Identifying Camouflaged Cuttlefish via Classical and U-Net-Based Image Segmentation

Carlos J. Anzola Castellanos, MS Robotics (2026)
Li Wang, MS Computer Systems and Engineering (2026)

December 6, 2025

Department of Electrical and Computer Engineering
Northeastern University

ABSTRACT

Cuttlefish camouflage makes automated segmentation challenging for marine research. We compare classical image processing techniques against deep learning for detecting camouflaged cuttlefish in underwater images. We tested three classical pipelines, including color-based, frequency-based, and their fusion, plus a U-Net model. Classical methods achieved IoU 0.7-0.8 on favorable cases but required 5-60 minutes of manual tuning per image and failed when color and texture both matched substrate very closely. U-Net achieved test IoU 0.96 automatically in seconds with smoother boundaries but failed on domain-shifted environments. Classical methods lack generalization without manual tuning. Deep learning needs representative training data. Both struggle when discriminative features are absent.

CONTENTS

I. Introduction

Camouflage in cuttlefish poses a difficult segmentation problem because their color, texture, and luminance patterns change based on environmental cues. Since cuttlefish can match the background very closely in color and texture, in some cases there are no obvious visual differences to rely on for proper identification and segmentation.

We first tested a set of pipelines using classical image processing methods designed to pick up differences in color and texture. We did not expect these methods to solve the problem in a general way, but expected they could work on a subset of scenes where some contrast or texture differences remained. We then trained a U-Net model to see whether a supervised deep architecture could learn useful features from a relatively small dataset and generalize across different environments.

The classical pipelines produced inconsistent results, were highly sensitive to image-specific parameters, and tended to underperform in cluttered backgrounds or extreme illumination. Their best cases required manual parameter tuning and still struggled in low-contrast regions. On the other hand, the U-Net achieved smoother segmentation quality with less dependence on the scene, though its performance dropped on images that differed strongly from the training dataset. Even with these limitations, the U-Net model consistently outperformed the classical pipelines.

II. Classic Image Analysis Methods

This section summarizes three classical pipelines that tested different hypotheses about what could separate the cuttlefish from the background. For all pipelines, we varied parameters and sometimes selected results manually, because automatic settings often failed. Evaluation was mostly qualitative due to inconsistent outputs.

The dataset used to test these pipelines consists of a 25-image sample of the Fathomnet [2] dataset. The images were chosen based on their characteristics. Images where cuttlefish were well camouflaged, and that had diverse environment conditions, were preferred over images where the cuttlefish were not camouflaged.

Although the original goal was a fully automatic pipeline, multiple steps proved difficult to generalize. The approaches required extensive fine-tuning and still failed on some images.

Therefore, in the interest of time, we allowed for user input in specific steps.

We report IoU, Dice, Precision, Recall, F1, and Pixel Accuracy mainly for completeness and to provide standard quantitative references. However, while these metrics are commonly used in computer vision, our actual tuning and comparisons relied mostly on visual inspection, with the metrics serving more as supplementary information. More detailed descriptions of the metrics can be found in the U-Net section.

A. Pipeline 1: Color-based Segmentation

The core hypothesis driving this pipeline is that camouflaged cuttlefish can exhibit subtle differences in color space from their environment's substrate. Specifically, we hypothesized that even when intensity values appear similar, the distribution across different color channels could reveal discriminative features. The cuttlefish might match the overall brightness of sand or rocks, but the saturation, hue, or chrominance components could differ enough to enable segmentation through clustering in multi-dimensional color feature space.

1. Architecture

The underwater imagery showed uneven lighting, color casts, and low contrast, which made segmentation harder. To reduce these effects, we applied a series of enhancements. Images were first corrected using either gray-world or black-patch, depending on which produced fewer artifacts for a given image. Gray-world assumes that the average color in a scene should be neutral, and adjusts channel values to reduce overall color bias. Black-patch uses dark regions in the image to normalize illumination, which can reduce the influence of strong directional lighting. Then, CLAHE was applied to increase local contrast.

As can be seen in Figure 1, some images have predominant yellowish or blueish tones. The color correction helps balance the image tone, while CLAHE enhances the details.

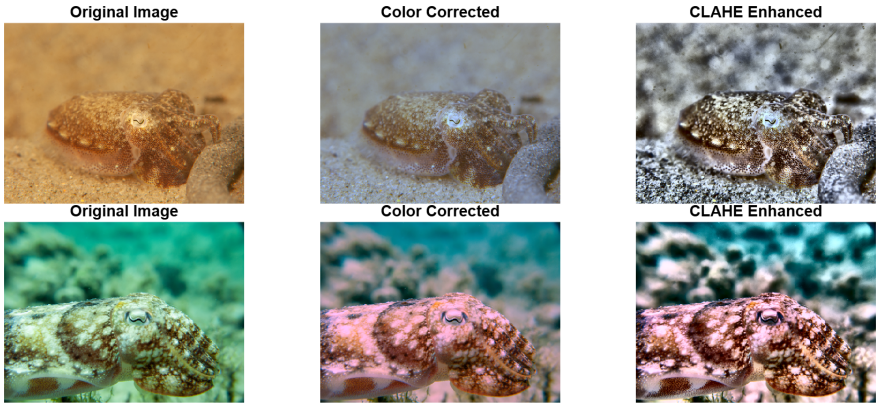

Fig. 1: Examples of image corrections in Pipeline 1

After correction, images were converted into multiple color spaces and split into individual channels. The user has the option to choose the color space(s) where it is easier to identify the cuttlefish. We noticed that, in most cases, channels a* and b* gave a noticeable difference between the cuttlefish and the background.

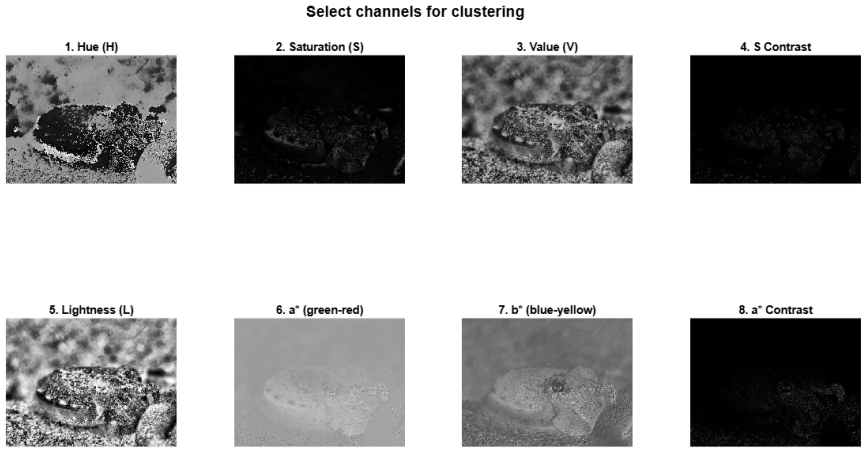

Fig. 2: Channel selection tool snapshot in Pipeline 1

Then, pixel values from selected channels were clustered using k-means. Increasing the number of clusters occasionally helped separate the animal from background noise, but also increased the computation time.

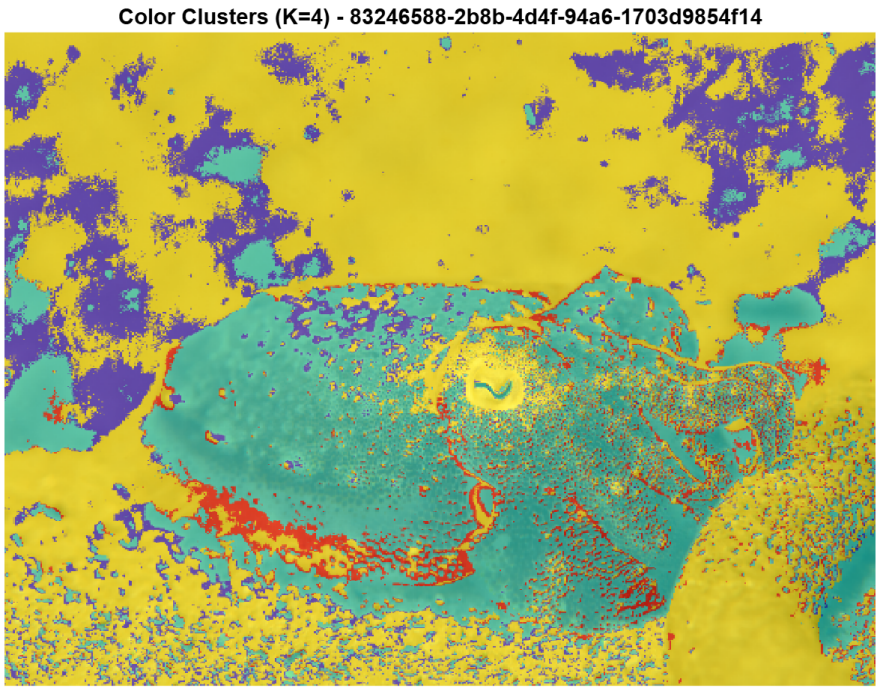

Fig. 3: Example of color clusters in Pipeline 1

Then, the user can specify the cluster(s) that best segment the cuttlefish (Figure 4). These masks are combined and cleaned with morphological opening and closing to remove small false positives and fill small gaps.

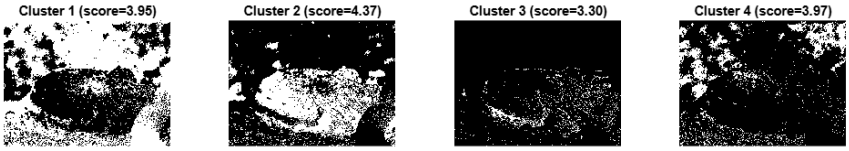

Fig. 4: Clusters for user review

2. Results

Figures 5-6 show two of the best results obtained. We determined these were good based on visual inspection, IoU, and the time spent tuning the pipeline parameters. In the case of the best results, tuning time was less than 5 minutes, and all the metrics were above 0.7. We found that the images requiring less fine-tuning were those where the cuttlefish, despite being camouflaged, still stood out a bit in color from the surrounding environment.

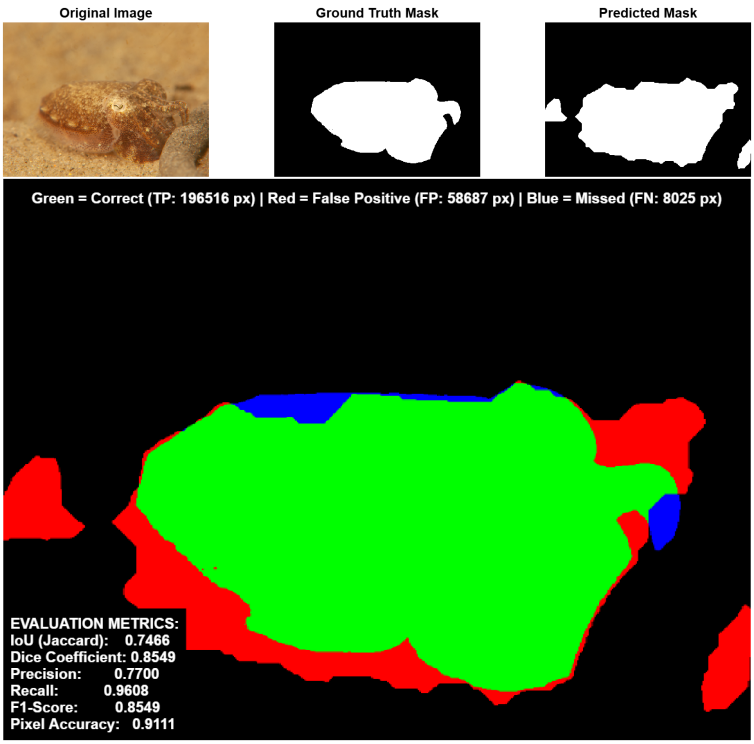

Fig. 5: Best segmentation result 2 Pipeline 1

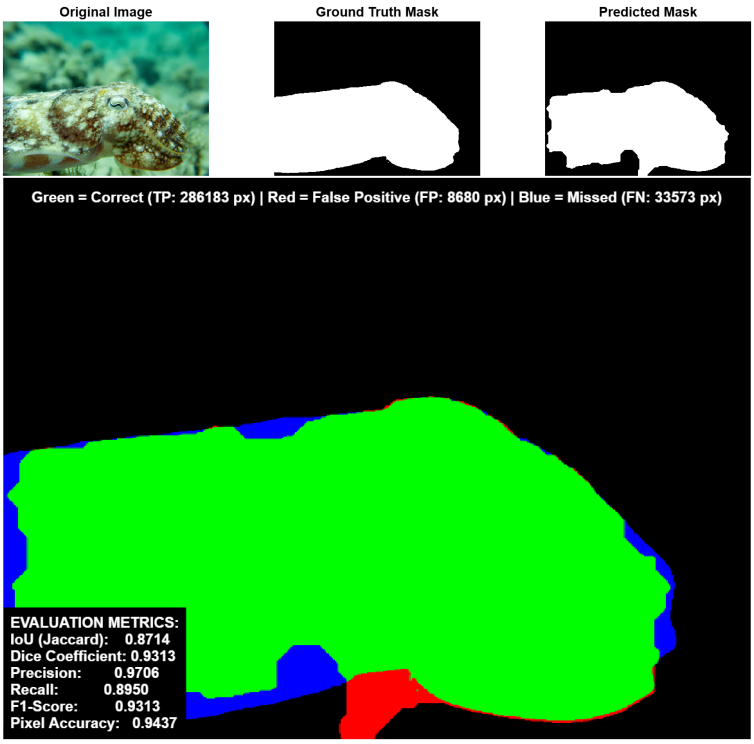

Fig. 6: Best segmentation result 2 Pipeline 1

Now, images with strong light beams required heavy fine-tuning with marginal improvements and not as good results. We found that black-patch enhancement performed better than gray-world probably because it focuses on correcting overexposed regions by emphasizing darker areas. This helps reduce the effect of bright streaks that otherwise distort the segmentation mask.

Increasing the number of clusters also helped mitigate the effects of strong lighting, but these images typically required about twice as many clusters as others, which increased computation time. Examples of these more difficult cases are shown in Figures 7 and 8.

Although Figure 8 shows relatively high precision and captures a reasonable portion of the cuttlefish, getting this result required about an hour of fine-tuning. In contrast, images without strong light beams reached acceptable results after only a few adjustments using the baseline parameters.

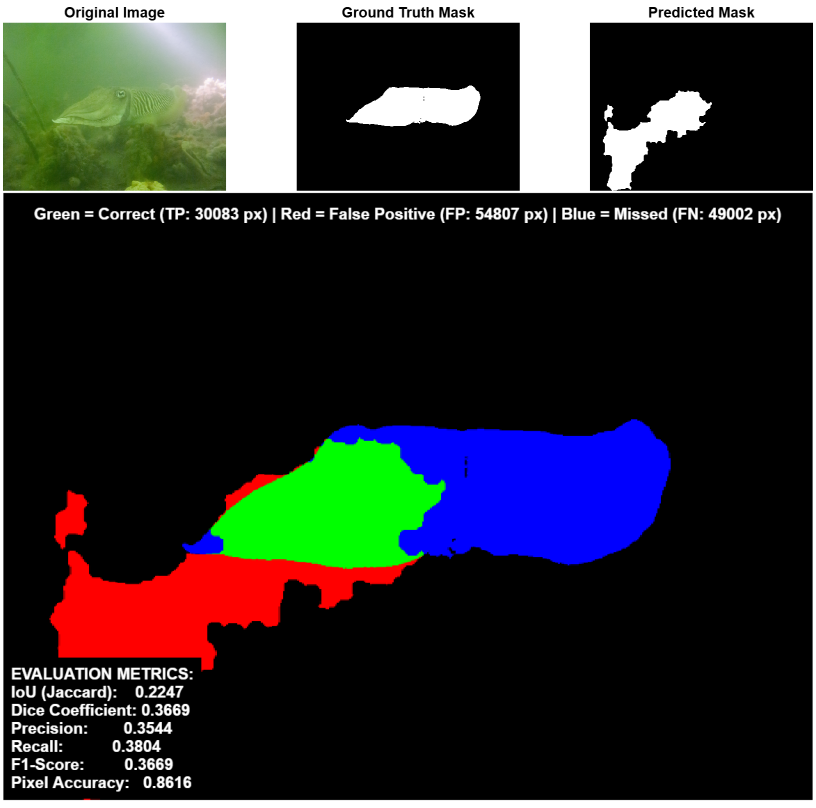

Fig. 7: Bad segmentation result 1: Heavy light

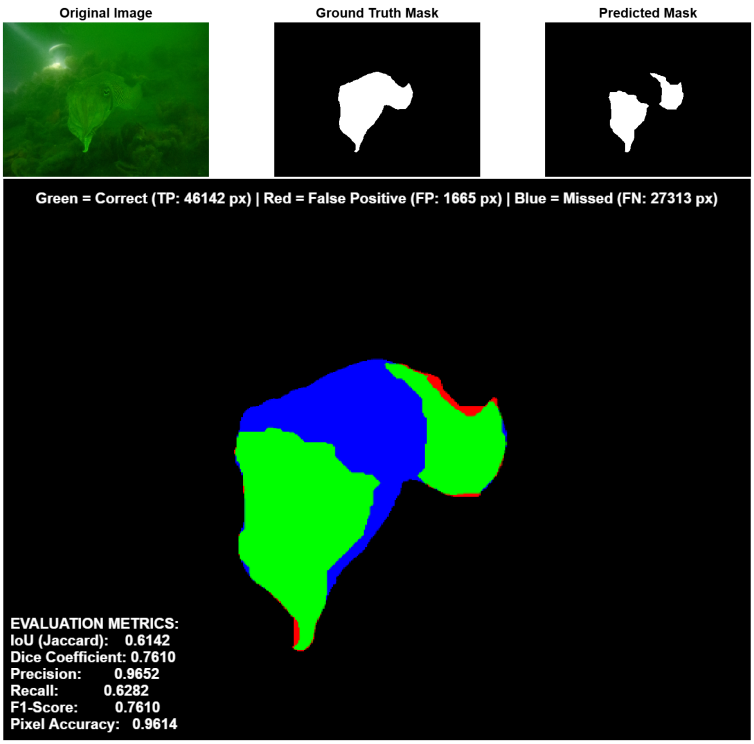

Fig. 8: Bad segmentation result 2: Heavy light

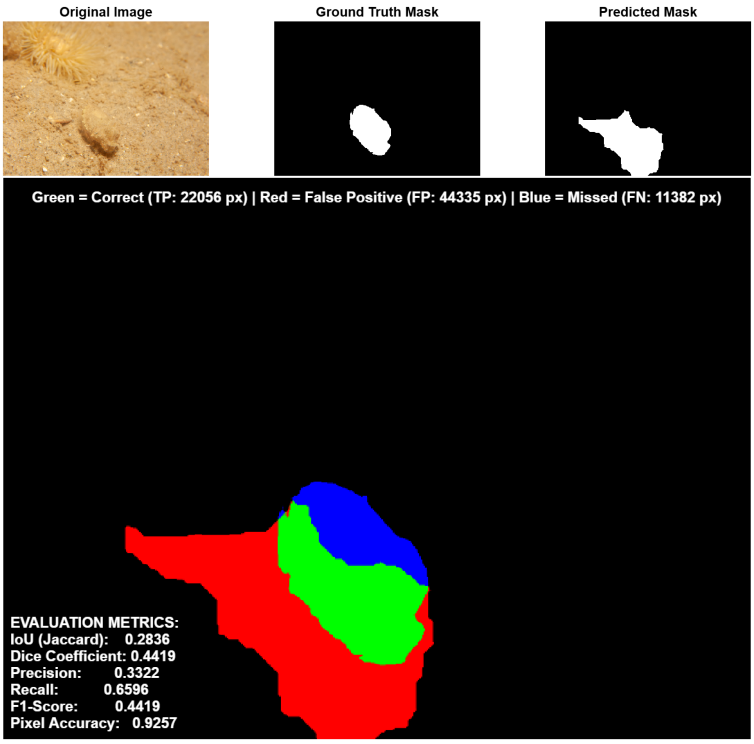

Fig. 9: Bad segmentation result 3: Identical color

Another limitation of the color pipeline is when the camouflage matches the soil color very well. Figure 9 shows an example of this. Even after long tuning, the best outcome captured only about half of the cuttlefish. However, we acknowledge that this half is mostly its shadow, and, therefore, not a reasonable result for the amount of tuning.

Based on our results, this pipeline succeeds easily when the cuttlefish's camouflage color is slightly different from the soil. We expected the pipeline to be able to capture minimal subtle differences in color, but it was not the case. Additionally, the illumination seemed to be the biggest issue, causing extensive tuning times and making parts of the image colors too extreme. For these reasons, we considered that color-based segmentation was not general enough for proper results, and we decided to try texture-based segmentation instead.

B. Pipeline 2: Texture-based Segmentation

This pipeline follows the hypothesis that even when cuttlefish match the color of their substrate, they may not perfectly replicate the spatial frequency characteristics. We thought that biological textures and patterns possess organized structure with characteristic frequencies, whereas natural substrates like sand and gravel may exhibit more random, broadband frequency distributions. We hypothesized that analyzing the local power spectrum would reveal these structural differences, allowing segmentation based on texture signatures rather than appearance features.

1. Architecture

For this pipeline, we didn't use the same enhancements from pipeline 1 because we wanted to try different techniques and compare. We start by applying a homomorphic filter to reduce the impact of uneven illumination, allowing the texture patterns of the cuttlefish to be more visible. Then, we do global and local frequency analysis. The global analysis identifies dominant spatial frequencies in the image, which can highlight textural differences between the cuttlefish and its background. Local frequency analysis captures variations within smaller regions, helping to detect finer structural details that are characteristic of the cuttlefish's skin patterns.

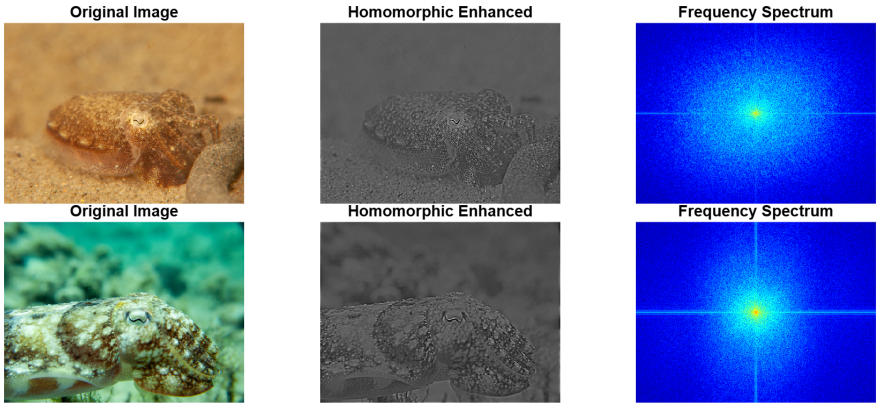

Fig. 10: Examples of image corrections in Pipeline 2

Similar to Pipeline 1, features were clustered using K-Means. The user then selects the most appropriate cluster or clusters, which are combined into a single mask and finally refined using morphological operations.

2. Results

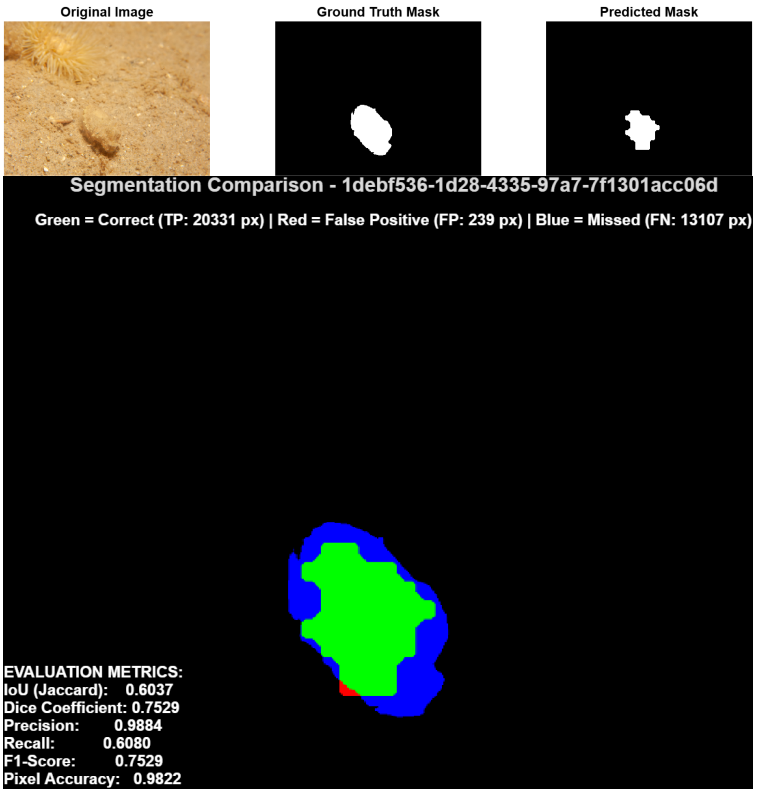

Fig. 11: Segmentation result Pipeline 2

Just as we expected, pipeline 2 did a better job at identifying cuttlefish whose camouflage replicated the soil color extremely well. Figure 11 shows an improvement in the segmentation mask on the same image as Figure 9, in which pipeline 1 struggled. For pipeline 2, it can be noticed that most of the cuttlefish was properly identified.

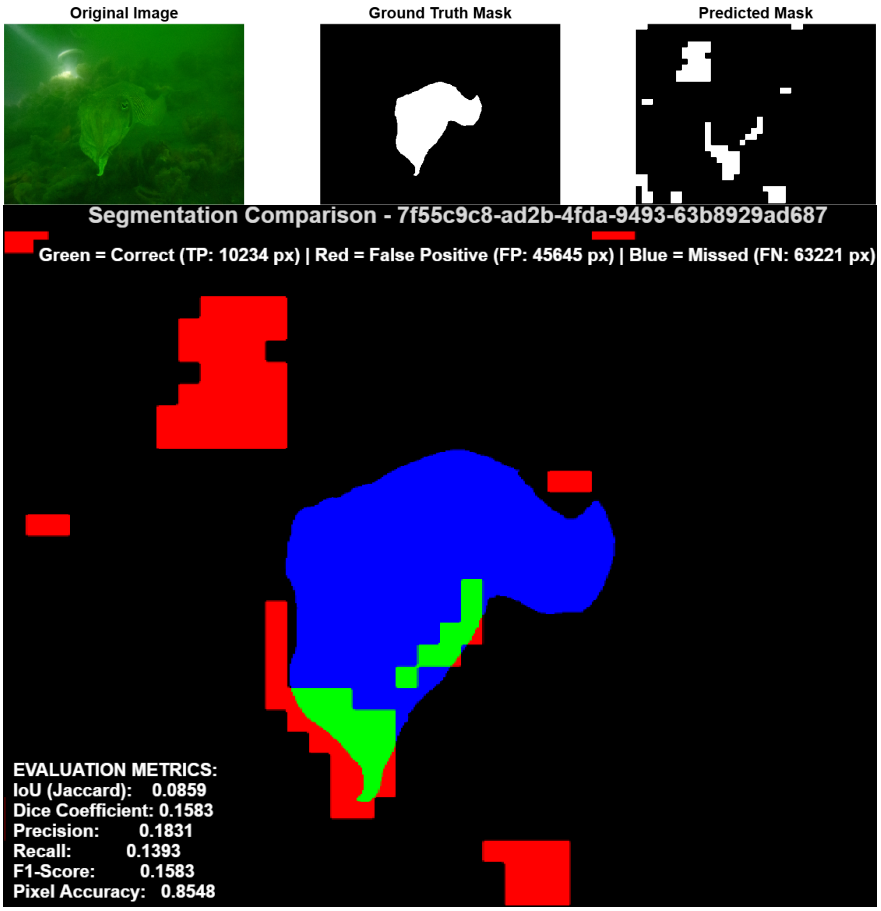

Fig. 12: Bad segmentation result: Heavy light

Images with strong light variations still didn't produce good results, as seen on Figure 12. For this particular image, the results are considerably worse than pipeline 1.

Additionally, while tuning these images, we noticed another limitation of this pipeline. Cuttlefish can have very different textures simultaneously when camouflaged. Some areas have have stripes, some have grains, some are very smooth. This variation makes clustering and segmentation more difficult, as the surrounding environment often contains similar textures, causing each cluster to capture substantial background noise.

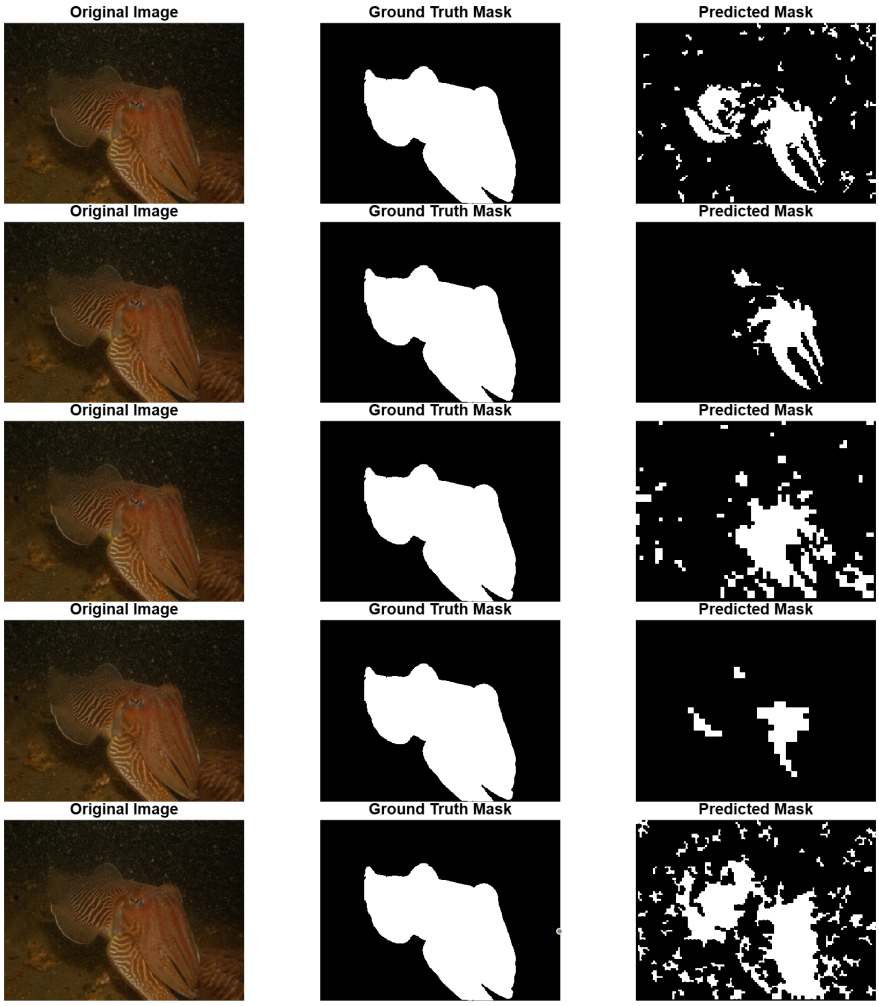

Fig. 13: Segmentation of a multi-texture cuttlefish camouflage

As shown in Figure 13, the cuttlefish exhibits multiple patterns along its body. We expected the pipeline to be able to detect each of these, and combine them properly, but that was not the case. The environment shared a lot of these textures as well, so fine-tuning and cleaning the masks became an even bigger problem. Even after applying morphological operations, these were the best results we could achieve after roughly an hour and a half of tuning.

C. Pipeline 3: Frequency and Color Segmentation

This pipeline is motivated by the observation that pipelines 1 and 2 each succeed on different image subsets. The central hypothesis is that by combining color-based and frequency-based features in a unified feature space, we can complement each approach's strengths to achieve a more robust segmentation across a wider range of conditions. When color

provides weak but non-zero discrimination and frequency analysis provides weak but non-zero discrimination, their combination may provide sufficient discrimination even though neither alone succeeds.

1. Architecture

We start with the same color correction from Pipeline 1 (either gray-world or black-patch depending on the lighting) followed by CLAHE enhancement to boost local contrast. Then, we extract color features by converting to HSV and Lab color spaces and let the user pick the best channels. At the same time, we run the homomorphic filter and local frequency analysis from Pipeline 2. These color and frequency features get normalized and combined into a single vector for each pixel. Similarly to pipelines 1-2, we then cluster these combined features using k-means, and the user selects which cluster(s) look the best. Finally, we apply the same morphological operations to clean up the mask.

The new parameters in this pipeline are the weights for the feature types. We included a weight for color-based features, and one for texture-based features.

2. Results

This pipeline had a lot of parameters to tune, yet it took less time to achieve a better result in the segmentation of images with intense light beams. This is shown in Figure 14. Additionally, the multi-texture problematic image yielded better results than in pipeline 2. Pipeline 1 results are not shown due to space limitations, but the best case for that pipeline suffered from considerable over-segmentation.

As seen in Figure 15, the complementary segmentation approach produced better results. For these images, the feature weights were set to 0.25 for frequency-based features and 0.75 for color-based features. Although tuning took longer than the easiest cases in Pipelines 1 and 2, it was still shorter than the most difficult cases in both of those pipelines.

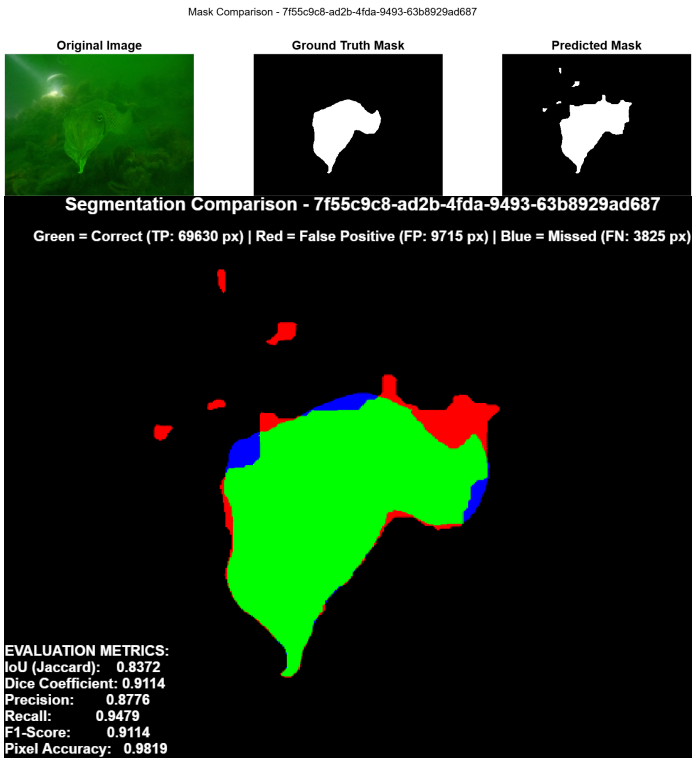

Fig. 14: Better segmentation result in Pipeline 3: Heavy light

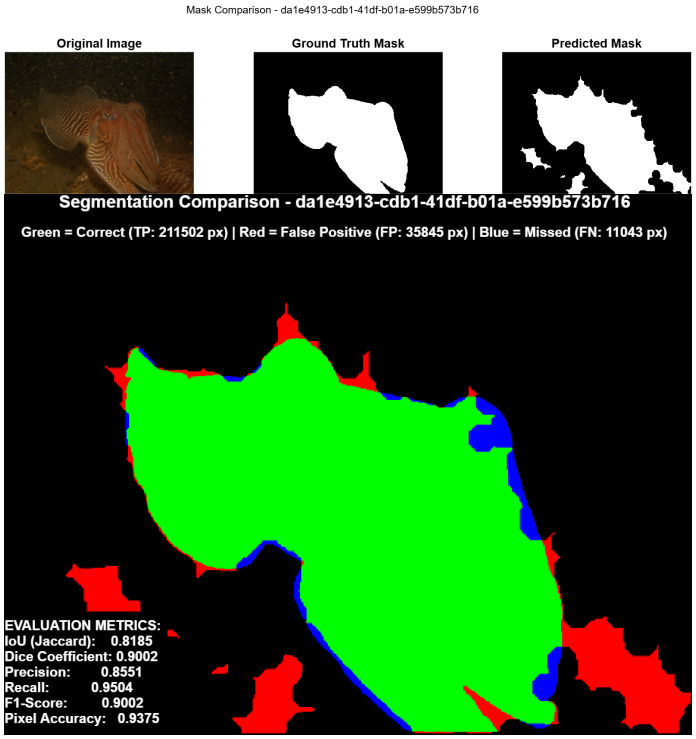

Fig. 15: Better segmentation result in Pipeline 3: Multi-texture

III. U-Net Model Based Segmentation

For U-Net-based segmentation [1], we developed an end-to-end deep learning pipeline for cuttlefish image segmentation. Due to the limited available annotated data, we downloaded all the 50 cuttlefish images with corresponding bounding boxes from FathomNet [2] and generated pixel-level segmentation masks using the Segment Anything Model (SAM) [3]. After image preprocessing and augmentation, there are 2,469 images available 5 unique underwater scenes and achieving a 76%/14%/10% train/validation/test split.

A. Datasets

The 50 cuttlefish images with bounding box annotations are downloaded from FathomNet. These images originated from 5 unique underwater scenes across different locations including Netherlands, Crete, and Malaysia, with the majority (38 images) coming from a single Netherlands dive session [2].

1. Image Preprocessing And Augmentation

We preprocessed images with color correction, contrast enhancement, and denoising, then applied various augmentations (flips, rotations, elastic transforms, color shifts, blur, and noise) to expand our dataset 40× from 50 to 2,469 samples.

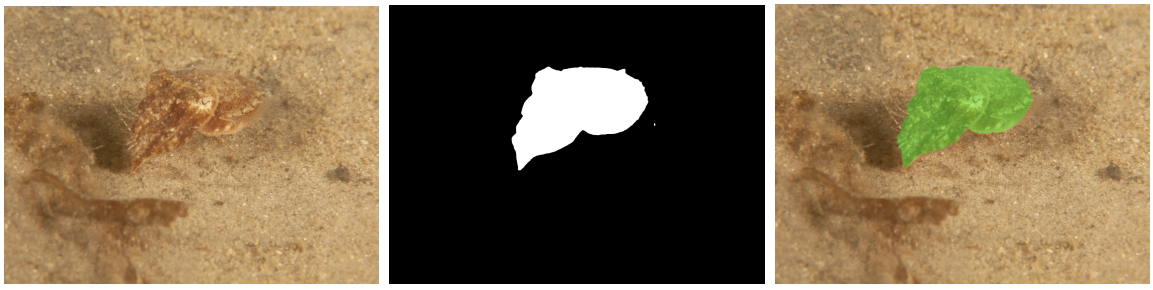

Fig. 16: Original Image, Binary Mask, SAM Mask

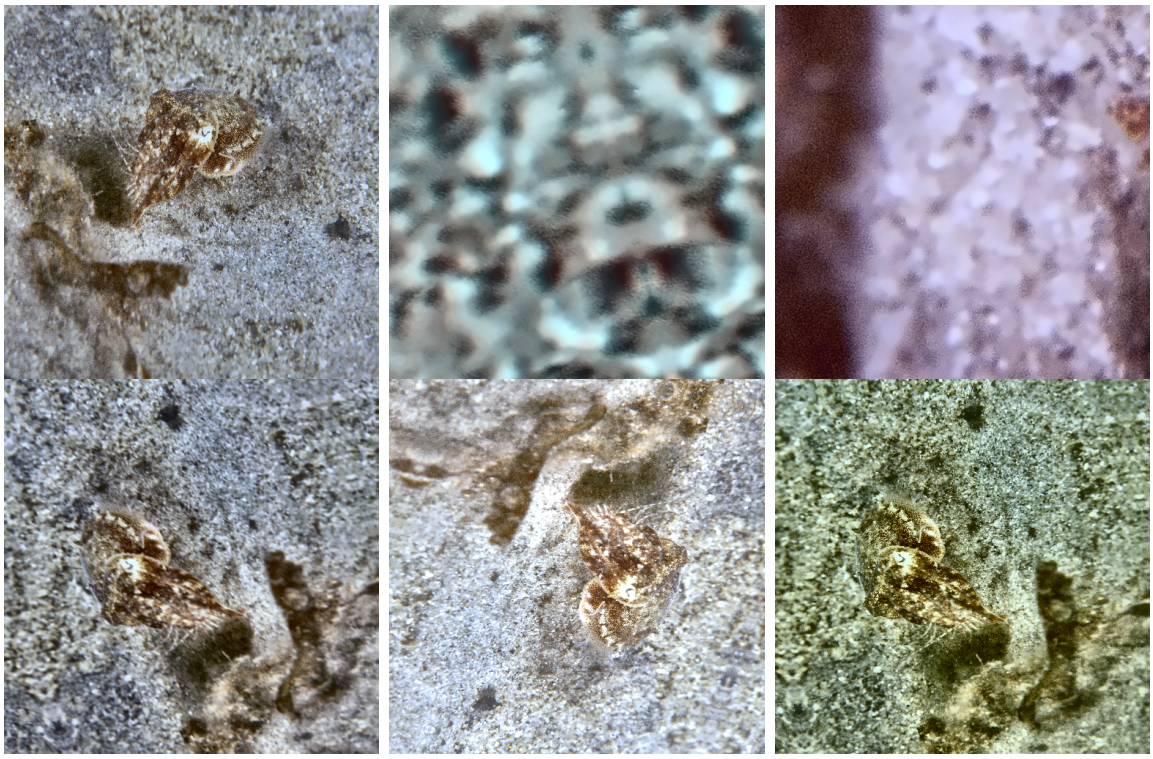

Fig. 17: Image preprocessing and augmentation: example 1

Fig. 18: Image preprocessing and augmentation example 2

2. Dataset Split

Since images from the same dive look too similar, we split by scene rather than randomly training on 38 Netherlands images, validating on 7 Crete images, and testing on 5 images from three other scenes (76%/14%/10% split). All the images are resized to a fixed input (i.e., 512 x 512) for the U-Net model. The dataset split distribution is shown in Table I.

TABLE I: Dataset split distribution

Split	**Original Images**	**Percentage**	**Augmented Images**	**Percentage**
Train	38	76%	1,727	70%
Validation	7	14%	442	18%
Test	5	10%	300	12%
Total	**50**	**100%**	**2,469**	**100%**

B. U-Net Model Based Approach Pipeline

1. The U-Net Model

U-Net is a neural network with two parts: an encoder that shrinks the image to learn what's in it, and a decoder that expands it back to make a pixel-by-pixel prediction. Our U-Net uses a ResNet-34 [4] encoder pretrained on ImageNet, allowing it to leverage features learned from over 1 million images. The model has about 24.4 million parameters, takes RGB images as input, and outputs a probability map indicating which pixels belong to the cuttlefish.

Three metrics are used to evaluate the segmentation model: Dice score (Equation 1), IoU (Intersection over Union) (Equation 2), and Boundary F1 (Equation 3).

Dice and IoU both measure how much the predicted mask overlaps with the ground truth. True Positives (TP) refers to pixels that are correctly identified as cuttlefish by the model. False Positives (FP) are pixels that the model incorrectly predicts as cuttlefish when they are actually background. False Negatives (FN) are cuttlefish pixels that the model fails to detect.

$$\text{Dice} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{1}$$

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{2}$$

Boundary F1 focuses specifically on how accurately the predicted edges match the ground truth boundaries, which is useful for evaluating fine details.

$$\text{BF1} = \frac{2 \times P_\partial \times R_\partial}{P_\partial + R_\partial} \tag{3}$$

where P_∂ is Boundary Precision and R_∂ is Boundary Recall.

2. Model Training

We resized all images to 512×512 pixels so the model receives a consistent input size. For training, we used a combined loss function that adds Dice loss (Equation 4) and Focal Binary Cross-Entropy (BCE) loss (Equation 5) together, each weighted at 0.5. The AdamW optimizer was used with a learning rate of 1×10^{-4} and weight decay of 1×10^{-4}, along with a cosine annealing scheduler that slowly lowers the learning rate as training progresses. Since cuttlefish pixels make up a small portion of each image compared to the background, we used Focal BCE loss with $\alpha = 0.25$ and $\gamma = 2.0$ to give more importance to the foreground pixels.

$$\mathscr{L}_{\text{Dice}} = 1 - \frac{2\sum_i \hat{y}_i y_i + \epsilon}{\sum_i \hat{y}_i + \sum_i y_i + \epsilon} \tag{4}$$

where $\hat{y}_i$ is the predicted probability and y_i is the ground truth label for each pixel.

$$\mathscr{L}_{\text{Focal}} = -\alpha(1 - p_t)^\gamma \cdot \log(p_t) \tag{5}$$

where p_t is the probability of the correct class, $\alpha = 0.25$ controls the weight given to foreground pixels, and $\gamma = 2.0$ is the focusing parameter. The term $(1 - p_t)^\gamma$ reduces the loss contribution from easy pixels that the model is already confident about, allowing it to focus on difficult pixels near boundaries.

3. Model Validation

After each epoch, we ran the model on the validation set to check how well it was learning. During validation, gradient computation was disabled since we were only evaluating, not training. We calculated three metrics on the validation set: Dice score, IoU, and Boundary F1. For example, if the validation Dice score stopped improving for 15 epochs in a row, training was stopped early to prevent overfitting. We saved the model with the best validation Dice score as our final model, which occurred at epoch 16.

4. Model Testing

For testing, we loaded the best model checkpoint and evaluated it on 300 test images from 3 scenes that were never seen during training. The model output a probability map for each image, where each pixel value represents the likelihood of being a cuttlefish. We converted the probability maps to binary masks using a threshold of $p > 0.5$, meaning any pixel with a probability above 0.5 was classified as cuttlefish. Finally, we cleaned up the masks by removing small connected components smaller than 100 pixels and applied morphological operations (closing followed by opening) to smooth out the edges and fill small holes. We also experimented with Otsu thresholding, which automatically selects the threshold based on the image.

C. Implementation and Results

The model was trained on a Lambda Labs cloud instance equipped with an NVIDIA A100-SXM4-40GB GPU. This GPU provides 40GB of memory and supports mixed-precision training, which uses 16-bit floating point numbers to speed up computation, allowing us to efficiently train our model with a batch size of 4 and an image resolution of 512×512. During training, approximately 2.1GB of GPU memory was used, and the entire training process took about 30 minutes to complete.

1. Model Trained Results

The model was trained for 68 epochs before early stopping was triggered, meaning the validation Dice score stopped improving for 15 consecutive epochs. The best model was saved at epoch 53, which achieved a validation Dice score of 0.97. At this epoch, the training loss was very low at 0.0057, and the training Dice score reached 0.9934, indicating that the model learned the training data well.

On the test set, which contained images from scenes the model had never seen, the results were lower than the validation scores. As shown in Table II, the Dice score ranged from 0.12 to 0.98 with a median of 0.78, and the IoU ranged from 0.08 to 0.96 with a median of 0.65. The Boundary F1 score ranged from 0.15 to 0.95 with a median of 0.72, showing that the model had moderate accuracy in predicting the exact edges of the cuttlefish. The large

gap between minimum and maximum scores indicates that the model performed well on some images but struggled significantly on others, likely due to varying levels of cuttlefish camouflage.

The gap between validation and test performance suggests that the model had some difficulty generalizing to new scenes because the test images came from different underwater environments than the training data. However, the post-processing step helped improve the final segmentation quality.

TABLE II: Training and test results

Metric	**Validation (Best)**	**Test Min**	**Test Median**	**Test Max**
Dice	0.97	0.12	0.78	0.98
IoU	0.95	0.08	0.65	0.96
Boundary F1	0.79	0.15	0.72	0.95

2. Model Prediction Results

We evaluated our trained model on two different datasets: the FathomNet test set (which the model had seen similar images during training) and a new unseen dataset with different environmental conditions.

On the FathomNet test set, the model performed reasonably well with a mean Dice score of 0.71 and mean IoU of 0.66 for raw predictions. After post-processing, the scores improved to a mean Dice of 0.76 and mean IoU of 0.73.

We also tested our model on 16 new images that were not from FathomNet. These images featured cuttlefish in bright sandy and pebbly environments, which were quite different from the darker underwater images in our training set. We compared two thresholding methods: fixed threshold ($p > 0.5$) and Otsu's automatic thresholding. The fixed threshold method worked reasonably well on most images. The model successfully detected cuttlefish in images with clear contrast between the animal and background. However, for highly camouflaged cuttlefish on sandy backgrounds, the model often over-segmented, marking large portions of the background as cuttlefish because the probability values were uniformly high across the image. On the other hand, the expected Otsu thresholding to help with over-segmented images by automatically finding a higher threshold. However, the results were mixed. In some cases where the probability map was mostly high values, Otsu selected a very high

threshold (around 0.85–0.95), which resulted in inverted or nearly empty masks. For images with good contrast in the probability map, Otsu performed similarly to fixed thresholding.

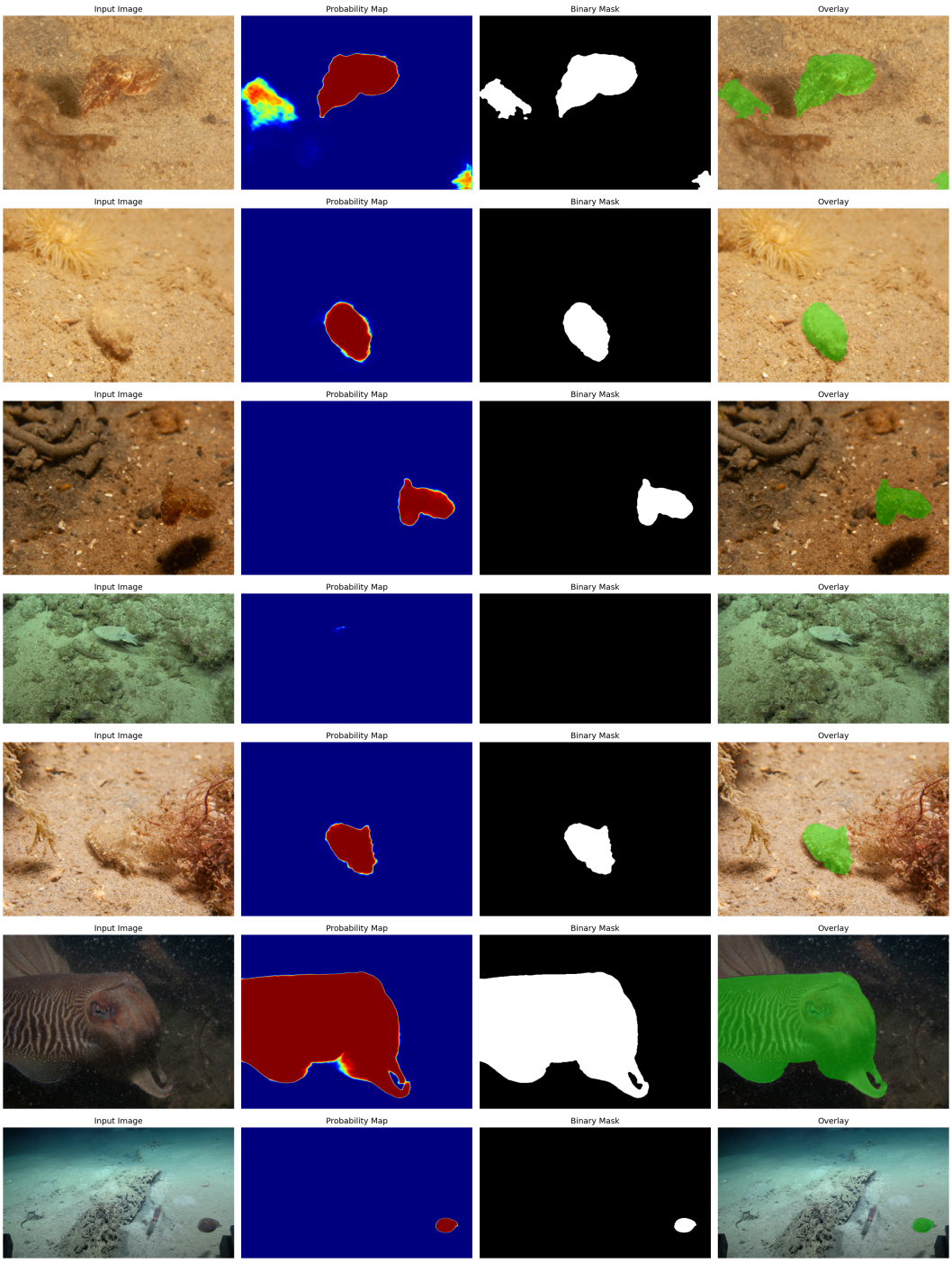

Fig. 19: Examples of the U-Net model based segmentation prediction results on Fathomnet

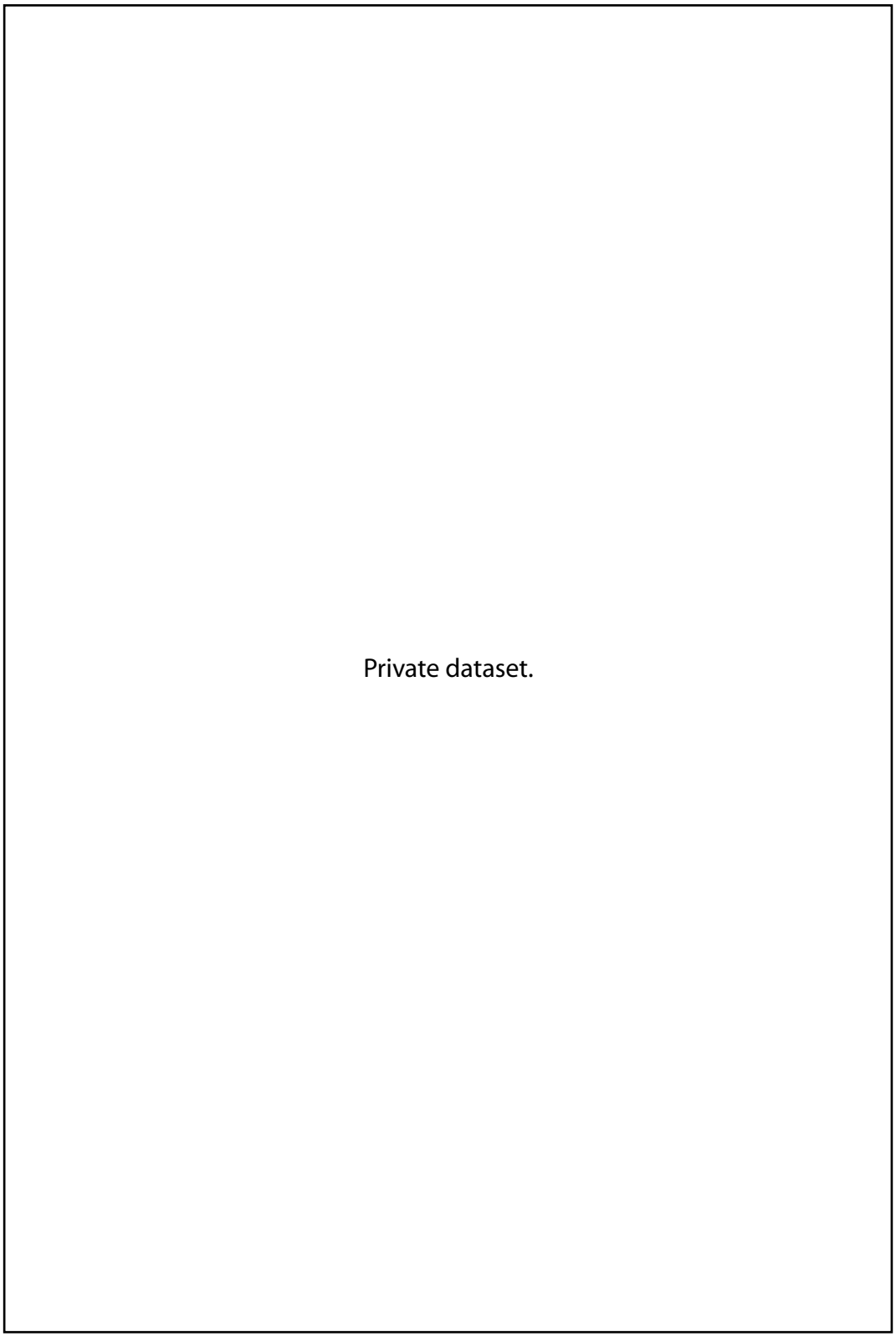

Fig. 20: Examples of the U-Net model based segmentation prediction results on unseen data with fixed threshold (>0.5)

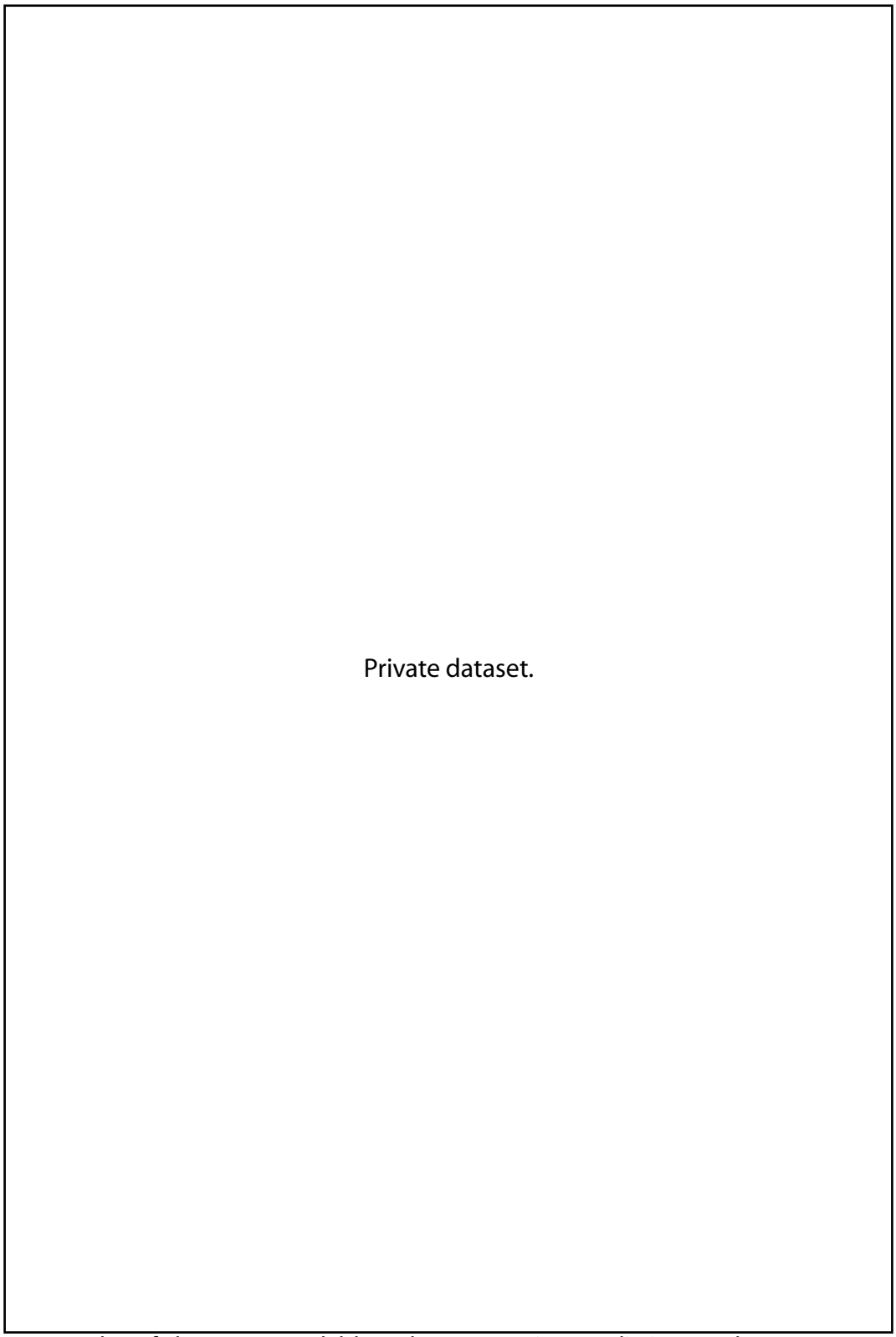

Fig. 21: Examples of the U-Net model based segmentation prediction results on unseen data with fixed threshold (>0.5)

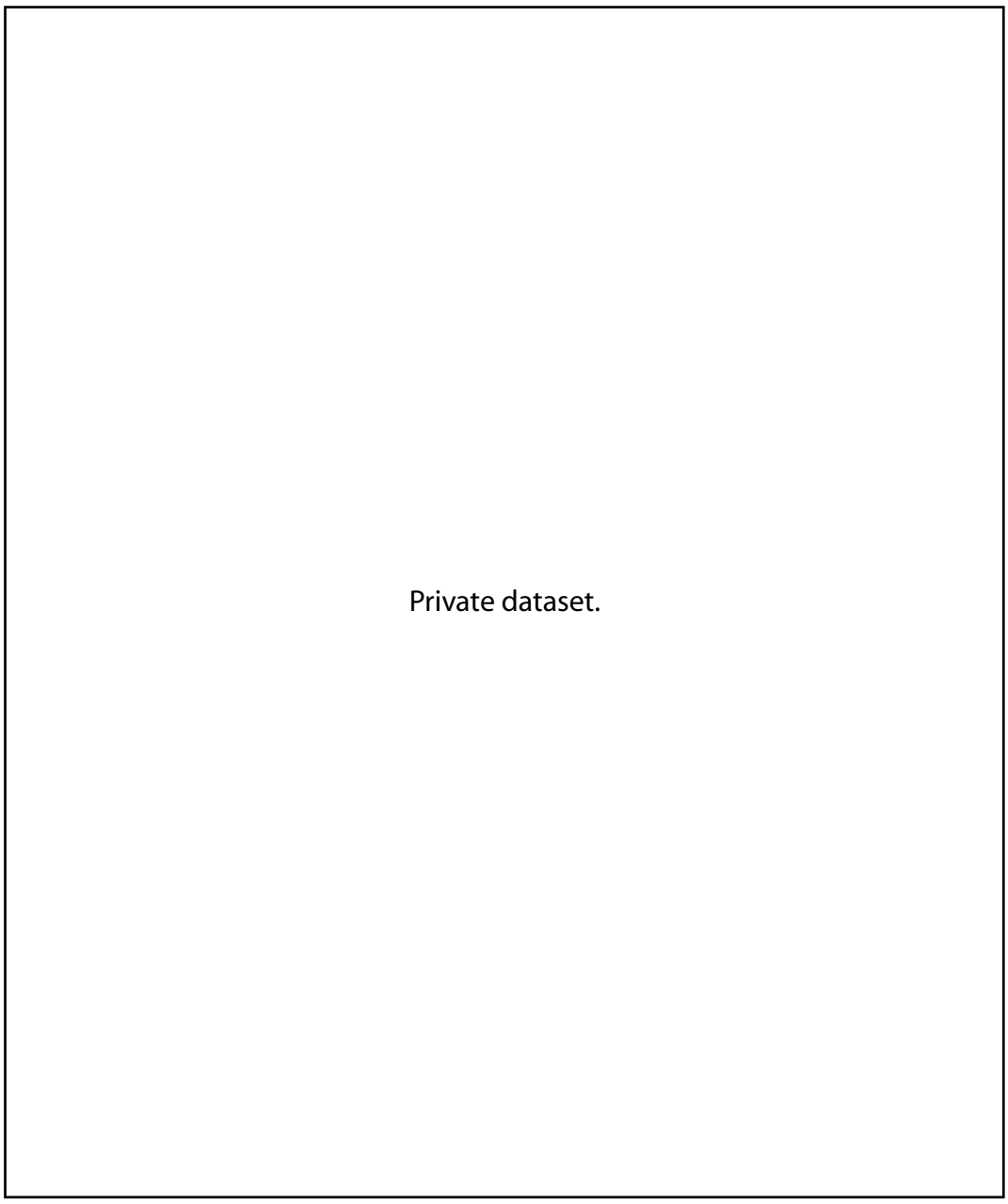

Fig. 22: Examples of the U-Net model based segmentation prediction results on unseen data with fixed and OTSU threshold methods.

IV. Comparisons And Discussions

A. Comparisons of The Two Methods

In general, the classical methods produced acceptable results only after extensive per-image tuning. Even the best cases (IoU 0.7-0.8) required 5-15 minutes of parameter adjustment

and manual cluster selection. Difficult cases with strong lighting or perfect camouflage took over an hour with marginal results (IoU 0.3-0.5). The U-Net model, once trained, processed images in seconds and achieved better results without any manual intervention.

However, the classical pipelines have the advantage of being interpretable (rather than a black-box). We can see exactly why a segmentation failed by looking at the color channels or frequency maps. They also work with minimal computational resources, and also don't need training data, which mattered early on when we had no labeled masks.

On the other hand, U-Net produced consistently smoother masks and more accurate boundaries compared to the classical methods, which showed rough edges, artifacts from morphological operations, and blocky boundaries from windowed processing. It also handled multiple challenging conditions better, such as multi-texture camouflage, strong illumination, and perfect color matching.

The biggest practical difference is generalization. Classical methods needed tuning for every image, making them impractical for large datasets. U-Net worked reasonably well across different scenes from the same domain without any adjustment.

B. Limitations of This Work

For the classic methods, the requirement for per-image parameter tuning is the biggest limitation. We couldn't find a single parameter set that worked across more than 2-3 similar images. This makes the classical approach impractical for automated processing. In addition, the interactive cluster and channel selection steps, while improving results, defeat the initial purpose of automation.

While our model performed well on the validation set, there are several limitations to consider. Our dataset only contained 50 original images from 5 scenes, and although we augmented them to 2,469 images, the model was still exposed to limited variety in backgrounds and lighting conditions. We also tested on 16 new images without ground truth masks, so we could only evaluate them visually without computing actual metrics. These new images had bright sandy backgrounds, which differed significantly from the darker underwater training images, causing the model to over-segment or miss camouflaged cuttlefish. Additionally, the test results showed high variance, with the model performing well

on some images but poorly on others. Finally, cuttlefish are naturally skilled at camouflage, making them inherently difficult to segment even for a well-trained model.

V. Future Work

A. Future Work

Regarding pipelines 1-3, bridging the gap between some user input and no user input could be beneficial and would enable a fully automated pipeline. To address the limitations in the U-Net model based method, future work could focus on collecting a larger and more diverse dataset that includes images and corresponding bounding boxes (or masks) from different environments such as sandy, pebbly, and bright backgrounds, not just underwater scenes. Additionally, creating ground truth masks for the unseen dataset would allow for proper quantitative evaluation, and exploring domain adaptation techniques could help the model generalize better across different imaging conditions.

References

[1] R. Azad et al., *Medical Image Segmentation Review: The Success of U-Net* in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 12, pp. 10076-10095, Dec. 2024, doi: 10.1109/TPAMI.2024.3435571.

[2] Fathomnet Database, *Fathomnet*, https://fathomnet.org/.

[3] Alexander Kirillov and Eric Mintun and Nikhila Ravi and Hanzi Mao and Chloe Rolland and Laura Gustafson and Tete Xiao and Spencer Whitehead and Alexander C. Berg and Wan-Yen Lo and Piotr Dollár and Ross Girshick, *Segment Anything*, https://arxiv.org/abs/2304.02643.

[4] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, *Deep Residual Learning for Image Recognition*, 2015, https://arxiv.org/abs/1512.03385.

Vita

Carlos A. is a M.S. in Robotics with a Computer Science concentration at Northeastern University. He is a research assistant at the Field Robotics Lab under Professor Hanumant Singh. His future goal is to work in space exploration, specifically in the sensing portion of space robots. He is currently focusing on the hardware portion of robotics, but is equally interested on the software side.

Li Wang is currently a graduate student in the Department of Electrical and Computer Engineering at Northeastern University, where she is pursuing her M.S. degree. Her research interests include image processing, computer vision, and machine learning, with a focus on applying deep learning techniques to real-world problems.