UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

BRUNO SIGNORI WUSTRO

MONITORAMENTO DE PEIXES-ZEBRA: RASTREAMENTO A PARTIR DE VÍDEOS

PATO BRANCO

2023

BRUNO SIGNORI WUSTRO

MONITORAMENTO DE PEIXES-ZEBRA: RASTREAMENTO A PARTIR DE VÍDEOS

Zebrafish Monitoring: Tracking from Videos

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Dalcimar Casanova

PATO BRANCO

2023

BRUNO SIGNORI WUSTRO

MONITORAMENTO DE PEIXES-ZEBRA: RASTREAMENTO A PARTIR DE VÍDEOS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 20/junho/2023

Dalcimar Casanova
Doutorado em Física Computacional
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Fábio Favarim
Doutorado em Engenharia Elétrica
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Anderson Luiz Fernandes
Mestre em Engenharia Elétrica, processamento paralelo e processamento de sinais
Centro Universitário Mater Dei (UNIMATER) - Pato Branco

PATO BRANCO
2023

RESUMO

Este estudo enfoca o desenvolvimento de um sistema de rastreamento robusto para peixes-zebra (*Danio rerio*), espécie amplamente utilizada em pesquisas científicas, aplicando técnicas de visão computacional, como o detector de objetos YOLO. Abordando as limitações de métodos tradicionais e a necessidade de controles rígidos do ambiente nos sistemas automáticos existentes. O trabalho propõe uma solução mais versátil e menos dependente de condições ambientais. Para a criação do sistema, uma análise aprofundada de literatura foi realizada, discutindo diversos métodos de rastreamento e a evolução das redes neurais convolucionais. A metodologia empregada envolveu a coleta de vídeos de peixes-zebra em variados ambientes, a rotulação manual das posições dos peixes, o treinamento da base do YOLOv4 e a aplicação de *data augmentation* para aprimorar a robustez do modelo. Os resultados indicam que a rotulação correta e a aplicação de técnicas de *data augmentation* melhoram a acurácia do modelo.

Palavras-chave: rastreamento de peixes-zebra; detector de objetos yolo; redes neurais convolucionais; *data augmentation*.

ABSTRACT

This study focuses on the development of a robust zebrafish (*Danio rerio*) tracking system, a species widely used in scientific research, by applying computer vision techniques, such as the YOLO object detector. It addresses the limitations of traditional methods and the need for strict environmental controls in existing automatic systems. The work proposes a more versatile solution, less dependent on environmental conditions. For the system creation, an in-depth literature review was conducted, discussing various tracking methods and the evolution of convolutional neural networks. The employed methodology involved collecting videos of zebrafish in various environments, manually labeling the fish positions, training the YOLOv4 base, and applying data augmentation to enhance the model's robustness. The results indicate that accurate labeling and the application of data augmentation techniques improve the model's accuracy.

Keywords: zebrafish tracking; yolo object detector; convolutional neural network; data augmentation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

ACDR	*Average Correct Detection Rate*
ADE	*Average Direction Error*
AEDR	*Average Error Detection Rate*
AFMM	*Augmented Fast Marching Method*
AODR	*Average Occlusion Detection Rate*
AP	*Average Precision*
ASPP	*Atrous Spatial Pyramid Pooling*
BB	*Bounding Box*
BoF	*Bag of Freebies*
BoS	*Bag of Specials*
CA	*Classification Accuracy*
CIR	*Correct Tracking Reason*
CNN	*Convolutional Neural Network*
CTR	*Correct Tracking Rate Index*
FPN	*Feature Pyramid Network*
FPS	*Frames Per Second*
GAN	*Generative Adversarial Network*
HOG	*Histogram of Oriented Gradient*
IA	*Identifying Accuracy*
IoU	*Intersection over Union*
mAP	*mean Average Precision*
MSE	*Mean Square Error*

PAN	*Path Aggregation Network*
PCA	*Principal Component Analysis*
RFB	*Receptive Field Block*
SPP	*Spatial Pyramid Pooling*
SVM	*Suport Vector Machine*
YOLO	*You Only Look Once*
YOLOv2	*You Only Look Once*
YOLOv4	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

Estima-se que, por ano, algumas centenas de milhões de animais são usados para fins científicos no mundo (TAYLOR; ALVAREZ, 2019). Em particular, o peixe-zebra (*Danio rerio*) é um organismo de grande relevância em pesquisas médicas e biológicas (DEAKIN *et al.*, 2019). Sua alta taxa de fecundidade e fertilidade permite que este peixe produza até duzentos embriões a cada sete dias (TAVARES; LOPES, 2013). Por esta razão, ele é amplamente utilizado como um modelo de estudo, especialmente em triagens de drogas e genes, experimentos biológicos e análises neurotoxicológicas (BAI *et al.*, 2018). Os espécimes são tipicamente submetidos a algum processo invasivo ou expostos a uma substância química, depois são colocados em um aquário, geralmente sozinhos, para permitir a recuperação e a cura. A seguir, são observadas as reações motoras dos indivíduos para estudo durante algumas horas. O comportamento dos animais pode variar bastante, dependendo do ambiente e dos procedimentos realizados, podendo ocorrer mudanças comportamentais sutis, como no início de uma experiência negativa, por exemplo (DEAKIN *et al.*, 2019), ou até mesmo movimentos bastante rápidos, como é o caso dos peixes na presença de alimento (BARREIROS *et al.*, 2021). A Figura 1 apresenta exemplos das características observáveis comuns do animal.

Devido à complexidade comportamental dos espécimes, é muito importante obter dados de rastreamento precisos para as análises (BAI *et al.*, 2018). Tradicionalmente, essas informações eram coletadas integralmente de forma manual, o que era extremamente impreciso, suscetível a erros e entediante, já que o pesquisador tinha que estimar a posição e velocidade do peixe, percorrendo quadro a quadro de vídeos, ou até mesmo a olho nu. Nesse contexto, com o intuito de melhorar e agilizar esse processo, cientistas propuseram diversos sistemas automáticos de rastreamento, e alguns já se tornaram bastante populares (MAASWINKEL; ZHU; WENG, 2013). Diversos métodos utilizando vídeos como fonte de dados foram propostos para realizar o rastreamento de peixes-zebra, porém a maioria, se não todos os sistemas propostos, apresentam pelo menos um tipo de controle no *habitat* artificial do animal, que favorece a detecção do objeto.

(a) Exemplo a **(b) Exemplo b**

Figura 1 – Fenótipo comum do peixe-zebra
Fonte: Autoria própria.

Embora a alteração do ambiente facilite o rastreamento, é evidente que um sistema capaz de realizar essa tarefa sem tais controles seria altamente útil. Isso permitiria a realização de pesquisas de maneira mais acessível, já que normalmente é necessária uma infraestrutura adicional para implementar os controles mais comumente utilizados. Tais práticas são exemplificadas a seguir.

É comum a prática de impedir que ocorram reflexos dos peixes, que potencialmente possam ser detectados como falsos positivos, tornando as paredes do aquário opacas (BARREIROS *et al.*, 2021) ou até mesmo reduzindo a altura do nível da água (BAI *et al.*, 2018). No entanto, essa última tem a desvantagem de limitar o espaço de movimentação dos alvos. Outra estratégia utilizada, é a de posicionar uma fonte de luz artificial embaixo do aquário com o objetivo de tornar o fundo da imagem uniforme e destacar os objetos, facilitando a detecção destes (BARREIROS *et al.*, 2021). Dessa maneira, é possível também minimizar prováveis reflexos na superfície da água que eventualmente ocultam os peixes. Apesar da eficácia dessa solução, manter o ambiente nestas condições pode ser difícil e, além disso, há indícios de que a exposição a algumas fontes de luz pode afetar o crescimento, desenvolvimento e locomoção de embriões do peixe-zebra (ÜSTÜNDAĞ *et al.*, 2019; VILLAMIZAR *et al.*, 2014). A consistência das características do ambiente e dos equipamentos (BAI *et al.*, 2018; BARREIROS *et al.*, 2021; MAASWINKEL; ZHU; WENG, 2013; DELCOURT *et al.*, 2011) também facilita a resolução do problema. Dados do mesmo aquário, filmados na mesma posição, ângulo e condições de iluminação tornam os dados mais homogêneos, favorecendo os métodos propostos em relação aos seus respectivos materiais.

Em consequência do ambiente ser bastante controlado, torna-se difícil desenvolver uma ferramenta de rastreamento que seja confiável em situações variadas. Outros pesquisadores podem não ter acesso aos materiais ou equipamentos necessários para reproduzir o ambiente onde o sistema rastreador é preciso e coeso. Neste contexto, seria útil superar algumas das limitações anteriormente mencionadas. Em particular, é desejável que fatores como as dimensões do ambiente, a presença de reflexos, oclusões e artefatos no aquário não afetem consideravelmente a qualidade do rastreamento. A questão chave neste trabalho é: diante do avanço da tecnologia de visão computacional, será possível desenvolver um sistema de rastreamento para o peixe-zebra, que seja menos dependente das condições do ambiente?

1.1 OBJETIVOS

O principal objetivo deste trabalho foi o desenvolvimento de um sistema de rastreamento para peixes-zebra, que seja robusto e menos dependente das condições ambientais específicas dos aquários. Isso foi alcançado através das seguintes metas específicas:

- Aplicar técnicas de visão computacional, particularmente o detector de objetos *You Only Look Once* (YOLO), para o rastreamento de peixes-zebra;

- Criar uma base de dados para treinar e avaliar o modelo detector, rotulando manualmente algumas milhares de amostras do animal;
- Avaliar a capacidade do sistema de operar em diferentes ambientes, sem a necessidade de controles estritos que limitam a utilidade dos sistemas de rastreamento atuais;
- Realizar testes para garantir a robustez e a precisão do sistema proposto.

Neste contexto, a expectativa é que o sistema proposto contribua para a acessibilidade e praticidade do rastreamento de peixes-zebra em diferentes cenários de pesquisa.

2 REFERENCIAL TEÓRICO

Para este trabalho, foi realizada a pesquisa de soluções relacionadas ao rastreamento do peixe-zebra, propostas por diversos pesquisadores, utilizando algum algoritmo de visão computacional. Foram exploradas as técnicas desenvolvidas e aplicadas, as funcionalidades, assim como suas limitações. Em particular, foi de interesse os controles de ambiente empregados para melhorar a qualidade dos dados utilizados em tais algoritmos. Foi também investigado diversos sistemas de visão computacional, em especial, os de detecção de objetos encontrados na literatura, desde sua história até os métodos modernos, considerados estado da arte.

2.1 TRABALHOS RELACIONADOS

Nesta seção, são contextualizados alguns trabalhos relevantes ao problema de detecção do peixe-zebra em ambientes pouco controlados, utilizando diferentes técnicas em ambientes variados.

Pérez-Escudero *et al.* (2014) desenvolveram o *idTracker*, que consiste em descobrir as impressões digitais de cada animal. Isso é feito encontrando porções do vídeo em que todos os indivíduos estão separados uns dos outros, que são então classificados e segmentados através de suas características texturais. No restante do vídeo, onde há oclusões, os quadros são segmentados e transformados em histogramas. A partir disso, as trajetórias são unidas por probabilidade dos mapas de contraste e intensidade dos pixels. Embora o processo de rastreamento seja 100% automático e o *idTracker* consiga identificar corretamente as trajetórias em uma média de 99,7% dos casos, o método requer milhares de amostras de cada peixe para extrair características relevantes para a identificação.

Qian *et al.* (2016) consideraram a movimentação dos peixes para estimar sua posição. Utilizando os primeiros quadros de um dado vídeo como *background*, foi possível determinar a região onde os peixes estão localizados, calculando a diferença entre os quadros subsequentes e a imagem de fundo original. Nessas regiões, é aplicado o *Augmented Fast Marching Method* (AFMM) para obter a linha central do peixe, que é utilizada para distinguir a cabeça do rabo. Assim, a partir do determinante da matriz Hessiana, que representa as informações estruturais locais da imagem, é estimada a posição e direção da cabeça de cada animal. O rastreamento é então calculado por uma função de custo, a fim de determinar a correta identificação do peixe após oclusões e cruzamentos, ou seja, garantir que as trajetórias de cada peixe presente no aquário, antes da obstrução, coincidam com os mesmos peixes das trajetórias, depois da obstrução. A validação do rastreamento foi feita em dois cenários, com o aquário iluminado por uma alta intensidade de luz e outro numa intensidade comum. Em ambos os casos, os peixes foram mantidos em um aquário com 3 cm de altura de água, contendo 20 e 40 animais, respectivamente. Os critérios de validação utilizados foram a *Average Correct Detection Rate* (ACDR), que é a razão entre o total de detecções verdadeiras-positivas e o número total de alvos detectados,

a *Average Error Detection Rate* (AEDR) que é a razão entre o total de detecções falsas-positivas e o número total de alvos, a *Average Occlusion Detection Rate* (AODR) que é a razão entre o total de oclusões verdadeiras positivas detectadas e o número total de oclusões ocorridas, e o *Average Direction Error* (ADE) que é o erro médio da direção dos alvos detectados como verdadeiros-positivos em relação à direção verdadeira. Estes critérios foram, respectivamente, maior que 97,1%, menor que 0,02%, maior que 79,5% e menor que 8,5° em ambos os cenários.

XU e Cheng (2017) utilizaram redes neurais convolucionais para a identificação de múltiplos peixes-zebra. Nomeado *CNNTracker*, o método identifica os peixes por meio de impressões digitais, que, diferentemente do *idTracker*, precisam apenas de características de texturas da cabeça do peixe. Essas características são obtidas por meio de técnicas de processamento de imagem, como o método de Otsu, que transforma um quadro do vídeo em uma representação binária dos peixes, e a *Principal Component Analysis* (PCA) na região da cabeça dos peixes para obter todas as características na mesma orientação. Estas, então, são introduzidas na rede convolucional para a detecção dos peixes. Os peixes foram mantidos em um aquário com 5 cm de altura de água, cujas paredes internas foram cobertas com papel branco e fosco para eliminar reflexões. O aquário foi também posicionado sobre um painel de matriz de *LED* com um filme difusor de luz. Diversas medidas de validação foram usadas, destacando-se a *precision*, dada pela razão entre a soma dos objetos identificados como verdadeiros positivos em todos os quadros e o número total de objetos em todos os quadros; a *recall*, que é a razão entre a soma dos objetos identificados como verdadeiros positivos em todos os quadros e a soma de todos os objetos detectados em todos os quadros; e a *F1-measure*, determinada pela média harmônica da *precision* e *recall*. Os pesquisadores avaliaram a rede em cinco vídeos e obtiveram medidas maiores de 99% em todas as avaliações mencionadas.

Bai *et al.* (2018) apresentaram o rastreamento de múltiplos peixes-zebra utilizando o algoritmo *Histogram of Oriented Gradient* (HOG) aprimorado para calcular um mapa de textura do peixe-zebra. Isso gera pequenos trechos de rastreamento que são classificados com uma *Suport Vector Machine* (SVM), rastreando os peixes automaticamente. Para melhorar a precisão, é realizada uma correção manual das trajetórias com erros, que são também localizados automaticamente. Os peixes-zebra foram filmados a uma altura de 50 e 100 cm do fundo de um aquário com 10 cm de profundidade de água. Para avaliação do algoritmo, foram definidas duas medidas: a *Identifying Accuracy* (IA), que denota a probabilidade de que o alvo seja identificado como verdadeiro positivo, e a *Classification Accuracy* (CA), que é a probabilidade de que os rótulos dos peixes sejam detectados como verdadeiros positivos em todos os quadros. A acurácia média dessas duas métricas foi de 99,27%.

Barreiros *et al.* (2021) propuseram um algoritmo de rastreamento que utiliza a rede neural convolucional de detecção de objetos *You Only Look Once* (YOLOv2). O método delimita e detecta a região da cabeça dos peixes-zebra individualmente e, posteriormente, um filtro de *Kalman* é utilizado para rastrear os peixes-zebra ao longo do tempo. Os peixes-zebra foram filmados em um aquário contendo 10 cm de altura de água, com as paredes externas tratadas

com um jato de areia, a fim de evitar o efeito da reflexão durante a detecção e rastreamento do peixe-zebra. O aquário foi também posicionado sobre uma fonte plana de luz branca, o que facilitou o rastreamento, já que os peixes-zebra ficam destacados. As métricas de avaliação de rastreamento utilizadas foram o *Correct Tracking Rate Index* (CTR), que descreve a porcentagem de quadros corretamente rastreados para um único peixe, e o *Correct Tracking Reason* (CIR), que representa a probabilidade de identificação verdadeira positiva de todos os peixes após uma oclusão. Dos oito vídeos utilizados na pesquisa, a rede neural convolucional YOLOv2 obteve uma precisão maior que 99,9% na detecção dos peixes-zebra, e as medidas CTR e CIR foram maiores que 95,0% na maioria dos vídeos.

Outros métodos para rastrear peixes também já foram explorados. Delcourt *et al.* (2011) utilizaram etiquetas coloridas e elastômeros visíveis implantáveis para rastrear animais aquáticos semitransparentes no escuro. Em particular, estudou-se o comportamento de enguias de atividade noturna durante um estágio de vida em que seus corpos são semitransparentes. Esses animais foram marcados com elastômeros e filmados em um tanque iluminado por uma luz ultravioleta. Isso permitiu a detecção usando técnicas de processamento de imagem, alcançando uma acurácia média superior a 95,0%.

2.2 DETECÇÃO DE OBJETOS

Como um dos problemas fundamentais em visão computacional, a detecção de objetos forma a base para várias outras tarefas, como segmentação de instâncias, legendação de imagens, rastreamento de objetos, entre outros (ZOU *et al.*, 2019). Este problema é fundamental, pois aplicações críticas como segurança, automação industrial e navegação autônoma de veículos dependem dele. Essas tarefas são triviais para seres humanos, que reconhecem milhares de objetos distintos em inúmeras situações. Crianças com apenas alguns meses de idade já conseguem reconhecer, identificar e classificar objetos comuns.

Programar computadores para realizar a detecção de objetos tem sido um objetivo há muitos anos devido às suas inúmeras aplicações, porém, vencer o desafio de identificar e localizar todas as instâncias de um objeto é muito difícil devido a diversas razões e situações (ZAIDI *et al.*, 2021). Como diferentes tarefas podem ter diferentes objetivos e restrições, suas dificuldades podem variar entre elas. Além das mais comuns, como as diferenças entre perspectivas, iluminação, número de objetos e variações intraclasses, os maiores desafios, mas não limitados a estes, incluem a rotação e escala dos alvos, localização precisa, detecção de objetos parcialmente ocluídos, velocidade de detecção, etc.

Do ponto de vista de aplicação, a detecção de objetos pode ser agrupada em dois tópicos de pesquisa: a detecção geral de objetos, onde se pretende simular a visão e cognição humana reconhecendo diversos tipos de objetos em uma estrutura unificada, a qual possibilita carros autônomos, por exemplo, e as aplicações de localização, que visam a detecção de alguns cenários específicos, como o reconhecimento de pedestres, faces, textos, animais, tráfego

e diversos outros. Devido o desenvolvimento dessas aplicações ser uma tarefa muito desafiadora, levou-se algumas décadas para obter um progresso considerável. Após o surgimento das *Convolutional Neural Network* (CNN), o assunto da detecção foi dividido em dois períodos históricos: a detecção de objetos tradicional e a detecção de objetos baseada em *deep learning*.

Na detecção de objetos tradicional, a maioria dos algoritmos eram baseados em características feitas à mão, tais como contornos, cores, texturas, entre outros. As pessoas não tinham escolha a não ser desenvolver representações sofisticadas das características que desejavam detectar (ZOU *et al.*, 2019). Já a detecção de objetos baseada em *deep learning* é agrupada em dois gêneros: os detectores de dois estágios, que passam a imagem por duas fases de processamento, e os detectores de um estágio, que realizam a detecção em uma única etapa.

Quanto a exemplos de detectores de dois estágios, tem-se a CNN com regiões, que divide a imagem de entrada em um conjunto de regiões, que contêm candidatos a objetos usando uma busca seletiva. Este conjunto é então redimensionado e alimentado a um modelo de CNN pré-treinado, e, finalmente, um classificador SVM é usado para prever a presença de um objeto em cada uma das regiões e reconhecer a categoria dos objetos.

Por essa abordagem, foi possível obter resultados muito melhores do que as técnicas anteriores, mas a detecção era bastante lenta, chegando a 14 segundos por imagem, utilizando GPU. Posteriormente, foi proposto um modelo de CNN com uma camada extra de *Spatial Pyramid Pooling* (SPP). Neste modelo, os pesquisadores tornaram os modelos invariantes de resolução e de proporção de aspecto das imagens, obtendo resultados semelhantes, porém mais eficientes.

Já os detectores de apenas um estágio têm-se de exemplo o YOLO. Proposto em 2015, esse modelo divide a imagem de entrada em grades de um determinado tamanho, e a célula onde o centro do objeto de interesse se encontra é responsável por detectá-lo. O detector *RetinaNet*, proposto em 2017, explorou uma função de perda diferente da comumente utilizada, para justificar o atraso de desempenho dos modelos de estágio único em relação aos de dois estágios (ZOU *et al.*, 2019).

O *EfficientDet* é construído em direção à ideia de detectores escaláveis, com maior precisão e eficiência. Ele introduz características de múltipla escala, rede piramidal de características bidirecionais e escala de modelo eficientes. *EfficientDet* possui melhor desempenho em relação a outros detectores, é computacionalmente mais leve e mais barato. Atualmente, é o modelo estado da arte para detectores de objeto de apenas um estágio (ZAIDI *et al.*, 2021). Diversos outros modelos foram propostos. Alguns, como o caso do YOLO, foram aprimorados várias vezes para torná-lo mais rápido, eficiente e preciso. Na atualidade, o YOLO está em sua quarta versão publicada, tornando-o estado da arte em detectores de estágio único em tempo real (ZAIDI *et al.*, 2021). Assim, levando em conta a popularidade e qualidade das CNN na tarefa de detecção de objetos, considerou-se que o YOLO é um candidato promissor para realizar o rastreamento de peixes-zebra.

2.2.1 MÉTRICAS DE AVALIAÇÃO

Para avaliar a qualidade dos métodos, é necessário definir algumas métricas. No caso dos detectores de objetos, são utilizados múltiplos critérios para medir seu desempenho. Esses critérios mudaram bastante ao longo dos anos devido às falhas em certos casos. Recentemente, algumas métricas se destacaram e são utilizadas frequentemente na avaliação dos modelos de detecção de objetos (ZOU *et al.*, 2019), tais como a *Frames Per Second* (FPS), que mede quantas imagens por segundo o modelo é capaz de processar para detecção, a *precision* e a *recall* (ZAIDI *et al.*, 2021), que são definidas pelas equações a seguir, logo após a explicação de alguns conceitos:

- Verdadeiros Positivos (VP): São os casos em que o modelo corretamente identifica a presença do objeto.
- Falsos Positivos (FP): Quando o modelo incorretamente identifica a presença do objeto.
- Falsos Negativos (FN): Casos em que o modelo falha em identificar a presença do objeto quando ele realmente está presente.

$$\textit{Precision} = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosPositivos} \tag{1}$$

$$\textit{Recall} = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosNegativos} \tag{2}$$

Como a *precision* é a medida de uma única imagem, torna-se necessário avaliar o modelo com uma maior diversidade, geralmente de maneira específica, como por exemplo de uma determinada categoria. Dessa forma, calcula-se a média das precisões (*Average Precision* (AP)). Para comparar o desempenho sobre todas as categorias, a *mean Average Precision* (mAP) é obtida fazendo a média das APs sobre todas as categorias, sendo frequentemente utilizada como a métrica final de desempenho (ZOU *et al.*, 2019).

Quando se trata de detecção de objetos, a localização de um objeto é comumente representada através de uma *Bounding Box* (BB), que é uma região retangular mínima que contém completamente o objeto. Para avaliar a precisão da localização de um objeto predito por um modelo, emprega-se uma métrica conhecida como *Intersection over Union* (IoU). A IoU quantifica o quão próxima uma previsão de localização de um objeto está em relação à sua localização real. Isso é feito calculando a área de intersecção (área comum) entre a caixa delimitadora predita e a real e dividindo-a pela área da união (área total coberta) destas duas caixas delimitadoras, conforme ilustrado na Figura 2. Para determinar a precisão na localização de um objeto, o valor da IoU é comparado a um limiar previamente estabelecido, que é geralmente determinado de forma experimental para cada caso específico. No contexto do modelo YOLO pré-treinado usado neste trabalho, por exemplo, o limiar foi definido como 0.213. Se a IoU exceder este limiar,

a previsão é classificada como um verdadeiro positivo. Caso contrário, a previsão é classificada como um falso positivo. Se o modelo não conseguir prever uma região que intercepte a caixa delimitadora real do objeto, tem-se um falso negativo (ZOU *et al.*, 2019; ZAIDI *et al.*, 2021).

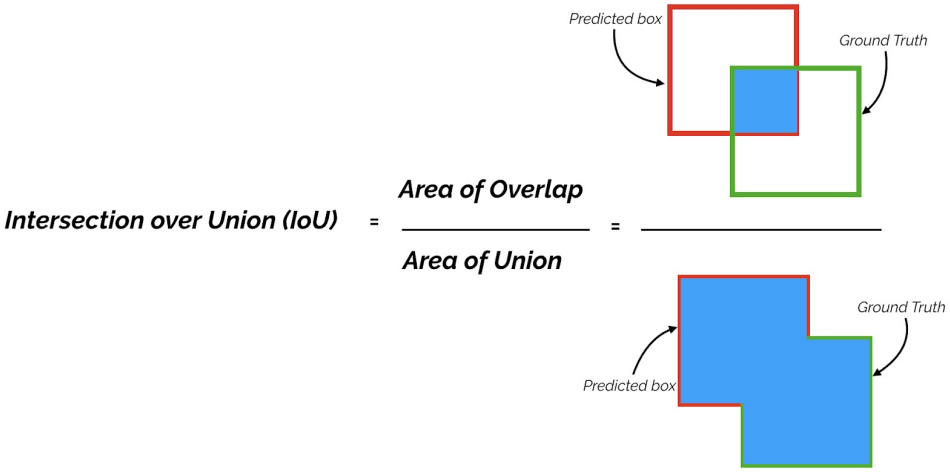


Figura 2 – Intersecção sobre União
A região em azul, representada no numerador, ilustra a área de intersecção entre a caixa delimitadora predita (em vermelho) e a real (em verde). No denominador, temos a área total abrangida pelas duas caixas (união), ou seja, a soma das áreas das duas caixas menos a área de intersecção (região em azul).
Fonte: Yohanandan (2020).

2.2.2 YOLOv4

O *You Only Look Once* (YOLOv4) é um detector de objetos baseado em CNN, atualmente em sua quarta versão. Ele implementa diversos aprimoramentos com o objetivo de melhorar sua precisão e velocidade, posicionando-se como estado da arte em detectores de estágio único em tempo real. Os detectores modernos são geralmente compostos de duas partes: a *backbone*, que é pré-treinada com a base de dados ImageNet em alguma arquitetura de classificação de imagens, como a VGG, ResNet, ResNeXt ou DenseNet quando se usa uma GPU, ou SqueezeNet, MobileNet ou ShuffleNet no caso de uso de CPU; e a segunda parte, a *head*, responsável por inferir as classes dos objetos e suas caixas delimitadoras BB. Nos detectores desenvolvidos mais recentemente, algumas camadas que formam o chamado *neck*, são frequentemente inseridas entre a *backbone* e a *head*. Essas camadas são usadas para coletar mapas de características de diferentes estágios. As redes equipadas com estes mecanismos incluem *Feature Pyramid Network* (FPN), *Path Aggregation Network* (PAN), *BiFPN* e *NAS-FPN*.

Como os detectores de objetos são geralmente treinados *offline*, os pesquisadores costumam desenvolver métodos de treinamento que podem melhorar a precisão do detector sem aumentar o tempo de inferência. Esses métodos, que aumentam apenas o tempo de treina-

mento, são chamados de *Bag of Freebies* (BoF). Na detecção de objetos, o *data augmentation* frequentemente se enquadra nessa definição. O objetivo do *data augmentation* é fornecer maior variabilidade nos dados de entrada, tornando o modelo mais robusto a imagens obtidas em ambientes diferentes. Por exemplo, distorções fotométricas como ajustes de brilho, contraste, matiz, saturação e ruídos, e também distorções geométricas como escala, recortes, inversões e rotações são métodos de *data augmentation* em imagens, muito comuns e beneficiam a tarefa de detecção de objetos. Além disso, alguns pesquisadores têm enfatizado técnicas que simulam a oclusão de objetos e outras que usam múltiplas imagens conjuntas para atuar como *data augmentation*, como a *CutOut* e a *grid mask*, que podem selecionar aleatoriamente um ou mais retângulos na imagem e preencher essas regiões com valores aleatórios ou até mesmo com zeros, a técnica *MixUp*, que superpõe duas imagens com coeficientes variados, a *CutMix*, que substitui uma região de uma imagem por outra, e outras que incluem até mesmo transferência de estilo, utilizando *Generative Adversarial Network* (GAN)s. Outro método que se enquadra em BoF, é a função objetivo de regressão das BBs. Tradicionalmente, era utilizada a função *Mean Square Error* (MSE) para estimar cada ponto da BB, os quais são tratados como variáveis independentes e consequentemente não consideram a integridade do objeto em si. Recentemente, foi proposta a função de perda IoU, que leva em consideração a cobertura da área da BB predita em relação à área da BB real. Além disso, como a IoU é invariante à escala da imagem, ela pode resolver o problema que alguns métodos tradicionais possuem, quando se é calculado função de regularização *l1* ou *l2*, que tornam esse valor muito maior, conforme a escala do objeto aumenta Bochkovskiy, Wang e Liao (2020, p. 3).

Os módulos de *plug-in* e métodos de pós-processamento, que incrementam o tempo de inferência em uma pequena proporção, mas que podem aprimorar significativamente a precisão do detector, são chamados de *Bag of Specials* (BoS). Estes são utilizados para melhorar certos atributos, como o aumento do campo receptivo, introdução de mecanismos de atenção, integração de características, entre outros. Módulos comuns, que ampliam o campo receptivo, são o SPP, *Atrous Spatial Pyramid Pooling* (ASPP) e *Receptive Field Block* (RFB), os quais implementam *kernels*, otimizados nas camadas de convolução do *backbone*.

Em relação aos mecanismos de atenção, dois representantes são o *Squeeze-and-Excitation* e o *Spatial Attention Module*, no entanto, o primeiro é mais recomendado para dispositivos móveis, uma vez que este incrementa o tempo de inferência na GPU em cerca de 10%. Desde que os métodos de predição multi-escala como o FPN se tornaram populares, vários outros módulos distintos de integração de características piramidais foram propostos, exemplos destes são o *Selective Feature Aggregation Module*, *Adaptive Spatial Feature Fusion BiFPN*, que utilizam diferentes técnicas para criar mapas de características em diferentes escalas.

O método de pós-processamento comumente utilizado em detecção de objetos baseada em *deep learning* é o *Non-Maximum Suppression*, que pode ser usado para filtrar as BBs que preveem incorretamente o mesmo objeto e retém apenas os candidatos com melhor resposta, este método foi aprimorado utilizando outras técnicas como o *greedy NMS* e *DIoU NMS*,

os quais implementam algum tipo de pontuação de confiança para levar em consideração o contexto da informação Bochkovskiy, Wang e Liao (2020, p. 4).

No decorrer do desenvolvimento do trabalho proposto por Bochkovskiy, Wang e Liao (2020), foram testadas diversas técnicas e métodos mencionados anteriormente. No caso da arquitetura do YOLO, foi escolhida como *backbone* a *CSPDarknet53*, pois notou-se que, embora a *CSPResNeXt50* possua uma precisão superior à da *CSPDarknet53* na classificação de imagens, o mesmo não se aplica para detecção de objetos.

No *neck* foi testada a influência de algumas BoSs, como PAN, RFB, SPP, *SAM*, *Gaussian YOLO* e *ASFF*, das quais SPP, *SAM* e PAN apresentaram melhor desempenho, e como *head* foi utilizada a mesma do *YOLOv3*. Dentre as BoF experimentadas, obteve-se uma melhor precisão utilizando *CutMix* e *Mosaic* como *data augmentation*, *Class label smoothing*, regularização *DropBlock* e função de perda *CIoU*. O modelo foi treinado com apenas uma GPU utilizando o *dataset MS COCO*, proporcionando um detector de objetos em tempo real mais preciso do que as alternativas disponíveis Bochkovskiy, Wang e Liao (2020, p. 5-7).

3 MATERIAIS E MÉTODOS

Considerando que o objetivo é rastrear um peixe-zebra em diversos ambientes, foi essencial obter uma coleção de vídeos desses animais nessas situações. Especificamente, era necessário que os vídeos incluíssem cenas contendo:

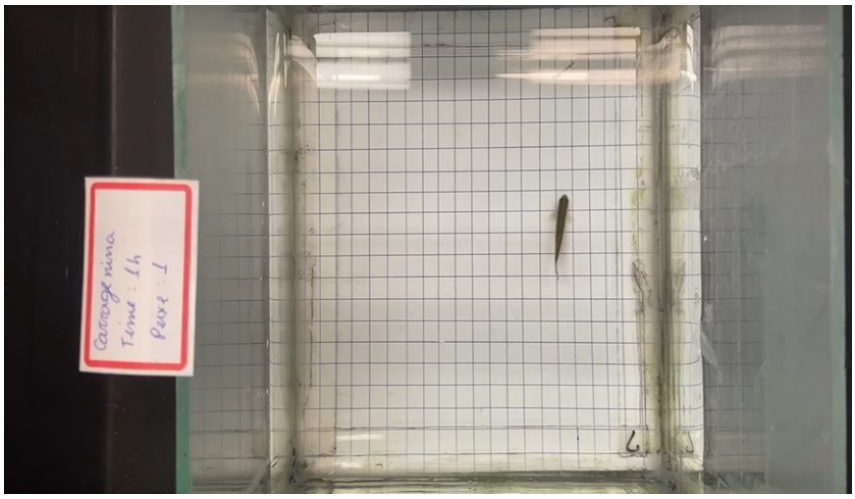
- Uma ou mais reflexões do peixe nas paredes do aquário;
- Agitações na superfície da água, as quais causam distorções na geometria do ambiente, em especial do peixe;
- Regiões nas quais o peixe está parcialmente ou totalmente ocluso na imagem, seja devido ao reflexo luminoso na superfície da água ou à presença do peixe em uma região do aquário que não é capturada pela câmera;
- Diferenças na imagem de plano de fundo entre os vídeos, incluindo cor, textura e artefatos ou objetos.

Para este trabalho, foram disponibilizados pelo Orientador Prof. Dr. Dalcimar Casanova, 32 vídeos. Houve um total de três aquários distintos, apresentados na Figura 3. Devido às filmagens dos aquários 3b e 3c serem de múltiplos peixes, foi necessário utilizar um software de edição de vídeos para extrair trechos contendo apenas um peixe, com alguns segundos de duração das filmagens, resultando em 24, 7 e 5 vídeos distintos para os aquários 3a, 3b e 3c, respectivamente.

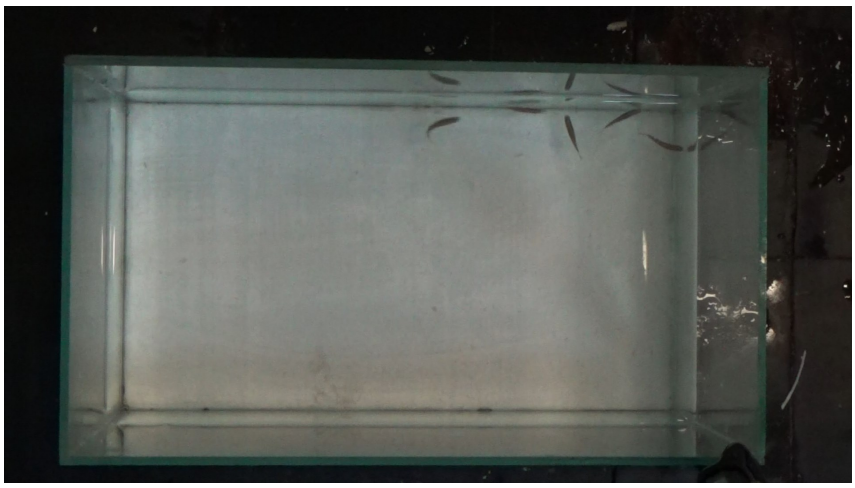
3.1 PRÉ-PROCESSAMENTO

Após a obtenção dos vídeos, foram selecionados trechos que continham alguma situação mencionada anteriormente, preferencialmente aqueles com várias situações ocorrendo simultaneamente. A seleção desses trechos foi feita manualmente após assistir aos vídeos e identificar os momentos de interesse. Posteriormente, foi rotulado a posição do peixe em cada quadro desses trechos selecionados. Inicialmente, disponibilizou-se para ser rotulado o conjunto de todos os quadros que formam cada um dos vídeos. O conjunto destas imagens foi criado utilizando um *script* em Python com a biblioteca OpenCV.

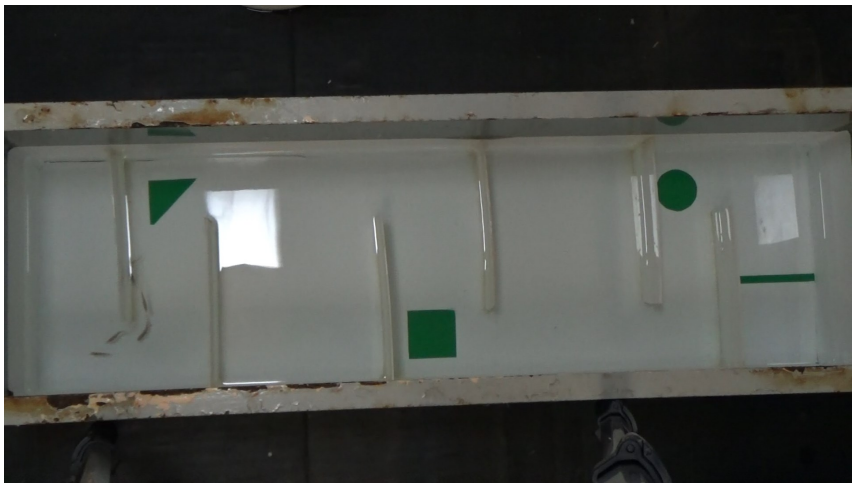
Este processo de seleção e rotulagem manual foi necessário, para garantir que os exemplos de treinamento incluíssem as várias situações e desafios, que o algoritmo de rastreamento teve que lidar.



(a) Aquário 1



(b) Aquário 2



(c) Aquário 3

Figura 3 – Exemplos de aquários disponibilizados

Fonte: Autoria própria.

3.1.1 ROTULAÇÃO

A rotulação foi realizada através de uma BB para cada objeto de interesse presente na imagem. As informações referentes a essas BBs foram salvas em um arquivo de texto junto com a imagem.

Recomenda-se o uso de uma base pré-treinada do YOLOv4 para aproveitar o conceito de transferência de aprendizado, o que pode acelerar a especialização na detecção de peixes. Portanto, foi necessário criar as bases de dados de treinamento e teste, contendo dados compatíveis com aqueles utilizados na base pré-treinada. As imagens e seus rótulos são representados pelas seguintes regras de formatação:

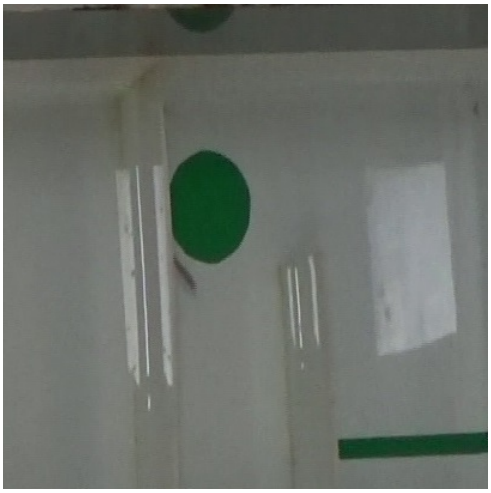
- Cada imagem deve possuir um nome único, no qual foi nomeado de acordo com o nome do vídeo e a posição do quadro no vídeo;
- Cada imagem terá um arquivo texto associado, de mesmo nome, contendo os rótulos dos objetos;
- Cada objeto contido na imagem será representado por uma linha do arquivo texto;
- Cada objeto é representado por uma BB que é formada por cinco valores, a classe do objeto, a posição horizontal central (*x*), posição vertical central (*y*), largura da caixa (*w*) e altura da caixa (*h*);
- Todos os valores (v) que formam a BB devem ser números reais tal que $0 < v < 1$, com exceção da classe que pode ser $0 \leq v \leq 1$.

A fim de melhorar a detecção, Bochkovskiy, Wang e Liao (2021) encorajam o uso de pelo menos duas mil amostras do objeto, com variedade em forma, ângulo de rotação, iluminação, escala, e outros. Um total de 3035 amostras foram rotuladas de forma manual para realizar o treinamento e teste, incluindo várias condições diferentes, além de uma quantidade adicional para validação. Algumas dessas condições são exemplificadas na Figura 4.

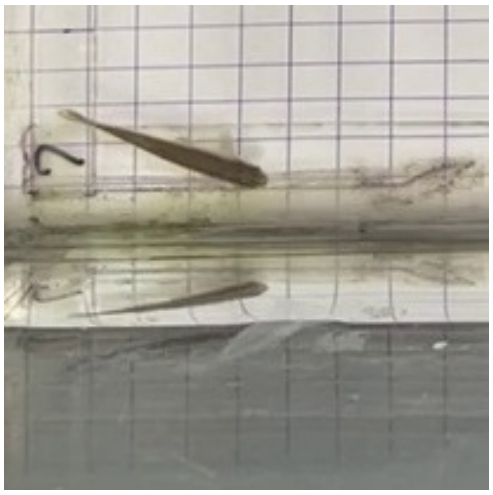
Outra forma recomendada por Bochkovskiy, Wang e Liao (2021) de aprimorar a detecção é aumentar o campo de receptividade do detector quando o objeto possui uma BB muito pequena. Como a imagem original é redimensionada para o tamanho do campo de receptividade, a BB pode possuir uma resolução menor de 16x16 pixels, o que se fez necessário então modificar alguns hiperparâmetros do detector YOLOv4. Primeiramente, foi incrementado o campo de receptividade de 416x416 para 608x608 pixels, alterou-se o número de filtros de convoluções e o passo do filtro em uma camada específica do modelo, conforme indica a documentação.

Como a rotulação teve de ser feita manualmente, foi desenvolvido um *script* em Python, que permitiu que fosse desenhado um segmento de reta na imagem, indicando a localização da cabeça e o corpo do peixe, ou seu reflexo. Dessa forma, considerou-se a curva que corta

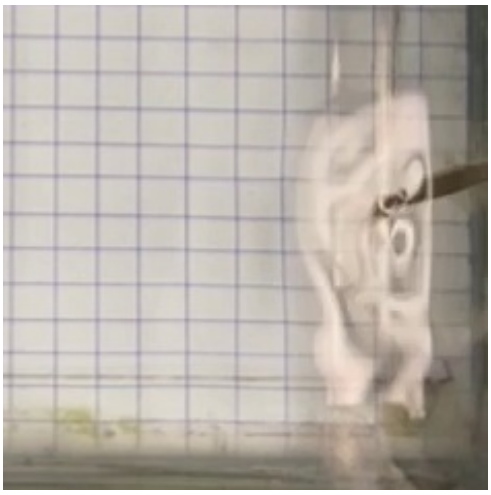
(a) Ocluso por reflexo



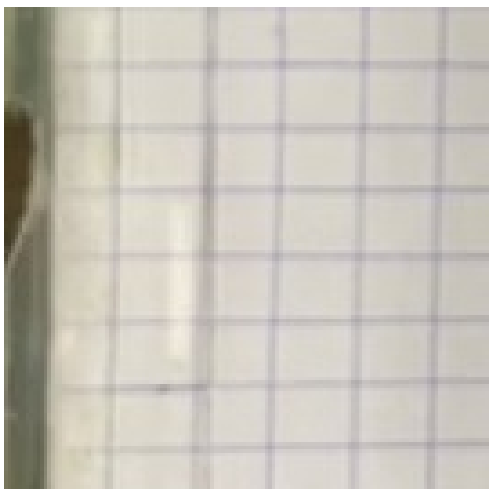
(b) Ocluso por artefato



(c) Reflexos nas paredes



(d) Distorções na superfície

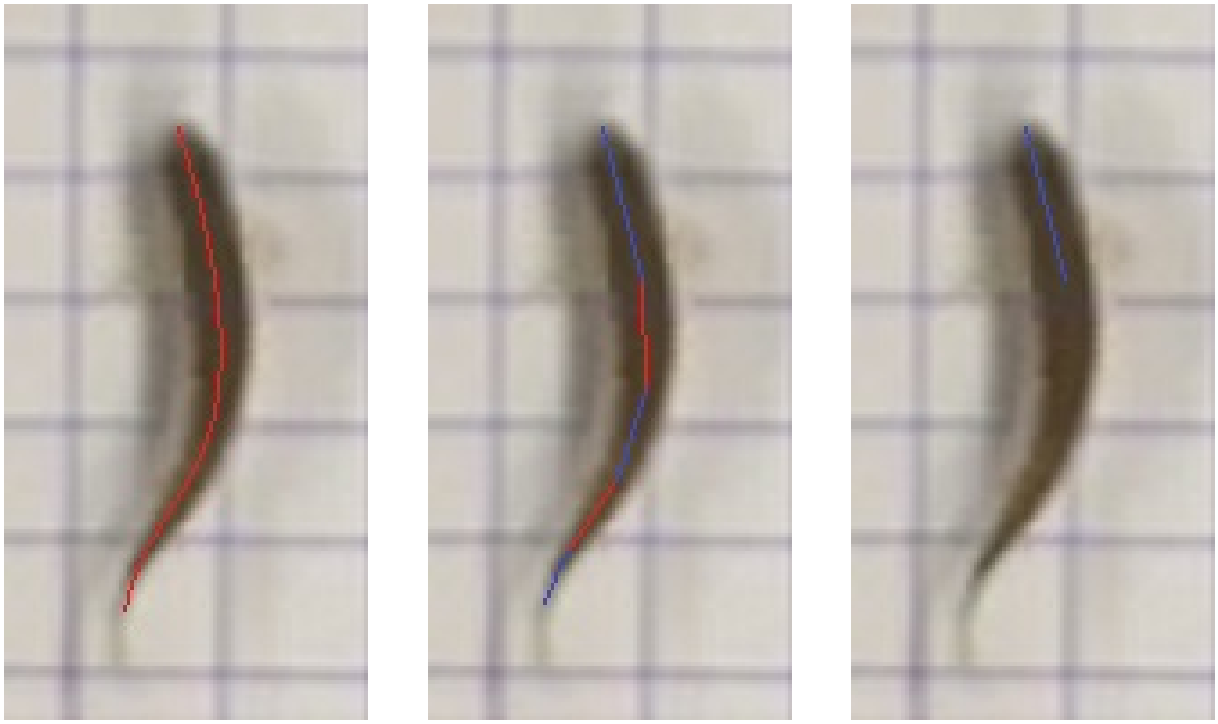


(e) Parcialmente fora do quadro

(f) Múltiplos reflexos

Figura 4 – Exemplos de situações de interesse

Fonte: Autoria própria.



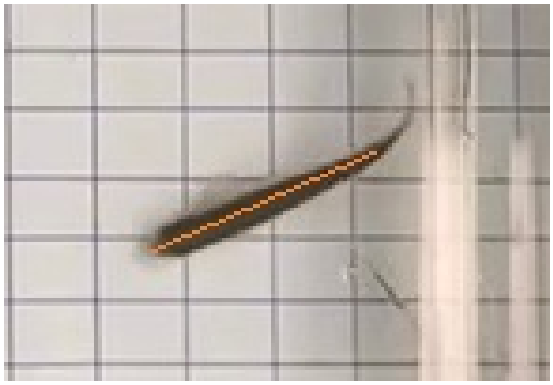
(a) Curva longitudinal **(b) Aproximação por segmentos de reta** **(c) Primeiro segmento**

Figura 5 – Exemplo de rotulação
Fonte: Autoria própria.

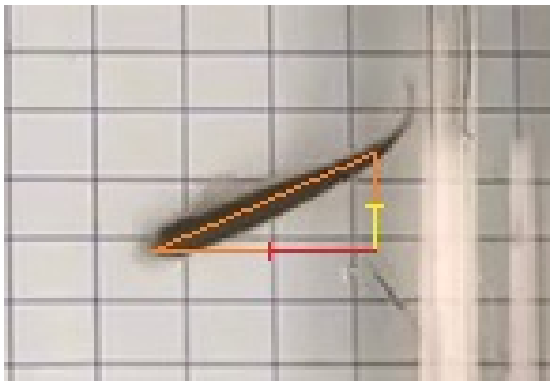
longitudinalmente o peixe e o divide em duas partes de mesma área, porém, devido às definições da BB, a curva teve de ser aproximada por segmentos de reta e limitada ao primeiro segmento, como mostra a Figura 5. Uma vez que os segmentos de reta foram desenhados, foi possível calcular diferentes BBs para a mesma amostra. Para este trabalho, foram consideradas variações de classe e BB, ou seja, se o reflexo será detectado ou não e a forma de desenhar a BB no peixe.

Foi crucial que os dados estivessem normalizados antes do treinamento, pois a representação de todas as características, na mesma grandeza numérica, que podem auxiliar a agilizar a generalização do modelo (RASCHKA; MIRJALILI, 2019, p.124), no caso das imagens, esse processo foi simples, uma vez que geralmente são representadas pelos canais *RGB* de um *byte* cada, assim, seus limites são bem definidos e puderam ser normalizados, simplesmente dividindo todos os seus valores por 255. Já para os rótulos, foi necessário apenas conhecer a quantidade total de objetos diferentes que serão classificados. Estes também puderam ser normalizados facilmente com uma divisão, juntamente com o processo de rotulação.

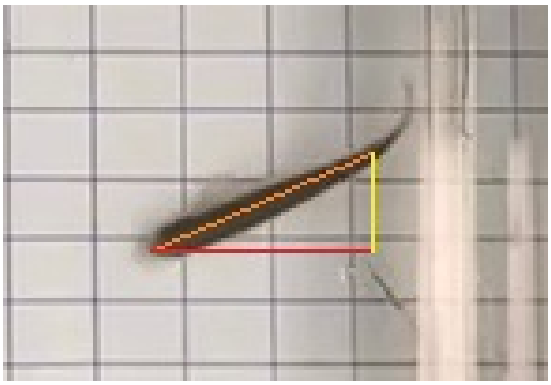
Quanto à forma de desenhar a BB no peixe, foram consideradas duas abordagens diferentes, que são exemplificadas na Figura 6. No primeiro caso, a BB é formada a partir do segmento de reta desenhado no peixe (Figura 6a), utilizando o comprimento até os pontos médios dos catetos do triângulo (Figura 6b) para formar a BB centralizada no meio do segmento de reta (Figura 6d). No segundo caso, foram criadas duas BBs, uma centralizada no início e



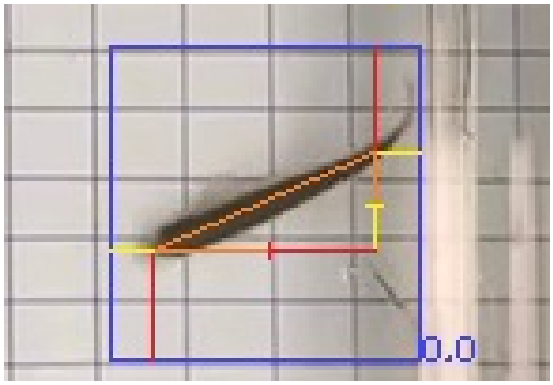
(a) Segmento de reta



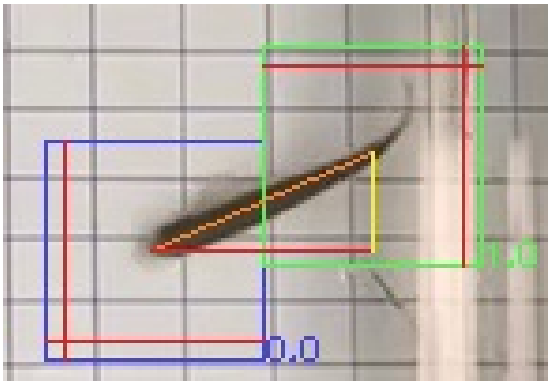
(b) Medianas dos catetos



(c) Medidas dos catetos



(d) BB centralizada no meio do segmento



(e) BB centralizada no início e outra no final do segmento

Figura 6 – Exemplos de caixa delimitadora
Fonte: Autoria própria.

outra no fim do segmento de reta, ambas possuindo como largura e altura o comprimento do maior cateto do triângulo formado pelos eixos x e y da imagem, tendo o segmento de reta como hipotenusa (Figura 6e). Dessa forma, foi possível investigar a possibilidade de obter a direção do peixe, detectando tanto a cabeça quanto o corpo. Levando em consideração a variação da detecção ou não do reflexo, foram treinados um total de cinco modelos distintos.

Uma vez determinada qual forma de rotular o peixe com a maior acurácia, tornou-se necessário investigar quão benéfica é a aplicação de métodos de *data augmentation*, para melhor diversificar o *dataset*.

3.1.2 *DATA AUGMENTATION*

Idealmente, o dataset composto por imagens e seus respectivos rótulos deveria incluir amostras de uma ampla variedade de peixes e ambientes. Como se teve acesso a vídeos de apenas três aquários e alguns peixes, recorreu-se a métodos de *data augmentation* para enriquecer o conjunto de dados e, consequentemente, melhorar a acurácia do modelo Bochkovskiy, Wang e Liao (2020, p. 3).

Diversas técnicas de processamento de imagens foram utilizadas para gerar imagens aumentadas através de deformações fotométricas e geométricas. Foram exploradas treze técnicas distintas, utilizando a biblioteca *CLoDSA* para transformações fotométricas como equalização do histograma (para melhoria do contraste), correção gamma (que ajusta o brilho através de transformações não lineares), variações nos canais RGB e HSV, *dropout* (simulando perda de dados em regiões aleatórias na imagem), e *sharpening* (para melhorar a nitidez da imagem).

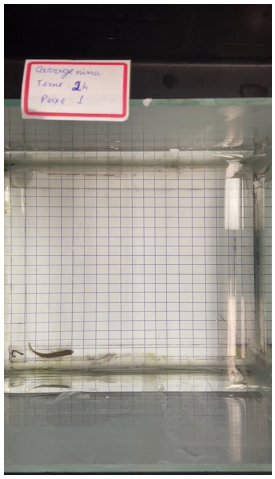
Para as transformações geométricas, utilizou-se as bibliotecas OpenCV e Numpy, aplicando nas imagens o espelhamento (vertical ou horizontal), a rotação (entre 90 e -90 graus), e a translação nos dois eixos. A Figura 7 apresenta exemplos de todas as transformações fotométricas e a Figura 8, todas as geométricas aplicadas.

No melhor dos casos, os conjuntos de validação e teste deveriam também refletir uma ampla variedade de peixes e ambientes. Para alcançar isso, incluiu-se imagens aumentadas nas bases de treinamento e teste, mas a avaliação foi feita tanto com, quanto sem *data augmentation* na base de teste. Decidiu-se dividir as bases de forma que os primeiros 75% das imagens rotuladas de cada vídeo fossem usadas para treinamento, e os 25% restantes para teste, quais representaram uma quantidade de 2275 e 760 nas bases de treino e teste respectivamente, quando não havia *data augmentation*, 4551 e 1519 quando somente uma técnica foi aplicada e, 22761 e 7589 nos ensaios com combinações de técnicas.

3.2 DETECÇÃO

Para treinar a CNN, foi necessário baixar a implementação YOLO disponível no GitHub, de autoria dos criadores do projeto. A instalação foi realizada no Windows 10, seguindo uma série de passos para o correto funcionamento do programa. Além disso, foram baixados e utilizados os pesos pré-treinados, para acelerar o processo de especialização do modelo para a detecção do peixe-zebra.

Com os *datasets* de treinamento e teste já preparados, foram executados algumas dezenas de treinamentos para chegar a um resultado satisfatório. Um *pipeline* dos passos de como treinar o modelo foi realizado. Primeiramente, como o YOLO é um programa já compilado, fez-se necessário mover os dados de treino e teste para um local específico na pasta de instalação do *software*, onde, através de um *script* em Python fornecido no GitHub, foram gerados dois arquivos de texto, contendo o nome do arquivo de todas as amostras disponíveis para treino e



(a) Original

(b) Equalização do Histograma

(c) Correção Gamma

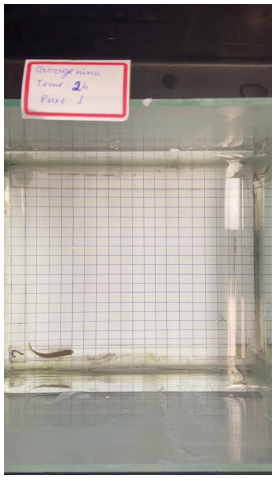
(d) Alteração no Canal Vermelho

(e) Alteração no Canal Verde

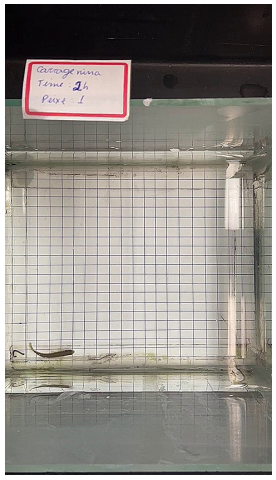
(f) Alteração no Canal Azul

(g) Alteração no Canal HUE

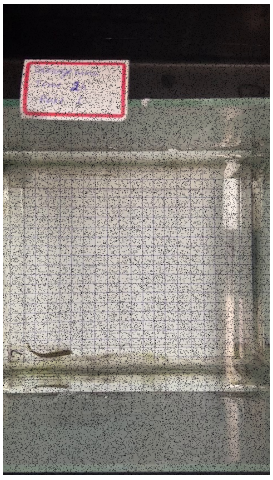
(h) Alteração no Canal Saturação



(i) Alteração no Canal Brilho



(j) *Sharpen*



(k) *Dropout*

Figura 7 – Exemplos de cada transformação fotométrica aplicada

Fonte: Autoria própria.

(a) Espelhamento **(b) Translação** **(c) Rotação**

Figura 8 – Exemplos de cada transformação geométrica aplicada
Fonte: Autoria própria.

teste. Após isso, foi preciso copiar dois arquivos, os quais contêm informações do número de classes que irão ser aprendidas pelo modelo e também o nome de cada classe.

Outro arquivo muito importante a ser ajustado, foi o de configuração da CNN, em que foi definida toda a capacidade do modelo, número de classes, tamanho do campo de receptividade, número de filtro de convolução e outros hiperparâmetros. Em particular, para este trabalho, foram ajustadas algumas configurações recomendadas para aproveitar melhor os recursos da GPU, aumentar o campo de receptividade e outros hiperparâmetros para poder detectar objetos muito pequenos, detalhes também disponíveis na documentação. Outra informação que teve de ser configurada para todo treinamento que foi realizado, é o número de classes a ser treinado e o número de filtros de convolução em algumas camadas específicas do modelo, que segundo a documentação, deve-se seguir algumas regras para que o número de filtros fique em função do número de classes, e o número de iterações de treinamento (*epochs*) deve ser também em função do número de classes, mas não menos que 6000 *epochs*. Para treinar o modelo com apenas uma classe, levou-se em média 12 horas em 6000 *epochs* na GPU utilizada, a qual tende a levar mais tempo, conforme o número de classes aumenta, o número de amostras para treinamento aumenta e consequentemente o número de epochs.

Os treinamentos, à princípio, foram realizados no Google Colaboratory, mas como há limitações na utilização de GPU gratuita e foi requerido um número grande de modelos treinados, optou-se por treinar localmente, utilizando uma placa de vídeo GeForce RTX 3060 com 12GB de memória dedicada, processador AMD Ryzen 7 1700x com 32 GB de memória RAM. Esta configuração, foi suficiente para realizar todos os treinamentos, porém, notou-se que ao aumentar o campo de receptividade de 416x416 para 608x608 pixels, foi preciso recorrer à memória compartilhada do sistema.

Uma vez que os arquivos estavam configurados e nos locais corretos do sistema, era possível realizar os treinamentos, utilizando alguns comandos do próprio YOLO. O programa,

por padrão, realiza algumas validações, para que se possa acompanhar a convergência do modelo durante o seu treinamento, e também salvar os pesos do modelo, quando a acurácia desse é aprimorada. Após treinar todas as *epochs*, obtém-se os pesos do modelo da melhor acurácia obtida durante o treinamento, os quais são então utilizados em conjunto com outro comando para medir a qualidade do modelo. As bases de treino e teste possuem amostras do mesmo aquário e mesmo peixe, porém, as amostras na base de teste nunca foram vistas durante o treinamento. A validação foi realizada no conjunto de testes utilizando a métrica AP para a classe do peixe, que no caso de apenas uma classe, AP é equivalente à mAP. Como os detectores processam uma única imagem por vez, após o treinamento do modelo, realizou-se a inferência em todos os quadros rotulados de um vídeo, para avaliar a precisão ao longo de uma gravação completa. Isso foi essencial para confirmar que o modelo é capaz de realizar detecções confiáveis e precisas em sequências de imagens, e não apenas em imagens individuais.

4 RESULTADOS

Este capítulo é dedicado à apresentação e discussão dos resultados obtidos nas diversas etapas de treinamento e validação que compõem este estudo. Esta análise foi realizada em duas partes principais. Primeiramente, foram analisadas as amostras rotuladas e a diversidade das situações que elas representam. Em seguida, foram explorados os resultados dos treinamentos realizados com diferentes técnicas de *data augmentation*.

4.1 ROTULAÇÃO

As amostras utilizadas para o treinamento do modelo de detecção de peixes-zebra foram o cerne deste estudo. Para realizar o rastreamento de forma eficaz em diversas situações, é recomendado que pelo menos duas mil amostras do mesmo objeto sejam rotuladas. Dessa maneira, foram coletadas um total de 3035 amostras, estas foram rotuladas de forma manual, cada amostra pôde conter mais de um objeto, quais foram categorizados de acordo com os diferentes cenários encontrados ao observar os vídeos. Dos vídeos observados, foram encontradas algumas situações de interesse em que, como por exemplo, havia o peixe totalmente visível, o peixe e um ou mais reflexos, o peixe parcialmente fora do quadro, e o peixe em baixa resolução. A distribuição destas categorias são apresentadas na Tabela 1.

Tabela 1 – Distribuição das amostras rotuladas

Descrição	**Quantidade**
Peixe completamente visível	1646
Peixe e pelo menos um reflexo	1219
Peixe parcial ou totalmente oculto	353
Peixe parcialmente fora do quadro	72
Peixe em baixa resolução	964

Fonte: Autoria própria

Considerando que, não foi possível obter muitas amostras de cenas de interesse e que teve-se que dividir as amostras rotuladas em treino e teste, optou-se por usar todas as 3035 amostras, sem realizar um balanceamento de amostras entre todas as categorias. Outro motivo de se utilizar todas as amostras, foi devido estas categorias serem processadas em classes, representadas por valores entre 0 e 1, de acordo com o método de rotulação e número de classes, para então serem aprendidas pelo detector.

4.1.1 BASES PARA TREINAMENTO

Inicialmente o desempenho do modelo foi investigado através do teste de diferentes métodos de rotulação da BB e variações de número de classe. Para esta análise, foram criadas

e testadas cinco bases diferentes, cujas características e resultados estão sintetizados na Tabela 2.

Tabela 2 – Bases para treinamento e acurácia

Rotulação da BB	**Classes**	**AP** (%)
Entre a cabeça e corpo	Peixe e reflexo	78.45
Entre a cabeça e corpo	Somente peixe	79.90
Cabeça e corpo	Cabeça, corpo, reflexo da cabeça e do corpo	78.37
Cabeça e corpo	Cabeça e corpo	79.10
Cabeça e corpo	Somente cabeça	77.04

Fonte: Autoria própria

Analisando os resultados obtidos, percebe-se que a variação da acurácia é relativamente pequena entre as diferentes bases testadas. No entanto, a maior acurácia foi obtida com a base de treinamento que usa uma única classe, o peixe, e rotula a BB entre a cabeça e o corpo. Esta opção parece oferecer um bom equilíbrio entre a complexidade do modelo e sua capacidade de generalização, por essa razão, esta forma de rotulação foi padronizada para o restante deste trabalho.

4.2 *DATA AUGMENTATION*

Com o objetivo de aumentar a diversidade dos dados e melhorar a acurácia do modelo YOLO, foram aplicados diversos métodos de *data augmentation* às imagens rotuladas. Para cada método de *data augmentation* utilizado, foi gerado uma base de dados distinta. Essas bases contêm tanto o conjunto original de 3035 amostras, quanto uma cópia dessas amostras com a técnica de *data augmentation* aplicada. Assim, cada base de dados individual para *data augmentation* contém um total de 6070 amostras, que foram divididas entre os conjuntos de treino e teste. "A Tabela 3 fornece um resumo dos resultados obtidos, mostrando a AP para cada técnica de *data augmentation* utilizada. A primeira linha da tabela apresenta a acurácia do modelo treinado sem técnica aplicada como valor de referência, já suas colunas representam os seguintes dados: o nome da técnica, a acurácia do modelo quando a base de teste inclui amostras com *data augmentation*, a melhoria de desempenho do modelo avaliado na base de teste que inclui amostras com *data augmentation* em relação ao modelo sem, a acurácia do modelo quando a base de teste não inclui amostras com *data augmentation* e a melhoria de desempenho do detector avaliado na base de teste que não inclui amostras amostras com *data augmentation* em relação ao modelo sem. Esses resultados são fornecidos para permitir uma comparação direta do impacto de cada técnica de *data augmentation* na desempenho do modelo.

Ao analisar os resultados, observa-se que a aplicação do *data augmentation*, aumentou significativamente a acurácia do modelo em todas as técnicas, indicando que a diversificação dos dados foi benéfica para o treinamento.

Tabela 3 – Resultados do *data augmentation* individual

Técnica	**AP com DA (%)**	**Melhoria com DA (%)**	**AP sem DA (%)**	**Melhoria sem DA (%)**
Ensaio sem *data augmentation*	-	-	79.90	-
Equalização do Histograma	94.74	18.57	96.46	20.73
Espelhamento	89.25	11.70	97.78	22.38
Correção *Gamma*	96.24	20.45	97.75	22.34
Alteração Canal Vermelho	97.03	21.44	98.26	22.98
Alteração Canal Verde	95.78	19.87	97.20	21.65
Alteração Canal Azul	96.47	20.74	97.18	21.63
Alteração no HUE	96.31	20.54	97.18	21.63
Alteração Saturação	96.31	20.54	97.42	21.93
Alteração Brilho	96.21	20.41	98.20	22.90
Rotação	83.74	4.81	96.77	21.11
Sharpen	96.23	20.44	97.19	21.64
Translação	88.64	10.94	96.82	21.17
Dropout	92.20	15.39	98.25	22.97

Fonte: Autoria própria

O uso de equalização do histograma, correção Gamma, *sharpen* e *dropout* por exemplo, mostraram um aumento notável na acurácia, tornando-se métodos promissores para ampliar a capacidade de detecção do modelo. Ajustes nos canais de cor também parecem ter impacto significativo na acurácia do modelo, com melhorias em todos os três canais (vermelho, verde e azul), assim como nos canais HUE, saturação e brilho.

Contudo, nem todos os métodos de *data augmentation* proporcionaram ganhos expressivos. A rotação, por exemplo, resultou na menor acurácia com *data augmentation* na base de teste (83.74%). E a translação, embora não tão prejudicial quanto a rotação, também resultou em um desempenho abaixo do esperado, uma vez que o peixe pode ser apresentado em todas as orientações e posições possíveis em uma filmagem. Isso indica que a inclusão de imagens rotacionadas ou transladadas, podem dificultar o aprendizado do modelo. Especula-se que as regiões em preto criadas na imagem quando rotacionadas ou transladadas, influenciem no aprendizado, porém isso não foi averiguado.

4.2.1 CONJUNTO DE *DATA AUGMENTATION*

Devido aos resultados promissores obtidos com as técnicas de *data augmentation*, foi proposto um teste utilizando um conjunto dessas técnicas. Inicialmente foi realizado um teste aplicando todas elas. O procedimento foi gerar, para cada amostra disponível, nove outras amostras com *data augmentation* aplicado, mais especificamente, foi determinado uma chance arbitrária de 50% de uma técnica em particular ser aplicada. Dessa maneira, cada nova amostra possui em média de 6.5 técnicas aplicadas, com desvio padrão de 1.8. Esta base foi então treinada, obtendo uma acurácia de 87.65% com *data augmentation* na base de teste e 94.49%

sem técnicas aplicadas, qual é menor de que todas as técnicas aplicadas individualmente, com exceção da técnica de rotação quando há *data augmentation* na base de treino.

Com base nestes resultados, questionou-se se o número de técnicas aplicadas em cada amostra, tem muita influência na capacidade de aprendizado. Dessa maneira, foram realizados novos ensaios similares ao anterior. Continuou-se gerando nove novas imagens para cada amostra, porém, cada nova amostra possui um e somente um método *data augmentation* aplicado e de forma aleatória. Assim, realizou-se testes, aumentando número de métodos, até que se houve-se cinco técnicas sendo aplicadas em cada amostra, obtendo os resultados na Tabela 4.

Tabela 4 – Resultados de múltiplos *data augmentation*s

Número de técnicas por amostra	AP com DA (%)	AP sem DA (%)
50% para cada técnica	87.65	94.49
1	95.66	97.48
2	93.24	96.85
3	91.84	96.30
4	89.52	95.96
5	89.13	93.85

Fonte: Autoria própria

Com base nestes resultados, observou-se que um excesso de técnicas aplicadas em uma única amostra, pode desfavorecer a generalização do modelo. Supõe-se que, devido ao número de novas imagens continuar sendo nove por amostra em todos os testes e a diversidade do modelo foi aumentando com o número de técnicas sendo aplicadas, não teve-se amostras suficientes para que o modelo generalizar-se. No entanto, este caso não foi averiguado.

4.2.2 VALIDAÇÃO EM SITUAÇÕES DE INTERESSE

Uma vez que se obteve um modelo detector de objetos satisfatório, desejou-se validar quão eficaz este é nas situações de interesse. Para isso, foram separados as amostras das situações apresentadas na Tabela 1 em conjuntos de treino e teste. Para que então pudesse se observar a diferença da desempenho nestas situações, em amostras que foram treinadas e amostras que nunca foram vistas pelo modelo. Ainda que, obteve-se modelos utilizando apenas uma única técnica de *data augmentation*, com os maiores valores de AP, para este ensaio, optou-se utilizar o modelo treinado com todas a técnicas e de melhor desempenho.

O modelo qual se aplicou apenas uma técnica, aleatória, por nova amostra, possui um valor de AP de 97.48%. Embora não seja o de melhor acurácia entre todos os modelos treinados, é o melhor obtido entre os que possuem uma maior diversidade em sua generalização, uma vez que todas as técnicas de *data augmentation* foram empregadas. Dessa maneira, obteve-se os valores da Tabela 5.

Tabela 5 – Distribuição das amostras rotuladas separadas em treino e teste

Descrição	Amostras de treino		Amostras de teste	
	Quantidade	AP (%)	Quantidade	AP (%)
Peixe completamente visível	1327	99.54	319	99.03
Peixe e pelo menos um reflexo	999	99.34	220	99.00
Peixe parcial ou totalmente oculto	153	97.62	200	95.91
Peixe parcialmente fora do quadro	72	33.19	0	-
Peixe em baixa resolução	723	96.75	241	96.94

Fonte: Autoria própria

Nota-se que em quase todas as situações, o modelo possui um ótimo desempenho, com exceção dos casos em que o peixe está parcialmente fora do quadro. Questiona-se a diferença quando comparada ao peixe que está parcialmente oculto por reflexo ou artefato, como mostra a Figura 4. Estipula-se que isso se ocorre devido a falta de informação, do contexto ao redor das regiões do peixe que estão visíveis no quadro. Porém não foi encontrado uma forma de validar este caso.

Finalmente, foram selecionados duas filmagens de dois aquários distintos, quais nunca apareceram nas bases de treino e teste. Foi realizado a detecção destes dois vídeos utilizando o mesmo modelo com valor de AP de 97.48% . Ao total, foram rotulados 1740 amostras dos 2728 quadros do vídeo com nome *CG1_1h.mp4* e 608 amostras dos 656 quadros do vídeo com nome *mts01.mp4*. Estes tiveram os valores 87.60% e 62.50% na medida AP dos vídeos respectivamente. Ambos os vídeos com o peixe detectado, estão disponíveis nos seguintes links para o YouTube CH1_1h.mp4 e mts01.mp4. Nota-se que estes valores de AP estão consideravelmente abaixo em relação ao modelo obtido, isso se da principalmente ao fato do modelo ser avaliado em amostras, que nunca estiveram presentes nem na base de treino quanto na de teste. Repara-se também, que no vídeo *mts01.mp4*, há vários quadros em que o peixe esta bem visível, porém não detectado. A hipótese é a de que o modelo ainda esta limitado para BBs muito pequenas, mas ainda é necessário investigar essa afirmação. Estes valores de 87.60 e 62.50%, em comparação aos trabalhos relacionados, estão ainda relativamente menores, considerando que na maioria dos casos a acurácia é muito próxima de 100%. Embora não tenha sido possível avaliar estes modelos com os dados utilizados neste trabalho, para realizar uma comparação assertiva, pode-se observar que ainda há espaço para melhorias, principalmente quando se trata da variabilidade dos dados para o treinamento e da evolução das tecnologias de visão computacional.

5 CONCLUSÃO

O objetivo deste trabalho foi desenvolver e avaliar um método para rastrear peixe-zebra em ambientes pouco controlados, utilizando técnicas de inteligência artificial, mais especificamente, métodos já desenvolvidos que possam ser especializados rapidamente, utilizando conceito de transferência de aprendizado.

Durante o desenvolvimento do projeto, várias metodologias foram experimentadas, que envolveram diferentes formas de rotulação das amostras e o uso de diversas técnicas de *data augmentation*. Os resultados obtidos foram significativos e permitiram obter *insights* importantes sobre a eficácia de cada abordagem.

No que diz respeito à rotulação, identificou-se que as maneiras de definir uma BB para o rótulo, podem ser pouco relevantes para a maximização da acurácia do modelo, no entanto, uma caixa delimitadora que engloba a cabeça e o corpo do peixe, e utiliza apenas uma classe, resultou na maior acurácia entre as testadas.

Em relação ao *data augmentation*, todas as técnicas testadas resultaram em um aumento significativo da acurácia do modelo. Em particular, a técnica de alteração no canal vermelho mostrou-se a mais eficaz, com uma acurácia (AP) de 98.26%, quando não há *data augmentation* na base de teste. No entanto, a técnica de rotação apresentou menor desempenho quando se considera *data augmentation* na base de teste, o que sugere que alterações na orientação do plano de fundo onde o objeto de interesse se encontra, podem influenciar a tarefa de detecção.

No geral, este projeto contribuiu para o avanço do uso de técnicas de inteligência artificial na biologia, mais especificamente, no rastreamento de animais em vídeos. Espera-se que os resultados aqui obtidos, possam servir de base para futuras pesquisas na área.

5.1 TRABALHOS FUTUROS

Este trabalho, embora tenha alcançado resultados significativos, possui muito espaço para futuras investigações. Por exemplo, novas técnicas de *data augmentation* e estratégias de rotulação podem ser exploradas. Métodos para inferir dados faltantes ou imprecisos, como por exemplo, filtro de Kalman ou redes neurais recorrentes, poderiam aprimorar bastante a acurácia para casos extremos, onde o peixe é completamente oculto por diversos quadros subsequentes. Outras melhorias que também poderiam ser investigadas, são as de averiguar o motivo das técnicas de rotação e translação não obterem resultados tão significativos quanto as outras técnicas. Assim como averiguar se aumentar o número de amostras, com múltiplos *data augmentations* aplicados, irá aprimorar o modelo. Seria interessante também, investigar os motivos do modelo ter dificuldade de detectar objetos parcialmente fora dos quadros. Novas tecnologias à serem exploradas, modelos menores, mais rápidos e robustos poderão abordar melhor objetos muito pequenos.

Além disso, o estudo poderia ser estendido para rastrear múltiplos peixes, ou até mesmo outras espécies de peixes ou animais, ampliando ainda mais o alcance das aplicações.

REFERÊNCIAS

BAI, Y.-X. *et al.* Automatic multiple zebrafish tracking based on improved hog features. **Scientific Reports**, v. 8, n. 1, p. 10884, 2018. Disponível em: https://doi.org/10.1038/s41598-018-29185-0. Acesso em: 12 de novembro de 2021.

BARREIROS, M. d. O. *et al.* Zebrafish tracking using yolov2 and kalman filter. **Scientific Reports**, v. 11, n. 1, p. 3219, 2021. Disponível em: https://doi.org/10.1038/s41598-021-81997-9. Acesso em: 12 de novembro de 2021.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. 2020. Disponível em: https://arxiv.org/abs/2004.10934. Acesso em: 12 de novembro de 2021.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. [*S.l.*]: GitHub, 2021. https://github.com/AlexeyAB/darknet Urlaccessdate = 21 de junho de 2023.

DEAKIN, A. G. *et al.* Automated monitoring of behaviour in zebrafish after invasive procedures. **Scientific Reports**, v. 9, n. 1, p. 9042, 2019. Disponível em: https://doi.org/10.1038/s41598-019-45464-w. Acesso em: 12 de novembro de 2021.

DELCOURT, J. *et al.* Video tracking in the extreme: A new possibility for tracking nocturnal underwater transparent animals with fluorescent elastomer tags. **Behavior Research Methods**, v. 43, n. 2, p. 590, 2011. Disponível em: https://doi.org/10.3758/s13428-011-0060-5. Acesso em: 12 de novembro de 2021.

MAASWINKEL, H.; ZHU, L.; WENG, W. Using an automated 3d-tracking system to record individual and shoals of adult zebrafish. **Journal of visualized experiments**, 2013. Disponível em: https://pubmed.ncbi.nlm.nih.gov/24336189. Acesso em: 12 de novembro de 2021.

PéREZ-ESCUDERO, A. *et al.* idtracker: tracking individuals in a group by automatic identification of unmarked animals. **Nature Methods**, v. 11, n. 7, p. 743–748, 2014. Disponível em: https://doi.org/10.1038/nmeth.2994. Acesso em: 12 de novembro de 2021.

QIAN, Z.-M. *et al.* An effective and robust method for tracking multiple fish in video image based on fish head detection. **BMC Bioinformatics**, v. 17, n. 1, p. 251, 2016. Disponível em: https://doi.org/10.1186/s12859-016-1138-y. Acesso em: 12 de novembro de 2021.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning, 3rd Ed.** Birmingham, UK: Packt Publishing, 2019. 748 p. ISBN 978-1789955750.

TAVARES, B.; LOPES, S. S. The importance of zebrafish in biomedical research. **Acta Médica Portuguesa**, v. 26, n. 5, 2013. Disponível em: https://www.actamedicaportuguesa.com/revista/index.php/amp/article/view/4628. Acesso em: 12 de novembro de 2021.

TAYLOR, K.; ALVAREZ, L. R. An estimate of the number of animals used for scientific purposes worldwide in 2015. **Alternatives to Laboratory Animals**, v. 47, n. 5-6, p. 196–213, 2019. PMID: 32090616. Disponível em: https://doi.org/10.1177/0261192919899853. Acesso em: 12 de novembro de 2021.

ÜSTÜNDAĞ, U. V. *et al.* White led light exposure inhibits the development and xanthophore pigmentation of zebrafish embryo. **Scientific Reports**, v. 9, n. 1, p. 10810, 2019. Disponível em: https://doi.org/10.1038/s41598-019-47163-y. Acesso em: 12 de novembro de 2021.

VILLAMIZAR, N. *et al.* Effect of lighting conditions on zebrafish growth and development. **Zebrafish**, v. 11, n. 2, p. 173–181, 2014. Disponível em: https://pubmed.ncbi.nlm.nih.gov/24367902. Acesso em: 12 de novembro de 2021.

XU, Z.; CHENG, X. E. Zebrafish tracking using convolutional neural networks. **Scientific Reports**, v. 7, n. 1, p. 42815, 2017. Disponível em: https://doi.org/10.1038/srep42815. Acesso em: 12 de novembro de 2021.

YOHANANDAN, S. **mAP (mean Average Precision) might confuse you!** 2020. Disponível em: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2. Acesso em: 12 de novembro de 2021.

ZAIDI, S. S. A. *et al.* **A Survey of Modern Deep Learning based Object Detection Models**. 2021. Disponível em: https://arxiv.org/abs/2104.11892v2. Acesso em: 12 de novembro de 2021.

ZOU, Z. *et al.* **Object Detection in 20 Years: A Survey**. 2019. Disponível em: https://arxiv.org/abs/1905.05055v2. Acesso em: 12 de novembro de 2021.