Vision-based Navigation Solution for Autonomous Underwater Vehicles

Tiago Alves
Institute for Systems and Robotics
Instituto Superior Técnico
Av. Rovisco Pais, 1
1049-001 Lisboa, PORTUGAL

Tiago Hormigo
Spin.Works
Avenida da Igreja 42 6º
1700-239 Lisboa, PORTUGAL

Rodrigo Ventura
Instituto Superior Técnico
Av. Rovisco Pais, 1
1049-001 Lisboa, PORTUGAL

***Abstract*—Vehicle navigation is fully autonomous when the system is capable of planing its path and execute it without human intervention. This research aims at introducing an AI-based approach for visual navigation in underwater environments. To achieve this, several challenges have to be overtaken, such as segmenting the images to filter the floating clutter typical in underwater environments. First, an annotated dataset with pairs of input images and segmentation grounds truths is essential for training a state-of-the-art AI model. Second, choosing a model adequate for image segmentation and training it. Finally, evaluate if this methodology improves the accuracy of visual navigation and scene reconstruction algorithms, such as online and offline SLAM. This approach achieved state-of-the-art results on the segmentation task, with 93% pixel accuracy and 85% IoU. At last, it was concluded that using the segmentation masks produced by the fully convolutional network improves the results of using offline and online SLAM algorithms.**

***Index Terms*—Segmentation, autonomous underwater vehicles, SLAM, Neural Networks**

I. Introduction

Autonomous underwater vehicles (AUVs) can revolutionize deep sea exploration, by changing the way data is acquired for further mapping and monitoring. Space exploration is another field that could benefit from AUVs, since one of the goals of several international organizations is to explore parts of our solar system potentially capable of hosting life, as is the case of ocean worlds, such as Enceladus, Europa or Titan. These ocean worlds could have conditions similar to those on the deep parts of our oceans, thus developing vehicles capable of operating autonomously in our seas could be a first step.

Developing a visual based navigation solution for underwater vehicles operating in deep ocean comes with several challenges, such as addresing the presence of floating particles around the environment, the low contrast typical of deep ocean cites and the appearance of vehicle parts in the image. Since the feature detection of this solution is based in deep learning, which requires annotated datasets for training, finding or developing a good dataset was the first task. The second challenge would be training a neural network suited for segmentation, in order tohave a robust feature detection tool. At last, assess if the proposed methodology helps improve the performance of SLAM algorithms.

To tackle the first problem, a dataset with pairs of input images and labeled segmentation ground truth needs to be created.

To address the segmentation problem, we used state of the art technology, robust enough to endure the difficult navigation conditions of a deep underwater environment. The approach chosen was using neural networks, as is the case of a Fully Convolutional Network, which learned how to segment between foreground and background.

To assess if the developed network contributes to SLAM algorithms, the outputs were tested, coupled with the input images, on both offline and online SLAM algorithms. Offline SLAM is used in post-exploration situations to reconstruct and map the visited sites. Online SLAM is applied for real-time autonomous navigation. Thus, evaluating on both tools is crucial in the scope of deep sea - and future ocean planets - exploration.

The main contributions of this work are:

- Development of an annotated dataset of deep underwater environments rich in hydrothermal vents and sea bed footage;
- Training and testing a state of the art AI-based semantic segmentation algorithm with the created dataset;
- Evaluate the contribution of introducing AI in feature detection to improve the performance of navigation and reconstruction algorithms, such as SLAM.

II. Background

Semantic segmentation is a very important technique in the field of computer vision, being used for problems like object detection and classification, development of self driving vehicles or virtual reality [1]. In the context of this project, semantic segmentation is the approach that makes the most sense, since we want to split images in regions containing obstacles, like sea bed, and background, such as areas containing only water.

This work was supported by the LARSyS - FCT Project UIDB/50009/2020

978-1-6654-8217-2/22/$31.00 ©2022 IEEE

A. Fully Convolutional Networks

Fully convolutional networks [2] are neural network architectures widely used for semantic segmentation tasks.

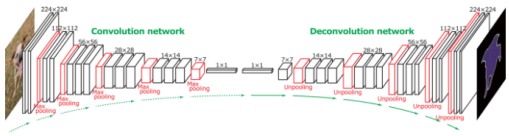


Fig. 1: Fully convolutional network. Source: [3]

Today, a good amount of the best performing state-of-the-art semantic segmentation methods relly on fully convolutional networks. The FCNs are able to yield dense pixelwise predictions on arbitrary sized inputs and can be trained end-to-end. Another advantage is that these networks are built on top of CNNs which are able to produce rich feature representations. These CNNs can be pretrained models on general purpose datasets, further reducing the time required to train the network.

These networks are denominated 'fully convolutional' because they do not have fully connected layers like CNNs and most artificial neural network architectures. Fully connected layers were removed for two reasons: first, they entail a single input size, not allowing the model to perform on arbitrary sized inputs; second, they only yield a single feature vector for the entire image. To replace the fully connected layer, fully convolutional layers were introduced, which are able to deal with the arbitrary size input.

1) From classifier to segmentation: The work of Long et. al [2] takes networks used for image classification problems, like GoogleNet, AlexNet or VGG16. In all the previous networks, the classifier is discarded and all the fully connected layers are converted to convolutions with 1×1 kernels. Convolutional layers with size 1 kernels are similar to fully connected layers but are capable of dealing with arbitrary size inputs. In a normal CNN, with fully connected layers, at the end of the classifier, the output would be a single predicted label. But since the fully connected layers are replaced by convolutional layers, what the network outputs is a feature map where features are assigned to a certain class.

The output of the fully convolutional layers has lower resolution than the input image, due to the pooling layers used to downsample. The method used by [2] uses transposed convolutions, or deconvolutions, to upsample the predictions. By doing this, the FCN's output has the same resolution as the original image.

From training and validating this architecture on the PASCAL dataset, Long et al [2] found the FCN with VGG16 was the one that yielded the best results.

III. Methodology

A. Dataset Preparation

The process of gathering visual data for the dataset consisted of three steps: first, going through all the videos and selecting periods with images suited for the task; second, sampling these video periods; last, from all the images collected from the sampling, clear the ones that were not good candidates, like images with too much noise or full of bubbles and sand.

The methodology followed to create the segmentation ground truths, after having all the inputs gathered, consists of three stages: applying contrast enhancement; produce an estimate segmentation mask, using edge detection tools; applying the necessary corrections, to yield ground truths as accurate as possible (Fig. 2). Let it be noted that almost all the images required some degree of manual correction.

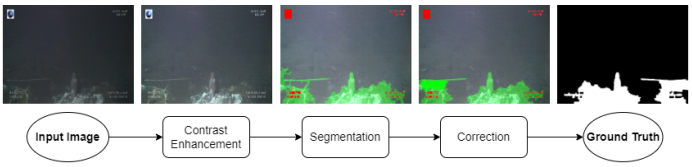


Fig. 2: Dataset creation flow chart.

B. Contrast Enhancement

When light travels in turbid mediums, like water or atmosphere, it suffers from absorption and scattering, which results in degraded images, with low contrast and poor colour quality. This scattering is not homogeneous across the scene, since this effect depends on the distance of scene points to the camera. Furthermore, in the conditions that this footage was produced, where natural lighting is non-existent, properly illuminating the entire captured scene is impossible, resulting in dark areas that can compromise the data analysis.

The Single Scattering Atmospheric Model (SSAM), models a hazy image, $I(x)$, as [4]

$$I(x) = J(x)t(x) + A(1 - t(x)), \tag{1}$$

where x is a coordinate vector of a given pixel, J is the hazeless image, A is the atmospheric light and t is the light that reaches the camera without scattering. If we calculate t and A, given the input I we can recover J and have the enhanced haze free image.

The first thing to be done is calculate the *dark channel*. The dark channel is formed by the pixels with the lowest intensity in one of the three RGB channels, in a patch of given size. The dark channel, J^{dark}, is given by

$$J^{dark}(x) = \min_{y \in \Omega}(\min_{c \in r,g,b} J^c(y)) \tag{2}$$

where Ω is a patch centered at x.

We can say that if J is a haze-free image, with recovered contrasts, then $J^{Dark} \rightarrow 0$..

The atmospheric light A, or the scene light, is calculated by choosing the 0.1% brightest pixels of the dark channel. This corresponds to the area with the most haze and lower contrasts, as the scattering is highly dependent on the distance travelled by light. The same pixels are retrieved from the original image I. Then, the mean of this group of pixels is calculated for each RGB channel.

Another necessary step for the improvement of image quality is to determine the transmission map, from the hazy image equation (eq. 1), which can be written as

$$t(x) = 1 - w \min_{y \in \Omega(x)} \Big(\min_{c} \frac{I^c(y)}{A^c} \Big) \tag{3}$$

where A^c is the atmospheric light in each color channel and w is a constant parameter ($0 < w \leq 1$), considering pixel intensities ranging from 0 to 1. Our understanding of human vision considers that haze allows the perception of distance and depth, an occurrence denominated aerial perspective.

If the transmission map was to be used just like it is yielded from equation 3, we would obtain an output image with undesired artifacts (halos or pixelated blocks) around objects present in the scene. In order to prevent this, the transmission map must be refined (filtered). The approach proposed by Tunai et al. [5] consists of using a guided filter. According to which, a filtered image q can be recovered from a guidance image I using

$$q_i = a_k I_i + b_k, \forall i \in w_k \tag{4}$$

with $\mathbf{i}$ being the pixel's index and and k the index of a local square window $\mathbf{w}$ of radius $\mathbf{r}$. The values of $\mathbf{a_k}$ and $\mathbf{b_k}$ can be determined using the methodology proposed by [5] and [6].

After determining the scene light $\mathbf{A}$ and the filtered transmission map $\mathbf{t(x)}$ we have the necessary information to recover the the enhanced image $\mathbf{J(x)}$, from equation 1 comes

$$J(x) = \frac{I(x) - A}{\max(t(x), t_0)} + A \tag{5}$$

where $\mathbf{t_0}$ is a constant introduced to establish a lower bound. For certain denominator values, the recovered image can be prone to noise in the most hazy regions, so it is good practice to introduce a constant ($\mathbf{t0}$) and limit the denominator value.

The parameter values chosen for this implementation were $w = 0.5$ (Eq. 3), $t_0 = 0.6$ (eq. 5), using patches of dimensions 15×15.

C. Operator Selection

Annotating an entire dataset by hand would be extremely time consuming and prone to human error. Thus, developing a methodology to help create the dataset is advantageous in what concerns efficiency and accuracy. Analysing the dataset, a characteristic becomes evident: most of the images are very rich in edges and corners, due to the presence of shellfish and algae on the seabed. Therefore, edge detection operators were chosen to extract those features from the images. Besides the edge detection approach, the k-means clustering algorithm for segmentation was also tested.

1) Sobel: Since the method that yielded the best results, was the Sobel approach, this document only covers this one. For this method, there are filters (kernel) that are convoluted accross the image. These kernels are an estimate of the derivative of pixel intensity, estimating the direction of the highest variation of pixel intensity (from bright to dark), which provides insights on how the the pixel intensities changes in each given point, and allows intensity gradient estimation. All these filters perform 2-D spatial gradient measurement, which means that a gradient is estimated across the horizontal (G_x) and vertical (G_y) dimensions and are then used to calculate the absolute gradient magnitude

$$G = \sqrt{G_x^2 + G_y^2}. \tag{6}$$

In theory, Sobel operator typically uses 3×3 kernels, however for improved segmentation results 5×5 kernels are also used. We used 5×5 kernels for better results. [7]

To prevent the detection of edges that appear in the image due to noise, the images must be filtered. The median filter from OpenCV was used with kernels of size $ksize = 5$. After filtering, the edge maps are retrieved by convoluting the kernels across the image. The edges alone are not a segmentation mask, because pixels are not labeled individually, instead only the pixels that are part of edges are labeled. However, if we perform morphological operations, such as closing and dilation, we are able to create a mask that, if overlayed in the original image, covers the entire object instead of just the edges. To remove unwanted areas, morphological transformations such as opening and eroding were implemented. Furthermore, a function was developed to erase areas that are bellow a certain size threshold, since most objects that would be segmented are large and unwanted areas of dust and other particles are small in comparison.

D. Fully Convolutional Network

The chosen framework for the implementation of the deep learning solution for semantic segmentation was Pytorch, developed by Facebook AI Research Lab and is seeing its usage increase at good pace. Comparing to other frameworks, such as Tensorflow or Keras, Pytorch is very flexible, offers good debugging capabilities and runs faster, meaning shorter training duration.

1) Data Loading and Preprocessing: The first step of the data loading process consists of cropping the images. The implemented model only accepts input images whose dimensions are multiples of 16. Since we have 1440 pixels of width, which already is a multiple of 16, the only dimension that should be cropped is the height. The original image's height is 1080 and it was converted to 1056.

Also, images should be converted from RGB format to BGR and pixel intensities for the three channels are normalized, so that they stay in the range $[-1; 1]$ instead of $[0; 255]$.

2) Architecture: The research by Long et al. [2] evaluates the performance of three different fully convolutional networks (FCNs), the FCN32s, FCN16s and FCN8s. The one that achieved the best results in [2] was the FCN8s, since it combines the output of deeper layers with outputs of shallower ones, preventing the loss of spatial information that is crucial to capture all the details. This operation of combining

https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga564869aa33e58769b4469101aac458f9

https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html

results of different layers will be further explained ahead. The implementation of the FCN8s can be split in two stages.

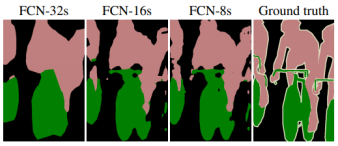


Fig. 3: Comparison between the results achieved by the FCN32s, FCN16s and FCN8s. Source: [2]

First, a transfer learning approach was used where a pretrained VGG16 network imported from Pytorch is integrated for the convolutional part of the network. The traditional VGG16 consists of 5 convolutional and pooling layers followed by fully connected layers. However, since we want the network to accept inputs of arbitrary size, the fully connected layers are replaced with another two convolutional layers, performing the convolutions with kernels of size 1×1. Second, the transpose convolution (deconvolution) layers are added at the output of the last convolutional layer, to upsample results to the size of the input image, providing an image where each pixel is assigned to a class.

This architecture yields good results because it combines results from different layers, making local pixel predictions while respecting global structure. The outputs of the 3^{rd}, 4^{th} and 5^{th} convolutional layers from VGG16 are combined with results from the transpose convolution layers(figure 4). When we go deeper in the network, we lose spatial information, due to all the convolution and pooling layers. So, if we combine the output of deeper layers with the output of shallower ones, we are adding location information, it is expected that the quality of the results increases, due to increased detail. This combination is an elementwise sum. The output of the 5^{th} convolutional layer is upsampled by a factor of 2 and summed with the output of the 4^{th} convoluional layer. The output of this operation is again upsampled by a factor of 2 and summed with the output of the 3^{rd} convolutional layer, which is later upsampled by a factor of 8 to yield the final result.

The chosen activation function used at the end of each layer was the rectified linear unit (ReLU), as it is the most commonly used for CNNs and FCNs. The selected loss function was binary cross entropy (BCE) and the optimization process is done through stochastic gradient descent, using the RMSProp optimization algorithm.

3) Network Parameters: Regarding the parameters chosen for the training phase, for computational capacity reasons the batch size was set to 1, as images have a very large number of pixels and consume a great amount of memory, even though the training was executed using GPUs.The learning rate was set to $1 \cdot 10{-}5$, weight decay $1 \cdot 10^{-5}$, step size 50 and momentum 0. The training dataset was iterated across 130 epochs. This configuration of parameters was the one that achieved the best results.

4) Dataset Management: The dataset consists of 1198 images, of which 798 were used for training and 150 for validation. After training, the model was tested with a dataset of 150 images, with examples that were not used in the training phase. Furthermore, since the model was trained several times with different parameters, a dataset with 100 unique examples from a different underwater site was stored to evaluate the model performance after the configuration of training parameters was settled. The purpose of this last test is to assess how robust the model is. It should be noted that after the first and only time this last test was performed, the network was not altered.

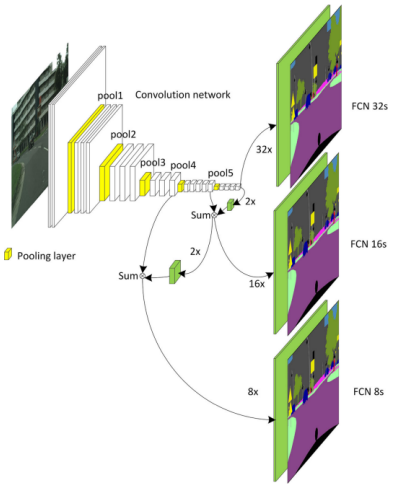


Fig. 4: Graphic representation of the combination of results from different layers. This process is used to improve the quality of results, by adding feature maps from deeper layers. Source: [8]

5) Evaluation Metrics: To evaluate the network's performance during training and testing sessions the selected performance metrics were mean pixel accuracy and intersections over union, as these are the ones considered most relevant in the work of Long et al. [2].

The mean pixel accuracy (MPA) can be defined as

$$MPA = \frac{CP}{TP} \tag{7}$$

where CP is the number of correct pixel classifications and TP is the total number of predictions. The number of correct pixels is a sum of the number of true positives and true negatives.

The intersections over union (IoU), sometimes also called Jaccard index, quantifies the overlap between the ground truth mask and our prediction mask. [9]

$$IoU = \frac{target \cap prediction}{target \cup prediction} \tag{8}$$

The intersections over union are determined for each class separately. Then the average of the IoUs is determined to provide a global metric of the segmentation predictions.

IV. Results

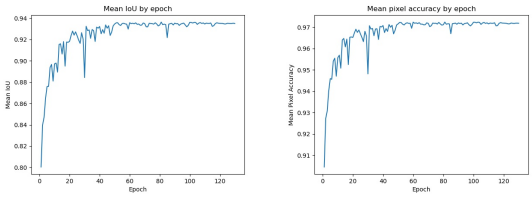
A. Fully Convolutional Network

The model was trained a total of 8 times, with different configurations of hyperparameters and correction of the dataset

between each training phase. After determining the configuration of parameters that achieves the most accurate results, the model is tested on a set of images, collected from a different site. The results of these last test are also exposed in this section.

1) Training an FCN with Different Parameters: These interim tests are the set of tests that aim at reaching the optimal configuration of network training parameters for this problem. All the tests were performed on the same dataset. After each epoch of training, a validation set is ran through the network to evaluate performance evolution throughout the epochs of training. For each training phase, the validation dataset is always a random subset of 150 images from the overall training set.

Throughout this testing phase, the network parameters, like the number of epochs, learning rate or weight decay have been tuned to evaluate the impact of changing these values on the model's performance. In total, 7 tests were conducted and there was no expressive changes in performance across them. We chose a configuration of parameters that presented good accuracy and IoU stability at the end of the training phase, indicating the algorithm had converged towards an optimal solution (figure 5). The achieved results on the validation and test sets are present in tables I and II.



(a) Mean IoU evolution for 7^{th} test.

(b) Mean pixel accuracy evolution for 7^{th} test.

Fig. 5: Evolution of the performance metrics, IoU and pixel accuracy, across epochs for the 7^{th} test.

TABLE I: Results Achieved on the Validation Set.

Pixel Accuracy	Mean IU
96.70%	92.53%

TABLE II: Results Achieved on the Test Set.

Pixel Accuracy	Mean IU
97.9%	94.61%

2) Final Test: For the last test, the model used was the one reached when training for test number 7. The dataset used for this test was significantly different from the one used for training, since the images were collected in a different underwater cite, with different conditions. The goal was to evaluate the model's capacity to generalize when the data being used is different.

Looking at table III, the network achieved very accurate results, with nearly 93% mean pixel accuracy and 85.21% IoU.

TABLE III: Results Achieved by the Fully Convolutional Network

Pixel Accuracy	Mean IU
92.80%	85.21%

The Sobel-based method achieved 91.5% mean pixel accuracy over the same samples. However, as an example, the FCN performs better than Sobel in scenes where parts of the vehicle are visible.

Another interesting metric, the FCN can yield segmentation predictions in around 0.01 seconds, which translates to 100 Hz. The video footage used to create the dataset for this paper had a frame rate of 25 frames per second. Meaning that our approach is 4 times faster than the frame rate of the camera.

B. Simultaneous Localization and Mapping

To evaluate the impact of the proposed methodology on real time navigation algorithms we have ran offline SLAM and online SLAM tools on sequences of images with no masks and with masks generated by the trained FCN model.

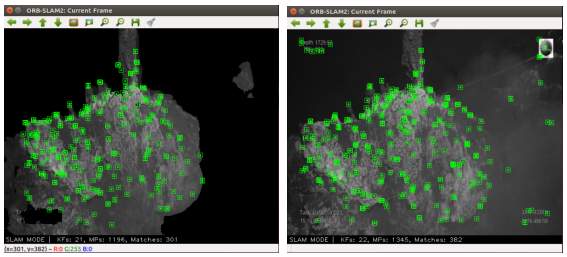
1) Offline SLAM: Agisoft Metashape Standard (Version 1.7.2)(Software) is a photogrammetry software, used for 3-D reconstruction of scenes. To evaluate the impact of semantic segmentation on offline SLAM, the program ran a sequence of 46 images. Figure 6 displays the software's output with and without using masks. Although there is no huge difference between the two cases, we can see that when no masks are used (6 **(b)**) the software builds the texture for a dust cloud that goes in front of the camera. The dust cloud has a brownish color. Furthermore, in the case without masks, the program also puts alpha-numerical characters, that appear on the corners of the input images, on the scene. When (6 **(a)**) masks are used, neither the dust cloud, nor the characters are built on the reconstructed scene.

(a) (b)

Fig. 6: **(a)** Output of Agisoft Metashape after processing a sequence of images with masks; **(b)**Output of Agisoft Metashape after processing a sequence of images without masks, in this case we can see the algorithm reconstructs a dust cloud.

2) Online SLAM: ORB-SLAM: On the scope of online SLAM, ORB-SLAM was ran on a sequence of 1000 images. To evaluate the impact of using masks, the images and their respective masks were overlayed so that only the foreground

was visible and everything else was 'deleted', by making every non-foreground pixel black, see figure 7 **(a)**.



(a) (b)

Fig. 7: Frame with features extracted by ORB-SLAM **(a)** with mask; **(b)** without mask.

On a more quantitative analysis, let us consider the trajectory's jerk, as the accuracy cannot be computed for the ground truth trajectory is not available. The jerkt is a physics concept that amounts to the rate at which an object's acceleration changes over time. It is denoted by j and its units are m/s^3. It is usually expressed as a vector, being the first derivative of acceleration. It can also be expressed as a third derivative of position.

Since ORB-SLAM outputs, for every keyframe, the coordinates of the camera's optical center, in respect to the world's coordinate system, jerk will be a 3-dimensional vector at any given point in time.

$$j(t) = \left(\frac{d^3x(t)}{dt^3}; \frac{d^3y(t)}{dt^3}; \frac{d^3z(t)}{dt^3} \right) \tag{9}$$

Besides the coordinates output, ORB-SLAM also provides the timestamps of those keyframes. This allows us to compute these derivatives through finite differences.

$$r'''(t_0) = \frac{-\frac{1}{2}r(t_{-2}) + r(t_{-1}) - r(t_{+1}) + \frac{1}{2}r(t_{+2})}{h_t^3} \tag{10}$$

where h_t is the time difference between each finite difference interval.

In robotics, a low jerk means the trajectory is smooth. In terms of the robot's control system, a smoother trajectory allows for less complex and more robust performance. Table IV establishes a comparison between the trajectory's jerk using masks and without masks. For a more accurate analysis, it should be noted that when running with segmentation masks, ORB-SLAM takes more time to initialize, since it detects less features per frame. However, it is able to find the same amount of keyframes (21) in a much smaller period. The results on table IV were computed by calculating the average of all jerks' norms. When running with masks, the jerk is approximately one order of magnitude lower than without masks.

TABLE IV: Average of Jerk Norms.

With Mask	Without Mask
0.021	0.258

V. Conclusions

The purpose of this work was to study the impact of introducing AI-based semantic segmentation tools on the performance of SLAM algorithms, for integration in autonomous underwater vehicles. An annotated dataset, with 1200 pairs of images and masks, was created to train a neural network to perform semantic segmentation and the results were tested on both online and offline SLAM algorithms.

To develop the dataset, images were enhanced to improve feature detection. Also, a tool was developed to yield an initial estimate of segmentation mask, using Sobel-based edge detection, which were hand corrected afterwards to create the segmentation ground truth for each image.

The chosen deep learning model for semantic segmentation was a Fully Convolutional Network, that was trained several times with different hyperparameters. With the network that yielded the best results, at around 93% accuracy, a sequence of frames was ran through the model to have masks to test on the SLAM algorithms.

The results were tested on offline SLAM, using Agisoft Metashape Standard (Version 1.7.2)(Software) and on online SLAM, with ORB-SLAM. On both cases the proposed methodology improved the quality of the results.

Future work should include the development of a larger and more diverse dataset and using it to train a Fully Convolutional Network, so the model can cope with conditions different from those of the proposed dataset. Also, for further validation, this methodology should be tested using an underwater autonomous vehicle on the context of real time ocean exploration.

References

[1] M. J. Islam, C. Edge, Y. Xiao, P. Luo, M. Mehtaz, C. Morse, S. S. Enan, and J. Sattar, "Semantic segmentation of underwater imagery: Dataset and benchmark," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 1769–1776.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[3] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015.

[4] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 12, Dec. 2011.

[5] T. P. Marques, A. B. Albu, and M. Hoeberechts, "Enhancement of low lighting underwater images using dark channel prior and fast guided filters," *Lecture Notes in Computer Science*, vol. 11188, 2018.

[6] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 6, Dec. 2013.

[7] H. Kekre and S. Gharge, "Image segmentation using extended edge operator for mammographic images," *International journal on computer science and Engineering*, vol. 2, no. 4, pp. 1086–1091, 2010.

[8] X. Liu, Z. Deng, and Y. Yang, "Recent progress in semantic image segmentation," in *Artificial Intelligence Review*, vol. 52. Springer International Publishing, 2019, pp. 1089–1106.

[9] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, and T. Isenberg, Eds. Cham: Springer International Publishing, 2016, pp. 234–244.