UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E COMPUTAÇÃO

ANDRÉ FELIPE DOS SANTOS CARAÍBA

Estimação de Distâncias de Objetos em Imagens de Câmeras Monoculares utilizando Redes Neurais Profundas

Goiânia
2024

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): André Felipe dos Santos Caraíba

Título do trabalho: **Estimação de Distâncias de Objetos em Imagens de Câmeras Monoculares utilizando Redes Neurais Profundas**

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 01/08/2024, às 19:52, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Andre Felipe Dos Santos Caraiba**, **Discente**, em 01/08/2024, às 19:54, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4713988** e o código CRC **9476E195**.

Referência: Processo nº 23070.016039/2024-04 SEI nº 4713988

ANDRÉ FELIPE DOS SANTOS CARAÍBA

Estimação de Distâncias de Objetos em Imagens de Câmeras Monoculares utilizando Redes Neurais Profundas

Trabalho de Conclusão apresentado à Coordenação do Curso de Engenharia de Elétrica do Escola de Engenharia Elétrica, Mecânica e Computação da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Bacharel em Engenharia de Elétrica.

Orientador: Prof. Dr. Álisson Assis Cardoso

Goiânia
2024

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Caraíba, André Felipe dos Santos
Estimação de Distâncias de Objetos em Imagens de Câmeras Monoculares utilizando Redes Neurais Profundas [manuscrito] / André Felipe dos Santos Caraíba. - 2024.
16 f.

Orientador: Prof. Dr. Alisson Assis Cardoso.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Engenharia Elétrica, Goiânia, 2024.
Bibliografia.
Inclui gráfico, tabelas.

1. Estimação de Distância. 2. Redes Neurais Profundas. 3. Câmeras Monoculares. 4. Detecção de Objetos. 5. Mapas de Profundidade. I. Cardoso, Alisson Assis, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Ao(s) 01 dia(s) do mês de agosto do ano de 2024 iniciou-se a sessão pública de defesa do Trabalho de Conclusão de Curso (TCC) intitulado "**Estimação de Distâncias de Objetos em Imagens de Câmeras Monoculares utilizando Redes Neurais Profundas**", de autoria de **André Felipe dos Santos Caraíba**, do curso de Engenharia Elétrica, do(a) Escola de Engenharia Elétrica, Mecânica e Computação da UFG. Os trabalhos foram instalados pelo(a) Dr. Álisson Assis Cardoso com a participação dos demais membros da Banca Examinadora: Dr. Flávio Henrique Teles Vieira (EMC/UFG) e do Dr. Gilberto Lopes Filho (SEINFRA/UFG). Após a apresentação, a banca examinadora realizou a arguição do(a) estudante. Posteriormente, de forma reservada, a Banca Examinadora atribuiu a nota final de **10,0**, tendo sido o TCC considerado **aprovado**.

Proclamados os resultados, os trabalhos foram encerrados e, para constar, lavrou-se a presente ata que segue assinada pelos Membros da Banca Examinadora.

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 01/08/2024, às 19:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Flavio Henrique Teles Vieira**, **Professor do Magistério Superior**, em 01/08/2024, às 19:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Gilberto Lopes Filho**, **Técnico de Laboratório**, em 01/08/2024, às 19:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4713940** e o código CRC **0E3342BB**.

Referência: Processo nº 23070.016039/2024-04 SEI nº 4713940

Estimação de Distâncias de Objetos em Imagens de Câmeras Monoculares utilizando Redes Neurais Profundas

André Caraíba
Escola de Engenharia Elétrica, Mecânica e de Computação
Universidade Federal de Goiás
Goiânia (GO), Brasil
Email: drecaraiba@gmail.com

***Resumo*—A estimação de distância é uma tarefa essencial para muitas aplicações, como navegação de robôs móveis, sistemas de navegação de veículos autônomos e dispositivos de assistência para pessoas com deficiência visual. Diferentes sensores, como LiDAR e câmeras estéreo, podem ser empregados nessa tarefa. No entanto, sensores de distância 3D, como LiDAR, possuem alto custo e apresentam problemas de sincronização. Câmeras de profundidade ou câmeras estéreo são comumente usadas, mas sofrem de dificuldades de calibração e geralmente não apresentam bons resultados em longos alcances. Câmeras monoculares oferecem uma solução econômica e versátil para realizar a estimação de distância. Com o avanço das técnicas de aprendizado profundo, diversos trabalhos propõem estimar a distância de objetos usando detecção de objetos e mapas de profundidade a partir de imagens de câmeras monoculares.**

Este trabalho se propõe a desenvolver uma rede neural capaz de estimar a distância de objetos usando imagens RGB capturadas por câmeras monoculares, foram testados diversos métodos e proposta uma nova arquitetura (CustomNet). O modelo proposto utiliza uma rede pré-treinada para estimação de profundidade, MiDaS, em conjunto com uma rede de detecção de objetos, YOLOv8, para extração de um vetor de características usado como entrada para uma Rede Neural Multicamadas que estima a distância final absoluta. Os resultados mostram que a abordagem proposta (CustomNet) supera abordagens alternativas em trabalhos anteriores, como DisNet, e supera outras abordagens desenvolvidas ao decorrer deste trabalho, como MLP simples e AlexNet. Dentre os modelos desenvolvidos neste trabalho, a rede convolucional VGG alcançou o menor erro na estimação da distância dentre os testes realizados.

Palavras-chave: estimação de distância, aprendizado profundo, detecção de objetos, câmeras monoculares, redes neurais profundas, MiDaS, YOLOv8, regressão, mapas de profundidade.

I. Introdução

A estimação de distância é uma tarefa essencial para muitas aplicações, como navegação de robôs móveis, em sistemas de navegação de veículos autônomos e dispositivos de assistência para pessoas com deficiência visual [1], [2]. Por exemplo, no contexto de navegação autônoma, a estimação de distância desempenha um papel fundamental no sistema de tomada de decisão do veículo, prevenindo colisões com outros veículos, pedestres e objetos ao redor [3].

Tem-se observado o desenvolvimento de novas tecnologias a partir de modelos de aprendizado profundo, como na detecção autônoma de objetos, classificação de objetos e segmentação [4]–[6]. Em contrapartida, comparativamente, observamos poucos estudos sendo realizados na estimação de distância de objetos.

Essa tarefa é igualmente importante, pois permite estimar a distância entre a câmera e os objetos reconhecidos na imagem, como carros, pedestres e ciclistas, fornecendo informações cruciais para o veículo evitar colisões e ajustar a velocidade para uma condução segura. Além de ser usada como dica para fusão de sensores e planejamento de trajetória. Diferentes sensores, como GPS, LiDAR [7], câmeras monoculares e estéreo [8], [9], e sensores ultrassônicos, têm sido empregados em sistemas avançados de direção, aumentando rapidamente o nível de automação do veículo. Apesar dos avanços, sistemas de detecção e estimação de distância enfrentam desafios significativos. Tecnologias baseadas em sensores de distância 3D, como LiDAR, são eficazes, mas possuem custos elevados e problemas relacionados à sincronização dos elementos ópticos e de imagem [10].

Geralmente, são empregados o uso de câmeras de profundidade ou câmeras estéreo para estimar distâncias, mas elas sofrem de alguns problemas, como a dificuldade de calibração estéreo para obter a relação de posição entre as duas câmeras [11]. Todo esse processo é tedioso e requer o uso de câmeras adicionais. Por outro lado, câmeras monoculares são soluções promissoras devido à sua relação custo-benefício e capacidade de integrar a estimação de distância com outras tarefas de visão computacional.

Atualmente, várias soluções e algoritmos têm emergido para estimar distâncias com eficiência. Nesse sentido, observa-se um destaque para os algoritmos de inteligência artificial e aprendizado de máquina, principalmente no campo de técnicas de aprendizado profundo [3], [11].

No caso de imagens que contêm múltiplos objetos, a detecção desses objetos torna-se uma tarefa essencial e indispensável. A detecção de objetos é tratada normalmente como um problema de aprendizado supervisionado, pois é necessário determinar a qual classe determinado objeto pertence. Um algoritmo de classificação pode identificar um objeto em uma imagem depois de ser treinado em um *dataset* de imagens,

apropriadamente rotuladas com as classes dos seus objetos. Redes Neurais Artificiais e suas variantes, como Redes Neurais Convolucionais (Convolutional Neural Networks - CNNs), são consideradas estado da arte para detecção de objetos e outras tarefas de classificação. Por exemplo, CNNs baseadas em região, ou R-CNNs [12]. Por sua vez, *You Only Look Once*, ou YOLO [13], é uma família de técnicas para detecção de objetos projetada para oferecer uma resposta rápida em aplicações em tempo real. Usualmente, a saída desses métodos são uma caixa delimitadora ao redor do objeto com um rótulo descritivo que classifica o objeto.

A distância do objeto pode ser estimada posteriormente, usando as diferentes dimensões das caixas delimitadoras decorrentes das diferentes distâncias ao objeto. Esse problema pode ser tratado como um problema de regressão, onde o relacionamento entre a caixa delimitadora e a distância pode ser aprendido.

Com a propagação das técnicas de aprendizado profundo, diversos trabalhos têm sido propostos para estimação de mapas de profundidade em câmeras monoculares a partir de uma única imagem. A aparência dos objetos muda significativamente com sua pose. Estimar um mapa de profundidade a partir de uma imagem 2D é um passo importante para determinar as coordenadas 3D de objetos presentes em uma cena. A estimativa de profundidade monocular baseada em métodos de aprendizado profundo pode ser implementada através de técnicas de aprendizado supervisionado ou não supervisionado [14], [15].

Na Figura 1 pode-se visualizar o modelo de estimação de distância de objetos proposto neste trabalho.

Foi proposto o desenvolvimento de uma Rede Neural para estimar distâncias de objetos utilizando imagens RGB estáticas capturadas por câmeras monoculares, a partir de mapas de profundidade usando Redes de Neurais Profundas, MiDaS [16], [17], juntamente com Redes Neurais Profundas para detecção de objetos, YOLOv8 [18]. E um regressor baseado em Redes Neurais Multicamadas (Multilayer Perceptron - MLP) customizado para realizar a tarefa de regressão final para predizer a distância do objeto.

A arquitetura criada neste trabalho se baseia em trabalhos anteriores, como DisNet [19], que utiliza um vetor de características extraídas a partir das caixas delimitadoras por um detector de objetos YOLOv3 e, logo em seguida, essas características são passadas para uma MLP que realiza a estimação de distância do objeto. Diferente do método proposto por eles, a metodologia proposta neste trabalho utiliza uma etapa adicional para extrair características da região de detecção do objeto a partir de um mapa de profundidade da imagem obtido por meio de um modelo de profundidade monocular.

Para esse fim, foi criado um *dataset* de treinamento e várias arquiteturas de Redes Neurais foram testadas em tarefas de estimação simplificadas, como enviar a imagem contendo o objeto como entrada da rede para que ela possa predizer a distância até o objeto. Nesse sentido, foram testados modelos baseados em Redes Neurais Multicamadas, MLP, Redes Neurais Convolucionais já bem conhecidas na literatura, como AlexNet [20] e VGG [21], bem como a própria DisNet.

Os resultados experimentais demonstram que o modelo proposto neste trabalho (CustomNet) é capaz de predizer distâncias com desempenho superior em relação a abordagens alternativas como DisNet, e também dentre as outras arquiteturas desenvolvidas neste trabalho, como a MLP e AlexNet. Enquanto a arquitetura convolucional VGG obteve o melhor desempenho dentre todos os modelos comparados.

Para quantificar as métricas de desempenho deste trabalho e outras abordagens, foi construído um *dataset* em ambiente de simulação, CoppeliaSim [22], onde as distâncias ao objeto foram calculadas usando funcionalidades do próprio simulador. Foram empregadas métricas de avaliação para a tarefa de estimação de distância. Os resultados quantitativos

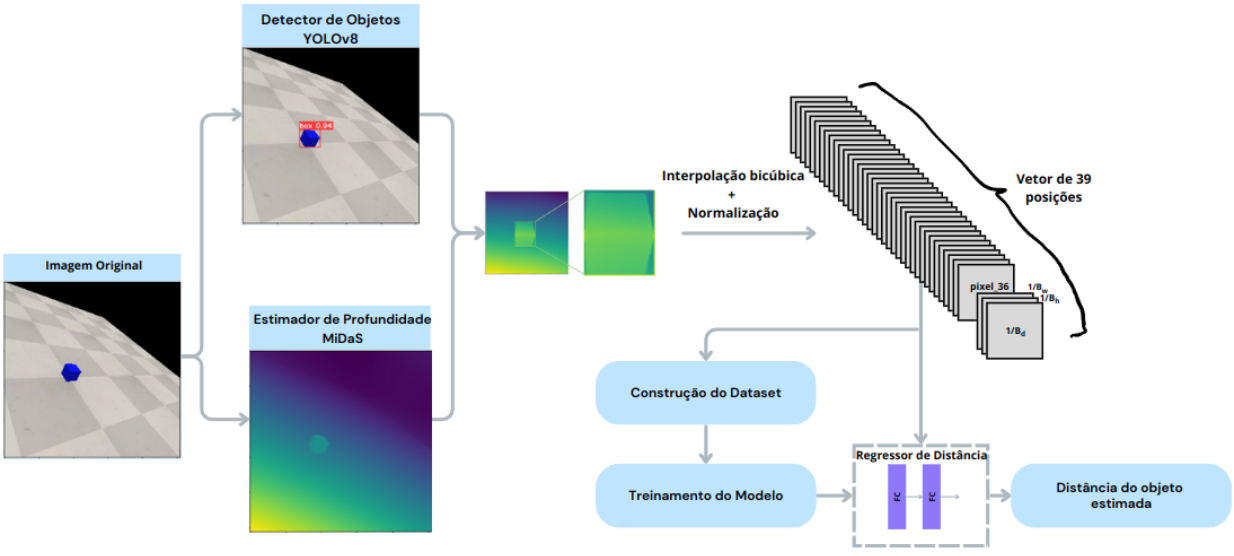


Figura 1. Arquitetura Customizada Proposta para estimação da distância de objetos por meio de uma câmera monocular

foram reportados, e alguns exemplos foram visualizados para comparação qualitativa. Os resultados experimentais demonstram que métodos baseados em redes neurais profundas podem prever com sucesso distâncias a objetos com desempenho superior a abordagens alternativas, como DisNet.

II. Revisão Bibliográfica

Nesta seção, é realizado um resumo a respeito das técnicas e métodos empregados ao decorrer deste trabalho, além de uma revisão de trabalhos anteriores tratando da tarefa de estimação de distância de objetos.

A. Redes Neurais Profundas

Aprendizagem Profunda ou *Deep Learning*, é um campo do Aprendizado de Máquina, que emprega algoritmos para processar dados e imitar o processamento feito pelo cérebro humano. Eles possuem o objetivo de modelar relações complexas a partir de grandes conjuntos de dados.

As Redes Neurais Profundas utilizam camadas de modelos matemáticos de neurônios para processar dados, compreender a fala humana e reconhecer objetos visualmente. Elas fazem isso através da obtenção de valores numéricos, chamados de pesos matemáticos, que são usados como parâmetros ajustáveis. Esses valores são multiplicados pelos dados de entrada, que são adicionados e passam por funções de ativação contidas em cada neurônio da rede. As informações são passadas através de cada camada, com a saída da camada anterior fornecendo entrada para a próxima camada, como representado na Figura 2.

A primeira camada em uma rede é chamada de camada de entrada, enquanto a última camada é chamada de camada de saída. Todas as camadas entre as duas são referidas como camadas ocultas. Se denomina Redes Neurais Multicamadas ou *Multilayer Perceptron* (MLP) aquelas redes que possuem pelo menos duas camadas ocultas totalmente conectadas, além das camadas de entrada e de saída.

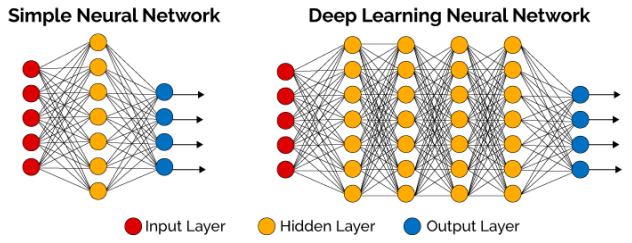


Figura 2. Rede Neural simples e Rede Neural Profunda [23]

As Redes Neurais Profundas têm sido usadas em diversos campos de aplicação. Em visão computacional, elas frequentemente usam Redes Neurais Convolucionais (Convolutional Neural Network - CNN) como um paradigma padrão. Como espinha dorsal dessas redes, elas incluem uma série de camadas convolucionais, camadas de agrupamento (*pooling*) e conexões residuais para extrair características, reduzir a dimensionalidade e prevenir o sobreajuste.

As CNNs têm alcançado bons resultados em uma variedade de tarefas de Visão Computacional, como classificação de imagens [24], detecção de objetos [24] e representação semântica [25].

B. Detecção de Objetos

A detecção de objetos é uma técnica importante dentro da visão computacional. Ela permite que sistemas detectem a caixa delimitadora em torno dos objetos. É um dos problemas mais desafiadores em visão computacional, pois o modelo de detecção de objetos é treinado para identificar objetos dentro de um conjunto de dados e não consegue identificar um objeto que não seja rotulado durante o treinamento. Essa é considerada uma de suas limitações. No entanto, os modelos de detecção de objetos treinados sempre podem ser treinados novamente para obter conhecimento sobre novos objetos.

Técnicas de detecção de objetos são usadas em aplicações como navegação de veículos autônomos, sistemas de vigilância por vídeo ou contagem de multidões. Alguns algoritmos populares de detecção de objetos são YOLO [13], R-CNN [12] e MobileNet [26]. Neste trabalho, o algoritmo YOLO foi escolhido para detecção de objetos. Esta escolha baseia-se na sua capacidade de equilibrar precisão e velocidade de inferência.

O modelo YOLO de detecção de objetos possui diversas versões lançadas ao decorrer dos anos. O artigo original do YOLOv1 foi publicado em 2015, e os modelos subsequentes foram publicados até o surgimento das versões mais recentes. O YOLOv8 [27] foi lançado no início do ano de 2023, sendo considerado um dos melhores devido à sua eficiência e desempenho.

A detecção de objetos é um problema importante dentro da Visão Computacional. Esse problema se refere a fornecer uma imagem como entrada do modelo e receber como saída uma caixa delimitadora de um objeto-alvo, o rótulo do objeto-alvo e uma probabilidade de confiança, que representa a probabilidade de que a classe correta exista na caixa. Existem duas maneiras principais de aplicar redes neurais profundas em tarefas de detecção de objetos: detecção de um estágio e de dois estágios.

Na detecção de um estágio, o método trata a tarefa de detecção de objetos como um único problema e realiza a localização do objeto e a classificação dele diretamente na imagem. Esse tipo de método usualmente produz muitos candidatos a caixas delimitadoras em cada posição da imagem de entrada e, então, aplica um classificador para determinar se existe um objeto em cada caixa candidata. YOLO [13] é uma das estruturas de detecção de objetos mais representativas que utiliza detecção de um estágio e famosa por sua habilidade em suportar predições em tempo real em reproduções de vídeo.

Na detecção de dois estágios, o método é decomposto em duas etapas. Primeiro, a imagem de entrada é enviada através da rede para predizer candidatos a caixas delimitadoras que podem conter o objeto. Depois, a imagem contida dentro da caixa é usada como entrada para classificação e ajuste fino. Nesse tipo de solução, Fast-RCNN é uma estrutura

amplamente utilizada e pode fornecer boa acurácia [28]. Por conter mais etapas, as técnicas de detecção de dois estágios geralmente possuem tempo de inferência superior comparados ao de um único estágio.

Técnicas de detecção de objetos têm sido empregadas em diversas aplicações, como em imagens térmicas para identificação de doenças [29].

O uso da detecção de objetos também tem sido realizado em diversos trabalhos anteriores para estimação de distâncias, como a DisNet [19], onde foi proposta uma estrutura que utiliza características extraídas das caixas delimitadoras detectadas em conjunto com uma rede neural multicamadas para estimar a distância de objetos em linhas de trem ferroviário.

C. Estimação de Profundidade

A estimação de profundidade também representa um desafio fundamental em visão computacional. Existem dois métodos diferentes para alcançar esse objetivo: modelos de aprendizado profundo supervisionado e não supervisionado.

Prever a profundidade de uma única imagem é uma tarefa naturalmente difícil, pois a mesma imagem pode projetar múltiplas profundidades concebíveis. Para prevenir isso, a profundidade predita precisa ter algum relacionamento com as cores da imagem. Existem várias abordagens, como fim a fim [30], amostragem de profundidade não paramétrica [31], fluxo óptico (*optical flow*) [32], e transferência de aprendizado [33]. Em métodos supervisionados, a profundidade original do mapa da imagem é usada para treinamento junto com as imagens coloridas. Isso ajuda o sistema a aprender de forma mais eficaz e, portanto, os resultados de métodos supervisionados usualmente possuem desempenho superior que métodos não supervisionados. Entretanto, na prática, construir mapas de profundidade originais em tempo real é inviável, sendo necessário o uso de câmeras estéreo ou LiDAR.

Para evitar esse problema, métodos não supervisionados têm sido usados para treinar os sistemas; apenas imagens originais e modelos pré-treinados, como DenseNet [34], ResNet [24] e ImageNet [35], são necessários. Em relação aos métodos não supervisionados, várias abordagens para a estimativa de profundidade foram propostas, como Redes Generativas Adversariais [36], informações temporais [37] e redes de pose separadas [38].

Um modelo que vem ganhando destaque nessa área é o MiDaS [16], considerado o estado da arte atualmente. Ele combina técnicas de aprendizado profundo com uma abordagem de treinamento supervisionado e não supervisionado, utilizando um conjunto diversificado de dados, incluindo pares de imagens estéreo, sensores de profundidade e informações de estrutura a partir do movimento (Structure for Motion - SfM).

D. Câmeras Monoculares

Câmeras monoculares são famosas por sua portabilidade, familiaridade, preço reduzido e aplicabilidade generalizada. Uma câmera monocular é um dispositivo de câmera que utiliza somente uma lente. Esse tipo de câmera captura imagens bidimensionais de uma cena 3D, possuindo informações da imagem apenas de uma única perspectiva.

Apesar dos seus benefícios, câmeras monoculares não conseguem estimar diretamente a distância de objetos devido à falta de informação de profundidade. Para esse fim, geralmente são usadas câmeras estéreo. Elas conseguem estimar distâncias de objetos devido ao princípio da visão estereoscópica, que é semelhante ao sistema de visão humana. A estereoscopia envolve o uso de duas câmeras posicionadas com uma separação horizontal fixa, permitindo a captura de duas imagens simultâneas da mesma cena a partir de dois pontos de vista ligeiramente diferentes. A diferença entre essas duas imagens é conhecida como disparidade. Ao analisar a disparidade entre pontos correspondentes nas duas imagens, é possível calcular a profundidade ou distância dos objetos em relação às câmeras. Esse método baseia-se na geometria epipolar e na triangulação, onde a distância entre as câmeras e o ângulo de disparidade fornecem as informações necessárias para determinar a profundidade [39].

Uma câmera monocular captura apenas uma imagem de uma cena, e, sem a informação de profundidade fornecida pela estereoscopia, é inviável determinar diretamente a distância dos objetos apenas com base na imagem bidimensional. A estimativa de distância com câmeras monoculares requer informações adicionais ou suposições sobre a cena, como o uso de pistas visuais (tamanho conhecido do objeto, perspectiva, oclusão) ou técnicas de aprendizado de máquina e aprendizado profundo que tentam inferir a profundidade com base em um treinamento prévio para essa finalidade.

E. Trabalhos anteriores

Muitos trabalhos foram realizados para tentar contornar o problema de estimação de distâncias em câmeras monoculares. Uma das maneiras clássicas de estimar a distância de um dado objeto foi converter os pontos da imagem em uma coordenada de vista aérea (*Bird's eye view*) usando o algoritmo de mapeamento de perspectiva inversa (Inverse Perspective Mapping - IPM) [40]. Outra abordagem [41] utilizou um modelo de regressão baseado em Máquinas de Vetores de Suporte (Support Vector Machine - SVM) a partir da largura e altura da caixa delimitadora dos objetos na imagem. Em DisNet [19], foi feita uma tentativa de construir uma rede neural onde os autores usaram a YOLO para prever as caixas delimitadoras em vez de usar as características das imagens aprendidas para estimar distâncias. Duman et al. [42] propuseram uma metodologia semelhante, usando caixas delimitadoras de corpo e face para estimar a distância de pessoas. Zhu et al. propuseram um método [3] usando R-CNN para estimação de distância diretamente a partir das imagens RGB. Masoumian et al. [43] propuseram uma metodologia que utiliza redes de detecção de objetos em conjunto com uma rede encoder-decoder para estimação de profundidade da imagem. No presente trabalho, além das características das caixas delimitadoras do objeto detectado na imagem, foram extraídas características de profundidade da imagem e, ao final, ambas as características de detecção e do mapa de

profundidade foram enviadas para uma rede MLP regressora final.

III. Metodologias

A rede neural proposta neste trabalho é mostrada na Figura 1. O procedimento realizado é descrito da seguinte forma: foram utilizadas duas redes em conjunto para extração de características e uma última arquitetura como regressor final, para estimar distância. A YOLOv8 foi o modelo usado para detecção de objetos, e o modelo MiDaS para estimar profundidade de objetos. A partir de uma imagem capturada, a profundidade de todos os elementos contidos na cena é extraída a partir do MiDaS. De forma semelhante, a YOLO detecta e localiza objetos retornando sua caixa delimitadora. A partir da localização exata de cada objeto definido pela caixa, a região correspondente é detectada na imagem de profundidade estimada. A partir das informações da caixa delimitadora, uma região de interesse é selecionada no mapa de profundidade, e finalmente é calculado um vetor de características que é passado como entrada para uma rede neural MLP, usada como regressor final para estimar a distância até o objeto. Nas próximas seções, serão dados mais detalhes a respeito de todos os procedimentos realizados.

Neste trabalho, o objetivo é que a rede neural desenvolvida seja treinada para estimar a distância de um objeto. Para isso, foi criado um ambiente de simulação contendo uma câmera monocular e um objeto. As imagens da cena foram extraídas e as distâncias do objeto até a câmera foram obtidas através de funcionalidades próprias do simulador para a construção do *dataset* de treinamento.

Após isso, foram utilizadas e desenvolvidas diferentes arquiteturas de redes neurais, dentre elas uma Rede Perceptron Multicamadas (Multilayer Perceptron - MLP), Redes Convolucionais onde foram testadas as arquiteturas AlexNet e VGG-16, uma implementação da DisNet baseada em [19], onde foi utilizada a rede YOLOv8 para detecção de objetos e extração de caixas delimitadoras do objeto na imagem. Essas características foram passadas como entradas para uma segunda MLP customizada.

Os modelos escolhidos e desenvolvidos foram treinados sobre o mesmo conjunto de dados e, ao final, eles foram avaliados e comparados a partir de diferentes métricas.

A. Criação do Dataset

A criação do *Dataset* consistiu, primeiramente, em desenvolver um ambiente de simulação responsável por simular o comportamento real de uma câmera monocular para obtenção das imagens dos objetos e extração da distância da câmera até o objeto em cada imagem capturada.

Além disso, foi necessário realizar as anotações do objeto em um conjunto de imagens para criação de um *dataset* para o treinamento da YOLOv8, usada para realizar a detecção de objetos.

1) Ambiente de Simulação: A primeira etapa da criação do *dataset* consistiu na criação do ambiente de simulação. O simulador CoppeliaSim [22] foi utilizado nessa ocasião. O CoppeliaSim é uma plataforma de simulação robótica amplamente utilizada para simulação de robôs e sistemas em ambientes virtuais 3D.

A plataforma inclui uma série de sensores virtuais (como câmeras, LiDAR e sensores de proximidade) para emular as condições do mundo real. Ela se destaca por ser muito intuitiva e altamente customizável, sendo possível interagir com ela através de scripts Python.

Nessa etapa, foi criada uma cena que contém um objeto, no caso, foi testado um bloco azul com dimensões de 10 cm x 10 cm x 10 cm. Juntamente, foi configurada uma câmera monocular com resolução de 256 x 256 pixels no formato RGB. Na Figura 3 é mostrada uma imagem contendo a representação visual de como a cena foi construída.

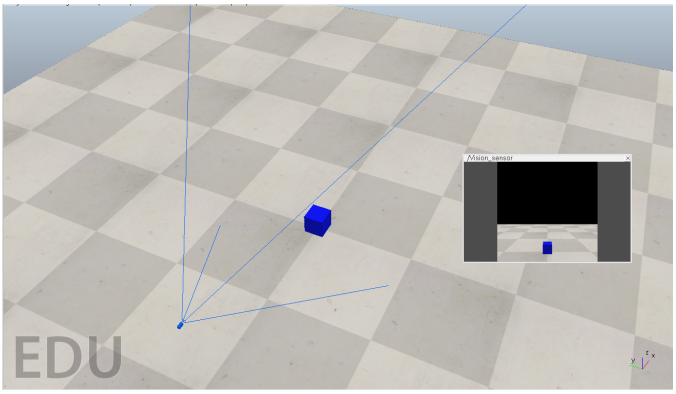


Figura 3. Cena criada em ambiente de simulação para criação de Dataset próprio

No desenvolvimento do *dataset*, é importante considerar um conjunto de imagens bastante diversificadas para a finalidade de obter a distância do objeto.

Dessa maneira, foi desenvolvido um script que posiciona a câmera em diferentes posições, apontando para o centro do objeto com uma distância máxima de 3 metros para captura da imagem e uma distância mínima de 0,1 metro. O valor da distância foi obtido utilizando uma funcionalidade própria do simulador que calcula a distância entre dois objetos, através da distância entre os dois pontos mais próximos de cada objeto, nesse caso, a câmera e o bloco.

Para adicionar mais aleatoriedade nas imagens, um ruído gaussiano na direção da câmera em relação ao objeto foi adicionado a cada captura. Para cada posição da câmera, foi capturada uma foto e extraída a distância do objeto usando funcionalidades próprias do CoppeliaSim. Cada imagem foi salva como um arquivo no formato .png com sua respectiva distância em um arquivo de texto. Ao final, foram capturadas cerca de 1894 imagens com suas respectivas distâncias alvo, como pode ser visualizado na Figura 4.

Para utilização do *dataset*, foram separados 70% dos dados para treinamento, 15% para teste e 15% para validação. Resultando, ao final, em 1325 arquivos para treino, 284 para validação e 285 de teste.

2) Anotação das imagens: Além disso, ainda foi necessário anotar essas imagens para utilizar no treinamento da YOLOv8.

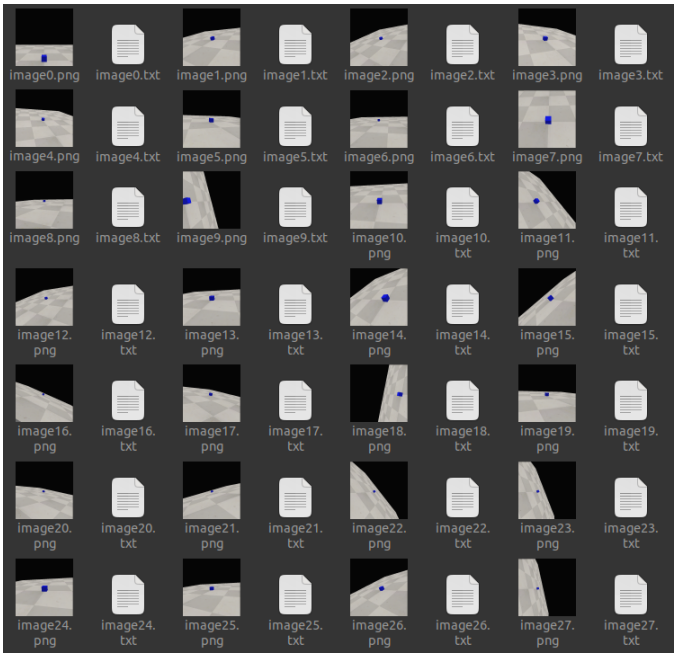


Figura 4. Dataset criado em ambiente de simulação contendo imagens do bloco em várias posições e ângulos diferentes

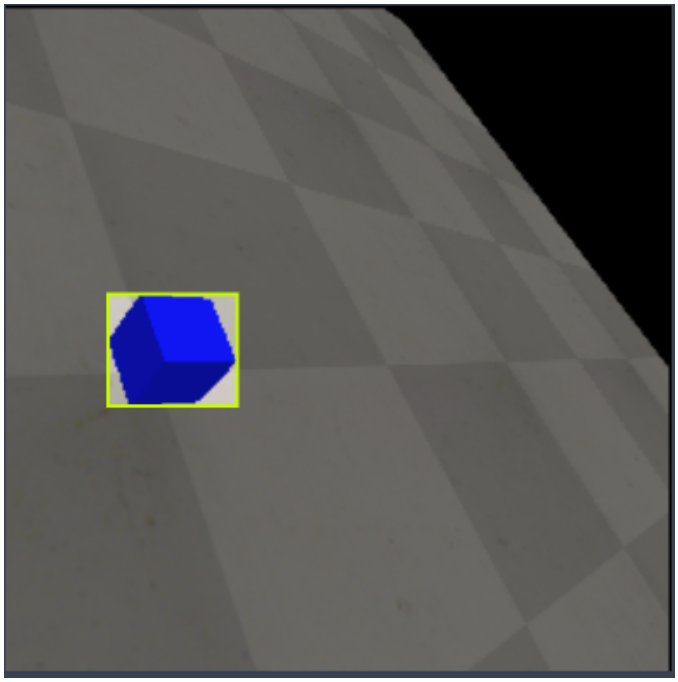


Figura 5. Exemplo de imagem anotada na plataforma Roboflow, a caixa delimitadora engloba todo o espaço ocupado pelo cubo

Para isso, foi utilizada a plataforma de anotações de imagens RoboFlow [44]. As imagens foram anotadas de forma a construir um retângulo que se encaixa em todas as dimensões do cubo. Ao final, foram anotadas 771 imagens selecionadas dentre as 1894 originais. Dessas 771, 540 foram usadas para o treinamento, 154 para validação e 77 para teste. Na Figura 5 um exemplo de anotação realizado no *dataset* é apresentado.

B. Estrutura da Rede Neural

Como citado anteriormente, foram testadas 5 arquiteturas diferentes: uma primeira que utiliza somente MLP, outras duas baseadas em CNNs, sendo elas AlexNet e VGG, a DisNet e a CustomNet.

As redes MLP, AlexNet e VGG foram pensadas de forma a executar a tarefa de estimação de distância da maneira mais simples possível. Nessa situação, as arquiteturas atuam como um regressor diretamente na imagem de entrada. Ou seja, os 256x256 pixels são passados como entrada da rede neural, e a distância em metros até o objeto é fornecida como alvo durante a sua fase de treinamento. Para o teste, é necessária somente a imagem RGB da câmera monocular contendo o objeto na imagem.

Observe que, nesse caso, essa estrutura possui uma limitação quanto à sua utilização. Ela utiliza a imagem como um todo, não identificando os objetos contidos na imagem individualmente, como realizado em abordagens que utilizam redes de detecção de objetos. De toda forma, foi observado um excelente resultado usando essa metodologia, o que significa que elas podem ser aplicadas para algumas situações específicas, caso necessário.

Além desses modelos, também foi realizada uma implementação da DisNet, baseada em trabalhos anteriores na literatura [19]. A DisNet consiste na utilização de características extraídas das caixas delimitadoras do objeto, que são passadas como entrada para um regressor que estima a distância final, nesse caso uma MLP. Essas caixas delimitadoras são obtidas através de um modelo de detecção de objetos. O estudo de referência utiliza o modelo YOLOv3; entretanto, para essa implementação, foi utilizada a YOLOv8, sendo esta a versão, aos recente disponível no momento da realização dos experimentos.

Por fim, também é apresentada uma arquitetura pessoal desenvolvida nesse trabalho, a CustomNet, que utiliza ideias desenvolvidas na DisNet, juntamente com a utilização de características de mapas de profundidade, como ilustrado na Figura 1. Nessa seção, serão detalhadas as arquiteturas de cada rede utilizada, bem como detalhes sobre o seu treinamento.

1) ***MLP****:* Foi desenvolvida uma MLP com duas camadas ocultas de 512 e 128 neurônios, sendo utilizada a função de ativação ReLU. A ReLU foi escolhida nessa ocasião porque ela evita que a rede escolha valores negativos em sua saída, condizente para o problema de estimar distâncias absolutas. Além de facilitar o processo de treinamento e evitar o desaparecimento do gradiente.

Para que a rede aprenda através dos dados, foi utilizado como função de perda o erro quadrático médio (MSE - Mean Squared Error), apresentado por,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

onde n é o número de exemplos no conjunto de dados, y_i é o valor real e $\hat{y}_i$ é o valor predito pela rede para o i-ésimo exemplo. A escolha da MSE como função de perda se deve ao fato de que ela é uma métrica simples que mede diretamente o erro médio quadrático entre as previsões do modelo e os valores reais. A natureza quadrática da MSE penaliza mais severamente os erros maiores. Isso é importante para a estimativa de distâncias, onde erros grandes podem ter um impacto significativo. Pensando no contexto de navegação, um erro grande poderia significar um acidente. Ao penalizar mais os erros maiores, a MSE ajuda a treinar a rede para ser mais precisa.

Além de utilizar a MSE como função de perda para treinar a rede, também foi utilizada a raiz do erro quadrático médio (RMSE - Root Mean Squared Error), apresentada por,

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

para avaliar o desempenho da rede durante o treinamento, onde n é o número de exemplos no conjunto de dados, y_i é o valor real e $\hat{y}_i$ é o valor predito pela rede para o i-ésimo exemplo.

A escolha da RMSE como métrica de avaliação se deve principalmente à sua facilidade de interpretação, uma vez que ela está na mesma unidade dos valores que estão sendo preditos, ou seja, a distância em metros. Isso facilita a interpretação dos resultados, permitindo que se compreenda diretamente o erro médio das previsões em termos de unidades de distância. Assim como a MSE, a RMSE penaliza mais severamente os erros maiores devido à sua natureza quadrática. Ela é sensível a *outliers*, o que significa que grandes erros terão um impacto significativo na métrica.

Com isso, a MLP foi treinada durante 500 épocas, com *batchsize* de 16 utilizando MSE como função de perda. Foi escolhida uma taxa de aprendizado de 0,0001 com otimizador SGD. O *dataset* criado foi utilizado para treinar e avaliar o modelo, onde foram extraídas a função de perda durante o treinamento bem como a RMSE. Além disso, foi utilizado um dropout de 30% na camada de entrada para evitar o sobreajuste do modelo. Na Figura 6, o gráfico obtido durante o treinamento do modelo pode ser visualizado.

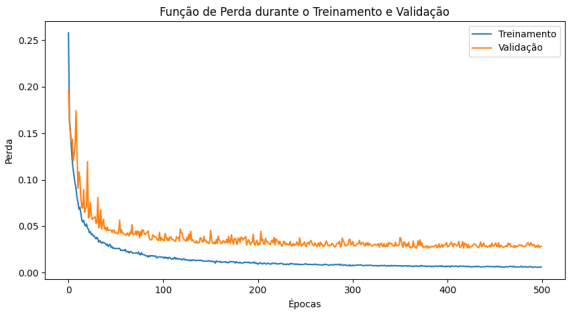


Figura 6. Função de perda observada nos dados de treino e validação durante o processo de treinamento da MLP

*2) **AlexNet**:* A arquitetura da AlexNet já é bem conhecida na literatura [20]. O grande diferencial dela são as camadas convolucionais que conseguem lidar melhor com imagens. Esse modelo é composto por cinco camadas convolucionais, seguidas por três camadas totalmente conectadas.

Para adaptar o modelo AlexNet à tarefa específica desse trabalho, foram realizadas algumas modificações. Primeiramente, a última camada totalmente conectada foi ajustada para corresponder a apenas uma saída, que será a distância calculada até o objeto. Além disso, o tamanho da entrada da rede foi ajustado para corresponder ao tamanho da imagem. Adicionalmente, incorporou-se mais uma camada de *average pooling* à sua estrutura interna. Manteve-se a função de ativação ReLU, devido à sua capacidade de lidar com valores negativos e auxiliar na convergência dos pesos da rede.

Na etapa de treinamento, foram utilizadas 700 épocas, com *batch size de 16*, utilizando MSE como função de perda. Foi escolhida uma taxa de aprendizado de 0,001 com otimizador SGD. O mesmo *dataset* usado para treinar a MLP e desenvolvido nesse trabalho foi utilizado para treinar e avaliar o modelo. Isso é importante para que, ao final, seja possível comparar os resultados obtidos em cada rede de forma que a influência do resultado final não seja proveniente de dados de treinamento diferentes. Durante essa etapa de treinamento, para verificar o desempenho da rede, foram extraídas a própria função de perda (MSE) e a RMSE a cada época. A cada época, as duas métricas também foram calculadas sobre os dados de validação para verificar sobreajuste da rede e seu desempenho em dados não vistos anteriormente. Os resultados obtidos no treinamento são apresentados na Figura 7.

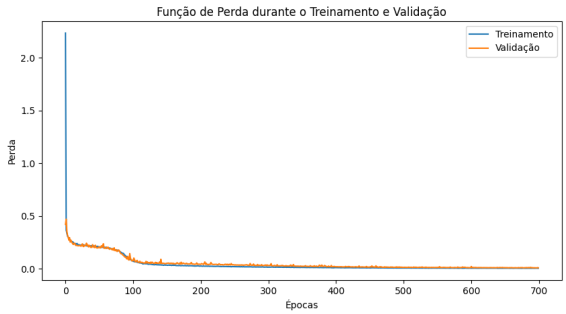


Figura 7. Função de perda observada nos dados de treino e validação durante o processo de treinamento da AlexNet

*3) **Rede DisNet**:* A DisNet pode ser pensada em duas etapas distintas. A primeira etapa consiste em uma rede de detecção de objetos que identifica o objeto e obtém a sua caixa delimitadora. A segunda etapa utiliza, como entrada, algumas características extraídas das caixas delimitadoras para estimar a distância final usando uma rede MLP.

Na primeira etapa, foi utilizada a YOLOv8. Ela é uma evolução das versões anteriores da YOLO, conhecida por sua eficiência, velocidade e precisão. Para a segunda etapa, que envolve a estimativa final da distância, foi utilizada uma MLP. Ela consiste de uma camada de entrada igual ao número de

características do vetor de entrada, duas camadas ocultas, cada uma com 150 neurônios com função de ativação ReLU, e um neurônio de saída para estimar a distância final.

Foi passado como entrada da rede um vetor de características $\mathbf{v}$ que contém as características das caixas delimitadoras dos objetos detectados na imagem da câmera, como apresentado na Figura 8. Para construir o *dataset* para treinamento dessa rede, as caixas delimitadoras foram extraídas nas imagens RGB de treinamento pela rede de detecção já treinada. Dessa forma, foram obtidos ao final 1325 vetores de características.

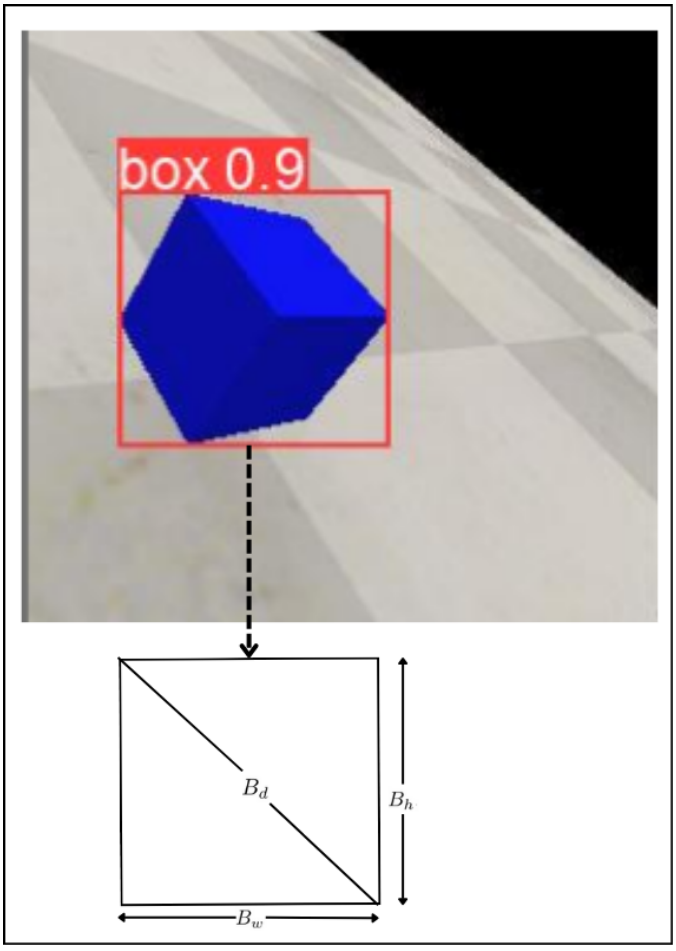


Figura 8. Objeto detectado pela YOLOv8 juntamente com as dimensões das caixas delimitadoras que serão usadas para criação do vetor de características

Para cada caixa delimitadora extraída do objeto, foi calculado um vetor de características $\mathbf{v}$ de três dimensões:

$$\mathbf{v} = \left[\frac{1}{B_h}, \frac{1}{B_w}, \frac{1}{B_d}\right]$$

onde cada característica é respectivamente:

- **Altura,** B_h: altura da caixa delimitadora do objeto em pixels / altura da imagem em pixels.
- **Largura,** B_w: largura da caixa delimitadora do objeto em pixels / largura da imagem em pixels.
- **Diagonal,** B_d: Diagonal da caixa delimitadora do objeto em pixels / Diagonal da imagem em pixels.

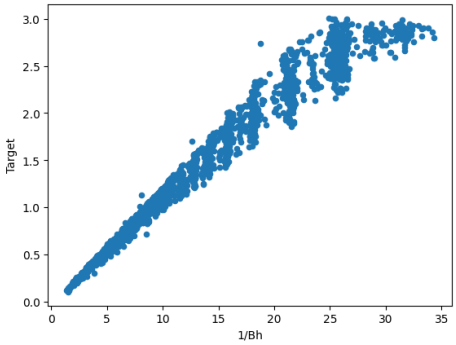
Utilizar as proporções relativas ao tamanho da imagem, como B_h, B_w e B_d, permite que o modelo treinado seja reutilizado com uma variedade de câmeras, independentemente da resolução da imagem.

Essas características foram selecionadas porque foi observado um relacionamento intrínseco entre elas e a distância real até o objeto em um intervalo de 0,2-3 m. Como pode ser observado na Figura 9. Geometricamente, espera-se que o tamanho da caixa delimitadora do objeto seja menor à medida que a distância do objeto aumenta. Portanto, é esperado que o inverso do tamanho da caixa delimitadora aumente à medida que a distância também aumenta. Inspecionar os dados confirma que esse é o caso e sugere que o relacionamento é aproximadamente linear. Esse é um bom embasamento para utilizar esse vetor de características para o treinamento da rede.

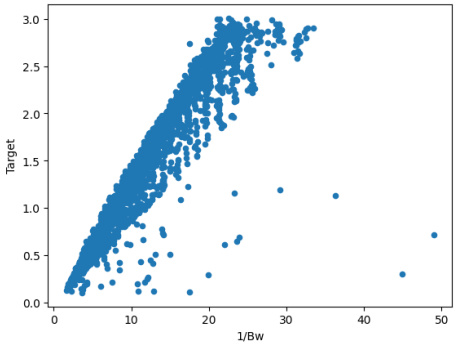
Para a etapa de treinamento, foram utilizadas 200 épocas utilizando MSE como função de perda, foi escolhida uma taxa de aprendizado de 0,001 com otimizador SGD. O *dataset* contendo as distâncias reais e o vetor de *features* foi usado, seguindo a mesma divisão de treino, validação e teste utilizadas nas outras arquiteturas, nessa ocasião o modelo foi treinado com um *batch size* de 4 elementos. Durante o treinamento, foram avaliadas a MSE e a RMSE a cada época nos dados de treino e de teste a fim de verificar o desempenho da rede e sobreajuste, como pode ser observado na Figura 10

4) ***Rede Proposta - CustomNet****:* Para a CustomNet, proposta nesse trabalho, foi usado um modelo pré-treinado de estimação de profundidade, MiDaS. O MiDaS possui diversas versões, com maior e menor quantidade de parâmetros. Nessa ocasião foi escolhido o modelo MiDaS v3 - Hybrid. O modelo foi testado no *dataset* próprio construído, explicado na seção de *Criação do Dataset*, para estimar a profundidade de cada imagem nos dados de treinamento. Ao mesmo tempo, o modelo de detecção de objetos detecta e localiza a caixa delimitadora do objeto. Baseado nas coordenadas da caixa delimitadora, a caixa exata foi localizada na imagem de profundidade predita pelo MiDaS. Na realidade, o modelo de profundidade estima os mapas de disparidade que representam a relação entre o movimento entre os pixels de entrada da imagem e os pixels da imagem de saída. Então, os mapas de disparidade são transformados em mapas de profundidade.

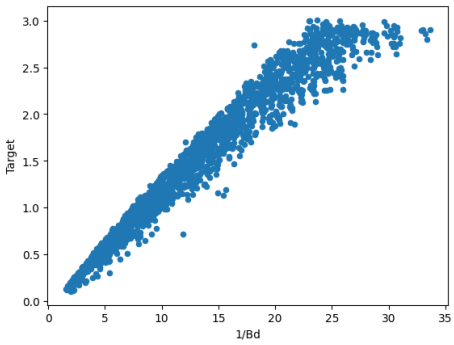
Logo após isso, os valores de todos os pixels dentro da caixa delimitadora no mapa de profundidade foram extraídos. Então, um algoritmo de interpolação bicúbica foi utilizado para redimensionar o tamanho dos pixels dentro da caixa delimitadora para uma imagem de tamanho fixo 6x6, que é achatado para um vetor de 36 características contendo as informações de interesse contidas no mapa de profundidade de forma resumida. Essa etapa é necessária para a próxima etapa, que de forma parecida com a DisNet utiliza uma rede MLP como regressor para estimar a distância final até os objetos através das características extraídas das caixas delimitadoras, como na DisNet, e adicionalmente com as características obtidas do mapa de profundidade, sendo passadas 39 características para



(a)



(b)



(c)

Figura 9. Distância do objeto em relação as características de entrada da caixa delimitadora. (a) Distância em relação ao inverso altura, (b) Distância em relação ao inverso do tamanho da largura e (c) Distância em relação ao inverso da diagonal.

a rede. Nessa ocasião a MLP regressora possui duas camadas ocultas de 150 neurônios. Foram utilizadas 250 épocas no processo de treinamento, com taxa de aprendizado de 0,001 utilizando otimizador SGD, *batchsize* de tamanho 4 e utilizada a MSE como função de perda.

Os resultados de treinamento podem ser visualizados na Figura 11:

5) ***Rede VGG****:* Por fim, foi avaliada a arquitetura baseada em Redes Neurais Convolucionais, VGG [21]. Essa também é uma arquitetura clássica na literatura e se destaca pela grande quantidade de camadas convolucionais que ela possui. Existem dois padrões comumente usados, a VGG-16 possuindo 16 camadas convolucionais e a VGG-19 com 19 camadas. Nesse

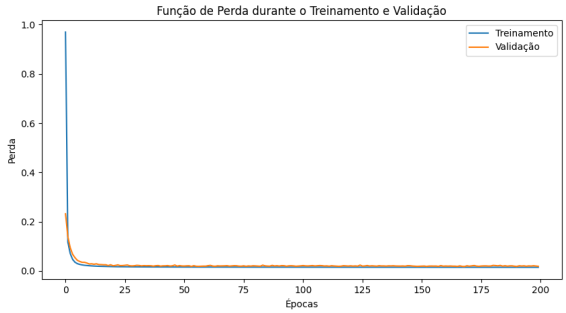


Figura 10. Função de perda observada nos dados de treino e validação durante o processo de treinamento da DisNet

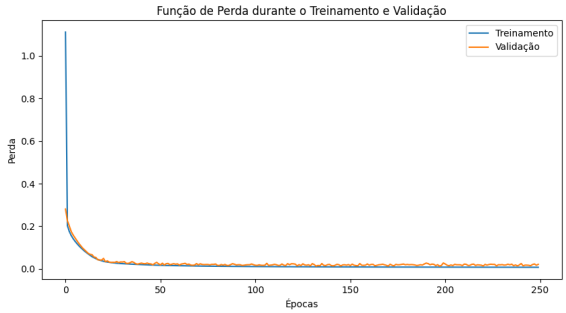


Figura 11. Função de perda observada nos dados de treino e validação durante o processo de treinamento da CustomNet

trabalho foi utilizado a VGG-16.

Esse modelo é relativamente grande, possuindo quantidade de parâmetros na ordem de centenas de milhares. Devido a isso, houve uma dificuldade maior no treinamento, portanto, algumas mudanças foram realizadas no fluxo de treinamento em comparação às arquiteturas apresentadas anteriormente.

Devido à quantidade de parâmetros, essa rede facilmente levava a explosões no gradiente, dificultando o treinamento. O tempo de treinamento requerido foi superior às demais arquiteturas.

Para resolver esses problemas, a rede foi treinada utilizando o otimizador Adam. Este otimizador possui características adaptativas. Tal propriedade reduz significativamente a necessidade de uma extensa busca manual pela taxa de aprendizagem ideal, facilitando o processo de treinamento. Adicionalmente, para diminuir o tempo de treinamento, o número de épocas utilizadas foi de 150, com tamanho do *batch* igual a 16. Os dados foram treinados com uma taxa de aprendizado de 0,001, onde foi usada como função de perda a MSE.

Além disso, foram aplicadas técnicas no treinamento como Dropout nas camadas totalmente conectadas para evitar o sobreajuste.

Os resultados da rede durante o treinamento foram avaliados em conjunto com os dados de validação, como apresentado na Figura 12. Observa-se no gráfico que as duas curvas de validação e teste estão próximas, indicando que o modelo não

se ajustou aos dados de treinamento.

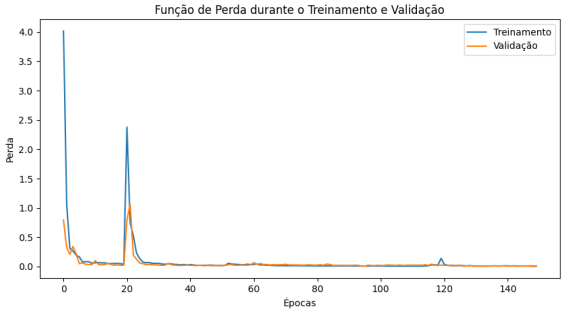


Figura 12. Função de perda observada nos dados de treino e validação durante o processo de treinamento da VGG

Todo o processo de treinamento dos modelos apresentados foi realizado em uma estação de trabalho com arquitetura x64, equipada com 16GB de memória RAM e uma GPU NVIDIA GeForce RTX 3070 Ti. Para o desenvolvimento e treinamento dos modelos, utilizou-se a biblioteca PyTorch (versão 2.2.1).

IV. Resultados

Após o treinamento das arquiteturas, seus desempenhos foram avaliados utilizando um conjunto comum de dados de teste, com o objetivo de compará-las. Foram utilizadas 285 imagens retiradas do simulador, juntamente com as respectivas distâncias verdadeiras. Os resultados são apresentados a seguir para cada arquitetura testada.

Nesta etapa, foram utilizadas outras métricas além da MSE e RMSE mencionadas anteriormente, com o objetivo de obter resultados mais robustos e garantir uma avaliação do sistema sob diferentes perspectivas.

A partir do valor predito e do valor real, foi possível avaliar o resultado da rede a partir de métricas como o erro médio absoluto (MAE - Mean Absolute Error), que realiza a média da diferença absoluta entre o valor predito e o valor real. A equação matemática que o descreve pode ser apresentada por,

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

A escolha do MAE como métrica de avaliação nesta etapa se deve a alguns fatores. Primeiro, é uma métrica simples e intuitiva que mede diretamente o erro médio absoluto entre as previsões do modelo e os valores reais. Isso facilita a interpretação dos resultados e a identificação de áreas onde o modelo precisa de melhorias. Outra característica do MAE é que ele não é excessivamente influenciado por grandes erros individuais, implicando em uma menor sensibilidade dos resultados devido a *outliers* presentes, fornecendo uma visão mais generalizada dos resultados.

Em conjunto com o MAE, também podemos utilizar o erro percentual médio absoluto (MAPE - Mean Absolute Percentage Error). Essa métrica basicamente fornece uma medida de erro em termos percentuais, o que facilita a interpretação dos resultados, permitindo uma compreensão imediata do desempenho do modelo em termos relativos ao valor real, sendo útil para análises comparativas. Além disso, o MAPE penaliza erros com base no tamanho relativo do valor real. Isso significa que erros em previsões de valores menores são mais significativos do que erros em valores maiores. A equação que descreve o MAPE é apresentada por,

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4}$$

Por fim, foi utilizado o coeficiente de determinação R^2. Essa é uma métrica amplamente usada em estatística e avaliação de modelos para avaliar a qualidade de um modelo de regressão. O R^2 quantifica a proporção da variabilidade na variável dependente que é explicada pela variável independente no modelo. A equação para calcular o R^2 é descrita como,

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5}$$

O valor de R^2 varia entre 0 e 1, onde o valor 1 indica que o modelo explica perfeitamente toda a variabilidade dos dados de resposta em torno da média, ou seja, todas as previsões do modelo são exatamente iguais aos valores reais. Quando é 0, indica que o modelo não explica nenhuma variabilidade dos dados de resposta ao redor da média. Neste caso, o modelo preditivo não é melhor que simplesmente usar a média dos valores reais como a previsão.

A partir dessas métricas os resultados para cada arquitetura foram compilados e apresentados na Tabela I.

Tabela I
Resultados obtidos para cada modelo

Modelo	MSE	RMSE	MAE	MAPE	R^2
MLP	0,0353	0,188	0,094	8,73%	0,945
AlexNet	0,0175	0,132	0,085	11,24%	0,973
DisNet	0,0162	0,127	0,098	8,76%	0,975
CustomNet	0,0107	0,103	0,077	6,23%	0,984
VGG	**0,0036**	**0,0598**	**0,045**	**4,70%**	**0,994**

A. Resultados MLP

Alguns dos resultados encontrados para o modelo de MLP podem ser visualizados a partir da Tabela I.

Ao traçar o gráfico dos valores reais em relação aos valores preditos sobre todos os exemplos dos dados de teste, é possível analisar como o modelo se comporta sobre diferentes perspectivas, como demonstrado na Figura 13, onde valor do coeficiente de determinação pode ser obtido.

Para esse modelo, foi observado um MAE de 0,094 m. Esse resultado significa que, em média, nossas predições erraram 9,4 cm do valor real da distância. Isso representa um erro percentual médio igual ao valor do MAPE, igual a 8,7%.

A partir dessa figura, é possível ver como o modelo avaliou cada amostra individualmente dentro do conjunto de teste. A partir dessa visualização, é possível observar o desempenho do modelo para cada região de distância e observar a presença de

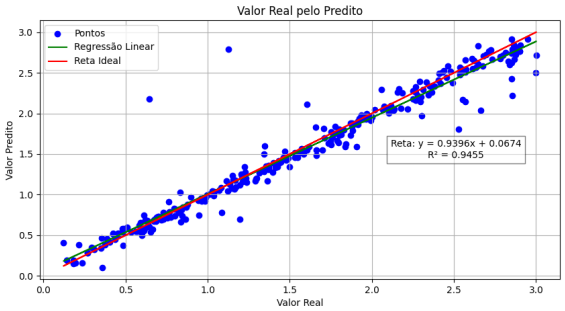


Figura 13. Gráfico da relação entre o valor real e o valor predito pelo modelo de MLP sobre os dados de teste

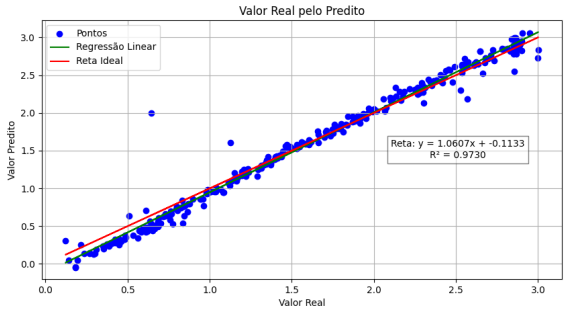


Figura 14. Gráfico da relação entre o valor real e o valor predito pelo modelo AlexNet sobre os dados de teste

valores discrepantes, *outliers*. Além disso, é possível observar o ajuste dos dados a partir do coeficiente de determinação.

Como observado no gráfico, existe a presença de muitos *outliers*. Apesar disso, de forma geral, os valores preditos são muito bem explicados pelos valores reais, uma vez que o valor de R^2 é de 0,945, próximo de 1.

O modelo também estimou de forma incorreta alguns valores negativos para objetos a pequenas distâncias. Esse comportamento não deveria ser esperado, uma vez que a distância ao objeto é um valor absoluto e maior que zero.

Os resultados obtidos podem ser explicados, em parte, pela incapacidade do modelo em generalizar os dados. Isso pode ser observado pela função de perda durante o treinamento do modelo. Ela mostra que os valores MSE e RMSE nos dados de validação foram superiores aos obtidos nos dados de treinamento; o desempenho nos dados de teste confirma isso.

Foram realizados vários ajustes a fim de melhorar o resultado do modelo para evitar o sobreajuste, como alterações na taxa de aprendizado, mudanças no otimizador utilizado, e mudanças na estrutura da rede, adicionando e removendo camadas e alterando o número de neurônios em cada camada.

Os resultados encontrados, ainda assim, mostram que o modelo não conseguiu capturar os padrões nos dados o suficiente para conseguir realizar generalizações.

B. Resultados AlexNet

De forma semelhante, os resultados encontrados para o modelo de AlexNet podem ser visualizados a partir da Tabela I.

Ao traçar a curva dos valores reais em relação aos valores preditos sobre todos os exemplos dos dados de teste, é possível analisar como o modelo se comporta sobre diferentes perspectivas, como apresentado na Figura 14.

Foi observado um MAE de 0,085 m, esse resultado significa que, em média, nossas predições erraram 8,5 cm do valor real da distância, isso representa um erro percentual médio igual ao valor do MAPE, igual a 11,2%.

Esses resultados foram interessantes porque mostram algumas coisas. Pode-se dizer que, a partir do MAE e do MAPE, a rede teve pior desempenho em média do que a MLP. Apesar disso, os valores de RMSE e MSE foram melhores, sendo encontrados menores valores.

Isso pode ser explicado analisando a Figura 14, onde é possível observar como o modelo avaliou cada amostra individualmente dentro do conjunto de teste e ver o ajuste dos dados a partir do coeficiente de determinação encontrado.

A quantidade de valores de erros acima do normal ou *outliers* foi inferior à quantidade encontrada pela MLP. Como as métricas RMSE e MSE levam muito em consideração os *outliers*, isso explica por que o modelo AlexNet teve valores inferiores nessas métricas, apesar de que, na média, não tenha estimado valores tão próximos dos valores reais. Isso também explica um valor de R^2 superior ao encontrado no modelo de MLP.

Os resultados obtidos mostram que a AlexNet teve menor quantidade de valores de erros acima do normal, entretanto não foi capaz de estimar com tanta fidelidade os valores das distâncias reais até o objeto em média, observado pelo maior valor do MAPE. Isso pode ter acontecido devido aos dados de testes utilizados, ou por uma deficiência do modelo em se ajustar perfeitamente aos dados representativos do problema; possivelmente, uma maior quantidade de dados de treinamento ajudaria o modelo a convergir mais rapidamente para o resultado desejado.

De forma geral, é possível dizer que esse modelo se mostrou mais robusto ao possuir menos *outliers*, apresentando resultados bastante sólidos, entretanto ele não se ajustou bem à tarefa de predição como o modelo de MLP. Isso pode ser observado comparando as retas de regressão linear com a reta ideal para cada modelo. No modelo de MLP, é possível observar uma maior aproximação entre as duas retas do que o observado no AlexNet.

C. Resultados DisNet

Nessa seção, são apresentados os resultados finais encontrados para a Rede DisNet, disponíveis na Tabela I.

A DisNet pode ser estruturada em duas etapas: a rede de detecção de objetos e o regressor de distância como MLP. Os resultados obtidos no treinamento do YOLOv8 para detecção de objetos podem ser visualizados na Figura 15.

O modelo de detecção foi treinado no *dataset* próprio de imagens anotadas. Nenhum pré-processamento foi realizado

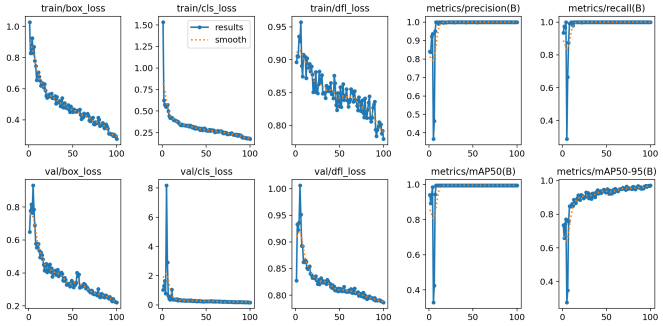


Figura 15. Resultados de treinamento da YOLOv8 no dataset próprio para detecção de objetos

nas imagens e o modelo, por fim, foi treinado em 100 épocas. Os resultados foram excelentes, com um mAP50 de 0,995 e mAP95 de 0,969 e tempo de inferência de 0,5 ms.

Para extrair mais informações acerca do resultado da arquitetura MLP usada como regressor, foi plotado um gráfico dos valores reais em relação aos valores preditos pelo modelo para cada imagem contida nos dados de teste, conforme apresentado na Figura 16.

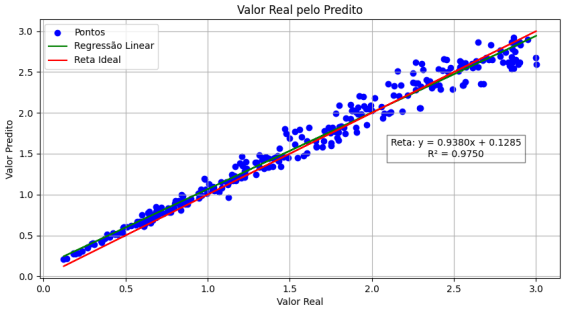


Figura 16. Gráfico da relação entre o valor real e o valor predito pela DisNet sobre os dados de teste

Foi observado um MAE de 0,098 m, ou seja, em média as predições erraram 9,8 cm do valor real da distância. Isso representa um erro percentual médio igual ao valor do MAPE, igual a 8,76%.

Isso pode ser observado pelo baixo valor do MSE e RMSE, respectivamente iguais a 0,0175 e 0,132. Esse efeito também é notado a partir da Figura 16. Nela, é possível observar como o modelo avaliou cada amostra individualmente. A partir desse gráfico, notamos a ausência de valores discrepantes de predição em relação aos valores reais, diferente da MLP que produziu muitos *outliers*, e nesse quesito foi superior a AlexNet, que também apresentou valores discrepantes.

Entretanto, ao observar o gráfico, nota-se que, à medida que a câmera se distancia do objeto e a distância aumenta, o modelo tende a ter pior desempenho. Os valores preditos começam a ter uma alta variância a partir de 1,5 metros de distância, fazendo com que os valores estimados não se aproximem dos valores reais.

Uma possível explicação para isso é obtida através dos gráficos de correlação das variáveis que o modelo utiliza em relação ao valor de saída, como foram observados. É possível notar uma baixa variância no valor das distâncias para curtas distâncias e, à medida que a distância ao objeto aumenta, essa variância também aumenta. A rede DisNet capturou essas características do vetor de entrada da rede.

A partir desses resultados, é possível concluir que a DisNet foi capaz de estimar com segurança os valores da distância até o objeto, sem apresentar erros muito discrepantes. A rede demonstrou uma menor quantidade de *outliers*, indicando uma maior consistência nas estimativas. Entretanto, o valor do MAPE sugere que, em média, a fidelidade dos valores de distância real até o objeto estimados pela DisNet foi equivalente àquela obtida pela MLP.

Isso pode ter acontecido devido aos dados de testes utilizados, ou por uma deficiência do modelo em se ajustar perfeitamente aos dados representativos do problema. Possivelmente, uma maior quantidade de dados de treinamento ajudaria o modelo a convergir mais rapidamente para o resultado desejado.

De forma geral, é possível dizer que esse modelo se mostrou mais robusto a cometer menor quantidade de estimativas de valores distantes dos valores esperados; o baixo valor do RMSE confirma isso. De forma geral, apresentou resultados bastante sólidos. Entretanto, deve-se tomar cuidado ao usá-lo para estimar objetos a longas distâncias, pois as estimativas começam a apresentar altas variâncias, não representando fielmente os valores reais.

D. Resultados CustomNet

Pode-se observar na Figura 17 alguns resultados qualitativos da rede proposta, demonstrados em alguns exemplos do *dataset* próprio construído em ambiente de simulação, incluindo a profundidade estimada na imagem, os resultados da detecção de objetos usando YOLOv8 e a parte relevante da estimação de profundidade, usadas para construção das características de entrada do modelo.

O modelo utilizado para estimação de profundidade possui várias versões disponíveis, com diferentes tamanhos. Dentre eles, foram testados os modelos MiDaS v3 - Large, MiDaS v3 - Hybrid e MiDaS v2.1 - Small. Dentre eles, a versão Small, apesar de possuir um tempo de menor, produziu resultados insatisfatórios para a aplicação. Em contrapartida, o modelo de maior tamanho produziu resultados semelhantes ao modelo de tamanho médio, porém com um tempo de inferência muito superior. Dessa maneira, o modelo MiDaS v3 - Hybrid foi escolhido por possuir boa acurácia e tempo de inferência mediano.

Assim como realizado nos modelos anteriores, na Figura 18 pode-se visualizar o gráfico relacionando os valores preditos no conjunto de teste em relação aos valores reais, adicionalmente contém uma reta de regressão nos pares de pontos para comparação dos resultados.

A partir do gráfico observado e dos resultados apresentados na tabela, nosso modelo foi capaz de superar em desempenho a

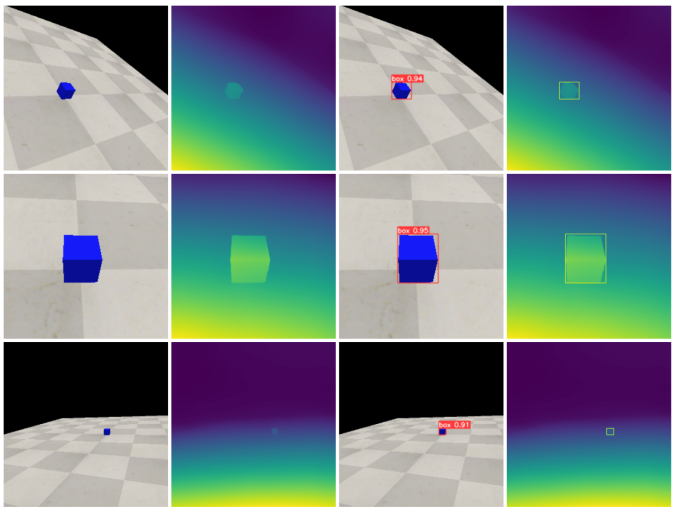


Figura 17. Etapas de processamento utilizadas na construção das features do modelo CustomNet

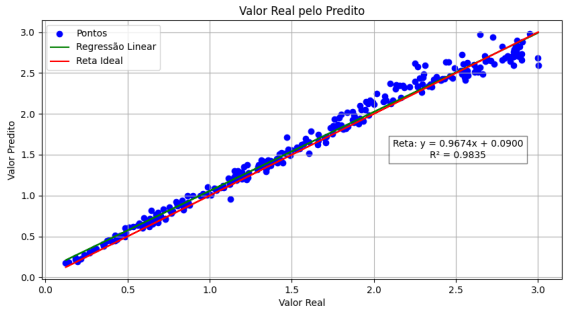


Figura 18. Gráfico da relação entre o valor real e o valor predito pelo modelo customizado, CustomNet, sobre os dados de teste

rede MLP, AlexNet e DisNet em todas as métricas. Possuindo um MAE de 7,7 cm e MAPE de 6,23%.

Além disso, nota-se que a rede não apresentou estimativas discrepantes em relação aos valores reais, desempenhando melhor que a DisNet nesse aspecto.

Em contrapartida, foi observado um aumento significativo no tempo de inferência da rede, devido à etapa extra de estimação de profundidade. Uma melhoria que pode ser realizada nesse sentido seria realizar a estimação de profundidade e a detecção de objetos em processamentos paralelos em vez de realizá-los de forma sequencial.

E. Resultados VGG

Por fim, são apresentados os resultados encontrados para a rede neural convolucional VGG. Todos os resultados obtidos estão disponíveis na Tabela I.

Assim como realizado para os resultados anteriores, foi plotado um gráfico da relação entre os valores reais e os preditos pelo modelo VGG, podendo ser observados na Figura 19.

A rede VGG obteve excelentes resultados, como pode ser visto na tabela e pelo gráfico gerado. Com um MAE de 0,045, produzindo predições que erram em média apenas 4,5 cm

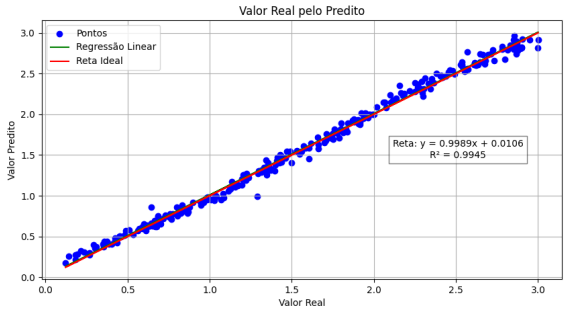


Figura 19. Gráfico da relação entre o valor real e o valor predito pelo modelo VGG sobre os dados de teste

do valor real da distância, isso representa um erro percentual médio igual ao valor do MAPE, de 4,70%.

Avaliando o resultado dessa arquitetura em comparação às anteriores, ela foi a que, de longe, melhor desempenho teve nos dados de teste em todos os quesitos.

Ao avaliar o MSE e RMSE, são observados valores bem inferiores, indicando que a rede quase não estimou valores discrepantes da realidade. A rede produziu resultados que explicam bastante a relação entre os dados preditos e os valores reais, confirmados a partir do alto valor do R^2, de 0,994, superior às outras arquiteturas apresentadas.

Os resultados realmente são positivos, mas devido à alta quantidade de parâmetros que a VGG possui, o tempo necessário para realizar inferências e o consumo de recursos como memória e armazenamento superior aos modelos AlexNet e MLP, sendo um ponto relevante a se considerar ao utilizar esse modelo. Em aplicações com baixa capacidade de processamento e armazenamento, algumas das redes apresentadas anteriormente podem ser mais apropriadas, a depender da aplicação.

Isso pode ser observado pelo baixo valor do MSE e RMSE, respectivamente iguais a 0,0175 e 0,132. Esse efeito também é notado a partir da Figura 19. Nela é possível observar como o modelo avaliou cada amostra individualmente. A partir desse gráfico, notamos a ausência de valores discrepantes de predição em relação aos valores reais, diferente da MLP que produziu muitos *outliers*, e nesse quesito se superou a AlexNet, que também apresentou valores discrepantes.

F. Comparação dos Resultados

Essa seção se dedica a realizar uma comparação final entre os modelos desenvolvidos neste trabalho.

A rede MLP apresentou um baixo erro médio absoluto, o que significa que ela se aproxima muito bem do valor da distância real ao objeto. Entretanto, ela possui muitos valores discrepantes, o que acaba afetando as outras métricas de forma negativa.

A rede AlexNet consegue estimar os valores com mais robustez. Entretanto, apresenta um erro médio absoluto maior entre todos os modelos, indicando que o modelo não apresenta resultados tão precisos quanto os outros em média, apesar de serem satisfatórios.

A DisNet, diferente dos modelos de MLP e AlexNet, não apresenta valores discrepantes e possui um baixo erro médio absoluto, assim como a MLP, indicando uma boa taxa de erro. Entretanto, à medida que a distância ao objeto aumenta, o erro aumenta significativamente.

O modelo proposto, CustomNet, alcançou bons resultados, superando em todos os sentidos os modelos de MLP, AlexNet e o próprio DisNet, usado como referência a ser superada. Apesar dos melhores resultados, observou-se um tempo de inferência superior às demais devido às etapas de processamento extras usadas para estimar a profundidade da cena e computar as características de profundidade.

Dentre todas as arquiteturas, a rede VGG apresentou os melhores resultados em todas as métricas. Em contrapartida, o consumo de recursos para utilização da rede é elevado. E ela não é capaz de detectar objetos, sendo seu uso limitado a cenas que possuam apenas um objeto.

A partir desses resultados, pode-se concluir que o modelo que obteve o melhor desempenho foi a VGG. Em contrapartida, essa rede possui desvantagens intrínsecas em comparação a arquiteturas que realizam detecção de objetos. Dessa maneira, a CustomNet se mostrou uma rede versátil, alcançando excelentes resultados a curtas, médias e longas distâncias, superando a DisNet em todos os sentidos. A DisNet obteve bons desempenhos apenas em curtas distâncias. A rede AlexNet apresentou resultados medianos. Por fim, a rede MLP não conseguiu generalizar bem os dados, podendo cometer muitos erros indesejados em dados de teste.

V. Conclusões

Nesse trabalho foram apresentados e comparados modelos de Redes Neurais Profundas para estimativa de distâncias em imagens de câmeras monoculares. Foi desenvolvido um modelo próprio para essa tarefa. Para esse fim, foram testadas arquiteturas baseadas em Redes Multicamadas, Redes Convolucionais e Redes para detecção de objetos, em conjunto com Redes de Estimação de Profundidade. Os resultados apresentados mostram que é possível estimar de forma confiável a distância de objetos a partir de uma única câmera monocular RGB utilizando Redes Neurais Profundas. Com estimativas de distâncias com erro médio inferior a 10 cm. Várias aplicações podem se beneficiar desse trabalho.

Além dissso, a arquitetura proposta nesse trabalho se mostrou superior ao modelo base [19], nos dados de teste do *dataset* construído. Entretanto, podem ser realizados melhorias em relação ao seu processamento acontecer de forma paralela em vez de sequencial para diminuir o tempo de processamento.

Trabalhos futuros podem estar relacionados à generalização dos resultados em dados que se aproximam melhor de ambientes reais. Uma possibilidade é usar o simulador CARLA [45], ou utilização de datasets com grandes conjuntos de imagens como KITTI [46] não se limitando a apenas um objeto em particular, mas vários.

Também podem ser realizadas melhorias no processo de obtenção das características que não foram abordadas nesse trabalho, como a utilização de redes neurais codificadoras *encoders* para obter a representação das características do mapa de profundidade em vez de fazer uma simples interpolação.

Agradecimentos

Gostaria de expressar minha gratidão a todas as pessoas que contribuíram para minha jornada durante a graduação e realização desse trabalho.

Agradeço à minha família por todo o suporte e apoio necessários para que eu pudesse realizar essa graduação.

Agradeço ao Professor Dr. Alisson Assis Cardoso por sua orientação e apoio constante ao longo do desenvolvimento deste trabalho.

Também expresso minha gratidão a tudo que pude vivenciar durante a faculdade, que ajudou a construir parte do que sou hoje. Dentre eles, ao Núcleo de Robótica Pequi Mecânico, onde tive a oportunidade de fazer pesquisa na área de robótica e vivenciar experiências únicas. Também ao Cerise (Centro de Excelência em Redes sem Fio e Serviços Avançados), onde tive a oportunidade de desenvolver projetos envolvendo navegação autônoma de robôs e sistemas imersivos.

Por fim, gostaria de agradecer a todos os professores da Escola de Engenharia Elétrica, Mecânica e de Computação da UFG e por todos os seus conhecimentos compartilhados comigo.

Referências

[1] N. Tripathi e S. Yogamani, "Trained trajectory based automated parking system using visual SLAM on surround view cameras", *arXiv preprint arXiv:2001.02161*, 2020.

[2] S. Zafar, M. Asif, M. B. Ahmad et al., "Assistive devices analysis for visually impaired persons: A review on taxonomy", *IEEE Access*, vol. 10, pp. 13 354–13 366, 2022.

[3] J. Zhu e Y. Fang, "Learning object-specific distance from a monocular image", em *Proceedings of the IEEE/CVF International Conference on computer vision*, 2019, pp. 3839–3848.

[4] A. Teichman e S. Thrun, "Practical object recognition in autonomous driving and beyond", em *Advanced Robotics and its Social Impacts*, IEEE, 2011, pp. 35–38.

[5] M. Yoshioka, N. Suganuma, K. Yoneda e M. Aldibaja, "Real-time object classification for autonomous vehicle using LIDAR", em *2017 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, IEEE, 2017, pp. 210–211.

[6] H. Hu, J. Gu, Z. Zhang, J. Dai e Y. Wei, "Relation networks for object detection", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3588–3597.

[7] K. Zhang, J. Xie, N. Snavely e Q. Chen, "Depth sensing beyond lidar range", em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1692–1700.

[8] N. Mayer, E. Ilg, P. Hausser et al., "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[9] H. Xu e J. Zhang, "Aanet: Adaptive aggregation network for efficient stereo matching", em *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1959–1968.

[10] H. G. Olanrewaju e W. O. Popoola, "Effect of synchronization error on optical spatial modulation", *IEEE Transactions on Communications*, vol. 65, n.º 12, pp. 5362–5374, 2017.

[11] H. Liang, Z. Ma e Q. Zhang, "Self-supervised object distance estimation using a monocular camera", *Sensors*, vol. 22, n.º 8, p. 2936, 2022.

[12] S. Ren, K. He, R. Girshick e J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, n.º 6, pp. 1137–1149, 2016.

[13] J. Redmon, S. Divvala, R. Girshick e A. Farhadi, "You only look once: Unified, real-time object detection", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[14] S. Abdulwahab, H. A. Rashwan, M. A. Garcia, M. Jabreel, S. Chambon e D. Puig, "Adversarial learning for depth and viewpoint estimation from a single image", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, n.º 9, pp. 2947–2958, 2020.

[15] C. Shu, K. Yu, Z. Duan e K. Yang, "Feature-metric loss for self-supervised learning of depth and egomotion", em *European Conference on Computer Vision*, Springer, 2020, pp. 572–588.

[16] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler e V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer", *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, n.º 3, pp. 1623–1637, 2020.

[17] R. Ranftl, A. Bochkovskiy e V. Koltun, "Vision transformers for dense prediction", em *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 179–12 188.

[18] J. Terven, D.-M. Córdova-Esparza e J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas", *Machine Learning and Knowledge Extraction*, vol. 5, n.º 4, pp. 1680–1716, 2023.

[19] M. A. Haseeb, J. Guan, D. Ristic-Durrant e A. Gräser, "DisNet: A novel method for distance estimation from monocular camera", *10th Planning, Perception and Navigation for Intelligent Vehicles (PPNIV18), IROS*, 2018.

[20] A. Krizhevsky, I. Sutskever e G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, vol. 25, 2012.

[21] K. Simonyan e A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

[22] E. Rohmer, S. P. N. Singh e M. Freese, "CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework", em *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, www.coppeliarobotics.com, 2013.

[23] D. S. Academy, *Deep Learning Book Brasil, Capítulo 3 - O que são Redes Neuraus Artificiais Profundas ou Deep Learning?*, https://www.deeplearningbook.com.br/o-que-sao-redes-neurais-artificiais-profundas/, Acesso em: Jul, 2024.

[24] K. He, X. Zhang, S. Ren e J. Sun, "Deep residual learning for image recognition", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] O. Ronneberger, P. Fischer e T. Brox, "U-net: Convolutional networks for biomedical image segmentation", em *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, Springer, 2015, pp. 234–241.

[26] A. G. Howard, M. Zhu, B. Chen et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017.

[27] G. Jocher, A. Chaurasia e J. Qiu, *Ultralytics YOLO*, versão 8.0.0, jan. de 2023. URL: https://github.com/ultralytics/ultralytics.

[28] R. Girshick, "Fast r-cnn", em *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[29] R. McManus, L. A. Boden, W. Weir et al., "Thermography for disease detection in livestock: A scoping review", *Frontiers in veterinary science*, vol. 9, p. 965 622, 2022.

[30] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari e N. Navab, "Deeper depth prediction with fully convolutional residual networks", em *2016 Fourth international conference on 3D vision (3DV)*, IEEE, 2016, pp. 239–248.

[31] K. Karsch, C. Liu e S. B. Kang, "Depth extraction from video using non-parametric sampling", em *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, Springer, 2012, pp. 775–788.

[32] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy e T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[33] I. Alhashim e P. Wonka, "High quality monocular depth estimation via transfer learning", *arXiv preprint arXiv:1812.11941*, 2018.

[34] G. Huang, Z. Liu, L. Van Der Maaten e K. Q. Weinberger, "Densely connected convolutional networks", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li e L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", em *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[36] A. Pilzer, D. Xu, M. Puscas, E. Ricci e N. Sebe, "Unsupervised adversarial depth estimation using cycled generative networks", em *2018 international conference on 3D vision (3DV)*, IEEE, 2018, pp. 587–595.

[37] V. M. Babu, K. Das, A. Majumdar e S. Kumar, "Undemon: Unsupervised deep network for depth and ego-motion estimation", em *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1082–1088.

[38] T. Zhou, M. Brown, N. Snavely e D. G. Lowe, "Unsupervised learning of depth and ego-motion from video", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1851–1858.

[39] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[40] S. Tuohy, D. O'Cualain, E. Jones e M. Glavin, "Distance determination for an automobile environment using inverse perspective mapping in OpenCV", 2010.

[41] F. Gökçe, G. Üçoluk, E. Şahin e S. Kalkan, "Vision-based detection and distance estimation of micro unmanned aerial vehicles", *Sensors*, vol. 15, n.º 9, pp. 23 805–23 846, 2015.

[42] S. Duman, A. Elewi e Z. Yetgin, "Distance estimation from a monocular camera using face and body features", *Arabian Journal for Science and Engineering*, vol. 47, n.º 2, pp. 1547–1557, 2022.

[43] A. Masoumian, D. G. Marei, S. Abdulwahab, J. Cristiano, D. Puig e H. A. Rashwan, "Absolute distance prediction based on deep learning object detection and monocular depth estimation models", em *Artificial Intelligence Research and Development*, IOS Press, 2021, pp. 325–334.

[44] B. Dwyer, J. Nelson, J. Solawetz et al., “Roboflow (version 1.0)[software]”, *URL: https://roboflow. com. computer vision*, 2022.

[45] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez e V. Koltun, “CARLA: An Open Urban Driving Simulator”, em *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[46] A. Geiger, P. Lenz e R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”, em *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.