Deep Learning Waterline Detection for Low-cost Autonomous Boats

Lorenzo Steccanella, Domenico Bloisi, Jason Blum, and Alessandro Farinelli

Department of Computer Science, University of Verona,
Strada le Grazie 15 - 37134 Verona, Italy
alessandro.farinelli@univr.it

Abstract. Waterline detection in images captured from a moving camera mounted on an autonomous boat is a complex task, due the presence of reflections, illumination changes, camera jitter, and waves. The pose of the boat and the presence of obstacles in front of it can be inferred by extracting the waterline. In this work, we present a supervised method for waterline detection, which can be used for low-cost autonomous boats. The method is based on a Fully Convolutional Neural Network for obtaining a pixel-wise image segmentation. Experiments have been carried out on a publicly available data set of images and videos, containing data coming from a challenging scenario where multiple floating obstacles are present (buoys, sailing and motor boats). Quantitative results show the effectiveness of the proposed approach, with 0.97 accuracy at a speed of 9 frames per second.

Keywords: robotic boats, autonomous navigation, deep learning, robot vision

1 Introduction

The water quality monitoring process costs every year more than 1 billion EUR at the European Union level. In particular, investigation looking for pollution in large rivers or lakes supplying drinking water are in the range of 150,000 to 400,000 EUR. The current approach, based on field sample and laboratory analysis, is unable to assess temporal and spatial variation in the contaminants of concern. This means that, in case of an accident, there is the risk of taking inadequate and late decisions for mitigating the pollution impacts.

The use of autonomous surface vehicles (ASVs) for persistent large-scale monitoring of aquatic environments is a valid and efficient alternative to the traditional manual sampling approach [1]. ASVs are capable of undertaking long-endurance missions and carrying multiple sensors (e.g., for measuring electrical conductivity and dissolved oxygen) to obtain water quality indicators [2].

There exist commercial ASVs specifically developed for water quality monitoring. An example is the Lutra mono hull boat produced by Platypus[1], which

[1] senseplatypus.com

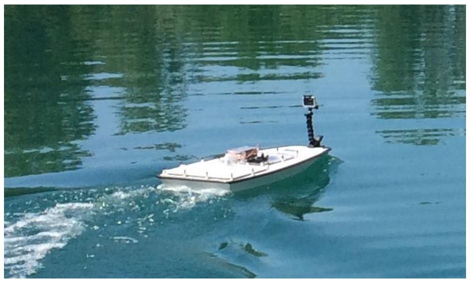

Fig. 1. IntCatch2020 project uses Platypus Lutra boats, about $1m$ long and $0.5m$ wide. A camera has been mounted on the front of the ASV.

can mount a submerged propeller (see Fig. 1) or an air fan for propulsion. Lutra boats are used in the EU-funded project IntCatch2020[2], which will develop efficient and user-friendly monitoring strategies for facilitating sustainable water quality management by community groups and non-governmental organizations (NGOs).

To achieve true autonomous navigation, an ASV must sense its environment and localize itself within that environment. We seek to develop vision-based sensing as a first step, focusing on the domain of small, low-cost ASVs. The low-cost design goals of the IntCatch boat preclude the use of sensors commonly utilized for these tasks, such as Lidar. Avoiding water-bourne obstacles can begin by segmenting an image into water and non-water pixels. From there, we can attempt to derive a relationship between the location of non-water pixels in the image and distance from the camera to those objects. Such a relationship would be heavily dependent on the pitch and roll angles of the ASV, and the low-cost gyroscope and accelerometer available on the IntCatch boat are not very precise. To address this, a more precise measurement of the pitch and roll angles of the ASV could be determined by tracking the horizon line. With limited on-board processing power, we seek methods that can address both the pixel-wise segmentation of the image to first identify obstacles, and track the horizon line to increase the precision of pitch and roll estimates.

The dynamic nature of water is capable of producing mirror-accurate reflections of the environment or a turblent, erratically textured surface. Recent advancements in deep learning methods for computer vision, particularly Convolutional Neural Networks (CNNs) [3], show promise for segmenting images captured by an ASV into obstacles and the water that surrounds them. While CNNs have been previously introduced into other mobile robotics domains (e.g., indoor wheeled and quadrotor), and ASV obstacle avoidance has been attempted with classical computer vision methods such as optical flow, the two have yet to be combined.

In this paper, we present a pixel-wise deep learning method for segmenting images captured by a camera mounted on a small ASV and extracting the hori-

[2] `www.intcatch.eu`

zon line, which we refer to as the *waterline* in this context. Another important contribution is the creation of a publicly available data set of images and videos, called IntCatch Vision Data Set[3]. This data set contains annotated data that can be used for developing supervised approaches for object detection.

The remainder of the paper is structured as follows. Related work is discussed in Section 2. The proposed method is presented in Section 3. Experimental evaluation is shown in Section 4. Finally, conclusions are drawn in Section 5.

2 Related Work

Monocular vision-based obstacle detection for low-cost Autonomous Surface Vehicles (ASVs) has previously received some interest. El-Gaaly et al. [4] utilize sparse optical flow, reflection rejection, and an occupancy grid overlaid on the image. This process produced a significant number of false positives when the water surface was disturbed. Sadhu et al. [5] used grayscale histograms of pixel neighborhoods as a descriptor of texture and saliency to detect logs floating on the surface of water. However, this method ignores the shoreline.

The inherent complexity of the water scenario makes unsupervised approaches, like the above ones, effective only in a limited set of situations. In this work, we apply a supervised approach, to obtain a mapping between image pixels and the two classes *water/not-water*. In particular, we use Convolutional Neural Networks (CNNs), a type of deep, feed-forward Artificial Neural Networks (ANNs). Since ANNs are able to approximate any continuous functional mapping, they can be employed when the form of the required function is unknown [6].

CNNs have shown impressive performance on image classification [7] and object detection [8]. Recognition approaches use machine learning methods based on pre-trained models to address the various categories of objects present in general scenes. Segnet [9] is an example of deep fully convolutional neural network architecture for semantic *pixel-wise* segmentation. Its segmentation engine relies on an encoding-decoding scheme and it can be employed for scene understanding applications.

Multiple mobile robotics domains have started to incorporate CNNs into obstacle detection and navigation. Giusti et al. [10] detect forest paths for a quadrotor to navigate. Chakravarty et al. [11] trained a CNN to produce depth maps from a single image in indoor enviroments. To the best of our knowledge, CNNs have yet to be implemented for ASV applications.

U-net [12] is a encoder-decoder type of network for pixel-wise predictions. In U-net, the receptive fields after convolution are concatenated with the receptive fields in a up-convolution process, allowing the network to use original features in addition to features after transpose convolution. Every step in the expansive path consists of *i)* an upsampling of the feature map followed by a 2×2 transpose convolution that halves the number of feature channels, *ii)* a concatenation with the correspondingly cropped feature map from the contracting path, and *iii)*

[3] `goo.gl/Kxt6HP`

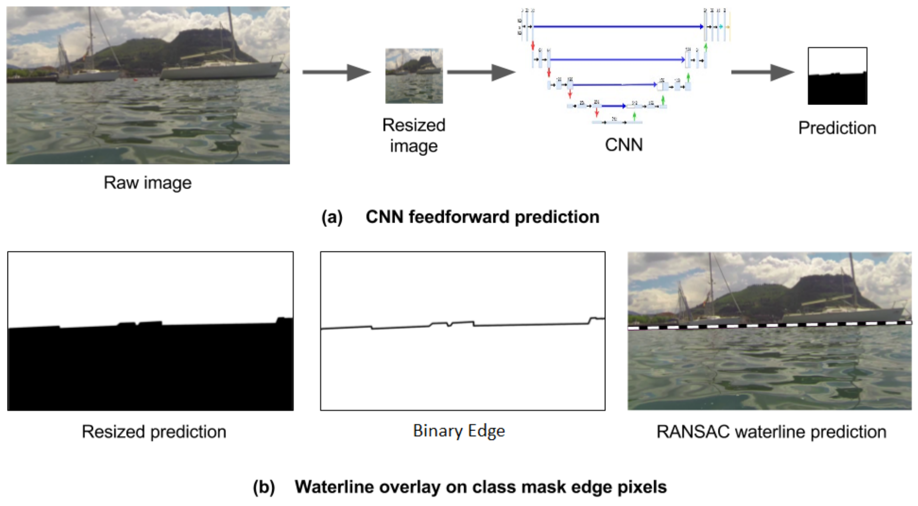
Raw image
Resized image
CNN
Prediction
(a) CNN feedforward prediction
Resized prediction
Binary Edge
RANSAC waterline prediction
(b) Waterline overlay on class mask edge pixels

Fig. 2. Waterline detection algorithm, idealized example. (a) Raw captured image is fed to trained neural network, producing the class mask. (b) Pixels on the border between classes are isolated by detecting the edges in the mask, and the waterline prediction is created with linear regression over the edge pixels.

two 3×3 convolutions, each followed by a ReLU. This results in overall better performance than a network that has access to only features after up-convolution. U-net is the starting point for building our CNN architecture.

3 Methods

We use a Fully Convolutional Neural Network [13] to segment images captured by an ASV, classifying pixels as water or non-water. After this, we process the segmentation mask to detect the waterline. Fig. 2 shows the overview of our approach.

3.1 Network Architecture

A U-net based architecture is used for the segmentation process [12]. The decision of choosing the U-net architecture has been taken for three main reasons:

1. U-net does not have any dense layer. This means that there is no restriction on the size of the input image.
2. The training stage can be carried out even with a limited amount of training data.
3. The receptive fields in encoding and decoding are concatenated. This allows the network to consider the feature after upconvolution together with the original ones.

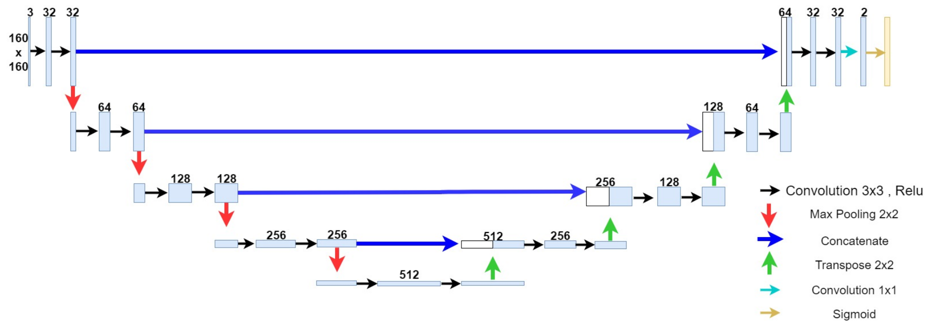

Fig. 3. Architecture for Full 160×160 network.

The architecture of the net is shown in Fig. 3. The input image is downsampled to obtain a 160×160 resized image. The encoding stage is needed to create a 512 feature vector, while the decoding stage is needed to obtain the predicted mask at 160×160 pixels. The encoding stage is made of ten 3×3 convolutional layers, and by four 2×2 max pooling operations with stride 2. In particular, there is a repeated application of two unpadded convolutions, each followed by a rectified linear unit (ReLU) and a max pooling operation. The expansive path (see the right side of Fig. 3) is made of eight 3×3 convolutional layers and by four 2×2 transpose layers. There is a repeated application of two unpadded convolutions, each followed by a ReLU and a transpose operation.

Increasing the speed of the net. Our eventual goal is autonomous obstacle avoidance for a low-cost ASV. When avoiding collisions while moving at full speed, processing time is at a premium and processing speed is critical. This motivates the pursuit of scalable methods that can be deployed with limited processing power. With the above described network architecture (denoted as *Full 160×160*), we are able to obtain a computational speed of about 4.5 frames per second (see next Section 4). We created three additional networks to explore the trade-off between performance and processing speed: Full 80×80, Half-Conv 160×160, and Half-Conv 80×80.

Full 80×80 is a network with the same architecture of Full 160×160 with an 80×80 reduced image in input. Half-Conv 160×160 and Half-Conv 80×80 are two reduced networks, meaning that they present an architecture where the convolution filter size is reduced by a factor of two. By reducing the input and convolutional filter size, it is possible to obtain a faster computation in terms of frames per second. Quantitative experimental results are shown in Section 4.

3.2 Data Set

To train the net we have used 191 labeled images coming from the IntCatch Vision Data Set[4], which has been created to store visual and sensor data collected

[4] `goo.gl/Kxt6HP`

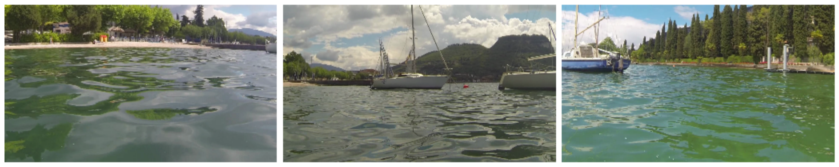
Fig. 4. Three frames captured at Lake Garda in Italy, which is one of the sites of the IntCatch2020 project.

during the IntCatch2020 project. At the moment, the data set contains 22 low and high resolution videos captured at different sites. The IntCatch Vision Data Set will be extended and updated until the end of the project, namely January 2020.

In this work, we use 8 videos coming from Lake Garda in Italy. Fig. 4 shows three frames captured at different times of the day.

Training has been performed on 191 labeled images and validated for early stopping on 40 labeled images taken from the first 6 videos namely:

- sequence lakegarda-may-9-prop-1
- sequence lakegarda-may-9-prop-2
- sequence lakegarda-may-9-prop-3
- sequence lakegarda-may-9-prop-4
- sequence lakegarda-may-9-prop-5
- sequence lakegarda-may-9-prop-6

While the experimental results have been obtained by considering 80 labeled images taken from:

- sequence lakegarda-may-9-prop-7
- sequence lakegarda-may-9-prop-8

Annotation. In order to perform supervised training of our network trough gradient descent and to evaluate the results, we annotated a total of 311 images by hand. The annotation has been performed with a custom tool created for the task. The tool takes advantage of super-pixel segmentation to give hints to the user that has to select the segments belonging to the water class. To perform super-pixel segmentation we used the SLIC algorithm [14]. This algorithm performs K-means in the 5d space of color information and image location. The mask created at this stage has been further redefined by a brush drag and drop with the mouse.

Data augmentation. Working with a data set of limited size presents a problem related to overfitting: Models trained with a small amount of data can have a limited generalization capacity. Data augmentation has become a usual practice to handle training on small data sets [15]. In this work, we have performed augmentation on the Lake Garda data to create a larger training data set. Fig. 5 shows some results of the data augmentation process.

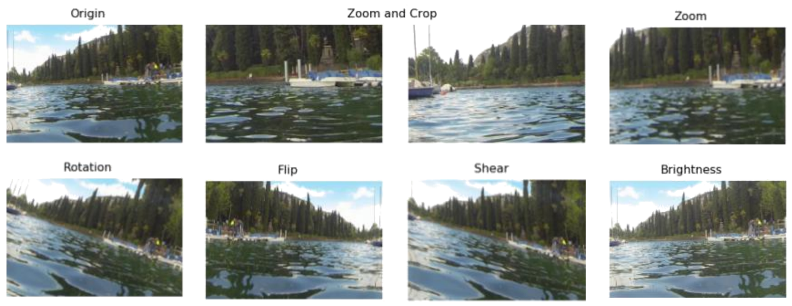

Fig. 5. Data augmentation.

Annotated ground truth masks were flipped horizontally, transformed through a random rotation within -20 to 20 degrees, sheared, zoomed, improved with a random brightness filter and patches from the full size original masks have been extracted producing 7 times the number of the original training samples. The dynamics of water frequently cause a small ASV to rotate (roll) significantly. While natural rolling motion is captured in the raw video footage, this effect is critical for performance, so we augment the data set with rotations as suggested in [12]. The masks obtained by flipping and rotating have been scaled down to 160×160 or 80×80 pixels, according to the input size of the CNN.

3.3 Training

The training step has been performed taking advantage of mini-batch gradient descent. In fact, even with our limited size data set, training over all the images (i.e., Vanilla Gradient Descent) was impossible on a commercial laptop.

Mini-batch gradient descent performs an update for every mini-batch of n training examples:

$$\theta = \theta - \alpha \bigtriangledown J(\theta; x^{(z:z+bs)}; y^{(z:z+bs)}) \tag{1}$$

where θ are the weights, α is the learning rate, bs is the mini-batch size and J the cost function, computed as:

$$J(\theta, b, x^{(z:z+bs)}, y^{(z:z+bs)}) = \frac{1}{bs} \sum_{z=0}^{bs} J(\theta, b, x^{(z)}, y^{(z)}) \tag{2}$$

We use an Adam optimizer [16] with a bs of 30 images. Adam Optimizer performs gradient descent with momentum, involving a weighted average. The hyper parameters chosen for training our net are: a learning rate of 0.001, a B_1 value of 0.9 and a B_2 value of 0.999. In order to prevent overfitting, the nets have been trained over 20 epochs performing early stopping by monitoring the loss over the validation set [17].

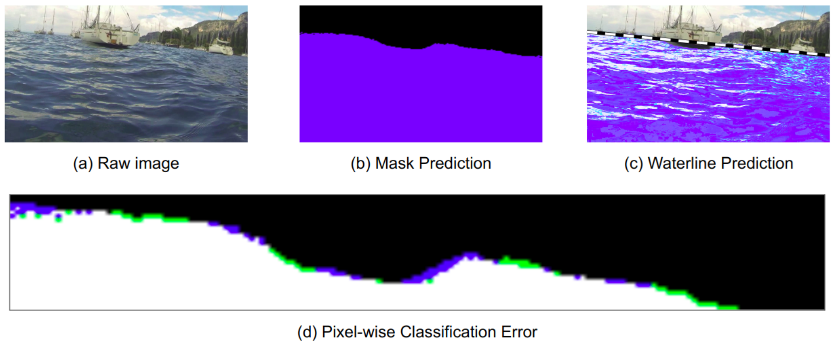

Fig. 6. Example classification and waterline result. In (d), black is true negative, white is true positive, green is false positive, and blue is false negative. The image is zoomed to the region around the waterline.

As a difference with respect to the U-net formulation given in [12], where a cross entropy loss function is used, we employ the Dice Similarity Coefficient (DSC) as loss function, which is defined as follows [18].

$$J = DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{3}$$

+where TP, TN, FP, and FN are the number of pixel-wise true positives, true negatives, false positives, and false negatives respectively. Using the DSC loss function allows to mitigate the class imbalance problem, i.e., pixel belonging to water class are more than other pixels.

3.4 Waterline Detection Algorithm

Fig. 2b shows an idealized example of the waterline detection algorithm. A raw test image is resized and fed through the CNN, producing a segmentation mask. The segmentation ideally follows the contour of the boundary between the water and any obstacles and the horizon line. To effectively extract the horizon line, which we refer to as the waterline in this context, we use post-processing methods that reduce the influence of sharp changes in the contour due to obstacles.

First a median filter is applied to the segmentation mask, reducing narrow protrusions (such as sailboat masts). Binary edge detection is then used to isolate the pixels on the contour between the two classes. The probabilistic Hough transform with line primitive is applied to filter out pixels that may be part shorter, closed contours that appear below the horizon line (a small buoy below the horizon line, for example). Only the pixels that compose a long edge are retained. Finally, linear regression is performed, utilizing RANSAC [19] to further reduce the influence of sharp changes in the contour. Fig. 6 shows an example result of the waterline detection algorithm.

Table 1. Segmentation Results

Network	Precision	Recall	Accuracy	F_1 Score	Fps
Half-Conv 80x80	0.932	0.982	0.954	0.956	16
Half-Conv 160x160	0.964	0.991	0.977	0.978	9
Full 80x80	0.972	0.987	0.979	0.979	10
Full 160x160	**0.984**	**0.993**	**0.988**	**0.988**	**4.5**

4 Experimental Results

Lake Garda represents a challenging scenario for a low-cost ASV due to 1) the presence of large waves (compared to the dimension of the ASV) and 2) a high number of floating obstacles (boats and buoys) on the water surface.

The four networks that we developed have been written in Python, using functions included in the libraries OpenCV, TensorFlow, and Keras. The complete source code is available as part of the IntCatch AI library downloadable at `goo.gl/KBSoQD`.

Quantitative evaluation. All the experiments have been carried out using a notebook with the following specifications: Intel Core i5-6300HQ CPU, 8 GB RAM, and a NVIDIA GeForce GTX 960M w/2GB GDDR5 GPU. Training time varies according to the complexity of the network and number of features extracted, it takes from a minimum of 4 hours to a maximum of 6 hours on our notebook.

We evaluate the performance of the water pixel-wise classification with multiple metrics: Precision (P), Recall (R), Accuracy (A), and F1-score (F_1) [20]. Using TP, TN, FP, and FN as the number of pixel-wise true positives, true negatives, false positives, and false negatives respectively, they are defined as

$$P = \frac{TP}{TP + FP} \tag{4}$$

$$R = \frac{TP}{TP + FN} \tag{5}$$

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

$$F_1 = 2\frac{P \times R}{P + R} \tag{7}$$

Table 1 shows the segmentation results for the four different network configurations tested. The video sequence used for computing the quality metrics for water segmentation and waterline extraction can be downloaded at `goo.gl/FHgwkV`.

Waterline error and speedup. A pixel-wise evaluation of classification error does not translate directly into an error in the waterline. To evaluate this, we calculated the maximum vertical distance in pixels between the ground truth

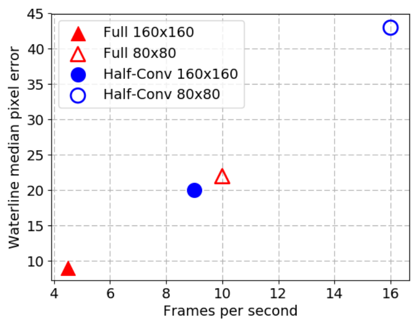

Fig. 7. Waterline median pixel error vs. frames per second.

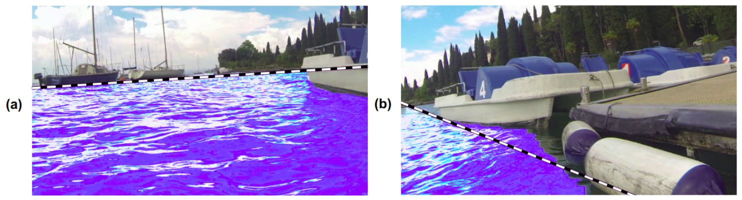

Fig. 8. Two examples demonstrating the limitations of a water line. (a) The contour of a boat begins to appear and is classified correctly. RANSAC line sticks to dominant horizon line. (b) Waterline construct breaks down completely, motivating the use of a water contour.

waterline (a RANSAC fit over the ground truth mask) and the predicted waterline for each original size test image. The median of this distance for the entire test set is plotted against the number of frames processed through the algorithm per second, shown in Fig. 7.

Discussion. We used the median pixel error between the ground truth waterline and the predicted waterline to provide a more geometric measure of error. This error scales in a roughly linear fashion with the frames per second processed by the algorithm (see Fig. 7). This demonstrates the scalability of the method. While this does not translate directly into a measure of obstacle avoidance performance, we achieve processing speeds that would be acceptable for a small, low-speed ASV. When the boundary between water and non-water pixels is dominated by the horizon line, assuming the boundary between water and obstacles is linear is reasonable. But as the distance to obstacles decreases, this assumption begins to break down. Fig. 8 shows two examples of this. At a certain point, a water boundary ”contour” becomes more useful. Deep learning methods, especially CNNs, could be used to learn to identify this contour directly from raw pixel data, circumventing the need for edge detection or other similar post-processing.

5 Conclusions

In this paper, we have shown the use of a deep learning based method for waterline detection on a low-cost ASV. Images captured by the ASV were segmented pixel-wise into water and not-water classes using a CNN. A line was fit to the edges in this binary class mask to create the waterline prediction. This can be used to infer the pose of the ASV and detect obstacles in front of the boat. The waterline extraction method has been tested on different sequences captured at Lake Garda in Italy. The chosen application scenario is challenging, due to the presence of large waves (compared to the dimension of our ASV) and a number of floating objects (buoys, sailing and motor boats).

We have demonstrated the effectiveness of the method with two different network architectures and two input layer sizes. The pixel-wise classification achieves good performance in all four cases, with accuracy and F_1 score ranging from 0.954 to 0.988. Reducing the input size and filter size of the CNN results in significant speedup without a significant reduction in the segmentation accuracy.

Future directions. This work represents a first proof of the feasibility of a deep learning approach to horizon line detection and water segmentation on images coming from a small ASV. It paves the way for investigating the use of different types of network architectures. A Recurrent Neural Network over the latent space on our network can be used to track the feature over time, and this could provide improvements in accuracy and smoothness of transitions between frame detection. The network itself could provide directly the horizon line attaching dense layers over the encoded latent space, and having a multiple output network.

The water detection scheme proposed in this paper will be used as starting point to develop an obstacle avoidance module on an embedded board mounted on the IntCatch boat. Moreover, we intend to expand the training input and experimental results to other operational environments within the IntCatch2020 project.

Acknowledgment

This work is partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 689341+.

References

1. M. Dunbabin and A. Grinham, "Quantifying spatiotemporal greenhouse gas emissions using autonomous surface vehicles," *Journal of Field Robotics*, vol. 34, no. 1, pp. 151–169, 2017.
2. D. L. Codiga, "A marine autonomous surface craft for long-duration, spatially explicit, multidisciplinary water column sampling in coastal and estuarine systems," *Journal of Atmospheric and Oceanic Technology*, vol. 32, no. 3, pp. 627–641, 2015.
3. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

4. T. El-Gaaly, C. Tomaszewski, A. Valada, P. Velagapudi, B. Kannan, and P. Scerri, "Visual obstacle avoidance for autonomous watercraft using smartphones," in *Autonomous Robots and Multirobot Systems workshop*, 2013.
5. T. Sadhu, A. B. Albu, M. Hoeberechts, E. Wisernig, and B. Wyvill, "Obstacle detection for image-guided surface water navigation," in *2016 13th Conference on Computer and Robot Vision*, 2016.
6. A. Castellini and V. Manca, "Learning regulation functions of metabolic systems by artificial neural networks," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference*, 2009, pp. 193–200.
7. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.
8. D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *CVPR*, 2014, pp. 2155–2162.
9. V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
10. A. Giusti, J. Guzzi, D. C. Cirean, F. L. He, J. P. Rodrguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, 2016.
11. P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. V. Eycken, "Cnn-based single image obstacle avoidance on a quadrotor," in *2017 IEEE Int. Conf. on Robotics and Automation*, 2017.
12. O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015, pp. 234–241.
13. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.
14. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.
15. L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.
16. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
17. R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Advances in neural information processing systems*, 2001, pp. 402–408.
18. T. L. Kline, P. Korfiatis, M. E. Edwards, J. D. Blais, F. S. Czerwiec, P. C. Harris, B. F. King, V. E. Torres, and B. J. Erickson, "Performance of an artificial multi-observer deep neural network for fully automated segmentation of polycystic kidneys," *Journal of Digital Imaging*, vol. 30, no. 4, pp. 442–448, 2017.
19. M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, 1981.
20. A. Pennisi, D. D. Bloisi, D. Nardi, A. R. Giampetruzzi, C. Mondino, and A. Facchiano, "Skin lesion image segmentation using delaunay triangulation for melanoma detection," *Computerized Medical Imaging and Graphics*, vol. 52, pp. 89 – 103, 2016.