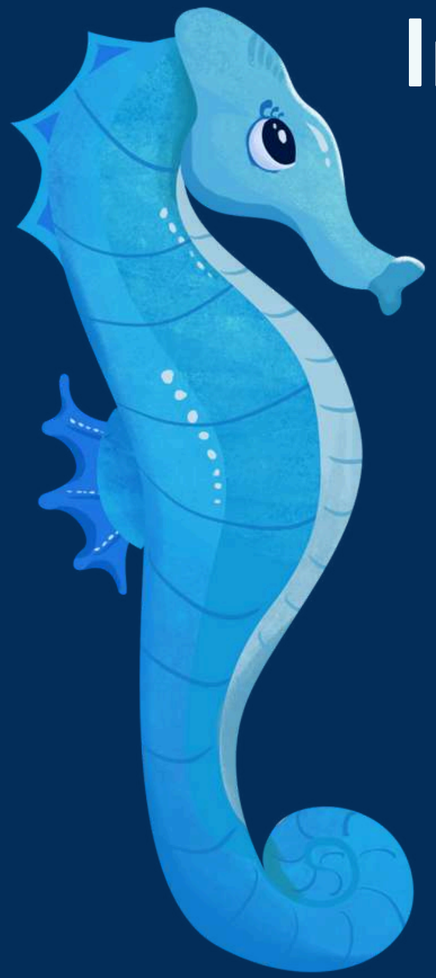
ANÁLISE E MONITORAMENTO DA BIODIVERSIDADE EM ECOSSISTEMAS MARINHOS

por Pedro Henrique F.F.

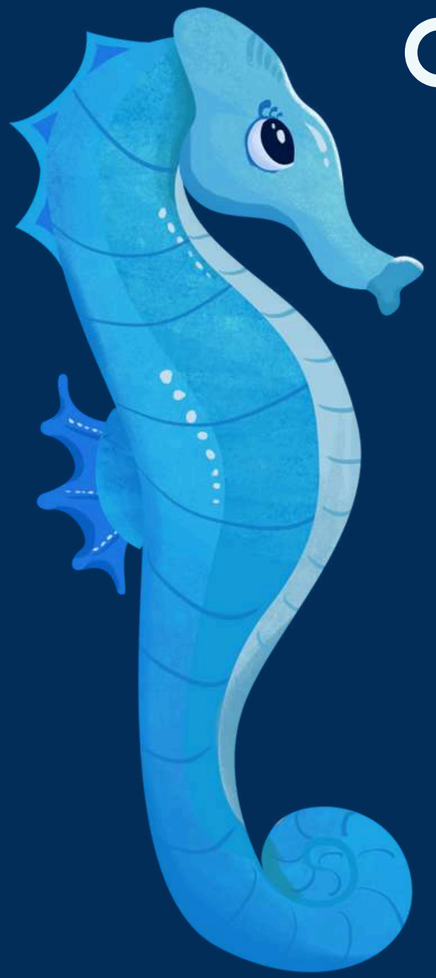
Sumário



Introdução

- Ao longo dos anos, os **ecossistemas marinhos** vem sofrendo com diversas ameaças, como poluição, mudanças climáticas e sobrepesca.
- **Monitorar a biodiversidade** nesses ambientes pode contribuir para a conservação, mas muitas vezes isso é feito de maneira manual e ineficiente.

Atividade petrolífera offshore e sua relação com os impactos ambientais nos ecossistemas marinhos. **[Nascimento et al., 2021]**



Objetivo

- O uso de câmeras subaquáticas e **técnicas de processamento de imagens** pode ajudar a automatizar a identificação e contagem de espécies, contribuindo para subsequente estudos de ecologia e conservação.
- O objetivo desse trabalho era desenvolver um método para realizar a **segmentação, identificação e contagem** de animais marinhos em imagens de ecossistemas subaquáticos, de preferência imagens com baixa iluminação.

Trabalhos Relacionados

Trabalho 01

Tracking Fish Abundance by
Underwater Image Recognition **[Marini et at., 2018]**

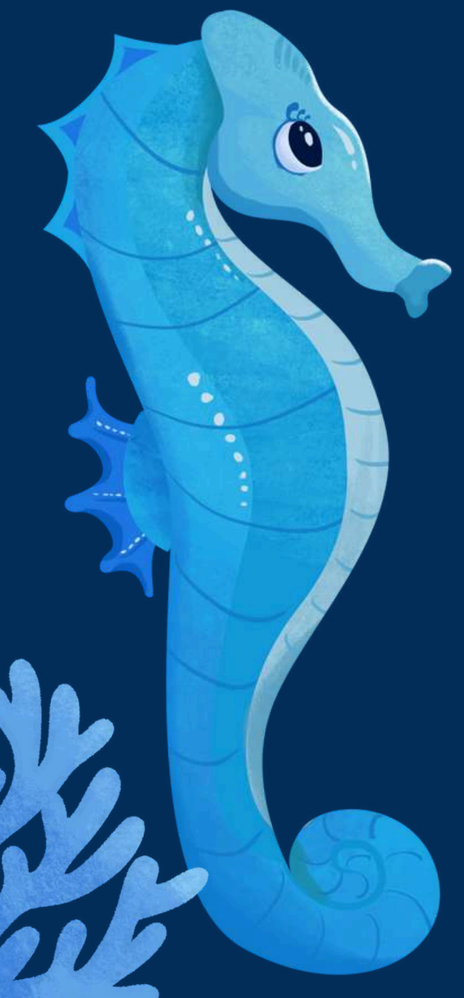
Trabalho 02

Underwater image enhancement: a comprehensive review, recent trends, challenges and applications **[Raveendran et at., 2021]**

Trabalho 03

Underwater image processing and analysis:
A review **[Jian et al., 2021]**

Tracking Fish Abundance by Underwater Image Recognition [Marini et at., 2018]



- O objetivo era capturar a dinâmica temporal da abundância de peixes. Foram processadas mais de 20.000 imagens que foram adquiridas em um cenário costeiro desafiador do mundo real no local de testes OBSEA-EMSO. As imagens foram coletadas de 30 em 30 minutos, continuamente durante dois anos. As condições ambientais altamente variáveis nos permitiram testar a eficácia de nossa abordagem sob mudanças na radiação luminosa, turbidez da água, confusão de fundo e crescimento de bioincrustação na caixa da câmera.
- Os resultados do reconhecimento automatizado foram altamente correlacionados com as contagens manuais e foram altamente confiáveis quando usados para rastrear variações do peixe em diferentes escalas de tempo horárias, diárias e mensais. Além disso, essa metodologia poderia ser facilmente transferida para outros vídeo-observatórios cabeados.

METODOLOGIA

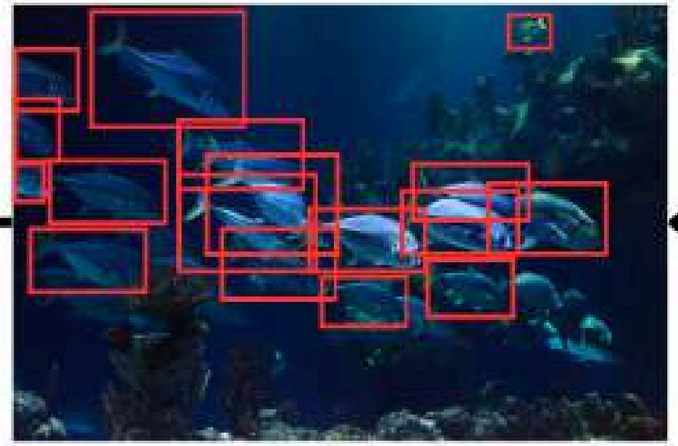
Método Proposto

Base de Imagens

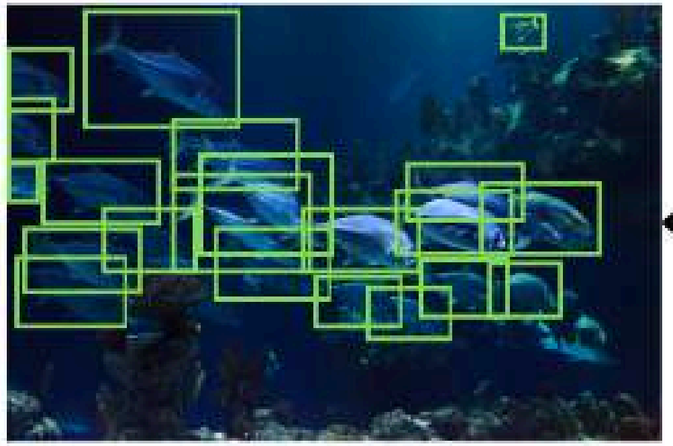
Pré-Processamento

Segmentação

Extração de Regiões



Definição das Bounding Boxes



Avaliação de Desempenho

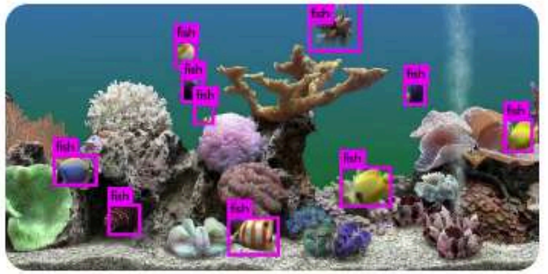
Search

SLAVKO PRYTULA · UPDATED 3 YEARS AGO

39 | New Notebook | Download

Underwater Object Detection Dataset

Yolov5 PyTorch format underwater life dataset for object detection



Data Card | Code (21) | Discussion (0) | Suggestions (1)

About Dataset

Info

The dataset contains 7 classes of underwater creatures with provided bboxes locations for every animal.

The dataset is already split into the train, validation, and test sets.

Data

It includes 638 images.

- Creatures are annotated in YOLO v5 PyTorch format

Pre-Processing

The following pre-processing was applied to each image:

Usability

7.50

License

CC0: Public Domain

Expected update frequency

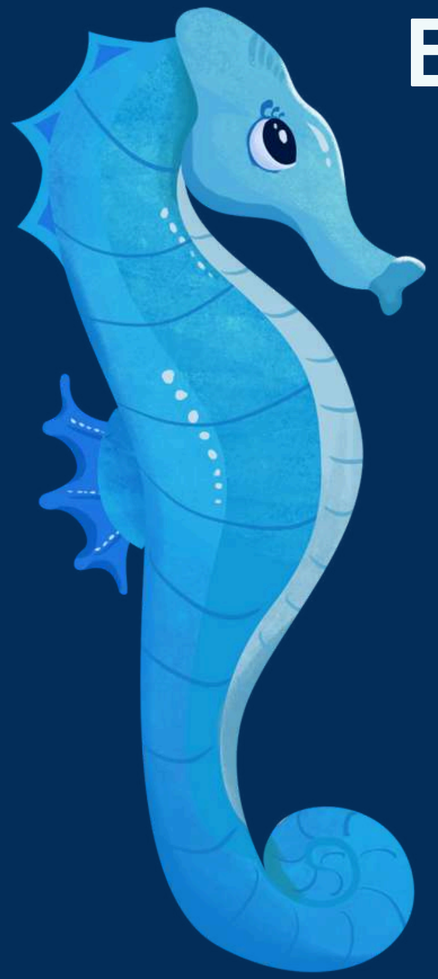
Not specified

Tags

Earth and Nature

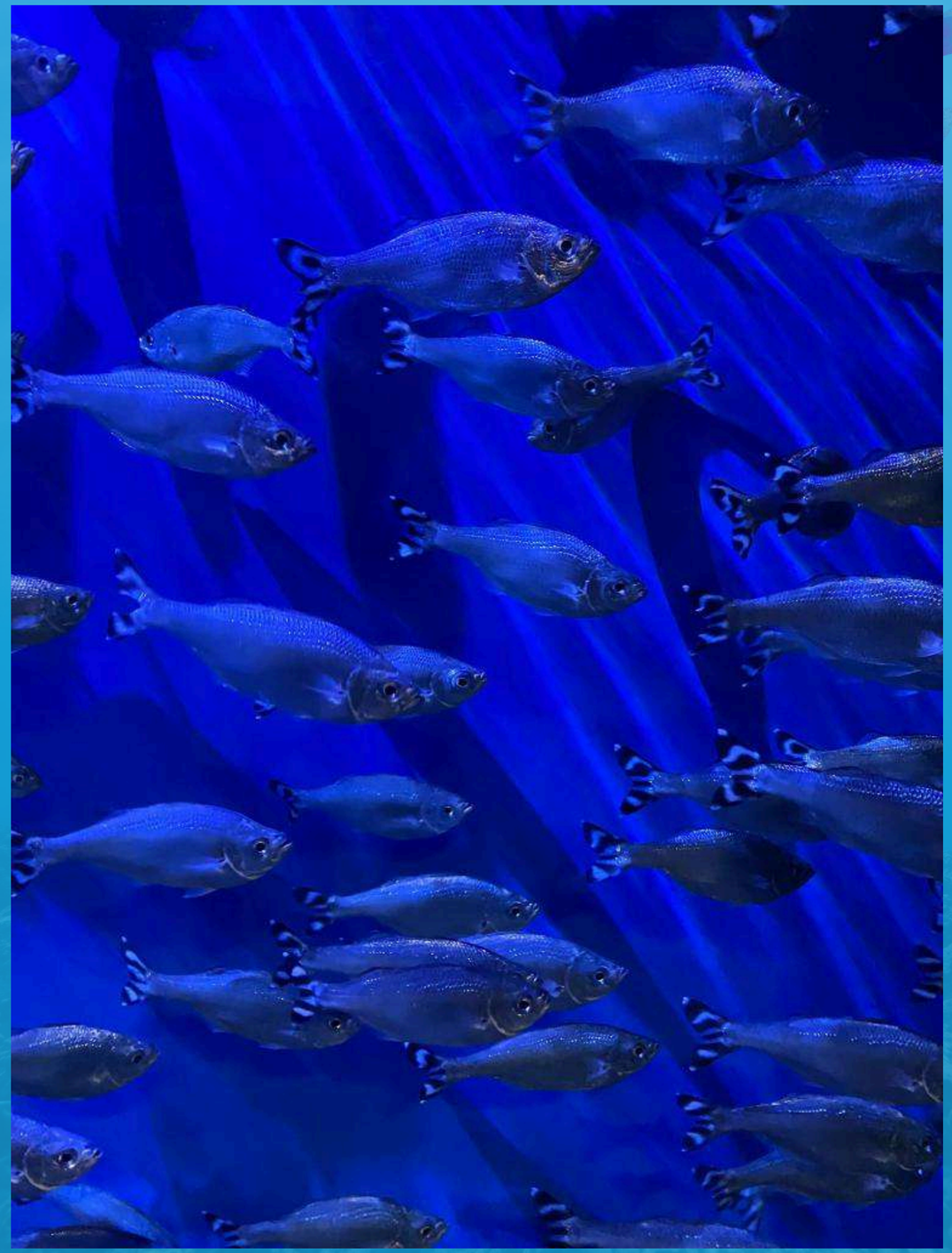
Arts and Entertainment

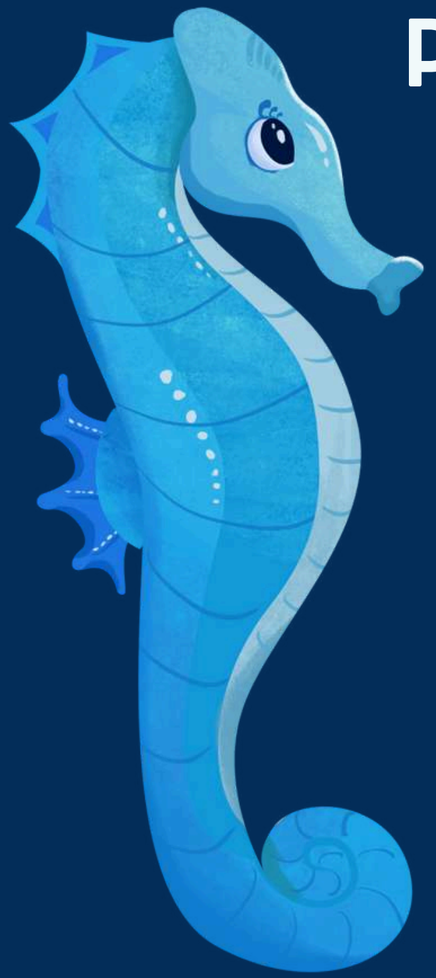
Animals | PyTorch



Base de Dados

- A base de dados usada para validar o método proposto foi a "Underwater Object Detection Dataset" do usuário Slavko Prytula encontrada no site Kaggle.
- Ela inclui 638 imagens de diferentes animais marinhos, todas em formato JPG, com o conjunto de dados já dividido em conjuntos de treinamento, validação e teste (448, 127 e 63 imagens respectivamente).
- Ela também inclui arquivos contendo as localizações das bounding boxes de cada animal para todas as imagens.





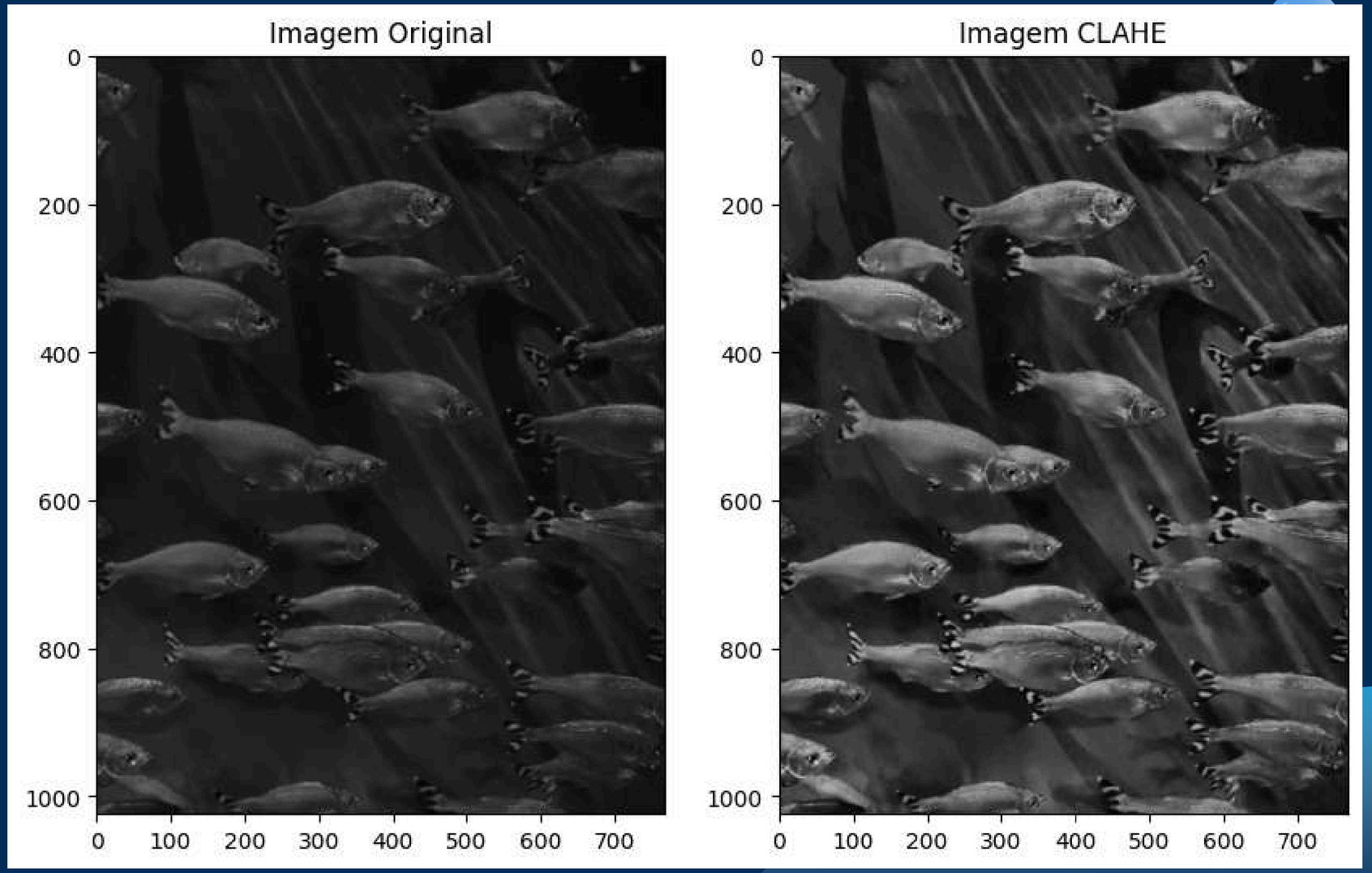
Pré-Processamento

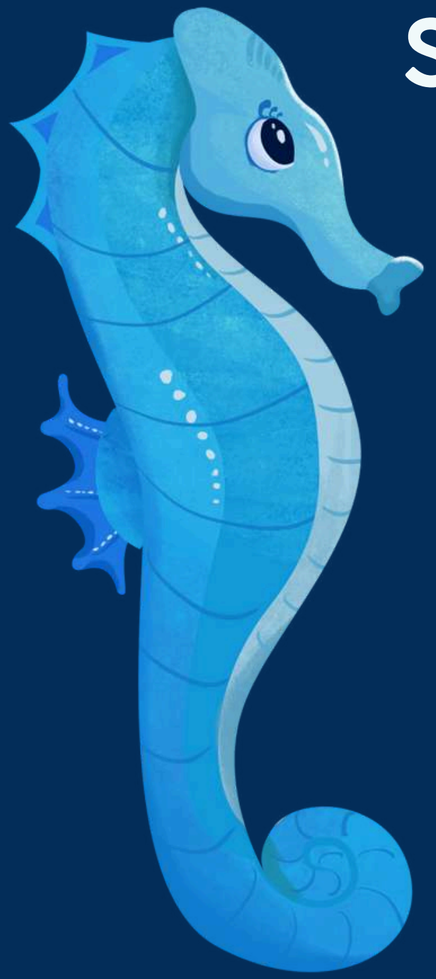
- Conversão para Escala de Cinza
- Desfoque Gaussiano
- Melhoramento de CLAHE (Contrast Limited Adaptive Histogram Equalization)

Underwater Image Segmentation using CLAHE Enhancement and Thresholding **[Rai et al., 2012]**

- É um algoritmo de equalização de histograma que particiona a imagem em regiões contextuais e aplica a equalização do histograma a cada um. Isso equilibra a distribuição dos valores de cinza usados e, assim, faz características ocultas da imagem mais visíveis.

```python
def apply_clahe(img, clahe):
    blured = cv.GaussianBlur(img,(3,3),0,borderType=cv.BORDER_REPLICATE)
    cl1 = clahe.apply(blured)
    return cl1
```

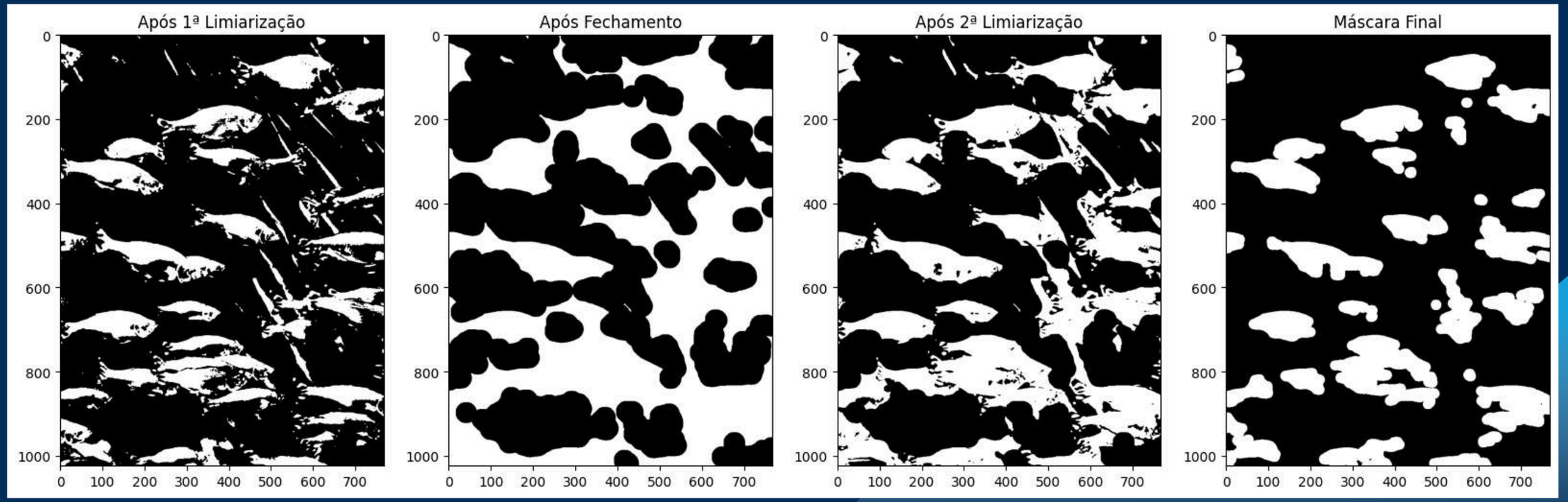


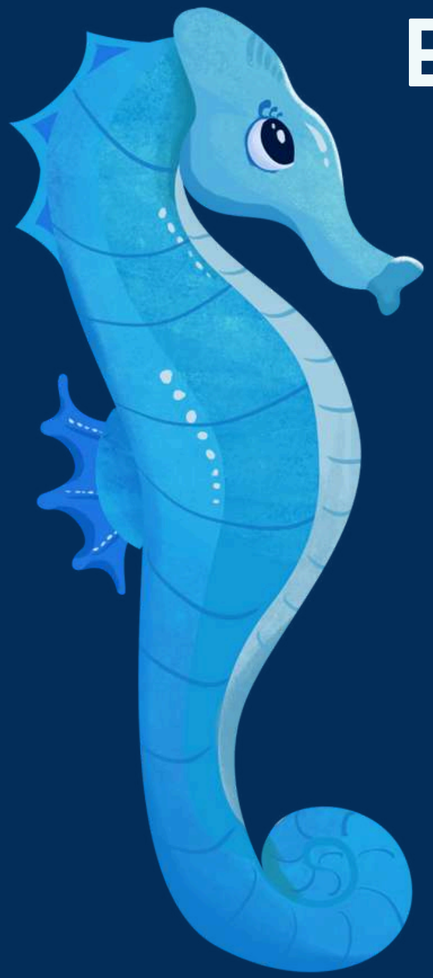


Segmentação

- Limiarização (Otsu)
- Transformação Morfológica
- Segunda Passagem

```python
def gen_mask(img):
    blured1 = cv.GaussianBlur(img,(3,3),0,borderType=cv.BORDER_REPLICATE)
    ret,mask1 = cv.threshold(blured1,0,255,cv.THRESH_OTSU)
    kernel1 = cv.getStructuringElement(cv.MORPH_ELLIPSE,(50,50))
    closed = cv.morphologyEx(mask1, cv.MORPH_CLOSE, kernel1)
    combined = cv.bitwise_and(blured1,blured1,mask = closed)
    blured2 = cv.GaussianBlur(combined,(3,3),0,borderType=cv.BORDER_REPLICATE)
    ret,mask2 = cv.threshold(blured2,0,255,cv.THRESH_OTSU)
    kernel2 = cv.getStructuringElement(cv.MORPH_ELLIPSE,(20,5))
    kernel3 = cv.getStructuringElement(cv.MORPH_ELLIPSE,(25,25))
    mask3 = cv.morphologyEx(mask2, cv.MORPH_OPEN, kernel2)
    final_mask = cv.morphologyEx(mask3, cv.MORPH_OPEN, kernel3)
    return final_mask
```



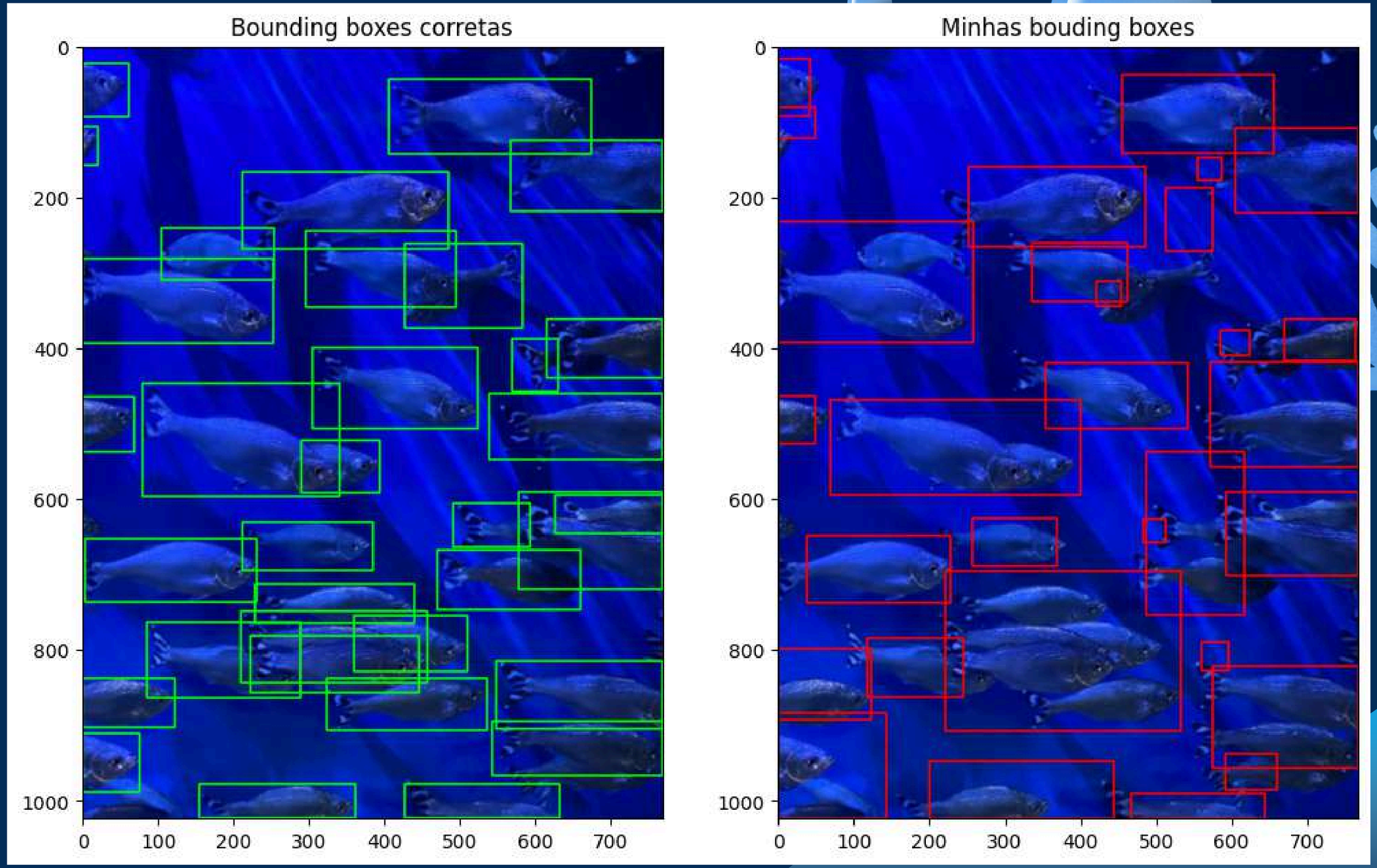


Extração de Regiões

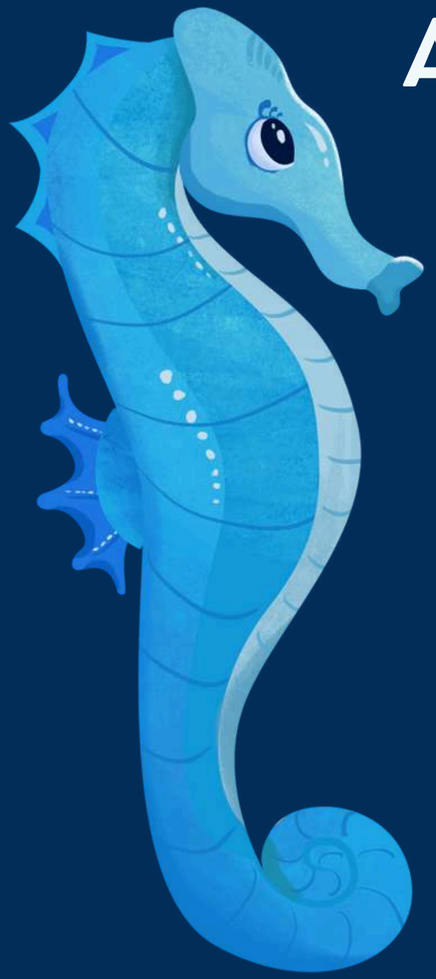
- Bounding Boxes
- Expansão

```python
def get_bb_lists(img, mask, path, expand):
    h, w, _ = img.shape
    box_list_right = []
    box_list_found = []
    with open(path, "r") as file1:
        for line in file1.readlines():
            class_id, x_center, y_center, width, height = map(float, line.split())
            x_min = int((x_center - width / 2) * w)
            y_min = int((y_center - height / 2) * h)
            x_max = int((x_center + width / 2) * w)
            y_max = int((y_center + height / 2) * h)
            box = (x_min, y_min, x_max, y_max)
            box_list_right.append(box)
    lbl_0 = label(mask)
    props = regionprops(lbl_0)
    box_list_found = []
    for prop in props:
        box = (prop.bbox[1], prop.bbox[0], prop.bbox[3], prop.bbox[2])
        box = aumentar_bounding_box(mask, box, expand)
        box_list_found.append(box)
    return box_list_found, box_list_right
```

```python
def aumentar_bounding_box(imagem, bbox, fator_aumento):
    altura_imagem, largura_imagem = imagem.shape
    x1, y1, x2, y2 = bbox
    cx, cy = (x1 + x2) / 2, (y1 + y2) / 2
    largura = x2 - x1
    altura = y2 - y1
    nova_largura = largura * fator_aumento
    nova_altura = altura * fator_aumento
    novo_x1 = int(max(0, cx - nova_largura / 2))
    novo_y1 = int(max(0, cy - nova_altura / 2))
    novo_x2 = int(min(largura_imagem, cx + nova_largura / 2))
    novo_y2 = int(min(altura_imagem, cy + nova_altura / 2))
    return novo_x1, novo_y1, novo_x2, novo_y2
```



fator_aumento = 1.2



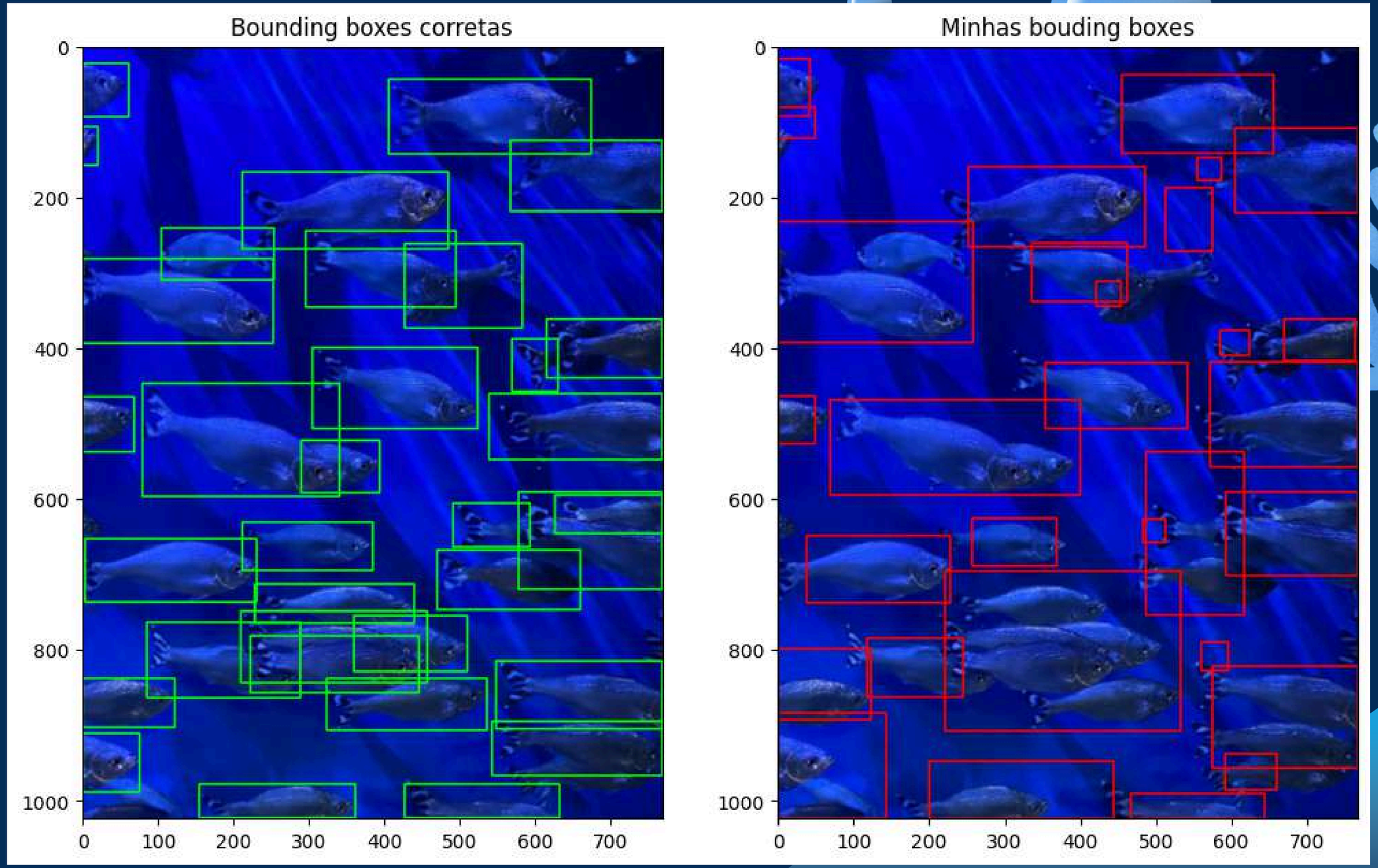
Avaliação de Desempenho

- Intersection over Union (IoU)

- Para cada imagem, calculamos o IoU das caixas delimitadoras similares entre as obtidas pelo método e as verdadeiras (Usando KNN).
- Fizemos então a média dos valores de IoU calculados a partir de todas as caixas encontradas pelo método.
- E por fim, multiplicamos essa média por um modificador que começa em 1 e diminui a medida que o número de caixas obtida se afasta do número de caixas real.

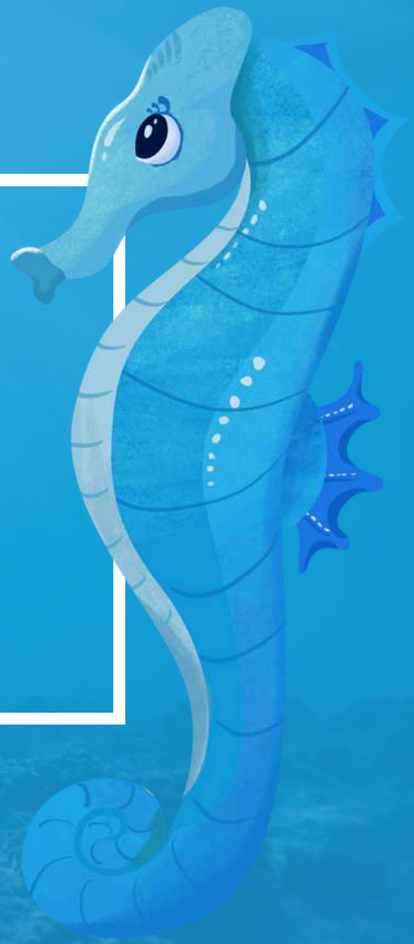
```python
def measure_accuracy(box_list_found, box_list_right):
    box_list_found = np.array(box_list_found)
    box_list_right = np.array(box_list_right)
    lf_size, lr_size = len(box_list_found), len(box_list_right)
    if len(box_list_right) > 0:
        classes = np.arange(len(box_list_right))
        knn = KNeighborsClassifier(n_neighbors=1)
        knn.fit(box_list_right, y=classes)
        matches = knn.predict(box_list_found)
        total = 0
        for i in range(lf_size):
            j = matches[i]
            current = get_iou(box_list_found[i], box_list_right[j])
            total += current/lf_size
    else:
        total = 1
    mod = 1-abs(lr_size-lf_size)*0.01
    total *= mod
    print("Acurácia para essa imagem foi de: {:.2f}%".format(total*100))
    return total
```

```python
def get_iou(bb1, bb2):
    assert bb1[0] <= bb1[2]
    assert bb1[1] <= bb1[3]
    assert bb2[0] <= bb2[2]
    assert bb2[1] <= bb2[3]
    x_left = max(bb1[0], bb2[0])
    y_top = max(bb1[1], bb2[1])
    x_right = min(bb1[2], bb2[2])
    y_bottom = min(bb1[3], bb2[3])
    if x_right < x_left or y_bottom < y_top: return 0.0
    intersection_area = (x_right - x_left + 1) * (y_bottom - y_top + 1)
    bb1_area = (bb1[2] - bb1[0] + 1) * (bb1[3] - bb1[1] + 1)
    bb2_area = (bb2[2] - bb2[0] + 1) * (bb2[3] - bb2[1] + 1)
    iou = intersection_area / float(bb1_area + bb2_area - intersection_area)
    assert iou >= 0.0
    assert iou <= 1.0
    return iou
```



Acurácia para essa imagem foi de: 41.57%

RESULTADOS



```python
def main(folder, expand):
    if folder == 'train': image_dir, label_dir = TRAIN_IMAGES, TRAIN_LABELS
    elif folder == 'valid': image_dir, label_dir = VAL_IMAGES, VAL_LABELS
    elif folder == 'test': image_dir, label_dir = TEST_IMAGES, TEST_LABELS
    elif folder == 'new': image_dir, label_dir = NEW_IMAGES, NEW_LABELS
    image_files = sorted(os.listdir(image_dir))
    number_img = len(image_files)
    total_accuracy = 0
    for image_file in image_files:
        img_path = os.path.join(image_dir, image_file)
        label_path = os.path.join(label_dir, image_file[:-4] + '.txt')
        img = cv.imread(img_path)
        img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
        gray = cv.cvtColor(img, cv.COLOR_RGB2GRAY)
        enhanced = apply_clahe(gray, clahe)
        mask = gen_mask(enhanced)
        box_list_found, box_list_right = get_bb_lists(img, mask, label_path, expand)
        accuracy = measure_accuracy(box_list_found, box_list_right)
        total_accuracy += accuracy/number_img
    print("Acurácia para o conjunto de imagens {} foi de: {:.2f}%".format(folder, total_accuracy*100))
```

Resultados

Tabela de Acurácia				
expand	train	valid	test	new
1	7,00%	6,10%	8,99%	12,21%
1.2	7,53%	6,53%	9,39%	12,99%
1.5	7,37%	6,42%	8,95%	12,30%
2	6,78%	6,04%	8,02%	10,60%

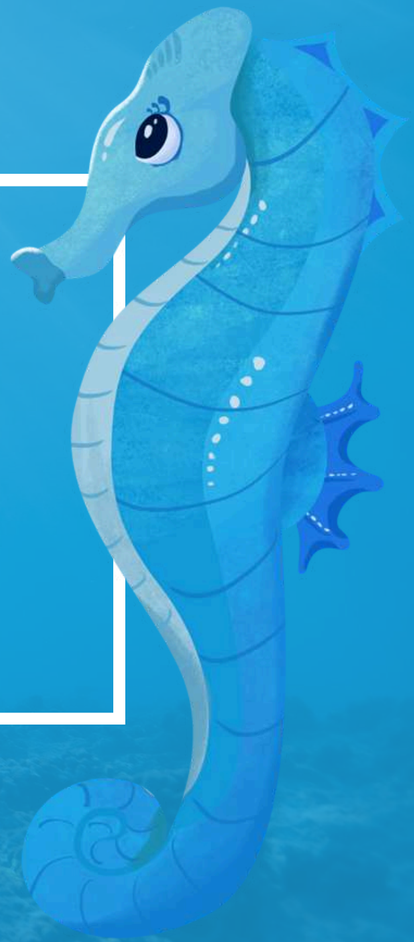
Resultados

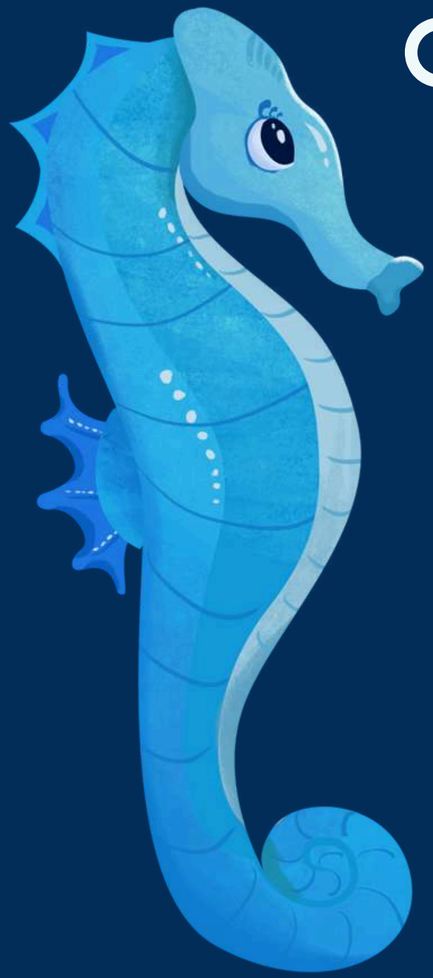
Validation Performance	Accuracy (std)	92% (0.02)
	True Positive Rate (std)	95% (0.03)
	False Positive Rate (std)	12% (0.04)

Table 1. Summary of the data acquired in the year 2012 that was used to train and validate the binary image classifier.

Tracking Fish Abundance by Underwater Image Recognition [Marini et at., 2018]

CONCLUSÃO





Conclusão

- Subestimamos o problema.
- CLAHE é um algoritmo bom para melhora de contraste.
- A variância da base de imagens atrapalhou o resultado.
- Apesar de tudo, o resultado foi aceitável.

OBRIGADO!

Aplausos!