UNIVERSIDADE FEDERAL DA BAHIA
DEEC - DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

ALLAN SOUZA ALMEIDA

Detecção Visual de Caminhos através de Aprendizado Profundo para a Navegação de Robôs Moveis

Dissertação de Mestrado

Salvador
16 de abril de 2025

UNIVERSIDADE FEDERAL DA BAHIA
DEEC - DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

ALLAN SOUZA ALMEIDA

Detecção Visual de Caminhos através de Aprendizado Profundo para a Navegação de Robôs Moveis

Trabalho apresentado ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. André Gustavo Scolari Conceição
Co-orientador: Prof. Dr. Tiago Trindade Ribeiro

Salvador
16 de abril de 2025

Ficha catalográfica elaborada pela Biblioteca Bernadete
Sinay Neves, Escola Politécnica - UFBA.

A447 Almeida,Allan Souza.

Detecção visual de caminhos através de aprendizado profundo para a navegação de robôs movies /Allan Souza Almeida. – Salvador, 2025.

78f.: il. color.

Orientador: Prof. Dr. André Gustavo Scolari Conceição.
Coorientador: Prof. Dr. Tiago Trindade Ribeiro.

Dissertação (mestrado) – Programa de Pós-Graduação em Engenharia Elétrica- Universidade Federal da Bahia - Escola Politécnica, 2025.

1. Redes neurais convolucionais. 2. Robótica móvel. 3.Tranferência de aprendizado. 4. Navegação autônoma.I. Conceição, André Gustavo Scolari. II. Ribeiro, Tiago Trindade. III. Universidade Federal da Bahia. IV. Título.

CDD: 006.3

ALLAN SOUZA ALMEIDA

Detecção Visual de Caminhos através de Aprendizado Profundo para a Navegação de Robôs Moveis

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Federal da Bahia.

Salvador, 16 de abril de 2025

Prof. Dr. André Gustavo Scolari Conceição
Universidade Federal da Bahia

Prof. Dr. Tiago Trindade Ribeiro
Universidade Federal da Bahia

Prof. Dr. Eduardo Simas
Universidade Federal da Bahia

Prof. Dr. Edmar Egídio Purcino de Souza
Universidade Federal da Bahia

Prof. Dr. Gustavo Medeiros Freitas
Universidade Federal de Minas Gerais

AGRADECIMENTOS

Agradeço a minha esposa, por me dar todo o suporte emocional e motivacional durante esse processo.

A minha filha, por iluminar a minha vida e me inspirar a buscar sempre a minha melhor versão.

Ao meu orientador e ao meu co-orientador pela paciência, disponibilidade e pelo apoio para fazer este trabalho acontecer.

A história de toda a sociedade até aqui é a história da luta de classes.

—KARL MARX E FRIEDERICH ENGELS (Manifesto do Partido Comunista, 1848)

RESUMO

Detectar um caminho a ser seguida por um veículo é uma das tarefas mais importantes na navegação de robôs autônomos. Nos últimos anos, técnicas baseadas em aprendizado profundo têm se destacado em relação aos métodos tradicionais de visão computacional na detecção de caminhos, devido à capacidade das redes neurais de extrair características diretamente dos dados, tornando a detecção mais robusta e precisa em diferentes cenários. Esta dissertação propõe uma abordagem baseada em transferência de aprendizado para resolver o desafio da detecção visual de caminhos na navegação de robôs móveis, utilizando uma rede neural convolucional residual fatorizada. O trabalho traz uma comparação entre o modelo proposto e outras arquiteturas de redes neurais convolucionais, além de um estudo de caso comparativo com um sistema baseado em visão computacional determinística. Os resultados experimentais indicam o potencial do modelo, que se mostra capaz de detectar caminhos com precisão, mesmo na presença de descontinuidades e variações de luminosidade, alcançando um F1 *score* de 0,9166 e tempo de inferência médio de 6,15 ms. A arquitetura proposta também encontra um bom equilíbrio entre acurácia e eficiência, o que torna o sistema adequado para aplicações de robótica móvel.

Palavras-chave: Robótica Móvel, Visão Robótica, Aprendizado Profundo, Redes Neurais Convolucionais, Transferência de Aprendizado.

ABSTRACT

Detecting the path that a vehicle should follow is an essential task in autonomous robot navigation. In recent years, deep learning-based approaches have outperformed traditional computer vision techniques in path detection, due to the ability of neural networks to extract features directly from data, making detection more robust and accurate across different scenarios. This work presents a transfer learning-based approach to tackle the problem of visual path detection for mobile robot navigation, using a factorized residual convolutional neural network. The work presents a comparison between the proposed model and other convolutional neural network architectures, as well as a comparative case study with a system based on deterministic computer vision. Experimental results indicate the promise of the proposed model, which is capable of detecting paths even in the presence of discontinuities and variations in lighting, achieving an F1 score of 0.9166 and an average inference time of 6.15 ms. The proposed architecture also achieves a good balance between accuracy and efficiency, making the system suitable for mobile robotics applications.

Keywords: Mobile Robotics, Robotic Vision, Deep Learning, Convolutional Neural Networks, Transfer Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

BCE	*binary cross entropy*
CBA	Congresso Brasileiro de Automática
CNN	*Convolutional Neural Networks*
CUDA	*Compute Unified Device Architecture*
FPS	*frames* por segundo
FCN	*Fully Convolutional Networks*
GPU	unidade de processamento gráfico
IFR	*International Federation of Robotics*
IoU	*intersection over union*
LaR	Laboratório de Robótica
MLP	*multi-layer perceptron*
ReLU	*Rectified Linear Unit*
RGB	*red-green-blue*
ROS	*Robot Operating System*

LISTA DE SÍMBOLOS

$\sigma(z)$	Função de ativação aplicada após cada convolução
θ_r	Erro angular no modelo visual utilizando um veículo virtual
$c(s)$	Curvatura do caminho no modelo visual utilizando um veículo virtual
H	Horizonte constante de predição no modelo visual utilizando um veículo virtual
h_{ij}	Saída de cada camada convolucional
I	Imagem utilizada na convolução
K	Filtro utilizado na convolução
L	Função entropia cruzada binária
L_{IoU}	Função de custo baseada no índice Jaccard
P	Imagem inferida pela rede
s	Comprimento do caminho no modelo visual utilizando um veículo virtual
$S(i,j)$	Resultado da convolução espacial discreta ou correlação cruzada
$s(t)$	Resultado da convolução temporal contínua
$s[n]$	Resultado da convolução temporal discreta
T	*Ground truth* utilizado no treinamento da rede
u_e	Entrada de controle no modelo visual utilizando um veículo virtual
Z	Deslocamento lateral no modelo visual utilizando um veículo virtual

SUMÁRIO

Capítulo 1

O presente documento consiste em uma dissertação de mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal da Bahia como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica. Neste capítulo serão apresentados a motivação, os objetivos do trabalho, a definição do problema e a revisão bibliográfica realizada.

INTRODUÇÃO

1.1. MOTIVAÇÃO

A robótica é um campo extremamente dinâmico com avanços prósperos em sua tecnologia. Os autores Bekey *et al.* (2008) afirmaram que

> junto com outras tecnologias emergentes, como a tecnologia da informação, biotecnologia e nanotecnologia, a robótica contribuirá para aumentar as oportunidades de crescimento econômico e terá um impacto significativo nas gerações futuras, com consideráveis repercussões sociais e econômicas.

A previsão dos autores tem se confirmado ao longo dos últimos anos. Segundo o estudo realizado por Jurkat, Klump e Schneider (2022) sobre a base de dados da *International Federation of Robotics* (IFR), o número anual de instalações de robôs industriais em 2019 foi próximo de 400 mil e o número de robôs em estoque ultrapassou 2,5 milhões de unidades. Contrariando o senso comum a respeito da tecnologia, o estudo de Doyle (2016) aponta uma relação direta entre o crescimento da indústria robótica e a criação de novos empregos.

Esse crescimento se deve em grande parte à evolução tecnológica e científica dos últimos anos. Os avanços mais recentes na área de inteligência computacional, principalmente no campo de aprendizado profundo, têm causado um impacto significativo na robótica, abrindo novas possibilidades e melhorando o desempenho de robôs em diversas aplicações.

Muitas dessas aplicações envolvem a utilização de robôs móveis. Um robô móvel é um dispositivo mecânico montado sobre uma base não fixa, que age sob o controle de um sistema computacional, equipado com sensores e atuadores que o permitem interagir

com o ambiente. A robótica móvel representa a área de pesquisa e aplicação que trata do controle de veículos robôticos autônomos, semi-autônomos e teleoperados (DUDEK; JENKIN, 2010). São diversas as aplicações da robótica móvel. Dentre elas, podem ser citadas exploração espacial e submarina, manufatura flexível, logística e armazenamento, agricultura de precisão, apoio a idosos e pessoas com mobilidade reduzida entre muitas outras.

Alguns dos desafios da robótica móvel são percepção, localização, planejamento de caminho, locomoção e desvio de obstáculos. A percepção é uma das tarefas mais importantes em sistemas autônomos de qualquer tipo, pois permite adquirir conhecimento sobre seu ambiente. Isso é feito através da realização de medições usando vários sensores e, em seguida, extraindo informações significativas dessas medições (SIEGWART; NOURBAKHSH, 2004). É, portanto, de significativa importância o processamento de informações obtidas a partir dos sensores a fim de garantir uma navegação precisa e segura.

O problema de detecção e estimação de caminhos abordado neste trabalho possui diversas aplicações dentro da robótica móvel, como a inspeção de linhas de transmissão e ativos subaquáticos, sistemas avançados de assistência veicular, controle de navegação de veículos autônomos e robótica de serviço, o que destaca a relevância do tema. Neste contexto, detecção de caminhos refere-se ao processo que identifica a presença de um caminho visível no ambiente, enquanto estimação de caminhos corresponde à extração dos parâmetros desse caminho por meio de uma função matemática. Espera-se que os resultados deste estudo não apenas contribuam para o avanço da pesquisa em robótica móvel, mas também tenham impactos práticos na melhoria da eficiência, segurança e versatilidade desses sistemas. Além disso, ao enfrentar desafios complexos na detecção e estimação de caminhos, este projeto também abre portas para futuras investigações e inovações nesta área em constante evolução.

1.2. OBJETIVOS

O objetivo central deste trabalho é apresentar uma solução para o problema de detecção de caminhos visuais no contexto de navegação robótica com emprego de técnicas de aprendizado profundo. O foco principal é o desenvolvimento de um sistema que permite detectar e estimar a existência, a forma e a posição de um caminho em condições diversas. Além disso, também serão apresentados uma comparação com outros três modelos de redes neurais convolucionais e um estudo de caso comparando o modelo proposto a uma técnica alternativa baseada em visão computacional determinística.

1.2.1. Objetivos específicos

Os objetivos específicos do presente trabalho são:

- Projetar um sistema de detecção de caminhos baseado em *deep learning* capaz de fornecer informações precisas sobre um caminho com base em dados visuais.

- Propor uma abordagem capaz de alimentar diferentes tipos de controladores baseados em modelo de erro.
- Validar o sistema proposto em ambiente simulado no contexto de navegação de robôs móveis.
- Validar o sistema utilizando imagens reais e modelos visuais, considerando o erro de controladores de seguimento de caminhos descritos na literatura.
- Comparar o modelo com outras arquiteturas de redes neurais convolucionais.
- Comparar a abordagem proposta com um método baseado em visão computacional determinística.
- Realizar a avaliação e discussão dos resultados obtidos visando contribuir com o acervo de pesquisas nas áreas de robótica e inteligência computacional.

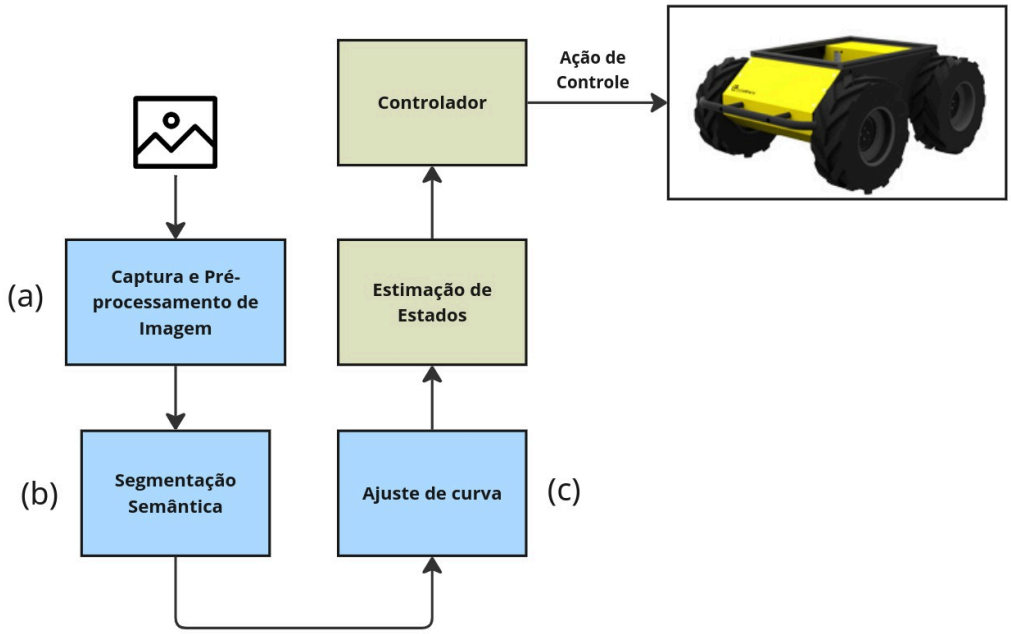
1.3. DEFINIÇÃO DO PROBLEMA

Nesta seção, o problema de detecção de caminho para navegação robótica é formalizado, delineando os fatores de estudo deste projeto. O ponto central do presente trabalho o desenvolvimento de um sistema de detecção de linhas de faixa para alimentar um controlador do robô móvel Husky UGV. O sistema de detecção de caminhos deve ser compatível com diferentes tipos de controladores baseados em modelo de erro, portanto o sistema fornece uma equação que parametriza o caminho e pode ser utilizada como base para a estimação de estados em modelos dinâmicos distintos. Para delimitar o escopo desta pesquisa, são considerados os seguintes aspectos:

- O foco é a detecção de linhas de faixa em ambientes interno e externo, onde diversas condições de iluminação podem ser encontradas, como alta ou baixa incidência de luz;
- O sistema deve ser capaz de detectar diferentes geometrias de faixas, incluindo linhas retas ou curvas, com faixas uniformes ou irregulares;
- Para garantir aplicabilidade prática, o sistema deve ser capaz de operar em tempo real, levando em consideração as restrições de tempo impostas pelo controlador;
- O sistema deve apresentar uma relação de compromisso equilibrada entre performance e acurácia, a fim de prover informações para um controle preciso.

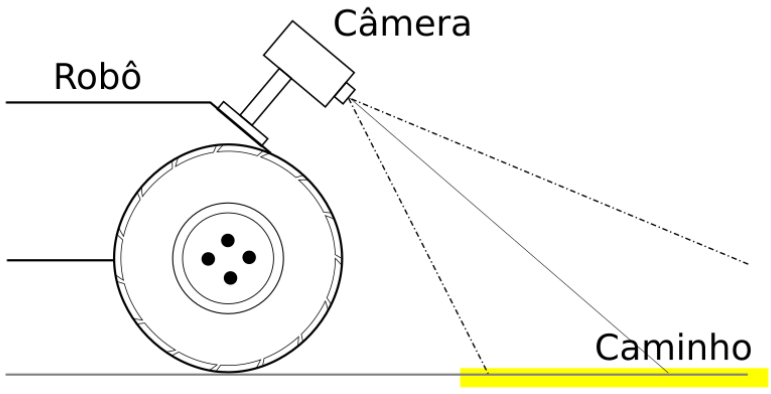
Como o objetivo do sistema-problema é o processamento de informação visual através de aprendizado de máquina para alimentar um controlador baseado em modelo, o *pipeline* proposto consiste em quatro estágios principais: (a) captura e pré-processamento de imagem; (b) segmentação semântica; (c) ajuste de curva; (d) estimação de estados. A Figura 1.1 ilustra as etapas supracitadas.

Figura 1.1: *Pipeline* do sistema-problema proposto.



Na etapa (a) é feita a aquisição da imagem utilizando uma câmera montada sobre o robô, como pode ser visto na Figura 1.2. Ainda nesta etapa, ocorre o pré-procesamento da imagem, onde são realizados o redimensionamento e a normalização dos valores de cada pixel.

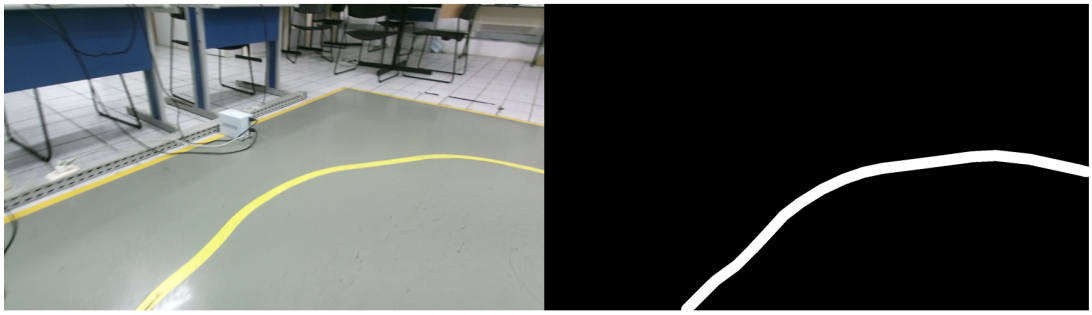
Figura 1.2: Representação esquemática da aplicação do sistema proposto.



A etapa seguinte, (b), é a segmentação semântica, que consiste em particionar a imagem em regiões que delimitam objetos significativos (SHOTTON; KOHLI, 2014). Neste trabalho, o tipo de segmentação semântica realizada é a binária, na qual os *pixels* são classificados como pertencentes a uma faixa ou não pertencentes, conforme a Figura 1.3.

Esta etapa é realizada com a utilização de técnicas de aprendizado de máquina.

Figura 1.3: Segmentação semântica binária. À esquerda, a imagem original e à direita, a imagem segmentada. Os *pixels* brancos correspondem à área que representa a faixa e os *pixels* pretos representam áreas não pertencentes à faixa.



Na etapa (c), última etapa do *pipeline proposto*, é feito o ajuste de curva, cujo objetivo é encontrar uma função matemática que melhor se adapte a uma série de pontos, proporcionando uma representação fiel e simplificada dos dados. Para isso, este trabalho utiliza funções polinomiais e o método de ajuste RANSAC, que lida eficientemente com valores atípicos.

As próximas etapas apresentadas na Figura 1.1 não fazem parte da solução proposta neste trabalho, porém são necessárias para a navegação robótica autônoma. Na etapa de estimação de estados busca-se estimar a posição e orientação do robô no espaço com relação ao caminho, utilizando as informações da etapa anterior e o modelo matemático do sistema robótico. Após esta etapa, os estados são fornecidos ao controlador, responsável por calcular e aplicar a ação de controle ao robô móvel.

Em síntese, este trabalho visa abordar os problemas a seguir:

- Navegação ao longo de caminhos visuais arbitrários e com descontinuidades;
- Robustez a condições diversas de iluminação;
- Estimação de função para caminhos visuais que possibilite a localização relativa do robô;
- Estimação de parâmetros do modelo de erro para o controle de seguimento de caminho.

1.4. REVISÃO BIBLIOGRÁFICA

A navegação autônoma de robôs e veículos enfrenta diversos desafios práticos de extrema relevância, como a detecção de obstáculos, veículos, pedestres e o planejamento eficiente de caminho. Este último depende da capacidade de mapear o ambiente, estabelecendo limites de navegação. Entre as técnicas utilizadas para o mapeamento de caminhos estão a segmentação de rodovias e a detecção de faixas de rodagem. A segmentação visa

identificar o formato do contorno de uma rodovia, separando os dados de entrada entre pertencentes ou não à área da estrada. Já a detecção de faixas consiste no processo de identificação e rastreamento das faixas presentes em uma via de tráfego. Essas tarefas são fundamentais para sistemas de assistência ao motorista e veículos autônomos, pois permitem que o veículo obtenha informações precisas sobre a localização e o formato das rodovias e faixas, auxiliando na navegação segura e no controle veicular.

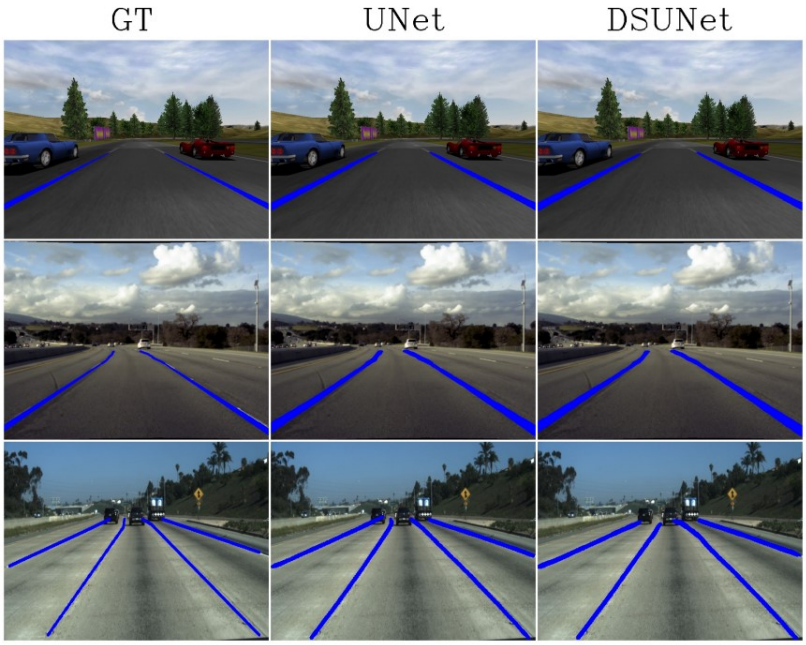
Tradicionalmente, a detecção de faixas e rodovias tem sido realizada utilizando técnicas determinísticas de processamento de imagens e visão computacional. Essas abordagens envolvem a análise de imagens capturadas por câmeras instaladas no veículo, às quais são aplicados algoritmos de segmentação baseados em regras predeterminadas para identificar as faixas presentes na estrada. O objetivo é extrair características relevantes das faixas, como sua posição, largura e orientação. No entanto, essa tarefa enfrenta desafios significativos devido a diversos fatores como a qualidade das imagens, condições ambientes como iluminação, chuva e neblina e oclusões parciais, que podem afetar significativamente a precisão da detecção (TANG; LI; LIU, 2021).

Diante desses desafios, têm sido exploradas abordagens avançadas com a utilização de técnicas de aprendizado profundo (*deep learning*), que são capazes de aprender padrões e características das faixas e estradas. O *deep learning* é uma subárea da inteligência artificial que se concentra na criação de redes neurais artificiais profundas que podem aprender a partir de grandes volumes de dados para realizar tarefas complexas, como a detecção de objetos em imagens. Para a detecção de faixas, as redes neurais profundas são treinadas em grandes conjuntos de dados que contêm imagens de estradas em diferentes condições de iluminação e clima. Estas redes podem ser capazes de detectar as faixas de rodagem de forma mais robusta do que as técnicas tradicionais de processamento de imagens (CHNG; LEW; LEE, 2020)

Os estudos apresentados a seguir destacam alguns dos avanços mais recentes no campo de pesquisa de detecção de faixas e segmentação de rodovias por meio de técnicas de aprendizado profundo. São apresentadas diferentes abordagens, compostas por técnicas como segmentação semântica, deslocamento de mapa de características, e transferência de aprendizado. Os resultados obtidos nos diferentes trabalhos apresentados confirmam a eficiência e robustez das técnicas de *deep learning* para solucionar o desafio proposto.

A relação de compromisso entre o custo computacional e acurácia dos modelos de aprendizado profundo é um ponto crucial para o projeto de bons sistemas de detecção de faixas e planejamento de caminho, uma vez que o objetivo final é, por diversas vezes, embarcar a solução em veículos autônomos. Nesse contexto, o trabalho de Lee e Liu (2022) propõe uma abordagem para a deteccção de faixas de pista e previsão de caminho para direção autônoma baseado em uma versão leve da UNet (RONNEBERGER; FISCHER; BROX, 2015), que utiliza convolução separável em profundidade (DSUNet). A arquitetura proposta substitui 17 camadas convolucionais 3×3 da UNet por 17 camadas convolucionais com filtros 3×3 monocanal em profundidade e 17 camadas com filtros 1×1 em ponto. A função de custo utilizada foi a entropia cruzada ponderada, que ajuda a lidar com números desbalanceados de amostras positivas (linhas de pista) e de fundo. Foi utilizado o algoritmo DBSCAN para agrupar os pontos detectados em faixas semânticas e, em seguida, foi aplicado o mapeamento de perspectiva para transformar as

Figura 1.4: Comparação entre UNet, DSUNet e *ground truth* no TORCS (primeira linha), LLAMAS (segunda linha) e TuSimple (terceira linha).



Fonte: Lee e Liu (2022).

faixas para o espaço de visão superior. O processo usa recursão de mínimos quadrados e impõe paralelismo para ajustar as linhas de faixa, além de usar um fator de escala para calcular a curvatura.

Três ambientes de treinamento e teste foram utilizados, TORCS, LLAMAS e TuSimple, totalizando em torno de 70.000 imagens, parte produzidas pelos autores e parte de *datasets* de terceiros. A Figura 1.4 ilustra a comparação da UNet e da DSUnet com o *ground truth* na detecção de faixas nos três ambientes. Os resultados do trabalho mostram que o DSUNet é comparável ao UNet em quatro medidas para detecção de faixas em quase todas as imagens no primeiro conjunto de dados de teste (com até 96% de F1 *score*). Além disso, a DSUNet melhora em 5,12x, 6,54x e 1,61x o tamanho do modelo, o número de operações de multiplicar-acumular e o número de *frames* por segundo (FPS), respectivamente.

Conforme mencionado previamente, fatores desfavoráveis como a distribuição desbalanceada de dados, oclusões e pouca iluminação podem ter um impacto negativo nos resultados de modelos de aprendizado de máquina que utilizam visão computacional para detecção de faixas e navegação autônoma. Tendo em vista esses desafios, o problema de detecção de faixas em condições de baixa iluminação foi abordado por Liu *et al.* (2020). A proposta do trabalho é realizar um aprimoramento de dados baseado em transferência de estilo para gerar imagens que representam condições de pouca luz a fim de compensar o desbalanceamento do *dataset* utilizado (CULane), que contém originalmente apenas 14% de imagens nessas condições.

Figura 1.5: Exemplos de imagens em condições normais de luminosidade e suas contrapartes correspondentes em condições de pouca luz após a transferência de estilo.

Fonte: Liu *et al.* (2020).

Para transferência de estilo de condições de luz, os autores propuseram uma SIM-CycleGAN, que produz imagens geradas em condições de pouca luz com maior fidelidade em comparação com a CycleGAN, além de resolver o problema de variação de escala causado pelo redimensionamento não proporcional. As imagens geradas usam os rótulos das imagens originais, portanto, não há necessidade de coleta de dados e anotação manual. Como modelo para detecçao, os autores utilizaram a ERFNet (ROMERA et al., 2018), uma rede convolucional residual fatorizada, com a adição de um ramo para detecção da existência de linhas, visando diminuir o tempo de convergência. A função de custo utilizada pra treinar o modelo possui uma parcela correspondente à probabilidade de logaritmo negativo na segmentação de instância e outra à entropia cruzada binária da existência da faixa. Os resultados experimentais apresentam F1 *score* de 69,4%, 76,2% e 91,8% nas condições de sombra, noite e dia, respectivamente, resultados superiores aos obtidos utilizando apenas a ERFNet. A Figura 1.5 mostra alguns exemplos de imagens em condições normais de luminosidade e suas contrapartes correspondentes em condições de pouca luz após a transferência de estilo.

Buscando a melhor relação entre a qualidade de segmentação e o tempo de execução de um modelo de rede neural convolucional utilizado para a tarefa de segmentação de estradas e faixas, o trabalho de Oliveira, Burgard e Brox (2016) propõe alguns refinamentos a fim de tornar o modelo aplicável à robótica. A arquitetura utilizada é baseada em *Fully Convolutional Networks* (FCN), redes neurais totalmente convolucionais, apresentando

Figura 1.6: As quatro diferentes tarefas de segmentação de faixa/estrada do *dataset* KITTI. A segmentação correta é mostrada em verde, enquanto os falsos negativos e falsos positivos são mostrados em vermelho e azul, respectivamente.

(a) UM LANE.

(b) UM ROAD.

(c) UMM ROAD.

(d) UU ROAD.

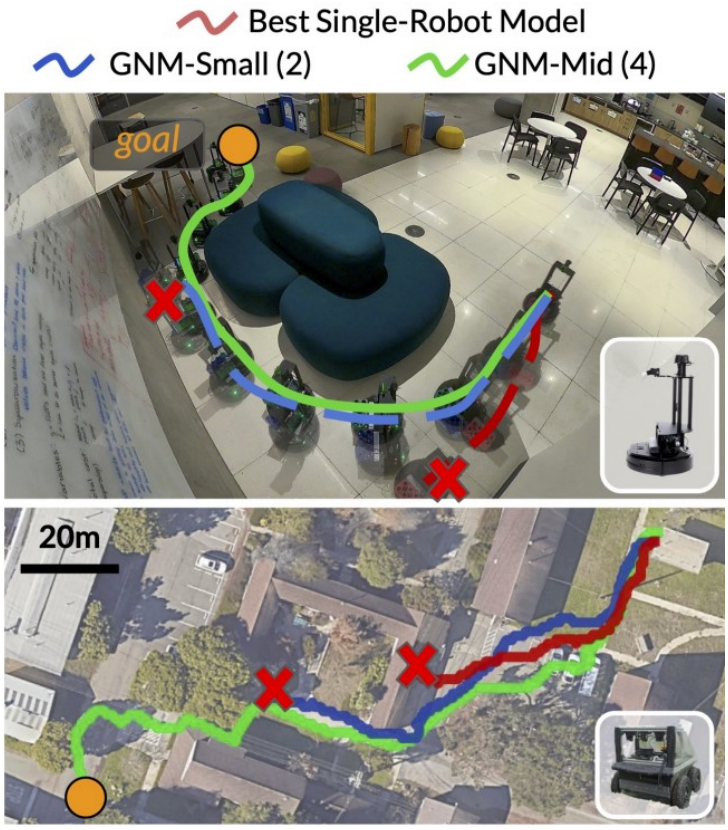
Fonte: Oliveira, Burgard e Brox (2016).

uma seção de contração, cuja saída é um mapa de características em baixa resolução, e uma seção de expansão capaz de produzir uma saída com a mesma resolução da imagem original com a classificação pixel a pixel a partir do mapa de características da seção de contração.

Algumas das melhorias propostas pelos autores foram a redução do número e da resolução dos filtros utilizados no modelo que serviu como base para a seção de contração, o VGG-16 (SIMONYAN; ZISSERMAN, 2015) e a utilização de múltiplos filtros por classe na parte expansiva da rede, baseada na UNet. O treinamento e a avaliação do modelo foram feitos utilizando o dataset KITTI, que possui imagens anotadas de vias urbanas em diferentes categorias. O trabalho apresenta resultados que superam em diversas métricas os melhores resultados anteriores no dataset KITTI, com ganhos de velocidade de mais de 20 vezes e acurácia igual ou superior em diferentes cenários do problema de segmentação. A Figura 1.6 apresenta imagens com resultados do sistema proposto nas quatro tarefas do *dataset* KITTI.

As abordagens comumente adotadas em soluções para controle de navegação autônoma requerem conjuntos de dados específicos para cada plataforma robótica, o que leva a uma descentralização no progresso das pesquisas, visto que cada pesquisador trabalha com seu próprio dataset e políticas de navegação. Na tentativa de superar esse desafio, Shah *et al.* (2023) apresentam o estudo do treinamento de modelos de aprendizado profundo em datasets mais gerais e reutilizáveis entre diferentes plataformas de robôs. São analisadas as decisões de design para garantir um compartilhamento efetivo de dados entre robôs distintos. Os autores agregaram 8 datasets coletados em plataformas robóticas em um único dataset heterogêneo e vasto, a fim de facilitar o aprendizado de políticas em larga escala. A Figura 1.7 mostra uma comparação qualitative entre políticas treinadas em conjuntos de dados com diferentes graus de heterogeneidade.

Figura 1.7: Comparação qualitativa entre políticas treinadas em conjuntos de dados com diferentes graus de heterogeneidade. *Datasets* mais diversos levam a melhores generalizações, como demonstrado para os robôs LoCoBot (parte superior) e Jackal (parte inferior), ambos controlados com a mesma política.



Fonte: Shat *et al.* (2023).

A arquitetura utilizada recebe como entradas a observação atual do robô e a observação alvo, e prevê na saída uma política de ações na forma de pontos de referência e distâncias normalizados. São utilizados dois encoders MobileNetV2 (SANDLER et al., 2018) separados para as duas entradas, cujas saídas são concatenadas e alimentadas a um conjunto de camadas totalmente conectadas que fazem a previsão do caminho e da distância. Após o treinamento do modelo, este é implantado em diferentes plataformas robóticas em uma variedade de ambientes internos e externos, visando avaliar a capacidade de generalização em novos robôs e ambientes e a robustez da abordagem proposta diante de perturbações e degradações aplicadas. Os resultados confirmam a superioridade da técnica implementada, com taxas médias de sucesso de navegação superiores a 93%, enquanto no treinamento em datasets isolados o melhor resultado foi de 68%. As principais contribuições do trabalho incluem a demonstração da superioridade de políticas aprendidas de datasets heterogêneos e a disponibilização de um modelo base pré-treinado para uma diversidade de aplicações de navegação.

1.5. CONTRIBUIÇÕES

Os estudos que resultaram nesta dissertação foram publicados em dois artigos. Um artigo foi publicado no *IEEE 20th International Conference on Automation Science and Engineering* (CASE) de 2024 (ALMEIDA; RIBEIRO; CONCEICAO, 2024b) e outro no Congresso Brasileiro de Automática 2024 (CBA) (ALMEIDA; RIBEIRO; CONCEICAO, 2024a):

- ALMEIDA, A. S.; RIBEIRO, T. T.; CONCEICAO, A. G. S. Transfer learning-based lane line detection system for visual path following control. In: *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024. p. 782– 787.
- ALMEIDA, A. S.; RIBEIRO, T. T.; CONCEICAO, A. G. S. Deteccção de faixa baseado em aprendizado profundo para seguimento de caminho visual. In: *Congresso Brasileiro de Automática 2024 (CBA)*, 2024.

1.6. ESTRUTURA DA DISSERTAÇÃO

A dissertação está organizada da seguinte forma: no Capítulo 2 é realizada uma fundamentação teórica sobre o funcionamento das redes neurais convolucionais na resolução do problema de segmentação semântica. A partir disto, no Capítulo 3 a abordagem proposta é apresentada, com os detalhes do modelo e da metodologia adotada. O Capítulo 4 traz os resultados experimentais obtidos através de simulação e a comparação com métodos alternativos. Por fim, o Capítulo 5 conclui o trabalho, discutindo os resultados e expondo as considerações finais, bem como trabalhos futuros e as referências bibliográficas.

Capítulo 2

Neste capítulo são discutidos conceitos fundamentais para o entedimento do problema proposto neste trabalho a respeito da aplicação de técnicas de aprendizado de máquina para extração de informações visuais.

FUNDAMENTAÇÃO TEÓRICA

O problema de detecção de caminhos em sistemas robóticos utilizando aprendizado profundo é de grande importância para a navegação autônoma, permitindo que robôs móveis sigam caminhos de forma precisa e eficiente. Neste contexto, a detecção de caminhos refere-se ao uso de redes neurais para reconhecer e segmentar áreas navegáveis no ambiente.

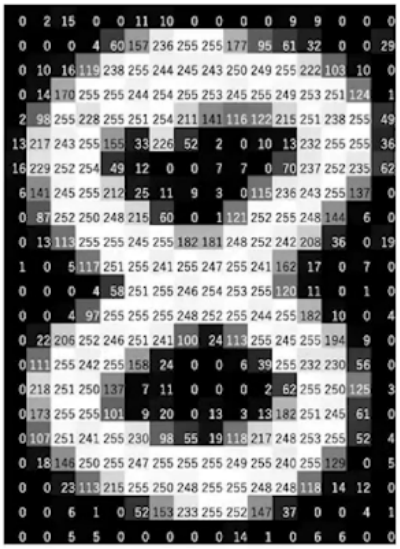
Neste capítulo, será feita uma introdução às redes neurais convolucionais, explicando os principais conceitos envolvidos no projeto e utilização dessas redes, bem como suas principais características. Além disso, será abordado o problema de segmentação semântica. Faz-se necessário este estudo a fim de entender a arquitetura e o funcionamento do modelo proposto neste trabalho.

2.1. REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks* (CNN), são um tipo especializado de rede neural para o processamento de dados que possuem uma topologia conhecida e semelhante a uma grade ou matriz (GOODFELLOW; BENGIO; COURVILLE, 2016). Alguns exemplos de dados deste tipo incluem séries temporais, que podem ser consideradas grades unidimensionais com amostras em intervalos regulares de tempo, e dados de imagens, que podem ser consideradas grades bidimensionais de *pixels*.

Os *pixels* são valores digitais que podem representar cores, iluminação, saturação ou outras variáveis dependendo do sistema de cores utilizado. Em um dos exemplos mais simples, a escala de tons de cinza, o valor de cada *pixel* representa a intensidade luminosa ou o nível de cinza correspondente, como pode ser visto na Figura 2.1. Um valor **0** representa o preto absoluto (sem intensidade de luz), enquanto o valor **255** (para sistemas de 8 *bits*) representa o branco absoluto (intensidade máxima de luz). Já no

Figura 2.1: Representação da imagem de um dígito em escala de tons de cinza.



Fonte: Singh (2023).

sistema de cores *red-green-blue* (RGB), um sistema aditivo baseado nas cores primárias de luz – vermelho, verde e azul –, são utilizadas três matrizes bidimensionais, uma para cada cor, resultando em uma imagem colorida.

As imagens possuem algumas propriedades que sugerem a necessidade de uma arquitetura de rede especializada. A primeira delas, conforme exposto acima, é a sua forma de matriz. Nos modelos de rede totalmente conectadas (ou densas), como o *multi-layer perceptron* (MLP), as camadas ocultas são geralmente maiores que a camada de entrada (PRINCE, 2023). Então, mesmo para uma imagem de baixa resolução e uma rede densa rasa (*i.e.*, poucas camadas ocultas), o número de neurônios necessários pode chegar facilmente a dezenas de bilhões, o que representa problemas práticos em termos de dados necessários para o treinamento e uso de memória computacional.

Outra propriedade é que os *pixels* vizinhos em uma imagem são estatisticamente correlacionados. No entanto, as redes totalmente conectadas não consideram nenhum tipo de relação baseada em proximidade, tratando todas entradas independentemente. Além disso, a interpretação de uma imagem é invariável a transformações geométricas. Uma imagem de um gato continua sendo a imagem de um gato se for deslocada lateralmente alguns *pixels* ou mesmo se for espelhada horizontalmente. Entretanto, para as redes densas essas operações alteram todas as entradas; consequentemente, a rede teria de aprender os padrões de *pixels* que representam um gato em cada posição possível, o que é claramente ineficiente (PRINCE, 2023). Em contrapartida, as redes neurais convolucionais – assim chamadas por realizarem a operação de convolução – processam cada região local

da imagem de maneira independente, utilizando parâmetros compartilhados em toda a sua extensão. Ademais, as camadas convolucionais utilizam menos parâmetros que as camadas totalmente conectadas, exploram as relações espaciais existentes entre *pixels* vizinhos, e não precisam reaprender a interpretação dos pixels em cada posição diferente.

Estas redes representam parte importante da área de estudo conhecida como *deep learning* (aprendizado profundo), que recebe esse nome devido à quantidade elevada de camadas utilizadas para realizar tarefas complexas em grandes volumes de dados. As CNNs se destacam como um exemplo de princípios neurocientíficos aplicados ao aprendizado de máquina. As pesquisas envolvendo a arquitetura destas redes avançam tão rapidamente que a cada poucos meses ou mesmo semanas uma nova "*melhor arquitetura*"para uma tarefa é proposta, tornando impraticável descrever a melhor arquitetura de fato. No entanto, a maioria destas arquiteturas possui blocos básicos de construção em comum, como camadas de convolução, *pooling* e deconvolução.

2.1.1. A Operação de Convolução

Em sua forma geral, a convolução é uma operação entre duas funções $f(t)$ e $g(t)$, com argumentos reais, que resulta em uma nova função $s(t)$, também com argumentos reais. A operação de convolução é comumente utilizada para combinar duas funções de forma a produzir uma nova função, que representa a interação entre as duas.

A convolução temporal contínua de duas funções $f(t)$ e $g(t)$ é definida pela seguinte equação:

$$s(t) = f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau) \cdot g(t - \tau)\, d\tau, \tag{2.1}$$

onde:

- $f(t)$ e $g(t)$ são as funções de entrada, que podem representar, por exemplo, sinais temporais, impulsos ou outras variáveis contínuas.
- τ é uma variável de integração, que representa uma forma de deslocamento entre as duas funções durante o processo de convolução.
- $*$ denota a operação de convolução, que combina as duas funções de maneira a medir a sobreposição entre elas ao longo do tempo.

Essa operação é amplamente utilizada em diversas áreas, como processamento de sinais, econometria, análise de sistemas dinâmicos, entre outras, para modelar como uma função $g(t)$ afeta a função $f(t)$ ao longo do tempo.

Em aplicações práticas de engenharia, os sinais do mundo físico são obtidos através de sensores que convertem variáveis mensuráveis em dados digitais ou analógicos, fornecendo muitas vezes o sinal de forma discretizada, ou seja, em amostras de tempos bem definidas. Para estes casos, define-se a convolução de tempo discreto como

$$s[n] = f[n] * g[n] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[n-k]. \tag{2.2}$$

Na terminologia de aprendizado profundo, a função f é geralmente definida como a *entrada* e a função g é chamada de *filtro*[1]. A saída, ou resultado da convolução, recebe o nome de *mapa de características*. Em grande parte das aplicações de aprendizado de máquina, a entrada é tipicamente um vetor multidimensional (ou matriz) de dados e o filtro é um vetor multidimensional de parâmetros, que são aprendidos pela rede (GOODFELLOW; BENGIO; COURVILLE, 2016).

Frequentemente a convolução é realizada em mais de uma dimensão, como é o caso da convolução de imagens. Se a imagem I utilizada como entrada na operação de convolução e o filtro K forem matrizes bidimensionais, a convolução espacial discreta entre as matrizes é definida por

$$S(i,j) = I(i,j) * K(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n). \tag{2.3}$$

Essa operação é comutativa, portanto (2.3) é equivalente a

$$S(i,j) = K(i,j) * I(i,j) = \sum_m \sum_n I(i-m, j-n) \cdot K(m,n). \tag{2.4}$$

A forma (2.4) é mais comumente usada em algoritmos de *machine learning* por possuir implementação mais simples, já que existem menos variações possíveis para a faixa de valores válidos de m e n. Segundo Goodfellow, Bengio e Courville (2016), a propriedade comutativa da convolução surge porque ocorre a reversão nos índices do filtro em relação à entrada, no sentido de que, à medida que m aumenta, o índice na entrada aumenta, mas o índice no filtro diminui. A única razão para reverter os índices do filtro, segundo os autores, é obter a propriedade comutativa. Embora a propriedade comutativa seja útil para escrever provas, ela geralmente não é uma propriedade importante na implementação de redes neurais. Em vez disso, muitas bibliotecas de redes neurais implementam uma função relacionada chamada *correlação cruzada*, que é a semelhante à convolução, mas sem reverter os índices do filtro:

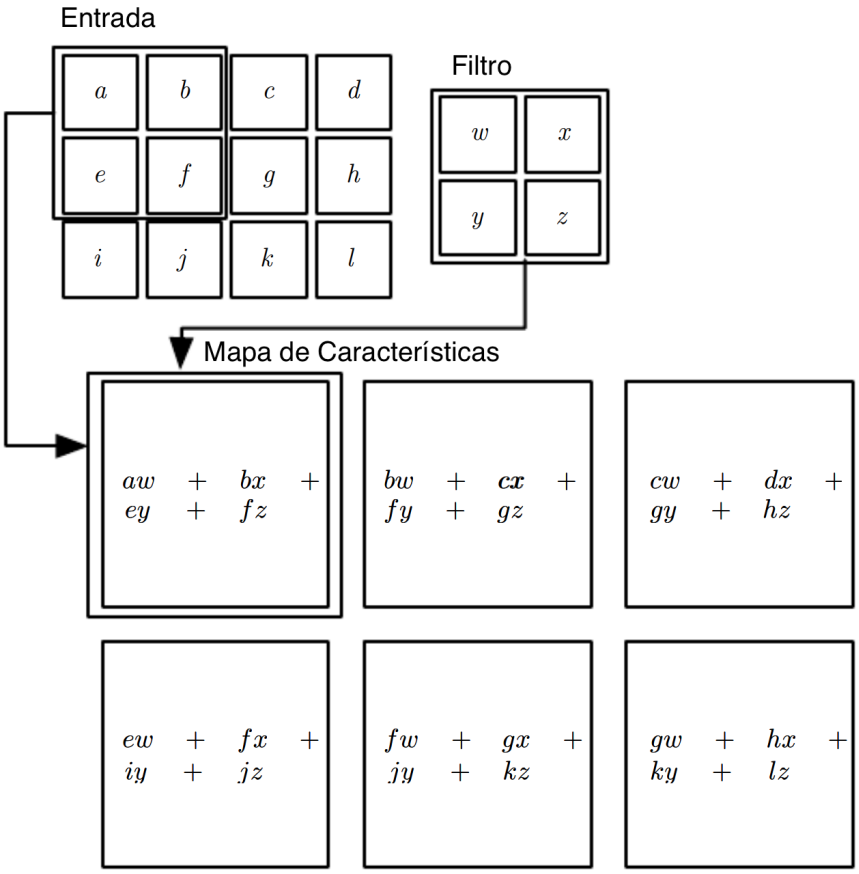
$$S(i,j) = K(i,j) * I(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n). \tag{2.5}$$

Neste texto, convenciona-se nomear ambas as operações de convolução, uma vez que são tratadas desta maneira na literatura. A Figura 2.2 ilustra o exemplo de uma convolução 2-D. Este tipo de convolução é chamada convolução *válida*, na qual o filtro está completamente dentro da imagem.

A convolução utiliza duas ideias importantes que podem ajudar a melhorar um sistema de aprendizado de máquina: interações esparsas e compartilhamento de parâmetros.

[1] No inglês, a terminologia comumente utilizada para o filtro é *kernel*, que seria núcleo em tradução livre.

Figura 2.2: Exemplo de convolução 2-D sem inversão do filtro. Neste caso a saída é restrita apenas às posições em que o filtro está completamente dentro da imagem, a chamada convolução *válida*.



Fonte: adaptada de Goodfellow, Bengio e Courville (2016).

Além disso, a convolução oferece um meio de trabalhar com entradas de tamanhos variáveis.

As interações esparsas – também chamadas de pesos esparsos ou conectividade esparsa – ocorrem em razão de o tamanho do filtro ser tipicamente menor que o tamanho da imagem. Consequentemente, ao processar uma imagem, esta pode conter milhares ou mesmo milhões de *pixels*, entretanto é possível detectar características menos complexas mas significativas, tais como contornos e formas primitivas, com filtros que ocupam apenas dezenas ou centenas de *pixels*. Dessa maneira, é necessário armazenar menos parâmetros, o que reduz os requisitos de memória do modelo e melhora sua eficiência estatítica (GOODFELLOW; BENGIO; COURVILLE, 2016). Isso também significa que o cálculo da saída da rede requer menos operações.

O segundo fator importante, o compartilhamento de parâmetros, refere-se à utilização de um mesmo parâmetro para mais de uma função em um modelo. Em uma rede neu-

ral densa, cada elemento da matriz de pesos é utilizado apenas uma vez ao calcular a saída da camada, sendo multiplicado por um único elemento da entrada e não mais revisitado. Já nas camadas convolucionais, cada membro do filtro é utilizado em todas as posições da entrada (exceto nos *pixels* das bordas da imagem, a depender do tipo de convolução adotada). Isso significa que em vez de aprender um conjunto de parâmetros para cada posição, é necessário aprender apenas um conjunto, o que não afeta o tempo de inferência do modelo mas reduz significativamente as necessidades de armazenamento de parâmetros.

2.1.2. Preenchimento

A equação (2.5) mostra que cada saída é calculada utilizando a soma ponderada dos *pixels* da região definida pelo filtro, o que introduz a questão sobre que acontece com os *pixels* das extremidades da imagem.

Existem duas abordagens comuns. A primeira é o preenchimento[2] da vizinhança externa da imagem com novos valores. A técnica *zero padding* consiste em preencher com valor **0** os pixels externos. Outra possibilidade é repetir os valores dos *pixels* das extremidades. Quando a quantidade de *pixels* utilizados para preenchimento permite que a saída tenha a mesma dimensão da entrada, a convolução é referida como *same*. A segunda abordagem é desprezar as regiões onde o filtro excede o intervalo de posições da entrada, a chamada convolução *válida*[3] Este tipo de convolução tem como vantagem a não necessidade de introduzir informação extra nas extremidades da entrada; em contrapartida, possui a desvantagem de reduzir o tamanho da representação. (PRINCE, 2023)

2.1.3. Passo, Tamanho do Filtro e Dilatação

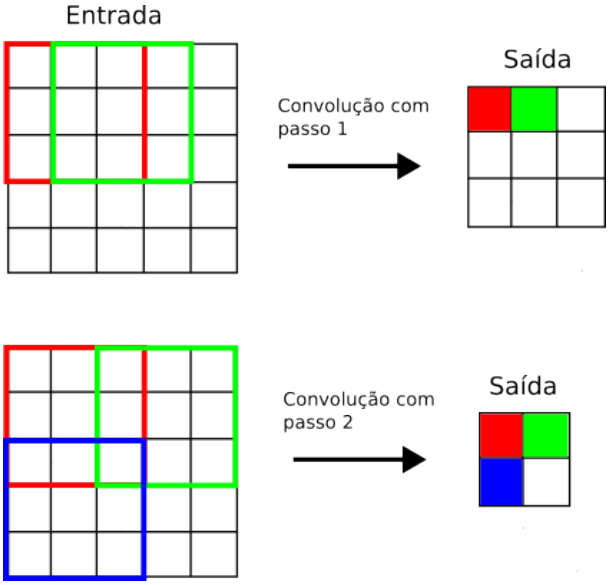
No exemplo de convolução 2-D da Figura 2.2, a saída é a soma do produto elemento a elemento dos 4 primeiros *pixels* da imagem de entrada com o filtro, que também possui 4 parâmetros. O filtro é, então, deslocado 1 *pixel* para a direita e o processo se repete até atingir as extremidades da imagem. No entanto, essa não é a única operação de convolução possível. As operações de convolução são determinadas por três fatores: o *passo*[4], o *tamanho do filtro*[5] e a *dilatação*[6]

O passo diz respeito a quantos *pixels* o filtro se desloca na operação de convolução. No exemplo dado, o passo é 1, visto que o filtro é deslocado apenas 1 *pixel* para a direita de cada vez. O passo pode assumir valores maiores que 1, o que ocasiona redução da resolução do mapa de características. A Figura 2.3 ilustra a operação de convolução com passo 1 e com passo 2.

O tamanho do filtro também pode ser configurado para realizar a convolução sobre

[2] O termo utilizado em inglês para o preenchimento das regiões externas à imagem é *padding*.
[3] O termo utilizado em inglês para convolução válida é *valid convolution*.
[4] O termo utilizado em inglês para passo é *stride*.
[5] O termo utilizado em inglês para tamanho do filtro é *kernel size*.
[6] O termo utilizado em inglês para dilatação é *dilation rate*.

Figura 2.3: Representação da operação de convolução com passo 1 (superior) e com passo 2 (inferior).



Fonte: imagem criada pelo autor.

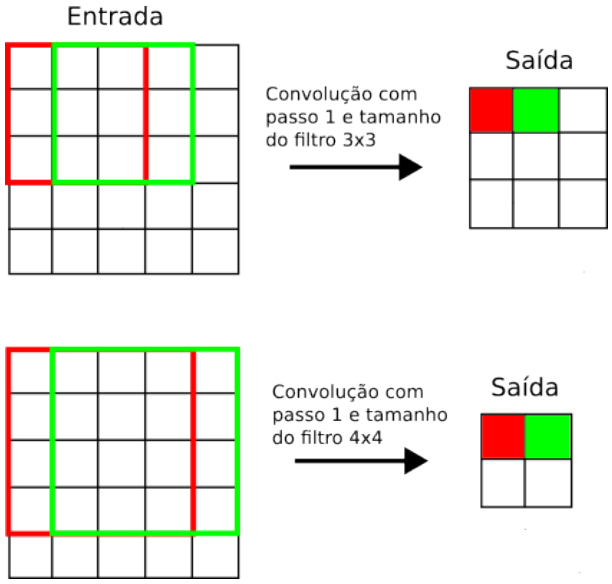
uma área maior ou menor. Aumentar o tamanho do filtro permite integrar informações de uma região maior da imagem, porém também significa aumentar o número de parâmetros da rede. O aumento do tamanho do filtro também causa uma diminuição da resolução do mapa de características para uma mesma entrada, como é possível observar na Figura 2.4, que ilustra a operação de convolução com dois tamanhos distintos de filtro.

Por fim, a dilatação permite agregar informações de uma região maior da imagem sem aumentar o tamanho do filtro e, consequentemente, o número de parâmetros. Para tanto, introduz-se lacunas entre os elementos do filtro. A quantidade de espaçamento entre os parâmetros do filtro é determinada pela taxa de dilatação. Por exemplo, é possível transformar um filtro 5×5 em um filtro 3×3 com taxa de dilatação 2 zerando alguns parâmetros de maneira alternada. A Figura 2.5 demonstra o conceito de dilatação na operação de convolução.

2.1.4. Função de Ativação

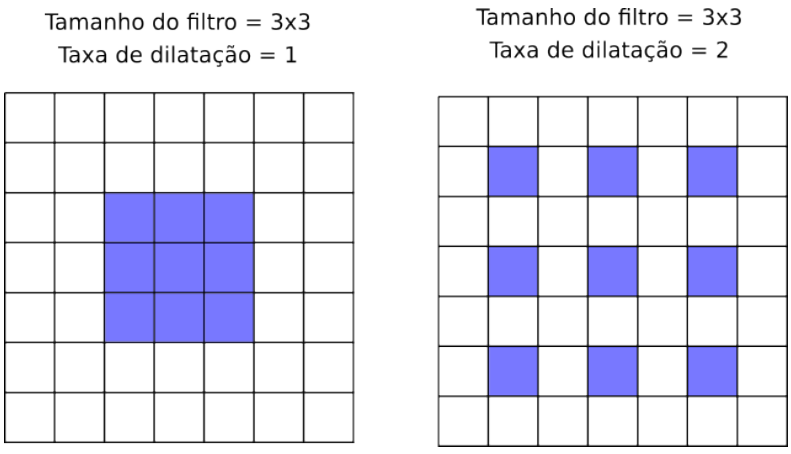
A operação de convolução representada em (2.5) é uma operação linear, visto que o mapa de características é formado por uma combinação linear da entrada com os parâmetros do filtro. Para introduzir uma não-linearidade à convolução e aumentar a capacidade da rede de aprender padrões complexos, utiliza-se uma função de ativação σ que é aplicada ao resultado da convolução ao final de cada passo de iteração. A saída de cada camada convolucional é então dada por

Figura 2.4: Representação da operação de convolução com tamanho do filtro 3×3 (superior) e 4×4 (inferior).



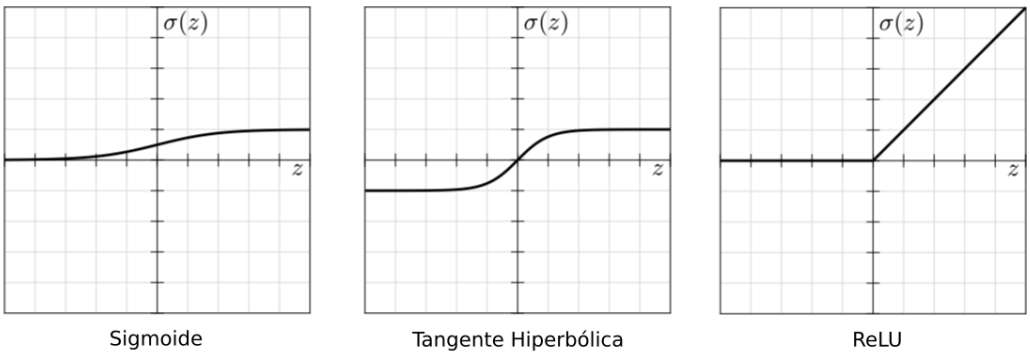
Fonte: imagem criada pelo autor.

Figura 2.5: Representação da operação de convolução com dilatação. O tamanho do filtro é mantido em 3×3 enquanto a taxa de dilatação é igual a 1 (esquerda) e 2 (direita).



Fonte: imagem criada pelo autor.

Figura 2.6: Funções de ativação $\sigma(z)$ comumente utilizadas. A resolução da grade é 1.



Fonte: adaptada de Roberts, Yaida e Hanin (2021)

$$h_{ij} = \sigma\left(\sum_m \sum_n I(i+m, j+n) \cdot K(m,n)\right), \tag{2.6}$$

onde σ é uma função não-linear. Algumas funções de ativação comumente utilizadas são a sigmoide, a tangente hiperbólica e a *Rectified Linear Unit* (ReLU), ilustradas na Figura 2.6 e representadas por (2.7), (2.8) e (2.9), respectivamente:

$$\sigma(z) = sig(z) = \frac{1}{1+e^{-z}}, \tag{2.7}$$

$$\sigma(z) = tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}, \tag{2.8}$$

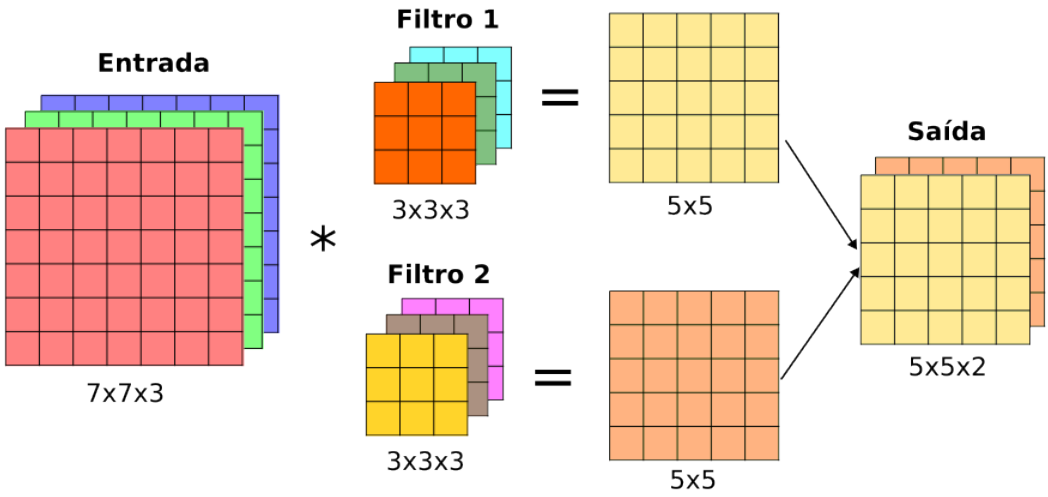
$$\sigma(z) = ReLU(z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases} . \tag{2.9}$$

2.1.5. Canais

Se apenas uma única convolução for aplicada, haverá perda inevitável de informações, uma vez que é feita a média dos *pixels* mais próximos, e algumas funções de ativação como a ReLU – uma das funções mais utilizadas em aplicações de *deep learning* – trunca os resultados que são menores que zero (PRINCE, 2023). Para evitar esse problema, é comum executar diversas convoluções em paralelo. Cada convolução é realizada com um conjunto de parâmetros e produz um conjunto de mapa de características, que é chamado de canal[7]. A Figura 2.7 detalha a operação de convolução utilizando dois filtros. Em geral,

[7] O termo em inglês utilizado para canal é *channel*.

Figura 2.7: Representação gráfica da operação de convolução utilizando dois filtros de dimensões $3 \times 3 \times 3$.



Fonte: imagem criada pelo autor.

as camadas de entrada e as camadas ocultas possuem múltiplos canais. O número de canais da saída C_o é igual ao número de filtros utilizados K, e o número de canais de cada filtro deve ser igual ao número de canais da entrada C_i. Por exemplo, para uma entrada com dimensões $30 \times 30 \times 3$ (largura $\times$ altura $\times$ canais), cada filtro deve ter dimensões $m \times n \times 3$. Se forem utilizados 10 filtros com essas dimensões, a saída terá 10 canais.

2.1.6. Camada de Agrupamento

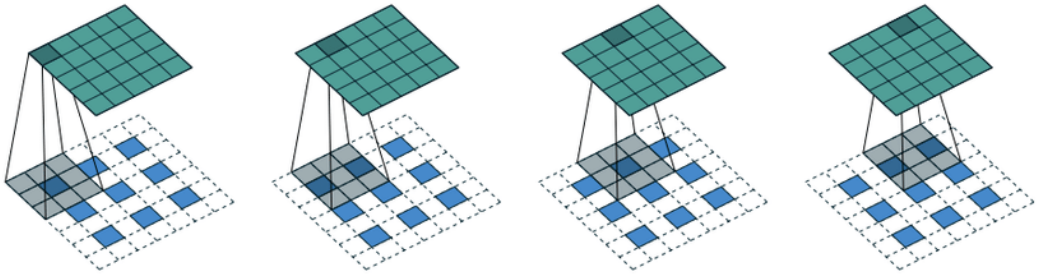
Um bloco típico de uma CNN consiste em três estágios. No primeiro estágio, a rede realiza várias convoluções em paralelo para produzir um conjunto de saídas lineares. Em seguida, como descrito, cada saída passa por uma função de ativação não-linear. No terceiro estágio, utiliza-se uma *camada de agrupamento* ou *camada de pooling*[8] para modificar a saída que alimentará a próxima camada.

Uma camada de *pooling* substitui a saída da rede em uma determinada região por um resumo estatístico das saídas próximas (GOODFELLOW; BENGIO; COURVILLE, 2016). As funções mais populares de *pooling* são o *max pooling*, que calcula a saída máxima dentro de um retângulo de busca, e o *average pooling*, que calcula a média aritmética dos valores dentro da região retangular. Outros tipos também utilizados incluem a norma L^2 e a média ponderada baseada na distância do *pixel* central da região de busca.

Em todos os casos, o objetivo do *pooling* é tornar a representação invariante a pequenas translações da entrada. A invariância à translação é a propriedade que conserva a

[8] O termo utilizado em inglês para camada de agrupamento é *pooling layer*.

Figura 2.8: Operação de convolução transposta para superamostragem. A resolução da saída (5×5) é maior que a da entrada (3×3).



Fonte: Bodnar (2018)

saída intacta mesmo que a entrada seja um pouco transladada. Essa propriedade pode ser especialmente útil quando se está mais interessado na existência de uma determinada característica do que na sua localização exata (GOODFELLOW; BENGIO; COURVILLE, 2016). De forma similar à operação de convolução, o *pooling* varre um filtro por todo o plano da imagem, no entanto esse filtro não possui parâmetros treináveis.

2.1.7. Superamostragem

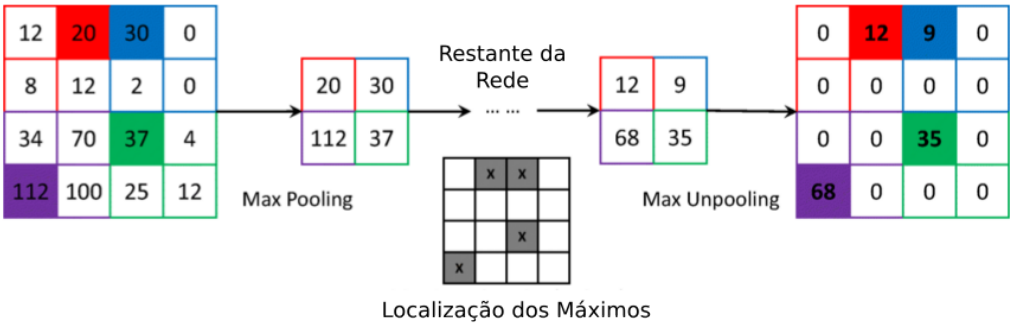
Nas etapas anteriores de convolução e agrupamento, o mapa de características possui, em geral, resolução menor que a da entrada. Esse tipo de redução da resolução é chamada subamostragem ou *downsampling*. No entanto, em muitas aplicações, como a segmentação semântica que será abordada a seguir, é necessário que a saída do modelo tenha a mesma resolução que a entrada. Nesses casos, são aplicadas as operações conhecidas como superamostragem[9]. A superamostragem consiste em inserir informações adicionais a uma entrada de forma a aumentar a sua resolução.

Uma das técnicas utilizadas para essa finalidade é *convolução transposta*, que consiste em aplicar a dilatação à entrada em vez do filtro antes de aplicar a operação de convolução. Dessa forma, a resolução da saída da camada pode ser superior à resolução da entrada. A Figura 2.8 demonstra a aplicação da convolução transposta em uma entrada de resolução 3×3 (taxa de dilatação $d = 2$) com um filtro de mesma resolução, que resulta em uma saída com resolução ampliada, 5×5.

Ainda entre as operações de superamostragem, existe também o desagrupamento[10]. O desagrupamento visa reverter o efeito do agrupamento, restaurando a resolução original que o mapa de características possuía antes do agrupamento. Algumas abordagens populares de desagrupamento incluem a *nearest-neighbor*, a interpolação bilinear e o *max unpooling*. A primeira consiste em replicar cada *pixel* de forma bidirecional, duplicando

[9] O termo utilizado em inglês para superamostragem é *upsampling*.
[10] O termo utilizado em inglês para desagrupamento é *unpooling*.

Figura 2.9: Representação das operações de *max pooling* e *max unpooling*.



Fonte: adaptada de Pan *et al.* (2022).

a resolução da saída. A interpolação bilinear é um método de interpolação no espaço bidimensional utilizado para estimar valores de uma função em pontos desconhecidos, baseando-se em valores conhecidos de pontos vizinhos. A terceira abordagem, o *max unpooling*, consiste em reverter o efeito da operação de *max pooling*, restaurando as posições dos *pixels* no mapa de características anteriores ao agrupamento. A Figura 2.9 ilustra a operação de *max unpooling*. Para realizar esta operação, é necessário armazenar as posições dos *pixels* de maior valor selecionados na operação de *max pooling*, a fim de retorná-los às posições originais durante o *max unpooling*.

2.2. SEGMENTAÇÃO SEMÂNTICA

O processamento de imagens inclui diversas tarefas como transformação, restauração, compressão, segmentação, detecção de regiões de interesse e classificação. Nesse contexto, a segmentação de imagens desempenha um papel muito importante, por ser a base para tarefas de alto nível em visão computacional. A segmentação de imagens tem sido usada na deteção industrial, análise de imagens médicas, observação da Terra por sensoriamento remoto, direção autônoma, entre outras aplicações.

A definição da segmentação de imagem é a divisão de uma imagem em distintas regiões não sobrepostas. O objetivo da segmentação é separar os elementos presentes em uma imagem para análises ou operações posteriores. A segmentação de imagem pode ser dividida, de forma geral, em três subgrupos:

- Segmentação semântica;
- Segmentação de instância;
- Segmentação panóptica.

Figura 2.10: Segmentação semântica de imagens de ambiente urbano.

Fonte: Lei e Nandi (2022).

A segmentação semântica é uma técnica que separa os *pixels* de uma imagem em diferentes classes semânticas, ou seja, atribui a cada *pixel* uma classe ou região específica. Em uma imagem de ambiente urbano, por exemplo, a segmentação semântica pode separar os *pixels* correspondentes a pessoas, árvores, carros, motocicletas e ao chão, conforme ilustrado na Figura 2.10.

Já a segmentação de instância é uma técnica que, além de separar os *pixels* em classes semânticas, identifica e separa cada instância (ou objeto individual) dessa classe. No mesmo cenário de ambiente urbano descrito, a segmentação de instância é capaz de identificar os *pixels* correpondentes a cada carro individualmente.

Por sua vez, a segmentação panóptica é a combinação da segmentação de instância e da semântica. Neste caso, além de identificar cada indivíduo distinto pertencente a cada classe, a segmentação panóptica diferencia os objetos de interesse das informações em plano secundário, como pavimentos ou cenas de fundo.

Cada uma das três tarefas possui sua importância e aplicações para as quais são mais adequadas dentro do campo da visão computacional.

Capítulo

3

Neste capítulo é apresentada a abordagem proposta para a solução do problema de detecção de caminhos visuais utilizando aprendizado profundo.

ABORDAGEM PROPOSTA

A abordagem apresentada e os resultados detalhados a seguir foram apresentados e publicados na *Conference on Automation Science and Engineering* (CASE) 2024, sediada na cidade de Puglia, na Itália e no XXV Congresso Brasileiro de Automática (CBA), sediado na cidade do Rio de Janeiro, ambos no ano de 2024.

O sistema-problema proposto neste trabalho tem por objetivo prover informações do ambiente para a tomada de decisão de um sistema de controle de navegação robótica. Para tanto, foi desenvolvido o *pipeline* ilustrado na Figura 3.1.

A primeira etapa do processo consiste na captura e pré-processamento de dados. Nessa fase, a câmera montada no robô captura a imagem e esta é armazenada na memória do computador como uma matriz tridimensional, representando os valores de cor no espaço RGB. Para otimizar o processamento e reduzir o consumo de memória, essa matriz é redimensionada para $176 \times 320 \times 3$, garantindo compatibilidade com a arquitetura da rede neural utilizada. Esse redimensionamento é uma prática comum em redes neurais convolucionais, pois facilita tanto o treinamento quanto a inferência do modelo, além de equilibrar a qualidade da informação visual e a eficiência computacional. No entanto, é importante destacar que o redimensionamento pode acarretar perda de detalhes, principalmente em áreas de alta resolução da imagem. Esse comprometimento, entretanto, é minimizado devido à capacidade das redes neurais em extrair características relevantes mesmo com entradas de resolução reduzida, garantindo a eficiência do modelo sem prejudicar significativamente seu desempenho.

Em seguida, a matriz é convertida em um tensor para as operações com a CNN. Tensores são entidades matemáticas que generalizam a noção de escalares, vetores e matrizes, sendo amplamente utilizados em aprendizado profundo para representar e manipular dados de forma eficiente.

O modelo recebe esse tensor e realiza a propagação direta (*feedforward*) através de suas camadas, aplicando sucessivamente operações de convolução, *max-pooling*, *upscaling* e funções de ativação. Ao final do processamento, a camada de saída fornece uma matriz

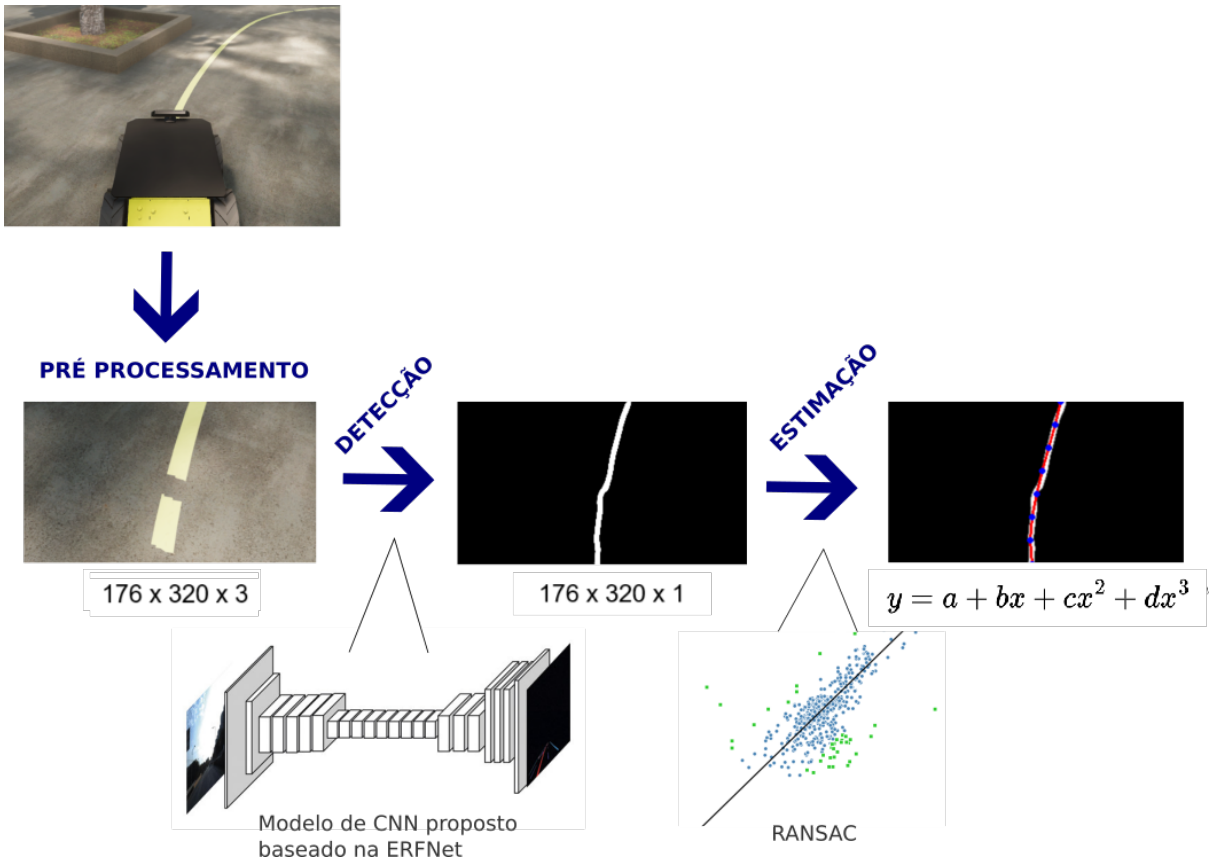


Figura 3.1: *Pipeline* de operações proposto para a solução do problema de detecção e estimação de caminhos navegáveis.

que representa o caminho detectado, de dimensões $176 \times 320 \times 1$, na qual os *pixels* com valor igual a 255 indicam a região correspondente ao caminho identificado.

Por fim, é realizada a estimação da função paramétrica que representa o caminho. Esse passo é essencial, pois os estados utilizados pelo controlador não podem ser fornecidos diretamente em formato de *pixels*, sendo necessário convertê-los para uma representação matemática adequada. Esta etapa fornece uma parametrização do caminho detectado, que pode ser utilizada pelo controlador para guiar o robô de forma eficiente e precisa. Assim, o *pipeline* proposto para a detecção de caminhos é capaz de transformar informações visuais em insumos para navegação que são diretamente aplicáveis a um sistema de controle, garantindo a autonomia do robô na realização de tarefas de navegação em ambientes dinâmicos.

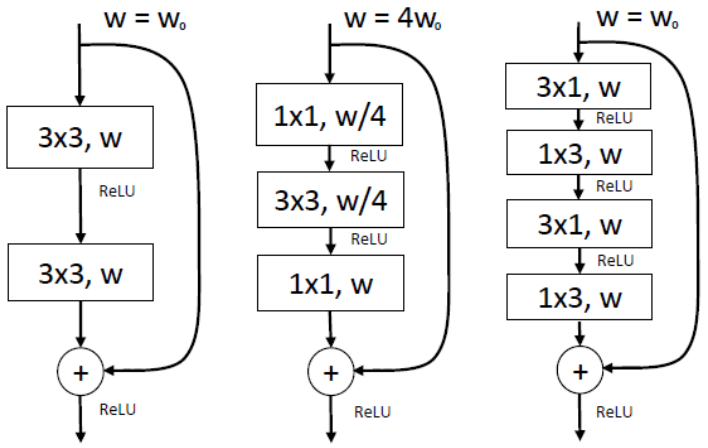
3.1. ARQUITETURA DA REDE

Com a crescente complexidade das tarefas de processamento de imagens e necessidade de encontrar um equilíbrio entre precisão e eficiência computacional, as redes neurais convolucionais têm se destacado como uma abordagem promissora. A visão computacional

baseada na CNN permitiu a realização do que era considerado impossível nos últimos anos, como reconhecimento facial, veículos autônomos, supermercados de autoatendimento e tratamentos médicos inteligentes (LI et al., 2022).

No contexto deste trabalho, a arquitetura escolhida como base foi a ERFNet, que é capaz de ser utilizada em tempo real e, ao mesmo tempo, fornecer segmentação semântica precisa (ROMERA et al., 2018). O ponto principal dessa arquitetura é uma nova camada nomeada *non-bottleneck-1D* (*non-bt-1D*) que utiliza conexões residuais e convoluções fatoradas para permanecer eficiente enquanto mantém uma precisão notável. Essa camada combina os pontos fortes dos *designs bottleneck* e *non-bottleneck* e sua estrutura é ilustrada na Figura 3.2. A solução apresentada neste trabalho adiciona uma camada de normalização de lote (*batch normalization*) após as convoluções dentro do bloco, como mostrado na Tabela 3.1, com o objetivo de estabilizar e acelerar o processo de treinamento, reduzindo o desvio interno de covariáveis.

Figura 3.2: Comparação entre as estruturas dos blocos (a) *non-bottleneck*, (b) *bottleneck* e (c) *non-bottleneck-1D*. w representa o número de mapas de recursos inseridos na camada, reduzido internamente em 4 vezes no *design bottleneck*. Nos blocos convolucionais, "d1 × d2, f" indicam seus tamanhos de kernel (d1, d2) e número de mapas de recursos de saída (f).



Fonte: Romera *et al.* (2018).

A normalização em lote, proposta por Ioffe e Szegedy (2015), é uma técnica comumente usada em redes neurais profundas para melhorar a eficiência do treinamento e o desempenho da generalização. A técnica consiste em normalizar as ativações dentro de um minilote – um subconjunto aleatório dos dados de treinamento – reduzindo a mudança de covariáveis internas e garantindo entradas estáveis para camadas subsequentes, levando a uma convergência mais rápida e estável. Também atua como regularizador, adicionando ruído à rede e reduzindo o *overfitting*, que ocorre quando o modelo se ajusta demais aos dados de treinamento, mas tem sua capacidade de generalização em novos

Tabela 3.1: Disposição de camadas do bloco *Non-Bottleneck-1D*

Camada	**Tipo**	**Tamanho do filtro**
1	Convolução	3x1
2	Convolução	1x3
3	Normalização de Lote	–
4	Ativação (ReLU)	–
5	Convolução	3x1
6	Convolução	1x3
7	Normalização de Lote	–
8	Ativação (ReLU)	–
9	Bloco de Adição de Camadas	–
10	Ativação (ReLU)	–

Fonte: Elaborada pelo autor com base em Romera *et al.* (2018).

dados degradada.

A arquitetura seguida é a *encoder-decoder*, utilizada na ERFNet e em outras redes neurais convolucionais profundas. Nessa estrutura, o segmento codificador (*encoder*) produz mapas de características reduzidos, enquanto o segmento subsequente do decodificador (*decoder*) aumenta esses mapas para corresponder à resolução de entrada. As camadas de 1 a 16 compõem o codificador, sendo compostas por blocos *non-bottleneck-1D* e blocos de *downsampling*, nos quais também foi incluída uma etapa de normalização em lotes. O bloco de *downsampling* realiza a redução por meio da concatenação das saídas paralelas de uma convolução 3x3 e de um *max-pooling* 2x2, ambos com passo 2. Essa redução permite que as camadas mais profundas capturem mais contexto e ajuda a diminuir o esforço computacional.

O segmento do decodificador é composto pelas camadas de 17 a 23 e tem como objetivo aumentar o tamanho da saída do codificador, refinando os detalhes. O bloco de *upsampling* presente no decodificador inclui uma deconvolução com passo 2, seguida por normalização em lotes e função de ativação ReLU. A vantagem de utilizar deconvolução em vez de *max-unpooling* está na simplificação dos requisitos de memória e computação, já que não requer compartilhar os índices de *pooling* do codificador (NOH; HONG; HAN, 2015).

Também foram intercaladas algumas convoluções dilatadas nos blocos *non-bottleneck-1D* com o objetivo de ampliar o campo receptivo sem aumentar o número de parâmetros, permitindo assim que o modelo capture um contexto mais amplo a partir dos dados de entrada. Essa técnica mostrou ser mais eficiente em termos de custo computacional e parâmetros do que o uso de filtros maiores (YU; KOLTUN, 2015). A disposição final das camadas da versão adaptada do ERFNet utilizada neste trabalho está detalhada na Tabela 3.2, na qual os blocos marcados como *dilated* tiveram seu segundo par de convoluções 3x1 e 1x3 substituído por um par de convoluções dilatadas, com a respectiva taxa de dilatação. A camada final do modelo é uma deconvolução com ativação sigmoidal,

que gera a máscara de predição.

Tabela 3.2: Disposição de camadas da adaptação da ERFNet utilizada.

Camada	**Tipo**	**Out-F**[1]	**Out-Res**[2]
1	Subamostragem	16	88x160
2	Subamostragem	64	44x80
3-7	5 x Non-bt-1D	64	44x80
8	Subamostragem	128	22x40
9	Non-bt-1D (dilatação 2)	128	22x40
10	Non-bt-1D (dilatação 4)	128	22x40
11	Non-bt-1D (dilatação 8)	128	22x40
12	Non-bt-1D (dilatação 16)	128	22x40
13	Non-bt-1D (dilatação 2)	128	22x40
14	Non-bt-1D (dilatação 4)	128	22x40
15	Non-bt-1D (dilatação 8)	128	22x40
16	Non-bt-1D (dilatação 16)	128	22x40
17	Aumento de amostragem	64	44x80
18-19	2 x Non-bt-1D	64	44x80
20	Aumento de amostragem	16	88x160
21-22	2 x Non-bt-1D	16	88x160
23	Deconvolução (*upsampling*)	1	176×320

[1] Out-F: Número de filtros da camada.
[2] Out-Res: Resolução na saída da camada para uma entrada de 176×320.
Fonte: Elaborada pelo autor com base em Romera *et al.* (2018).

3.2. TRANSFER LEARNING

A transferência de aprendizado, ou *transfer learning*, é um método de *machine learning* no qual um modelo desenvolvido para uma tarefa é reutilizado como ponto de partida para um modelo em uma segunda tarefa. É uma abordagem popular em problemas de aprendizado profundo, onde modelos pré-treinados são utilizados como ponto de partida em tarefas de visão computacional e processamento de linguagem natural, dadas as vastas demandas computacionais e de tempo necessárias para desenvolver modelos de redes neurais para esses problemas e os grandes avanços de habilidade que eles oferecem em problemas relacionados.

O aprendizado por transferência imita o sistema de visão humana ao fazer uso de quantidades suficientes de conhecimento prévio em outros domínios relacionados ao executar novas tarefas no domínio fornecido. No aprendizado por transferência, tanto os dados de treinamento quanto os dados de teste podem contribuir para dois tipos de domínios: o domínio alvo e o domínio de origem (SHAO; ZHU; LI, 2015).

A transferência de aprendizado tem como objetivo melhorar o desempenho dos aprendizes alvo em domínios alvo transferindo o conhecimento contido em domínios de origem diferentes, mas relacionados (ZHUANG et al., 2020), o que pode ser particularmente útil ao lidar com dados limitados para a nova tarefa.

3.2.1. Pré-treinamento

Inicialmente, a rede apresentada foi treinada em um conjunto de dados amplamente utilizado para detecção de faixas chamado TuSimple, composto por 6408 imagens capturadas por câmeras veiculares em rodovias. Essas imagens exibem condições climáticas variando entre boas e medianas, apresentam entre duas e cinco faixas, e mostram diferentes condições de tráfego. A distribuição das imagens para a fase de pré-treinamento foi feita em 3204 para treinamento, 1602 para validação e 1602 para teste. Antes do treinamento, todas as imagens foram redimensionadas para 360x640 *pixels* com três canais – RGB – e organizadas em lotes de quatro imagens.

O otimizador Adam foi usado para treinar a rede, pois é computacionalmente eficiente, invariante ao redimensionamento diagonal de gradientes e é adequado para problemas grandes em termos de dados/parâmetros (KINGMA; BA, 2017). Durante o pré-treinamento a função de custo entropia cruzada binária, do inglês *binary cross entropy* (BCE), também chamada de perda logarítmica, que é definida da seguinte forma:

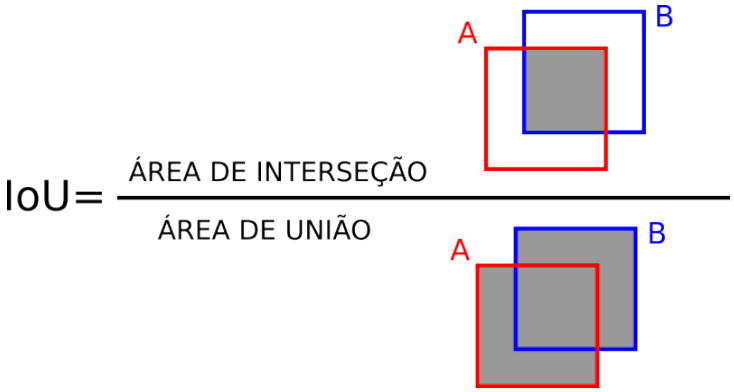
$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)), \tag{3.1}$$

onde y_i é o rótulo verdadeiro para o *pixel* i e $p(y_i)$ é o valor predito, realizando assim uma classificação binária *pixel* a *pixel*.

O tempo de treinamento do modelo foi de 42 minutos em uma unidade de processamento gráfico (GPU) NVIDIA RTX A1000 com 2048 *Compute Unified Device Architecture* (CUDA) *cores*. Após o treinamento, foram avaliadas algumas métricas de desempenho do modelo. A acurácia média avaliada, que mede o número de predições corretas com relação ao número total de predições do modelo, foi 0,98. No entanto, a *intersection over union* (IoU), métrica bastante utilizada para avaliação de modelos em tarefas de segmentação, foi 0,5078. Essa discrepância pode ser explicada pelo fato de que a acurácia considera os *pixels* igualmente, e como a maioria dos pixels nas imagens avaliadas representa trechos que não são parte das faixas (possuem valor 0), o sistema consegue atingir uma alta acurácia mesmo que ele não encontre nenhuma faixa na imagem. Por outro lado, o índice de Jaccard, também chamado de IoU, é uma medida de sobreposição espacial. Essa medida representa o cálculo da interseção entre as áreas de faixa preditas e verdadeiras com relação à sua união, como é representado na Figura 3.3. Quando a maioria da área rotulada representa regiões sem faixa, ou seja, uma pequena porção dos *pixels* da imagem corresponde de fato às faixas, uma sobreposição pouco precisa entre a área predita e a área verdadeira de faixa pode degradar significativamente o IoU.

Apesar do baixo valor de IoU, de maneira geral, a rede proposta foi capaz de detectar faixas de trânsito corretamente, mesmo na presença de pequenas oclusões. A Figura 3.4

Figura 3.3: Representação visual do cálculo do índice de Jaccard, o IoU.



Fonte: imagem criada pelo autor.

mostra a comparação entre o valor verdadeiro das faixas (*ground truth*) e as predições realizadas pelo modelo para algumas imagens do conjunto de teste do *dataset* TuSimple.

3.2.2. Transferência do Aprendizado

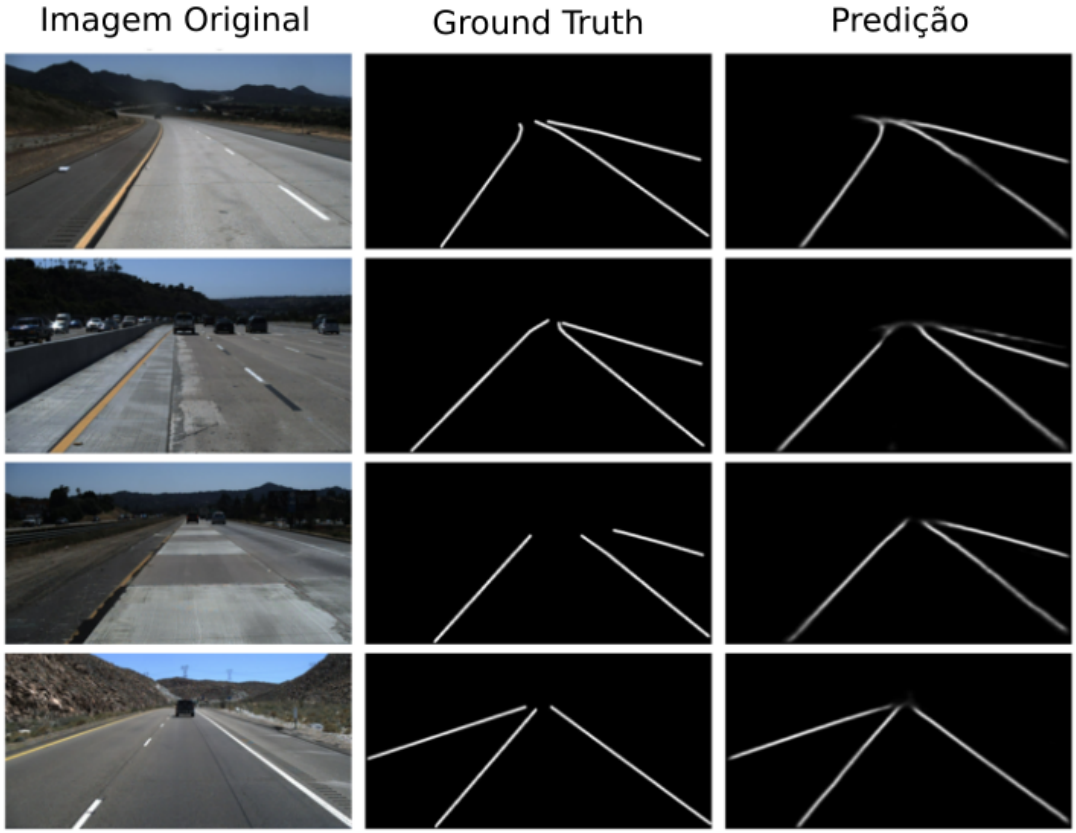
Depois que o modelo foi pré-treinado no conjunto de dados de faixas de estrada, foi construído um novo *dataset* contendo imagens coletadas especificamente para a tarefa atribuída, que é a detecção de caminhos visuais para o controle de um robô móvel. Dois cenários foram escolhidos para construir o conjunto de dados, um interno e outro externo, ambos na Escola Politécnica da Universidade Federal da Bahia. Os caminhos visuais foram delineados usando fita adesiva colorida no chão. A coleta de dados foi realizada conduzindo com auxílio de um controle remoto o robô equipado com uma câmera ao longo dos caminhos, como mostrado na Figura 3.5. O processo de anotação do *dataset* foi realizado com o auxílio do *software* de código aberto LabelMe (WADA, 2016).

Inicialmente, o *dataset* foi construído com 657 imagens, abrangendo os cenários interno e externo. Para aumentar o número de imagens disponíveis, foi aplicada uma técnica chamada *data augmentation*, que consiste em aumentar artificialmente o conjunto de dados criando cópias modificadas de um *dataset* usando dados existentes. Foi aplicada a inversão horizontal das imagens e o *color jitter*, uma técnica de ampliação de dados onde altera-se aleatoriamente o brilho, contraste e saturação das imagens na tentativa de melhorar a robustez e generalização do modelo. Após a ampliação dos dados, o número total de imagens subiu para 2628.

Como as métricas de IoU alcançadas durante a fase de pré-treinamento foram insatisfatórias, decidiu-se utilizar uma nova abordagem para atualizar os pesos do modelo durante a transferência de aprendizado. Em vez de usar a função de custo BCE, foi seguida a estratégia proposta por van Beers (2018) para otimizar o modelo diretamente em uma aproximação do IoU.

A equação original para o IoU é obtida da teoria de conjuntos e pode ser expressa

Figura 3.4: Comparação entre o valor verdadeiro das faixas (*ground truth*) e as predições realizadas pelo modelo para algumas imagens do conjunto de teste do *dataset* TuSimple



Fonte: montagem realizada pelo autor a partir de imagens extraídas do TuSimple e imagens geradas durante os testes do modelo.

como:

$$IoU = \frac{|T \cap P|}{|T \cup P|}, \tag{3.2}$$

na qual T representa a imagem com o *ground truth* e P é a imagem predita. Apesar de ser uma boa representação da qualidade da predição, essa equação não é diferenciável e, portanto, não pode ser utilizada diretamente para treinar o modelo, uma vez que o treinamento por técnicas de gradiente descendente requer o cálculo da derivada da função de custo. Além disso, para utilizar essa equação é necessário que T e P sejam binários, 1's e 0's absolutos. Embora isso seja verdade para o *ground truth*, a imagem predita P contém valores entre 0 e 1, devido à ativação sigmoidal na última camada de *upsampling* da rede.

Para resolver essa limitação, van Beers (2018) propõe uma aproximação do IoU usando probabilidades, descrita pela seguinte equação:

Figura 3.5: Plataforma experimental. (a) Husky UGV. (b) Exemplo de imagem do *dataset*.

(a)

(b)

$$IoU' = \frac{|T \times P|}{|T + P - (T \times P)|}. \tag{3.3}$$

T e P permanecem os mesmos em (3.3), no entanto $T \times P$ representa o produto *pixel* a *pixel* de T e P. No numerador, isso resulta numa aproximação da interseção por culminar na probabilidade de P_n quando T_n é 1, e 0 caso contrário, sendo n cada *pixel* da imagem. O denominador é a soma de T e P com a dedução da interseção, assim como na definição tradicional da união, para mitigar o efeito de contar a área sobreposta duas vezes. Com essas alterações, essa versão aproximada do IoU torna-se diferenciável e pode ser utilizada como função de custo.

Quanto maior o valor do IoU', melhor a predição, portanto a otimização do modelo deve buscar maximizar o seu valor. Considerando-se que no problema de aprendizado supervisionado o objetivo é a minimização do erro, então define-se o custo por:

$$L_{IoU} = -IoU'. \tag{3.4}$$

Esse custo é aplicado a cada elemento em um lote de treinamento e somado, resultando em um valor entre 0 e o tamanho do lote quando o valor do IoU para cada exemplo se aproxima de 1 ou 0, respectivamente (BEERS, 2018).

3.3. AJUSTE DE CURVA

Uma vez concluído o processo de segmentação semântica, o passo seguinte é o ajuste de curva, uma etapa crítica em muitas áreas de análise e processamento de dados. O objetivo do ajuste de curva é encontrar uma função matemática que se alinhe o mais precisamente possível a uma série de pontos de dados, criando uma representação simplificada e, ao mesmo tempo, fidedigna dos padrões subjacentes aos dados observados.

O ajuste de curva é especialmente relevante na detecção de caminhos visuais para seguimento servo visual, pois permite transformar o conjunto de dados capturados — possivelmente disperso ou fragmentado devido a ruídos e variações no ambiente, mesmo após uma segmentação semântica precisa — em uma representação contínua do caminho a ser seguido. Essa função ajustada possibilita que o sistema de fornece uma base consistente para o controle de movimento do robô. Ao encontrar uma curva que se ajusta bem aos dados detectados, o sistema pode extrair informações essenciais sobre a direção e a inclinação do caminho, o que facilita ajustes em tempo real e permite que o robô antecipe mudanças na rota. Esse ajuste permite ao robô interpretar a faixa de forma confiável, mesmo em áreas de visibilidade parcial, e realizar interpolação em trechos onde a continuidade do caminho não é visível, garantindo um seguimento servo visual seguro e preciso.

Para este fim, são utilizadas neste trabalho uma função polinomial de terceira ordem e um método de ajuste robusto, o Random Sample Consensus (RANSAC). O algoritmo, proposto por (FISCHLER; BOLLES, 1981), consiste em ajustar uma curva a amostras aleatórias de um conjunto de dados, utilizando o método dos mínimos quadrados. O ajuste de uma curva de terceira ordem consiste em encontrar o coeficientes a, b, c e d de um polinômio da forma:

$$y = a + bx + cx^2 + dx^3, \tag{3.5}$$

para que o somatório dos quadrados das diferenças entre os valores observados y_i e os valores ajustados pela função polinomial seja minimizado. Ou seja, minimiza-se

$$S = \sum_{i=1}^{n} (y_i - (a + bx + cx^2 + dx^3))^2. \tag{3.6}$$

Para minimizar a soma dos erros ao quadrado, deve-se resolver um sistema de equações normais, que é obtido derivando-se a função erro em relação aos coeficientes do polinômio de terceira ordem – a, b, c e d – e igualando essas derivadas a zero. Esse procedimento gera um sistema linear de equações, que pode ser representado em forma matricial, relacionando os valores conhecidos das variáveis de entrada e os termos associados aos coeficientes a serem determinados:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 \\ \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \\ \sum x_i^3 y_i \end{bmatrix}. \tag{3.7}$$

Esse sistema de equações fornece os valores ótimos para os coeficientes do polinômio de terceiro grau que minimizam a diferença ao quadrado entre o conjunto aleatório de valores observados e os valores ajustados.

Em seguida, o algoritmo verifica quantos pontos dos dados restantes se ajustam ao modelo dentro de uma tolerância definida. Esse processo é repetido várias vezes para encontrar o melhor conjunto de pontos que se ajusta bem ao modelo, ignorando os valores atípicos. O modelo final é então recalculado usando todos os pontos do melhor conjunto.

Com a aplicação do RANSAC sobre a imagem segmentada e binarizada, o caminho no plano da imagem é parametrizado através dos coeficientes de uma função polinomial, o que torna possível a estimação de estados para um sistema de controle, seja no plano da imagem ou no sistema de coordenadas do mundo.

Capítulo

4

Neste capítulo são apresentados os resultados experimentais e detalhes de implementação do sistema de detecção caminhos visuais.

RESULTADOS EXPERIMENTAIS

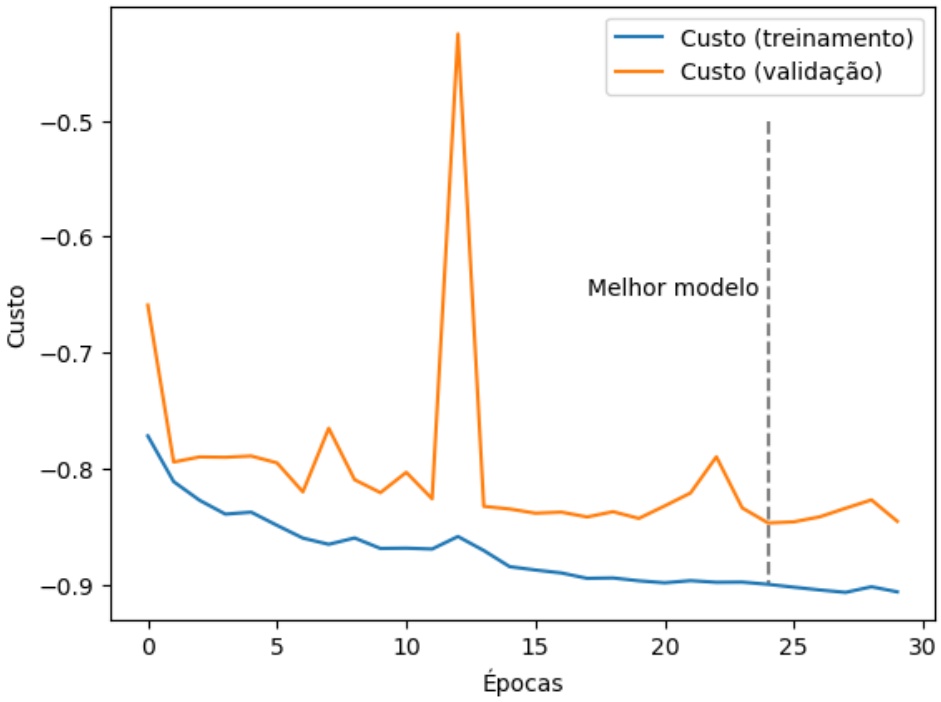
4.1. AVALIAÇÃO DAS MÉTRICAS DE SEGMENTAÇÃO SEMÂNTICA

O estágio de *transfer learning* foi realizado utilizando o *dataset* construído contendo imagens dos cenários interno e externo, em uma divisão de 70% para treinamento, 15% para validação e 15% para teste. A resolução das imagens foi diminuída nesta etapa a fim de reduzir o consumo de memória. Devido a limitações de *hardware* na máquina utilizada para o treinamento, o tamanho de lote máximo possível de ser utilizado foi de 16 imagens, considerando uma resolução de 176×320 *pixels*. O modelo foi treinado por 30 épocas, mas após a época 24 o custo de validaçao parou de decair, resultando em uma parada antecipada do treinamento com recuperação dos melhores pesos, conforme a Figura 4.1.

Após o treinamento, a acurácia, o IoU e F1 *score* atingidos foram 0,9895, 0,8617 e 0,9166, respectivamente. As métricas avaliadas aumentaram significativamente durante a aprendizagem por transferência em comparação com o estágio de pré-treinamento. O tempo de treinamento foi de 38 minutos e o tempo médio de inferência foi de aproximadamente 6,15 ms, com um desvio padrão de 1,24 ms. O modelo demonstrou boa precisão durante o teste em imagens de ambos os cenários, conforme refletido pelas métricas apresentadas na Tabela 4.1 e pelas previsões durante o teste mostradas na Figura 4.2. Os resultados sugerem que o modelo adquiriu com sucesso conhecimento de um conjunto de dados diversificado e heterogêneo, sendo capaz de identificar caminhos com precisão mesmo em cenários desafiadores marcados por sombras, oclusões e holofotes intensos.

A Figura 4.3 mostra o gráfico do tempo de inferência para uma determinada janela de tempo durante os testes com o modelo em tempo real e a Figura 4.4 apresenta um histograma com a distribuição do tempo de inferência. É possível observar variações no tempo de inferência do modelo dentro da janela de predição. Essas oscilações são esperadas em sistemas embarcados que operam em tempo real e podem ser atribuídas a diversos fatores. Um dos principais refere-se ao conteúdo das imagens processadas: variações na

Figura 4.1: Curva da função de custo de treinamento e de validação durante o treinamento do modelo.



complexidade visual da cena podem influenciar a quantidade de ativações geradas nas camadas intermediárias da rede, afetando sutilmente o tempo de execução. Adicionalmente, a presença de outros processos concorrentes no sistema, a gestão dinâmica de recursos computacionais e mecanismos de controle térmico do *hardware* também podem contribuir para essas flutuações. Tais variações, no entanto, mantêm-se dentro de limites aceitáveis, considerando as limitações de taxa de atualização do controlador, e não comprometem o desempenho em tempo real da aplicação.

4.1.1. Comparação com Métodos Alternativos baseados em Aprendizado de Máquina

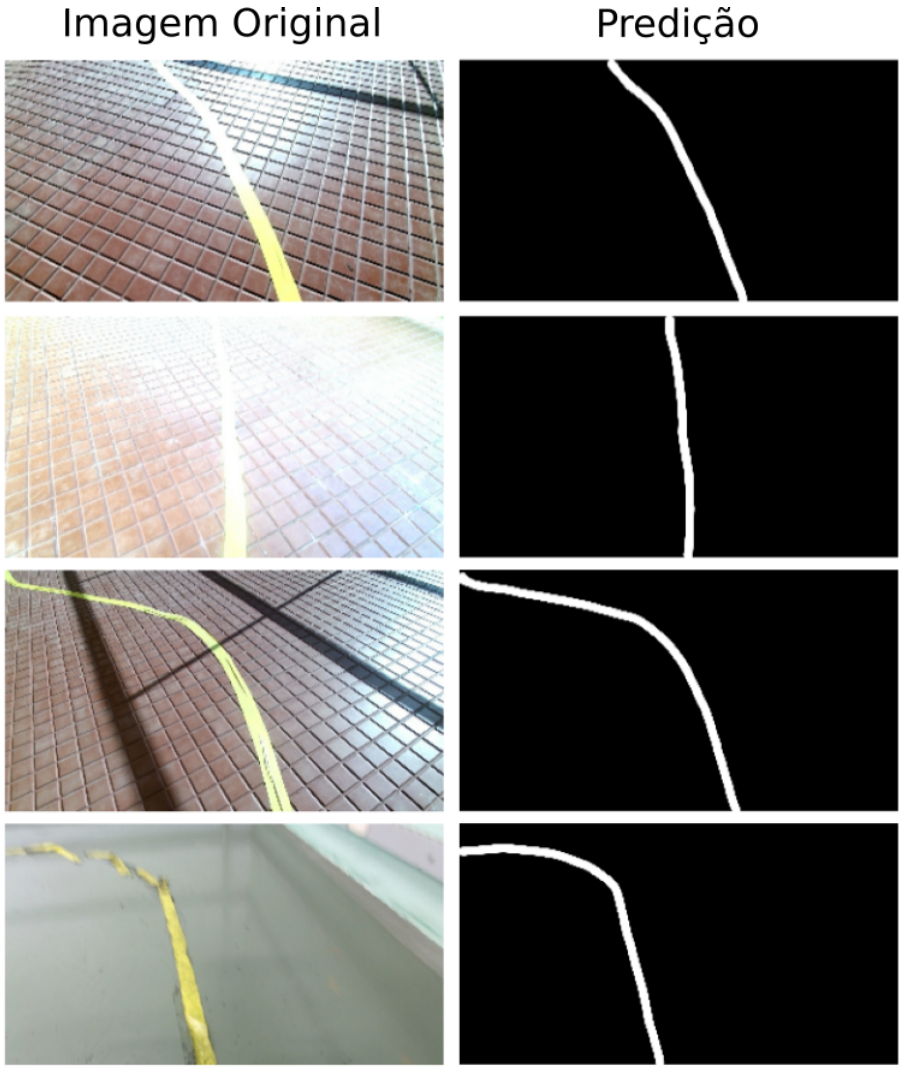
A fim de verificar a eficácia do sistema de detecção de caminhos visuais proposto, foi realizada uma comparação com outros métodos baseados em aprendizado de máquina. Foram selecionadas abordagens bem sucedidas na literatura na tarefa de segmentação semântica para navegação robótica. Os modelos testados foram a UNet (RONNEBERGER; FISCHER; BROX, 2015), a FCN-VGG16 (LONG; SHELHAMER; DARRELL, 2015) e a PSPNet (ZHAO et al., 2017).

A fim de garantir uma homogeneidade na comparação, foram utilizados os mesmos conjuntos de dados de treinamento e teste, além de métricas padronizadas como IoU, acurácia, F1 *score* e tempo de inferência. Todos os modelos seguiram as arquiteturas apresentadas nos artigos (adaptando-se apenas os tamanhos de entrada e saída) e foram

Tabela 4.1: Métricas avaliadas no modelo após os estágios de pré-treinamento e *transfer learning*.

	Pré-treinamento	**Transfer learning**
Função de custo	BCE	IoU'
Resolução	360×640	176×320
IoU	0,5078	**0,8617**
F1-*score*	0,1566	**0,9166**
Acurácia	0,98	**0,9895**
Tempo de treinamento (min)	42	38
Tempo de inferência (ms)	28	**6,15**

Figura 4.2: Predições do modelo após o estágio de *transfer learning*.



Fonte: montagem realizada pelo autor.

Figura 4.3: Tempo de inferência do modelo durante o teste.

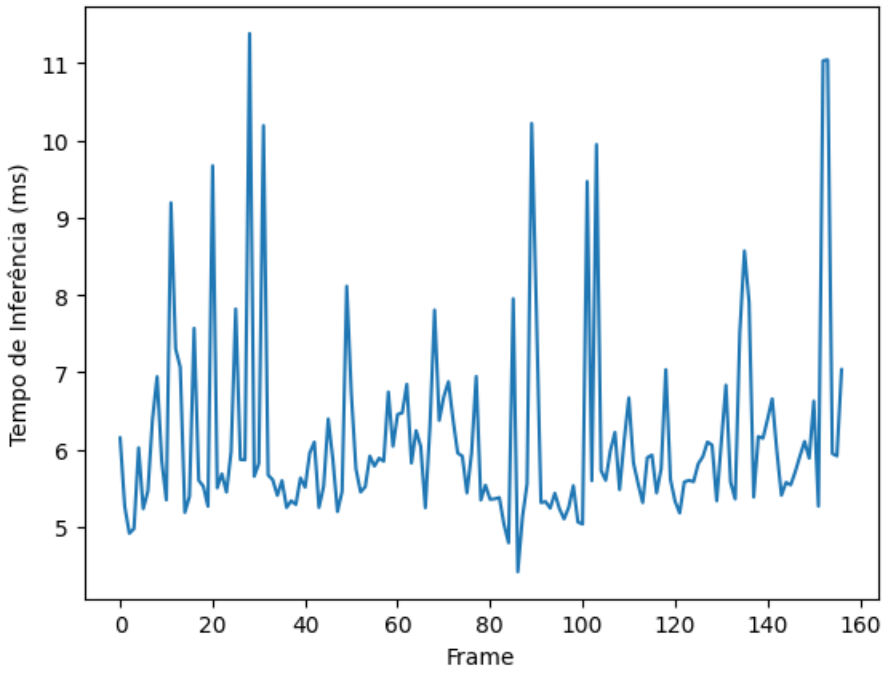


Figura 4.4: Histograma com a distribuição dos tempos de inferência.

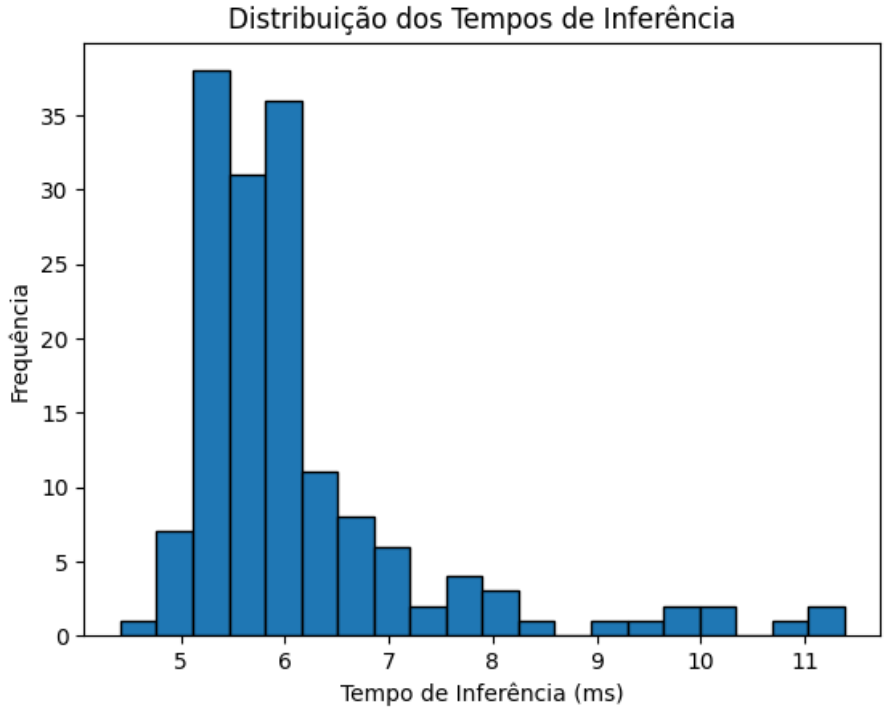


Tabela 4.2: Métricas da abordagem proposta e de outros modelos de redes convolucionais

	Modelo proposto	**UNet**	**FCN-VGG 16**	**PSPNet**
IoU	**0,8616**	0,7857	0,8198	0,7805
F1-*score*	**0,9166**	0,7984	0,8494	0,7830
Acurácia	**0,9895**	0,9842	0,9867	0,9839
Tempo médio de inferência (ms)	**6,15**	**2,59**	4,07	3,81
Tempo médio de treinamento (min)	**38**	38	32	126

implementados utilizando as bibliotecas TensorFlow e Keras. As configurações de treinamento, incluindo o número de épocas, o otimizador e a função de custo, foram mantidas consistentes entre todos os modelos, sendo a função de custo a IoU'. O treinamento foi realizado em duas etapas com o uso de *transfer learning* do primeiro conjunto de dados para o segundo, assim como na abordagem proposta. Ademais, a quantidade de épocas e os critérios de parada adotados foram os mesmos para todos os modelos.

Os resultados, dispostos na Tabela 4.2, evidenciam que o modelo proposto apresentou um desempenho superior em termos de qualidade preditiva, com maiores valores de IoU, F1-*score* e acurácia em comparação com os demais modelos. Em contrapartida, esse ganho foi acompanhado por um aumento no tempo de inferência, indicando um possível *trade-off* entre a precisão na tafera de detecção e a eficiência computacional. No entanto, desde que o tempo de inferência esteja dentro dos limites desejados levando-se em consideração a aplicação e o controlador utilizado, a qualidade da detecção pode ser um fator determinante para a confiabilidade do sistema, especialmente em cenários onde a precisão é essencial, como na navegação de robôs móveis.

Além disso, analisando as segmentações fornecidas por cada modelo, percebe-se uma capacidade maior de abstração na detecção do caminho no modelo proposto, como é possível observar na Figura 4.5. Essa abstração é fundamental para uma detecção mais robusta, permitindo ao modelo compreender o caminho de forma global, mesmo em situações onde as marcações apresentam falhas ou ruídos, semelhante ao modo como um ser humano interpreta o ambiente.

Dessa forma, os experimentos sugerem que a utilização do modelo baseado na ERFNet para a detecção de caminhos visuais não apenas melhora a acurácia da segmentação, mas também viabiliza informações para um controle mais robusto e eficiente de robôs móveis em ambientes dinâmicos.

4.2. ESTUDO DE CASO: ESTIMAÇÃO DE ESTADOS

Nesta seção, apresenta-se um estudo de caso para demonstrar a acurácia do sistema de detecção de caminhos proposto em comparação com um método baseado em visão computacional determinística. O experimento foi conduzido em um ambiente simulado, utilizando o robô móvel Husky UGV, da Clearpath, e avaliou a estimação dos estados de

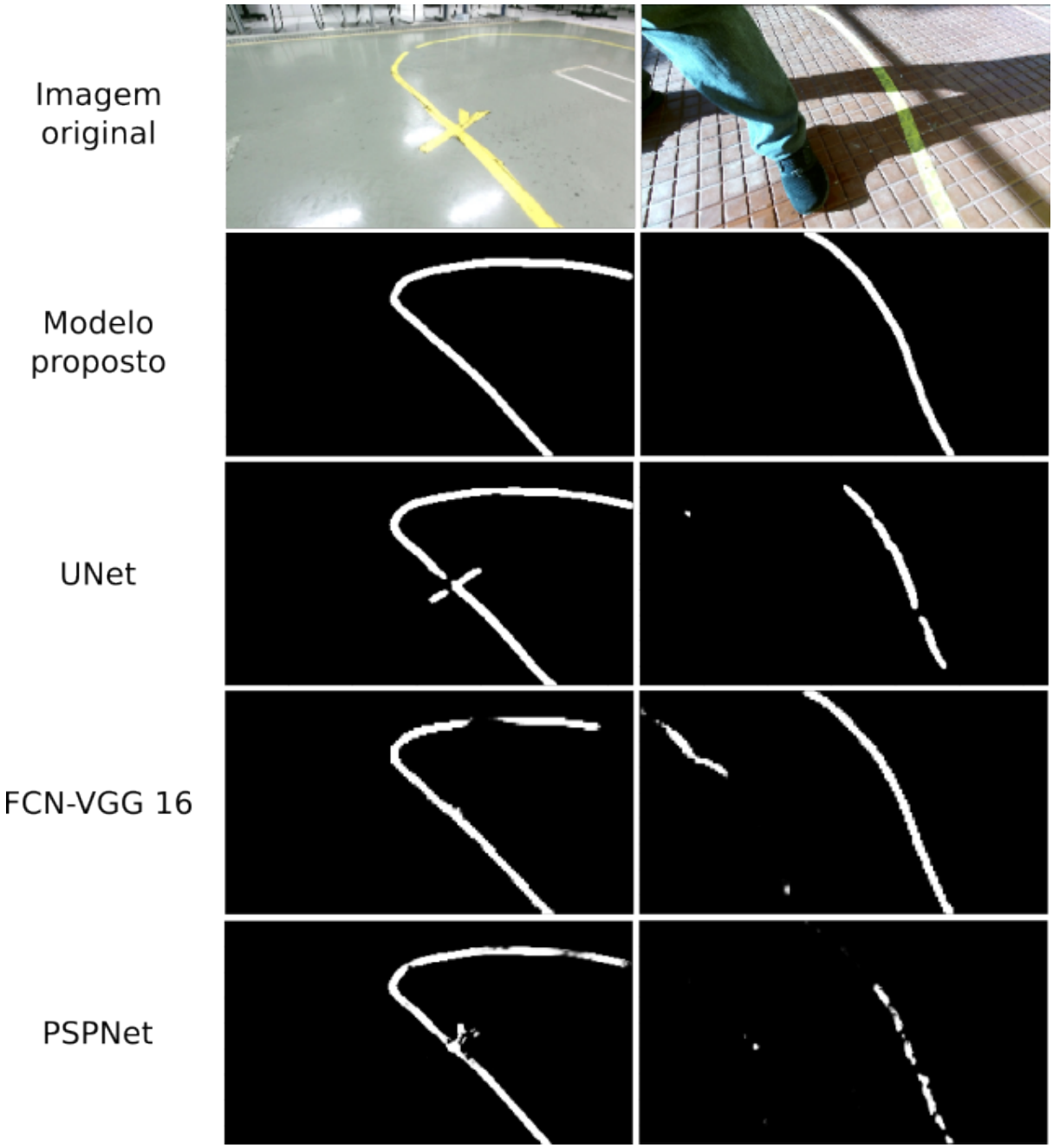


Figura 4.5: Comparação da qualidade de segmentação de caminho entre os diferentes modelos de aprendizado

um modelo de movimentação baseado em imagem para comparar o desempenho dos dois sistemas.

4.2.1. Ambiente de Simulação

Existem diversos *softwares* para simulação de sistemas robóticos, desde opções com recursos mais limitados até plataformas mais avançadas, com funcionalidades abrangentes e alto grau de personalização. Dentre os simuladores mais utilizados, destacam-se Gazebo, CoppeliaSim, Webots e Isaac Sim. Além dessas ferramentas desenvolvidas especificamente para robótica, algumas *game engines* vêm ganhando notoriedade nesse contexto, principalmente devido ao alto realismo gráfico, permitindo simulações mais próximas da realidade. Entre as opções mais populares, destacam-se Unity e Unreal Engine, amplamente conhecidas no desenvolvimento de jogos 2D e 3D.

Neste trabalho, o Unity foi escolhido como simulador devido à qualidade das renderizações, à extensa documentação disponível para simulações robóticas e à simples integração com o *Robot Operating System* (ROS). O ROS é um *framework* amplamente utilizado na robótica, oferecendo um ecossistema modular que facilita a comunicação entre diferentes componentes do sistema, permitindo a implementação e o teste de algoritmos de percepção, controle e navegação em ambientes simulados antes da implantação em hardware real.

A Figura 4.6 ilustra duas cenas ambientadas no LaR. A primeira (imagem superior) foi criada no Gazebo e a segunda (imagem inferior) no Unity. É possível observar que a cena do Unity possui detalhes visuais como iluminação, sombras e reflexos mais realistas.

4.2.2. Estimação de Estados

O problema de controle visual, conforme proposto por Franco, Ribeiro e Conceição (2021), tem seu sistema de coordenadas ilustrado na Figura 4.7. Nesse modelo, para um perfil de velocidade linear pré-determinado, os estados são características visuais extraídas a cada iteração que representam o erro entre a velocidade angular do robô e a de um veículo virtual posicionado no horizonte visual. O modelo é descrito pelas seguintes equações:

$$u_e = \omega - c(s) \cdot \frac{(v + \omega Z)}{cos(\theta_r)}, \tag{4.1}$$

$$x_e = \begin{bmatrix} \dot{z} \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \omega H + (\omega Z + v) \cdot tan(\theta_r) \\ u_e \end{bmatrix}, \tag{4.2}$$

onde u_e é a entrada de controle, s é o comprimento do caminho, $c(s)$ é a curvatura do caminho, Z é o deslocamento lateral, θ_r é o erro angular e H o horizonte constante. Nesse modelo, Z e θ_r são as variáveis de estado e podem ser calculadas diretamente do plano da imagem utilizando relações trigonométricas.

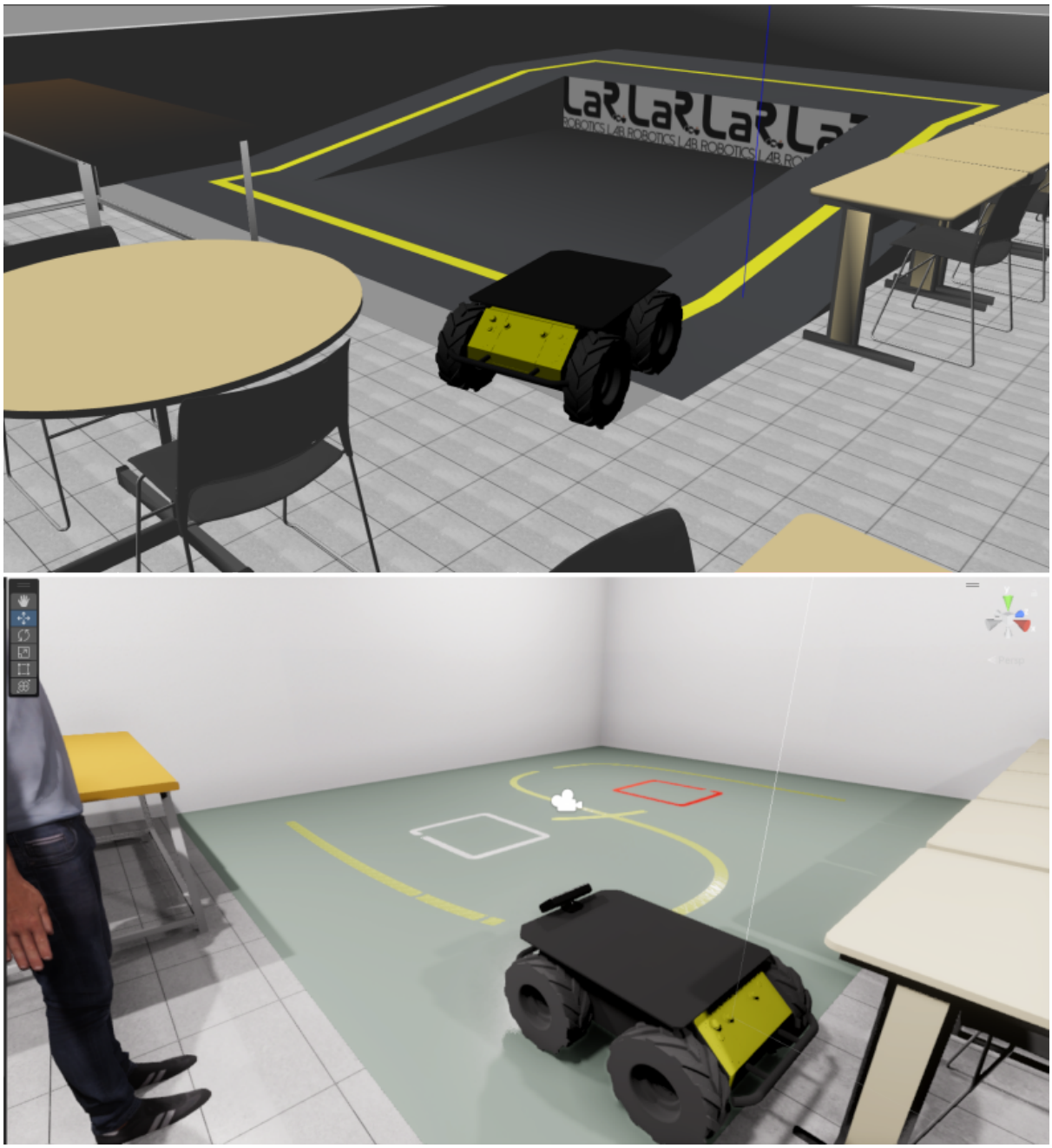
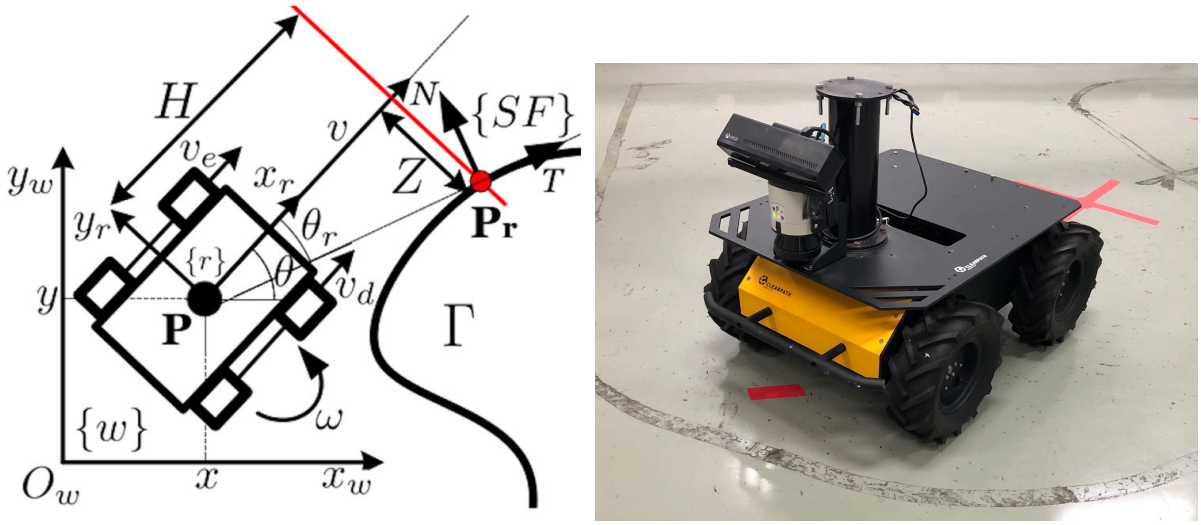


Figura 4.6: Simulação do LaR no Gazebo (imagem superior) e no Unity (imagem inferior)



(a) Sistema de coordenadas.

(b) *Clearpath Husky A200 UGV.*

Figura 4.7: Representação do sistema de coordenadas para o problema de controle visual e foto do robô móvel utilizado.

Este modelo extrai os estados a partir do caminho parametrizado pelo método RANSAC, conforme explicado no Capítulo 3.

4.2.3. Comparação com Método Alternativo baseado em Visão Computacional Clássica

Para avaliar o desempenho do sistema de detecção, este foi comparado ao sistema apresentado por Franco, Ribeiro e Conceição (2021), que utiliza métodos de visão computacional clássica para detectar caminhos. Foram utilizadas imagens geradas a partir do ambiente de simulação construído com a *engine* Unity, conforme a Figura 4.8, onde o robô percorre uma faixa para realizar a detecção e a estimação de estados utilizando ambas abordagens. O ambiente de simulação foi estruturado para oferecer condições desafiadoras para o sistema de detecção, com reflexos e focos de luz no solo, sombras e caminhos faltosos, e texturas bastante realistas, a fim de testar o sistema sob circunstâncias bem próximas das reais. No canto superior esquerdo da Figura 4.8, observa-se a imagem capturada com a câmera montada sobre o robô, enquanto no canto superior direito encontra-se o caminho detectado sobreposto pela função polinomial que parametriza o caminho.

Um nó do ROS foi implementado para fazer a estimação dos estados a partir da função polinomial utilizando as relações trigonométricas do sistema de coordenadas exibido na Figura 4.7a.

A Figura 4.9 apresenta o gráfico do estado Z estimado para uma janela de tempo, enquanto a Figura 4.10 mostra alguns exemplos de detecção obtidos utilizando ambas abordagens.

O método baseado em visão computacional clássica apresentou regiões de instabilidade

Figura 4.8: Simulação do *Husky UGV* utilizando a engine Unity. Um video com a demonstração do sistema pode ser visto no link ⟨https://youtu.be/V6X5VH_C8pY⟩.

na estimação dos estados, por detectar erroneamente trechos faltosos no caminho devido a diferenças de iluminação. Por se tratar de um método determinístico de visão computacional, a capacidade de generalização do sistema é severamente afetada por condições de luminosidade diferentes daquelas para as quais o sistema foi projetado.

Como consequência, são encontradas regiões de pico e maiores oscilações no estado estimado, o que pode levar um sistema de controle à instabilidade, fazendo com que o robô deixe de detectar e seguir o caminho. É possível observar, no gráfico da Figura 4.9, o comportamento instável descrito. Em $t = 15,3$ segundos, nota-se um pico no deslocamento lateral estimado. Este instante está representado no primeiro exemplo (primeira linha) da Figura 4.10. Observa-se também que em $t = 20$ segundos o estado estimado utilizando o método original diverge, tendendo a valores elevados que não condizem com o estado real.

Enquanto isso, o método proposto detectou com precisão os caminhos, mesmo os faltosos ou com sombras e focos de luz, demonstrando uma maior robustez diante de condições adversas. Isso indica que o modelo proposto possui uma capacidade superior de generalização, permitindo a identificação do caminho mesmo em cenários onde as marcações estão parcialmente obstruídas ou degradadas.

Além disso, nota-se que a estabilidade do estado estimado foi mantida ao longo do tempo, sem picos abruptos ou oscilações significativas, o que contribui diretamente para a confiabilidade do sistema de navegação. Esse comportamento é essencial para garantir que o robô se mantenha no caminho de forma segura e previsível, reduzindo o risco de falhas na detecção ou de tomadas de decisão errôneas pelo controlador.

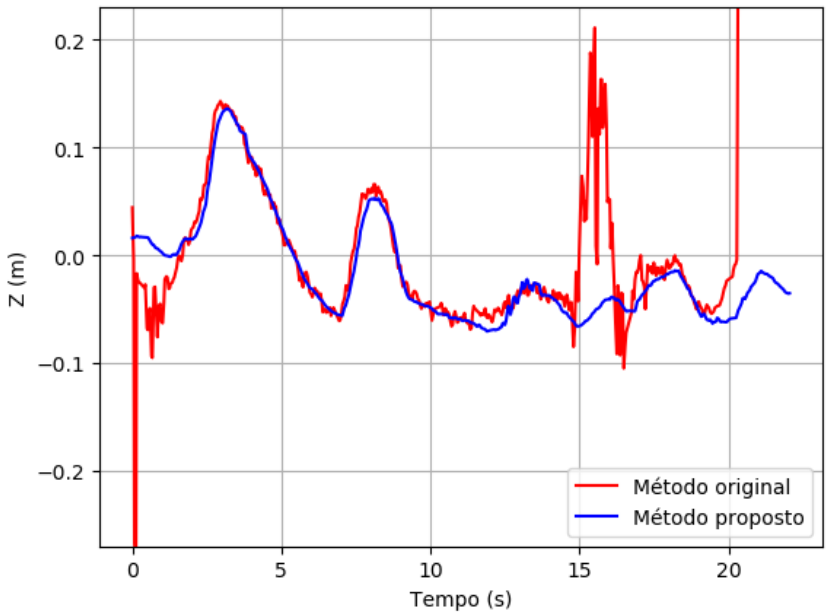


Figura 4.9: Estimação do deslocamento lateral (Z) em uma janela de tempo. Em vermelho, o estado estimado utilizando o método original, em azul, utilizando o método proposto neste artigo.

Dessa forma, os resultados sugerem que a abordagem proposta não apenas melhora a qualidade da segmentação dos caminhos, mas também impacta positivamente a estabilidade do sistema como um todo. Isso é fundamental para aplicações de navegação autônoma, onde precisão e consistência na detecção são essenciais para um desempenho seguro e eficiente.

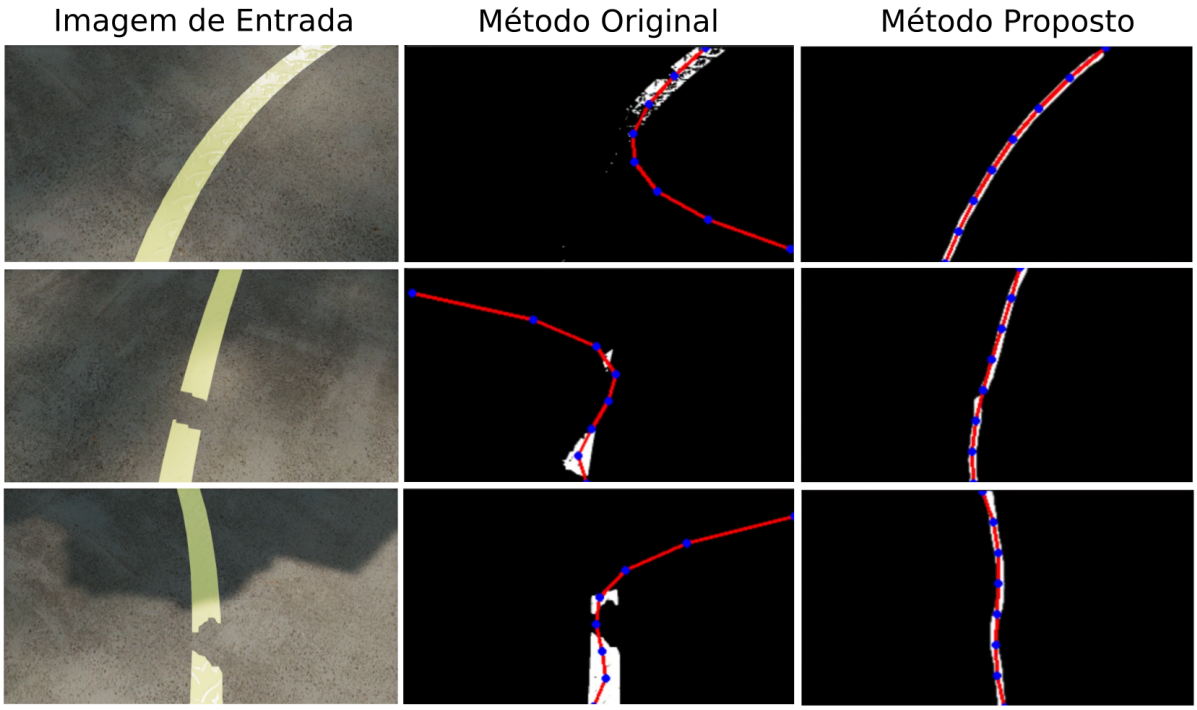


Figura 4.10: Comparação entre os resultados do método original (segunda coluna) e do método proposto (terceira coluna) na detecção de caminhos para diferentes imagens de entrada (primeira coluna).

Capítulo 5

Este capítulo apresenta as considerações finais e as discussões de possíveis melhorias e trabalhos futuros

CONCLUSÕES

5.1. CONCLUSÕES SOBRE A EFICÁCIA DO SISTEMA PROPOSTO

Este trabalho apresentou um sistema de detecção visual de caminhos através de aprendizado profundo para a navegação de robôs móveis. O sistema foi desenvolvido utilizando rede neural convolucional baseada na ERFNet, que possui uma seção de *encoder*, onde ocorre a extração de características relevantes da imagem, e uma seção de *decoder*, responsável pela reconstrução da segmentação do caminho a partir das informações extraídas. A arquitetura escolhida foi otimizada para alcançar um equilíbrio entre precisão e eficiência computacional, permitindo a execução em tempo real em GPUs de entrada.

Além disso, o treinamento foi realizado utilizando a técnica de transferência de aprendizado, aproveitando o conhecimento previamente adquirido em um conjunto de dados amplamente utilizado para detecção de faixas. Inicialmente, a rede foi pré-treinada no conjunto TuSimple, permitindo que aprendesse representações relevantes para a tarefa de segmentação de caminhos. Esse processo possibilitou um melhor aproveitamento dos recursos computacionais e uma convergência mais eficiente do modelo, uma vez que as camadas convolucionais puderam ser inicializadas com pesos previamente ajustados em um domínio semelhante.

A utilização do *transfer learning* mostrou-se importante para a robustez do modelo proposto, permitindo que ele generalizasse melhor para cenários variados, incluindo condições desafiadoras como iluminação irregular, sombras e faixas desgastadas. Além disso, a abordagem mitigou o impacto da limitação de dados do conjunto de treinamento final, já que a rede iniciou o aprendizado a partir de um ponto mais avançado, em vez de partir do zero. Dessa forma, a transferência de aprendizado contribuiu significativamente para a melhoria dos resultados obtidos, reforçando sua importância na aplicação de aprendizado profundo em tarefas de visão computacional.

Quando comparado a outros modelos de CNN na tarefa de segmentação de caminhos, como a UNet, a FCN-VGG 16 e a PSPNet, o modelo proposto demonstrou um desempenho superior em termos de qualidade preditiva. Conforme evidenciado na Tabela 4.2,

a abordagem baseada na ERFNet obteve maiores valores de IoU, F1-*score* e acurácia em relação às demais arquiteturas analisadas. Esse resultado destaca a capacidade do modelo em realizar uma segmentação mais precisa, o que é essencial para aplicações de navegação robótica.

Além disso, a análise qualitativa das segmentações geradas por cada modelo revelou que a abordagem baseada na ERFNet apresentou uma capacidade superior de abstração na detecção do caminho, como ilustrado na Figura 4.5. Essa característica é fundamental para lidar com condições adversas, como falhas nas marcações, sombras e variações na iluminação, tornando a navegação do robô mais confiável. Assim, os experimentos realizados sugerem que a utilização da ERFNet para a detecção de caminhos visuais não apenas melhora a acurácia da segmentação, mas também pode contribuir para um controle de trajetória mais eficiente em ambientes dinâmicos.

Por fim, um estudo de caso foi realizado para comparar o desempenho do método proposto com uma abordagem alternativa baseada em visão computacional clássica. Utilizando um ambiente de simulação na engine Unity, o robô percorreu uma faixa para detectar e estimar os estados do caminho sob diferentes condições adversas, como variações de iluminação, sombras e trechos faltosos. Os resultados demonstraram que o método tradicional apresentou instabilidades na estimação dos estados, resultando em oscilações e erros significativos que poderiam comprometer a navegação autônoma. Em contrapartida, a abordagem proposta mostrou maior robustez, mantendo a estabilidade da estimativa ao longo do tempo e permitindo a detecção precisa do caminho, mesmo em cenários desafiadores. Esses achados reforçam a eficiência e confiabilidade do modelo desenvolvido, evidenciando seu potencial para aplicações em sistemas de navegação autônoma.

5.2. DISCUSSÃO DE POSSÍVEIS MELHORIAS E TRABALHOS FUTUROS

Embora o modelo proposto tenha apresentado um desempenho superior, foi observado um aumento no tempo de inferência em comparação com as abordagens tradicionais, evidenciando um possível *trade-off* entre precisão e eficiência computacional. Todos os modelos foram treinados e testados no mesmo conjunto de dados, sob as mesmas condições, incluindo hiperparâmetros idênticos e a mesma estratégia de *transfer learning*, garantindo uma avaliação justa. Apesar desse aumento no tempo de inferência, em aplicações de robótica móvel, a precisão na segmentação pode ser mais crítica do que a velocidade de processamento, desde que esta permaneça dentro de limites aceitáveis para o controlador. Trabalhos futuros podem explorar otimizações no tempo de inferência, seja por meio de modelos mais leves ou técnicas de compressão de redes neurais, sem comprometer a robustez da segmentação.

Como continuidade deste trabalho, pretende-se explorar o uso de sensores adicionais, como LiDAR ou câmeras estéreo, para aprimorar a detecção de caminhos em terrenos não planos, acidentados ou inclinados, ampliando a aplicabilidade do sistema para diferentes cenários de navegação autônoma. Além disso, a integração de um controlador ao sistema de detecção desenvolvido permitirá avaliar seu impacto direto na estabilidade e

no desempenho da navegação, possibilitando ajustes dinâmicos na trajetória do robô com base nas informações extraídas. Essas melhorias contribuirão para tornar o sistema mais robusto e versátil, viabilizando sua aplicação em ambientes mais desafiadores.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, A. S.; RIBEIRO, T. T.; CONCEICAO, A. G. S. Detecção de faixa baseado em aprendizado profundo para seguimento de caminho visual. In: *Congresso Brasileiro de Automática 2024 (CBA)*. [S.l.: s.n.], 2024.

ALMEIDA, A. S.; RIBEIRO, T. T.; CONCEICAO, A. G. S. Transfer learning-based lane line detection system for visual path following control. In: *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. [s.n.], 2024. p. 782–787. Disponível em: ⟨https://ieeexplore.ieee.org/abstract/document/10711739⟩.

BEERS, F. van. *Using Intersection over Union loss to improve Binary Image Segmentation*. Dissertação (Mestrado) — University of Groningen, 2018. Disponível em: ⟨https://fse.studenttheses.ub.rug.nl/18139/⟩. Acesso em: novembro de 2023.

BEKEY, G. et al. *Robotics: State of the Art and Future Challenges*. [S.l.]: Imperial Colege Press, 2008.

BODNAR, C. *Text to Image Synthesis Using Generative Adversarial Networks*. Tese (Doutorado) — University of Cambridge, 04 2018. Disponível em: ⟨https://www.researchgate.net/publication/324783775_Text_to_Image_Synthesis_Using_Generative_Adversarial_Networks⟩. Acesso em: julho de 2024.

CHNG, Z. M.; LEW, J. M. H.; LEE, J. A. Roneld: Robust neural network output enhancement for active lane detection. *Proceedings - International Conference on Pattern Recognition*, p. 6842–6849, 2020. ISSN 10514651. Disponível em: ⟨https://doi.org/10.48550/arXiv.2010.09548⟩. Acesso em: junho de 2023.

DOYLE, B. Do robots create jobs? the data says yes! In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. [s.n.], 2016. p. 1–5. Disponível em: ⟨https://ieeexplore.ieee.org/abstract/document/7559099⟩. Acesso em: setembro de 2023.

DUDEK, G.; JENKIN, M. *Computational Principles of Mobile Robotics*. 2. ed. [S.l.]: Cambridge University Press, 2010.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 24, n. 6, p. 381–395, jun 1981. ISSN 0001-0782. Acesso em: novembro de 2023.

FRANCO, I. J. P. B.; RIBEIRO, T. T.; CONCEICAO, A. G. S. A novel visual lane line detection system for a NMPC-based path following control scheme. *Journal of Intelligent*

& Robotic Systems, v. 101, p. 12, 2021. Disponível em: ⟨https://link.springer.com/article/10.1007/s10846-020-01278-x⟩. Acesso em: março de 2025.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. (Adaptive Computation and Machine Learning series). ISBN 9780262035613. Disponível em: ⟨https://books.google.com.br/books?id=Np9SDQAAQBAJ⟩.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*, 2015. Disponível em: ⟨https://arxiv.org/abs/1502.03167⟩. Acesso em: novembro de 2023.

JURKAT, A.; KLUMP, R.; SCHNEIDER, F. Tracking the rise of robots: The ifr database. *Jahrbücher für Nationalökonomie und Statistik*, v. 242, n. 5-6, p. 669–689, 2022. Disponível em: ⟨https://doi.org/10.1515/jbnst-2021-0059⟩. Acesso em: setembro de 2023.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Disponível em: ⟨https://arxiv.org/abs/1412.6980⟩. Acesso em: novembro de 2023.

LEE, D.-H.; LIU, J.-L. End-to-End Deep Learning of Lane Detection and Path Prediction for Real-Time Autonomous Driving. *Signal, Image and Video Processing*, fevereiro de 2022. Disponível em: ⟨https://link.springer.com/article/10.1007/s11760-022-02222-2⟩. Acesso em: abril de 2023.

LEI, T.; NANDI, A. *Image Segmentation: Principles, Techniques, and Applications*. Wiley, 2022. ISBN 9781119859000. Disponível em: ⟨https://books.google.com.br/books?id=wwmQzgEACAAJ⟩.

LI, Z. et al. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, v. 33, n. 12, p. 6999–7019, 2022. Disponível em: ⟨https://ieeexplore-ieee-org.ez10.periodicos.capes.gov.br/stampPDF/getPDF.jsp?tp=&arnumber=9451544⟩. Acesso em: novembro de 2023.

LIU, T. et al. Lane Detection in Low-light Conditions Using an Efficient Data Enhancement: Light Conditions Style Transfer. *IEEE Intelligent Vehicles Symposium, Proceedings*, Institute of Electrical and Electronics Engineers Inc., p. 1394–1399, 2020. Disponível em: ⟨https://ieeexplore.ieee.org/abstract/document/9304613⟩. Acesso em: maio de 2023.

LONG, J.; SHELHAMER, E.; DARRELL, T. *Fully Convolutional Networks for Semantic Segmentation*. 2015. Disponível em: ⟨https://arxiv.org/abs/1411.4038⟩. Acesso em: março de 2025.

NOH, H.; HONG, S.; HAN, B. Learning deconvolution network for semantic segmentation. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. [s.n.], 2015. p. 1520–1528. Disponível em: ⟨https://ieeexplore.ieee.org/document/7410535⟩. Acesso em: novembro de 2023.

OLIVEIRA, G. L.; BURGARD, W.; BROX, T. Efficient deep models for monocular road segmentation. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [s.n.], 2016. p. 4885–4891. Disponível em: ⟨https://ieeexplore.ieee.org/document/7759717⟩. Acesso em: agosto de 2023.

PAN, J. et al. Enhanced fcn for farmland extraction from remote sensing image. *Multimedia Tools and Applications*, v. 81, p. 1–28, 04 2022. Disponível em: ⟨https://www.researchgate.net/publication/360150806_Enhanced_FCN_for_farmland_extraction_from_remote_sensing_image⟩. Acesso em: julho de 2024.

PRINCE, S. J. *Understanding Deep Learning*. The MIT Press, 2023. Disponível em: ⟨http://udlbook.com⟩. Acesso em: julho de 2024.

ROBERTS, D. A.; YAIDA, S.; HANIN, B. The principles of deep learning theory. *CoRR*, abs/2106.10165, 2021. Disponível em: ⟨https://arxiv.org/abs/2106.10165⟩. Acesso em: julho de 2024.

ROMERA, E. et al. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, v. 19, n. 1, p. 263–272, 2018. Disponível em: ⟨https://ieeexplore.ieee.org/document/8063438⟩. Acesso em: junho de 2023.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: NAVAB, N. et al. (Ed.). *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015. p. 234–241. ISBN 978-3-319-24574-4. Disponível em: ⟨https://arxiv.org/abs/1505.04597⟩. Acesso em: agosto de 2023.

SANDLER, M. et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [s.n.], 2018. p. 4510–4520. Disponível em: ⟨https://ieeexplore.ieee.org/document/8578572⟩. Acesso em: agosto de 2023.

SHAH, D. et al. Gnm: A general navigation model to drive any robot. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. [s.n.], 2023. p. 7226–7233. Disponível em: ⟨https://ieeexplore.ieee.org/abstract/document/10161227⟩. Acesso em: agosto de 2023.

SHAO, L.; ZHU, F.; LI, X. Transfer learning for visual categorization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, v. 26, n. 5, p. 1019–1034, 2015. Disponível em: ⟨https://ieeexplore-ieee-org.ez10.periodicos.capes.gov.br/document/6847217⟩. Acesso em: novembro de 2023.

SHOTTON, J.; KOHLI, P. Semantic image segmentation. In: IKEUCHI, K. (Ed.). *Computer Vision: A Reference Guide*. Boston, MA: Springer US, 2014. p. 713–716. ISBN 978-0-387-31439-6. Disponível em: ⟨https://doi.org/10.1007/978-0-387-31439-6_251⟩. Acesso em: outubro de 2023.

SIEGWART, R.; NOURBAKHSH, I. *Introduction to Autonomous Mobile Robots*. Bradford Book, 2004. (A Bradford book). ISBN 9780262195027. Disponível em: ⟨https://books.google.com.br/books?id=gUbQ9_weg88C⟩.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: . Computational and Biological Learning Society, 2015. p. 1–14. Disponível em: ⟨https://arxiv.org/abs/1409.1556⟩. Acesso em: agosto de 2023.

SINGH, H. *How Do Computers Store Images?* 2023. Disponível em: ⟨https://www.analyticsvidhya.com/blog/2021/03/grayscale-and-rgb-format-for-storing-images⟩. Acesso em: julho de 2024.

TANG, J.; LI, S.; LIU, P. A review of lane detection methods based on deep learning. *Pattern Recognition*, Pergamon, v. 111, março de 2021. ISSN 0031-3203. Disponível em: ⟨https://www.sciencedirect.com/science/article/abs/pii/S003132032030426X⟩. Acesso em: abril de 2023.

WADA, K. *Labelme: Image Polygonal Annotation with Python*. 2016. Disponível em: ⟨https://github.com/wkentaro/labelme⟩. Acesso em: outubro de 2023.

YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. Disponível em: ⟨https://arxiv.org/abs/1511.07122⟩. Acesso em: novembro de 2023.

ZHAO, H. et al. *Pyramid Scene Parsing Network*. 2017. Disponível em: ⟨https://arxiv.org/abs/1612.01105⟩. Acesso em: março de 2025.

ZHUANG, F. et al. *A Comprehensive Survey on Transfer Learning*. 2020. Acesso em: dezembro de 2023.