than 60% recall; all of them tend to be mistakenly classified as *none*. We can also see that the classifier gets confused by similar-looking gestures. For example, 15% of *start* samples get classified as *end* and vice versa. Similarly, 15% of *one* samples get mistaken for *two*. Finally, perhaps as a consequence of our BOVW model's inclination towards *none*, it is able to capture actual negative samples with ease, posting 85% accuracy for true negatives.

The HOG model performs much better in terms of recognizing gestures. It has an overall accuracy of 84.53% compared to BOVW's 64.03%. From the confusion matrix (Fig. 6, right), it is able to correctly recognize 7 gestures (*start, end, down, backward, carry, three,* and *four)* at least 90% of the time. Interestingly, unlike BOVW, HOG does not confuse *start* and *end* despite them looking very similar. *Photo* and *one* are a bit challenging to recognize as HOG confuses both of these gestures with *two.* This difficulty may be attributable to *two*'s uncanny resemblance with *photo* and *one* as shown in the upper left and upper right images of Fig. 7. Similarly, *five*'s resemblance with *four* (Fig. 7, bottom) can explain HOG's difficulty with recognizing the former. In fact, *five* is the most challenging gesture for HOG as it only posted 33% recall.

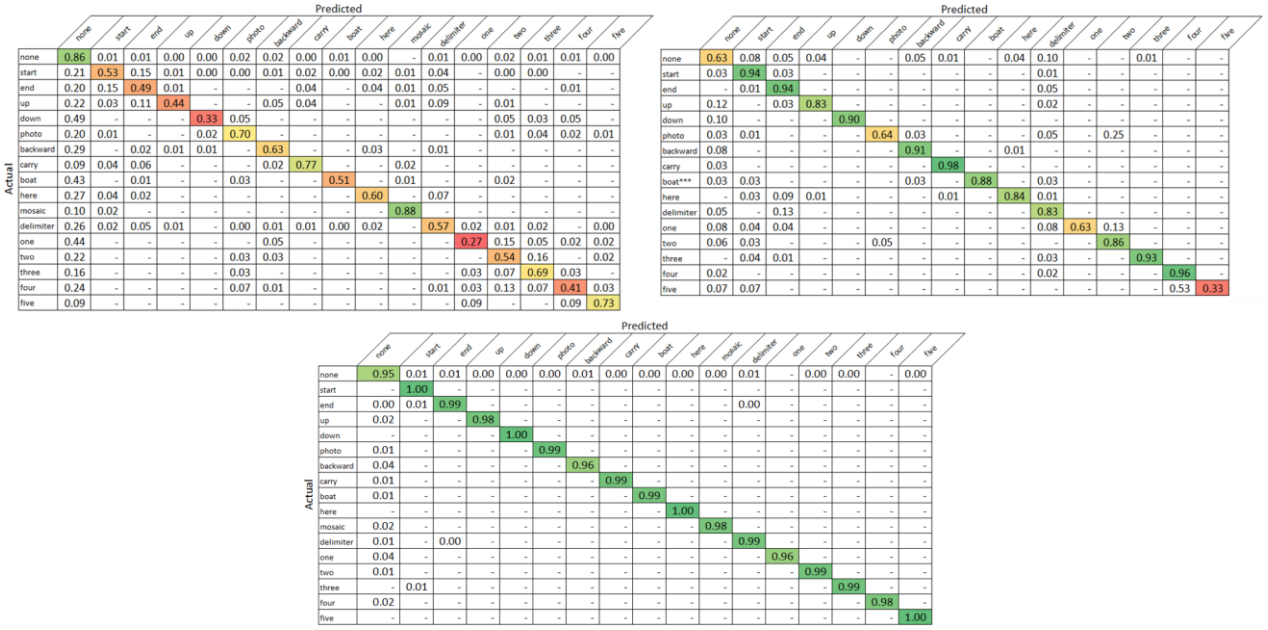


Figure 6. BOVW (upper left), HOG (upper right), and CNN (bottom) confusion matrices. Color codes indicate relative performance: green means good while red means poor.

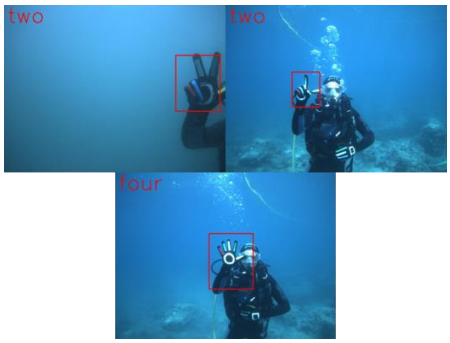


Figure 7. HOG misclassified samples. Upper left: *photo* misclassified as *two*. Upper right: *one* misclassified as *two*. Bottom: *five misclassified* as *four*.

Despite HOG's better overall performance, it is not without limitations. One challenge is choosing the right patch size for the sliding window. As indicated in the appendix, we are using patches with a portrait orientation since we observed that most gesture RoIs have a larger height than width. However, this configuration fails for *boat*, posting only 0.01% recall despite sliding over multiple image scales. Changing the patch orientation to landscape, however, results to *boat* recall rising to 88%. Another limitation for HOG is that patch-based classification is limited to the locality of the patch. Therefore, if an RoI is not a contiguous block (e.g. *mosaic*, which involves two hands raised far apart from each other), HOG may have a hard time capturing the correct label. This is why we did not include *mosaic* in our HOG experiments. Thirdly, while HOG generally outperforms BOVW, the latter is better at correctly recognizing negative samples. HOG posts only 63% accuracy for the *none* class. It seems that the locally limited patch-based classification makes it more difficult to disregard 'meaningless' hand positions. Most of the false positive cases are negative samples being classified as *delimiter*. Since *delimiter* is simply a closed fist held up against the camera (with the square white tape face facing the camera), the classifier may be classifying a patch with this appearance as *delimiter* even though the diver's hand is just resting. Lastly, HOG's reliance on the sliding window technique means that inference time would be less than ideal in real-time HRI systems. In our case, one test image takes around 30 seconds to process.

As for the results of the CNN model, the confusion matrix is shown in Fig. 6 (bottom), the per-gesture accuracy is high for all classes. All positive gestures have at least 96% accuracy. The CNN model posts an overall accuracy of 97.06%, the highest among all the 3 models we tested. Most of the errors involve *none* (i.e. positive gestures being incorrectly classified as *none* or *none* samples being incorrectly classified as a positive gesture). Nonetheless, the performance towards negative samples is still high with a 95% correct recognition on *none* frames. Lastly, our CNN model does not seem to suffer the same issue regarding confusion over similar gestures, which the SIFT+BOVW and HOG models exhibit.

B. *Inference Speed*

Despite HOG's decent performance, one of its weaknesses is its extremely slow inference due to its reliance on sliding windows. From Table I, one test image takes HOG around 30 seconds to process. BOVW is much faster at 10 frames per second (fps). For both of these models, testing was done in a machine with i7-7500 CPU @ 2.70GHz. As for our CNN model, it can perform inference at 84 fps. However, it should be noted that in testing the CNN, we used a different machine with an Nvidia RTX 2070 GPU and i5-9400k CPU @ 2.90 GHz.

TABLE I. INFERENCE TIME PER MODEL

	Model		
	HOG	BOVW	CNN-Resnet50
Frames per second (fps)	0.03 fps	10 fps	84 fps

V. Conclusion

In this paper, the authors tackled the problem of gesture recognition in the underwater environment. The recently released CADDY public dataset was utilized and both classical (Histogram of Gradients and SIFT + Bag of Visual Words) and deep learning-based computer vision techniques were employed. Out of all the models, the CNN model with a ResNet-50 backbone performed the best, achieving as high as 97.06% accuracy.

As far as the authors know, this is one of the earliest attempts, if not the first, to do gesture recognition on the CADDY dataset. We hope our results will serve as a benchmark for future work on the said dataset and other research on underwater gesture recognition. For future work, we recommend the following: 1) try other feature extraction techniques for Bag of Visual Words, 2) explore how to speed up HOG's inference time perhaps by first localizing where the person is in the image to minimize the search space for the sliding window, 3) test all three models on a computer (e.g. Raspberry Pi) that can be mounted on an underwater robot to be able to better compare their inference speeds.

Appendix A HOG Implementation Details

Parameter	Setting
Patch (window) size	64x80
Cell size	8x8
Block size	16x16
Block stride	8
Total HOG descriptor vector shape per patch	2268
Image scales for sliding window	480x640 (normal) 768x1024 (large) 300x400 (small)
SVM kernel	Linear
SVM *c*	0.1

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

M. A. Martija and J. I. Dumbrique provided equal contribution in conceptualizing the problem, conducting experiments, analyzing the results, and writing the paper. Dr. Naval supervised the project and provided invaluable insights and recommendations along the way. All authors had approved the final version.

References

[1] C. O. Ancuti, C. Ancuti, C. D. Vleeschouwer, and P. Bekaert, "Color balance and fusion for underwater image enhancement," *IEEE Trans. Image Process*, vol. 27, no. 1, pp. 379-393, Oct. 2017.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770-778.

[3] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE Trans. Intelligent Transportation Systems*, vol. 5, no. 6, pp. 2368-2377, Aug. 2014.

[4] K. S. Abhishek, C. F. Qubeley, and D. Ho, "Glove-based hand gesture recognition sign language translator using capacitive touch sensor," in *Proc. IEEE International Conf. on Electron Devices and Solid-State Circuits*, 2016, pp. 334-336.

[5] R. Wen, L. Yang, C. K. Chui, K. B. Lim, and S. Chang, "Intraoperative visual guidance and control interface for augmented reality robotic surgery," in *Proc. IEEE International Conf. on Control and Automation*, 2010, pp. 947-952.

[6] G. Devineau, W. Xi, F. Moutarde, and J. Yang, "Deep learning for hand gesture recognition on skeletal data," in *Proc. 13th IEEE International Conf. on Automatic Face & Gesture Recognition*, May 2018, pp. 106-113.

[7] Q. D. Smedt, H. Wannous, J. P. Vandeborre, J. Guerry, B. L. Saux, and D. Filliat. "3D hand gesture recognition using a depth and skeletal dataset," in *Proc. Workshop on 3D Object Retrieval*, Apr. 2017, pp. 33-38.

[8] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time and gesture detection and classification using convolutional neural networks," in *Proc. IEEE International Conf. on Automatic Face & Gesture Recognition*, 2019.

[9] G. Strezoski, D. Stojanovski, I. Dimitrovski, and G. Madjarov, "Hand gesture recognition using deep convolutional neural networks," in *Proc. International Conf. on ICT Innovations*, 2016, pp. 49-58.

[10] M. J. Islam, M. Ho, and J. Sattar, "Understanding human motion and gestures for underwater human-robot collaboration," *Journal of Field Robotics*, Apr. 2018.

[11] S. Ren, K. He, R. Girschick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, June 2017.

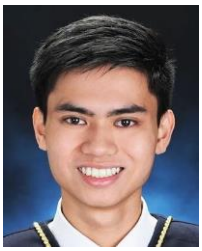
[12] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. on Neural Networks*, vol. 4, pp. 570-578, July 1993.

[13] E. Zereik, A. Babic, A. G. Chavez, A. Ranieri, and A. Birk, "Caddy underwater stereo-vision dataset for human-robot interaction (HRI) in the context of diver activities," *Journal of Marine Science and Engineering*, vol. 7, no. 1, p. 16, Jan. 2019.

[14] N. Dalal and B. Triggs, "Histogram of oriented gradients for human detection," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[15] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd International Conf. on Learning Representations*, Dec. 2014.



Mygel Andrei M. Martija received a B.S. Management Engineering degree in 2016 from the Ateneo de Manila University, Quezon City, Philippines. He is currently pursuing his M.S. degree in Computer Science at the University of the Philippines, Quezon City, Philippines. He is also serving as a research assistant at the Computer Vision and Machine Intelligence Laboratory in the Department of Computer Science in the same university. His research interests include underwater computer vision, multi-agent reinforcement learning, and deep learning.

Jakov Ivan S. Dumbrique is currently pursuing his M.S. degree in Computer Science at University of the Philippines-Diliman. He earned his B.S. and Master degree in Applied Mathematics with specialization in Mathematical Finance from Ateneo de Manila University in 2016 and 2017, respectively. He is currently an instructor at the Mathematics Department of Ateneo de Manila University. His research interests include deep learning, computer vision, medical image processing, and financial machine learning.

Prospero C. Naval, Jr. holds a Ph.D. in Electrical Engineering from the University of the Philippines. He acquired his B.S. and M.S. degrees in Electrical Engineering from the same university. He also holds an M.Eng. in Information Science from Kyoto University, Japan.

He is currently a professor at the Department of Computer Science of the University of the Philippines, where he heads the Computer Vision and Machine Intelligence Lab.

His research interests include underwater computer vision, data analytics for healthcare, environment, and education, deep reinforcement learning, deep learning, probabilistic machine learning, and swarm robotics.