

**PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO
CIENTÍFICA E TECNOLÓGICA**

**UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
DARCY RIBEIRO**

*Centro CCT
Labotatório LCMAT*

Relatório do período: Junho 2024 - Junho 2025

Relatório Anual PIBIC

Bolsista: Daniel Terra Gomes

Matrícula: 00119110484

Orientadora: Profa. Dra. Annabell Del Real Tamariz

Curso: Bacharelado em Ciência da Computação

Titulo do Projeto: Project-driven Data Science: Aprendendo e Mapeando

Título do Plano de Trabalho: Exploração e Aplicação de Conceitos Fundamentais de
Carros Autônomos.

Fonte Financiadora: PIBIC/CNPq

“Behind me lies a farm.

*I wonder if there is bread above the hearth
and if I will ever return.”*

(Pantheon, League of Legends)

Resumo

Os Veículos Autônomos (VAs) representam um avanço tecnológico significativo, prometendo transformações na mobilidade urbana e segurança viária por meio de tecnologias embarcados. No entanto, desafios persistem na detecção de objetos e na interpretação de cenários, especialmente para atingir níveis de automação (Nível 5 SAE). Neste contexto, este estudo propõe o desenvolvimento de um sistema de assistência à condução em VAs, com foco na detecção e reconhecimento de placas de trânsito, proporcionando feedback visual ao condutor e contribuindo para uma navegação mais segura. Para tal, foram empregadas simulações computacionais no ambiente CARLA, nas quais foi implementado e avaliado um modelo baseado no algoritmo YOLO (You Only Look Once) para a identificação de sinalizações viárias. O treinamento do modelo foi conduzido com um conjunto de dados sintético, gerado a partir de cenários virtuais, permitindo a avaliação do sistema sob diferentes condições ambientais e situações adversas, sem os riscos e custos inerentes a testes e coleta de dados em ambientes reais. Os resultados obtidos demonstram que a abordagem proposta foi eficaz na detecção de placas de trânsito, apresentando precisão superior a 90%. Além disso, observou-se que soluções de visão computacional, como o algoritmo YOLO, viabilizam sistemas rápidos e eficientes para aplicações de sistemas autônomas. Contudo, a interpretabilidade desses algoritmos ainda se apresenta como uma limitação, o que ressalta a necessidade de investigações futuras voltadas ao desenvolvimento de soluções com algoritmos mais explicáveis. Por fim, este estudo proporcionou uma compreensão aprofundada e uma aplicação prática dos conceitos de estimativa e localização de estado, percepção visual e planejamento de movimento aplicados em carros autônomos pela indústria automobilística.

Palavras-chave: Carros autônomos. Detecção de Objetos. Controle Veicular. Simulação. CARLA. YOLO.

Abstract

Autonomous Vehicles (AV) represent a significant technological advancement, promising transformations in urban mobility and road safety through embedded technologies. However, object detection and scene interpretation challenges persist, especially in achieving complete automation levels (SAE Level 5). In this context, this study proposes the development of a driver assistance system for AV, focusing on the detection and recognition of traffic signs, providing visual feedback to the driver, and contributing to safer navigation. To this end, computational simulations were conducted in the CARLA environment, in which a model based on the YOLO (You Only Look Once) algorithm was implemented and evaluated for traffic sign identification. The model was trained using a synthetic dataset generated from virtual scenarios, allowing the system to be evaluated under different environmental conditions and adverse situations without the risks and costs associated with testing and data collection in real-world environments. The results demonstrate that the proposed approach was effective in traffic sign detection, achieving an accuracy of over 90%. Additionally, it was observed that computer vision solutions, such as the YOLO algorithm, enable fast and efficient systems for autonomous applications. However, the interpretability of these algorithms remains a limitation, highlighting the need for future research focused on developing more explainable algorithmic solutions. Finally, this study provided an in-depth understanding and practical application of concepts related to state estimation and localization, visual perception, and motion planning, as applied to autonomous vehicles in the automotive industry.

Keywords: Autonomous Vehicles. Object Detection. Vehicle Control. Simulation. CARLA. YOLO.

Listas de ilustrações

Figura 1 – Método Científico (CRISTINA; WAZLAWICK, 2021, p. 4).	16
Figura 2 – Etapas do projeto para o 3º ano de pesquisa.	23
Figura 3 – O Sistema de Detecção YOLO. O processamento de imagens com YOLO é simples e direto. O sistema (1) redimensiona a imagem de entrada para 448×448 , (2) executa uma única rede convolucional na imagem e (3) limita as detecções resultantes pela confiança do modelo. (REDMON et al., 2016, p. 1).	24
Figura 4 – Comparações com outros em termos de trade-offs de latência-precisão (esquerda) e tamanho-precisão (direita). Medimos a latência de ponta a ponta usando os modelos oficiais pré-treinados. (WANG et al., 2024, p. 1).	25
Figura 5 – Arquitetura do Software. Elaborado pelo Autor.	26
Figura 6 – Arquitetura de Software da Percepção do Ambiente. Baseado em University of Toronto (2018a).	27
Figura 7 – Arquitetura de Software do Planejador de Movimento. Baseado em University of Toronto (2018a).	27
Figura 8 – Arquitetura de Software do Controlador do Veículo. Baseado em University of Toronto (2018a).	28
Figura 9 – Fluxo da Execução da Solução. Elaborado pelo Autor.	28
Figura 10 – Planejamento Hierárquico (University of Toronto, 2018b, Module 2 - Lesson 1: Driving Missions, Scenarios, and Behaviour. 10min55s).	30
Figura 11 – Máquinas de Estados Finitos (University of Toronto, 2018b, Module 2 - Lesson 4: Hierarchical Motion Planning. 7min54s).	31
Figura 12 – Lattice Planner (University of Toronto, 2018b, Module 2 - Lesson 4: Hierarchical Motion Planning. 15min30s).	32
Figura 13 – Conformal Lattice Planner (University of Toronto, 2018b, Module 2 - Lesson 4: Hierarchical Motion Planning. 15min30s).	32
Figura 14 – Imagem a partir da Camera: Scene final (CARLA Simulator, 2024).	34
Figura 15 – Modelo. O sistema modela a detecção como um problema de regressão. Dividindo a imagem em uma grade $S \times S$ e para cada célula da grade prevê caixas delimitadoras B , confiança para essas caixas e probabilidades de classe C . Segundo os autores, essas previsões são codificadas como um tensor $S \times S \times (B \times 5 + C)$. Redmon et al. (2016, p. 2).	36

Figura 16 – Arquitetura. A rede de detecção tem 24 camadas convolucionais seguidas por 2 camadas totalmente conectadas. Camadas convolucionais alternadas 1×1 reduzem o espaço de recursos das camadas precedentes. Nós pré-treinamos as camadas convolucionais na tarefa de classificação do ImageNet na metade da resolução (imagem de entrada 224×224) e então dobramos a resolução para detecção (REDMON et al., 2016, p. 3).	36
Figura 17 – Esquema geral de um sistema de detecção e reconhecimento de sinais de trânsito usando o algoritmo de detecção de objetos YOLO. Imagem de (FLORES-CALERO et al., 2024, p. 2).	38
Figura 18 – Trecho do Mapa 1 (CARLA, 2018).	44
Figura 19 – Estrutura do projeto (Elaborado pelos autores).	45
Figura 20 – Captura de Tela da execução da solução no simulador CARLA (Elaborado pelos autores).	48
Figura 21 – Trajetória percorrida pela solução (Elaborado pelos autores).	51
Figura 22 – Imagem capturada da saída do YOLO durante a simulação.	52
Figura 23 – Certificado de conclusão da Especialização em Self-Driving Cars.	55
Figura 24 – Certificado de participação no CONFICT 2024.	56
Figura 25 – Certificado de participação no minicurso de Robótica Competitiva durante o CONFICT 2024.	57

Lista de trechos de código

4.1	Exemplo da documentação CARLA para adicionar uma câmera “Scene final” no simulador (CARLA Simulator, 2024)	34
5.1	Inicializando o simulador Carla na pista Town 1	47
5.2	Executando a solução no simulador Carla	47
5.3	Configuração e carregamento do modelo YOLOv3 utilizando o OpenCV DNN (Elaborado pelos autores)	48
5.4	Loop de detecção de objetos com uso do modelo YOLOv3 sobre imagens da câmera no CARLA (Elaborado pelos autores)	49

Lista de abreviaturas e siglas

VA	Veículo Autônomo
VAs	Veículos Autônomos
IA	Inteligência Artificial
SAE	Society of Automotive Engineers
GPS	Global Positioning System
ICR	Instantaneous Center of Rotation
RPM	Rotação Por Minuto
P	Proporcional
PI	Proporcional Integral
PD	Proporcional Derivativo
PID	Proporcional Integral Derivativo
Feedback	Realimentação
Feedforward	Antecipação
2D	Bidimensional
3D	Tridimensional
MPC	Model Predictive Controller

Sumário

1	INTRODUÇÃO	11
1.1	Problema	13
1.2	Hipótese	13
1.3	Objetivos	14
1.3.1	Objetivos Específicos	15
1.4	Justificativa	15
1.5	Método	15
1.6	Organização do Trabalho	21
2	ETAPAS PROPOSTAS NO PLANO DE TRABALHO	23
3	DETECÇÃO E CLASSIFICAÇÃO DE OBJETOS	24
4	MODELAGEM CONCEITUAL PARA SIMULAÇÃO DE CARROS AUTÔNOMOS	26
4.1	Planejamento de Movimento de Carros Autônomos	29
4.2	Percepção Visual de Carros Autônomos	33
4.2.1	Câmeras	33
4.2.2	You Only Look Once - YOLO	35
4.2.2.1	Arquitetura	35
4.2.2.2	Treinamento	37
5	IMPLEMENTAÇÃO	40
5.1	Configurando o Simulador Carla	41
5.2	Implementação do Controle de Veículos Autônomos	43
5.3	Integração da Solução YOLO ao Simulador CARLA	48
6	RESULTADOS E DISCUSSÃO	51
7	CONSIDERAÇÕES FINAIS	53
8	PERSPECTIVA DE CONTINUIDADE	54
9	PARTICIPAÇÃO EM CONGRESSOS E TRABALHOS PUBLICADOS OU SUBMETIDOS E OUTRAS ATIVIDADES ACADÊMICAS E DE PESQUISA	55
10	DATAS E ASSINATURAS	58

10.1	Data e assinatura do bolsista (assinatura digitalizada)	58
10.2	Data e assinatura do orientador (assinatura digitalizada)	58
REFERÊNCIAS		59
APÊNDICES		64
APÊNDICE A – MATERIAL COMPLETO		65
ANEXOS		66
ANEXO A – MATERIAL DE RELEVÂNCIA		67

1 Introdução

Veículos Autônomos (VAs) representam um dos avanços tecnológicos mais promissores para transformar a mobilidade urbana e a segurança no transporte (European Commission, Directorate-General for Mobility and Transport, 2021), (OTHMAN, 2021). Seu surgimento acompanha o crescimento do mercado de veículos elétricos (SEBO, 2024), permitindo que empresas como ZOOX®, WAYMO® e LUME ROBOTICS® invistam significativamente em sistemas de condução autônoma. Os quais, devido a tecnologias embarcadas de visão computacional, sensores de proximidade, entre outros, prometem maior eficiência, segurança e conveniência nos sistemas de transporte público e privado (BRATZEL; CAM, 2022), (PAREKH NISHI PODDAR, 2022; DREIBELBIS, 2023). A partir da incorporação em ampla escala dessa tecnologia será possível notar impactos transformadores na infraestrutura urbana, na configuração das cidades e no comportamento social das pessoas, afetando positivamente a mobilidade cotidiana de diversos grupos populacionais, incluindo crianças, idosos e pessoas com mobilidade reduzida (LAGE, 2019; KPMG International, 2020). Desses impactos, um dos mais significativos seria em redução nos acidentes de trânsito. Estudos indicam que erros humanos são responsáveis por mais de 90% dos acidentes de trânsito (National Highway Traffic Safety Administration, 2018), fazendo da tecnologia de VAs uma solução promissora para reduzir significativamente esses incidentes. Visto que esses veículos têm o poder de tomar ações por conta própria, e/ou dar assistência ao motorista em casos de situações perigosas (OKPONO et al., 2024). Estimativas sugerem que a adoção em larga escala de VAs poderia evitar até 585.000 mortes ao longo de uma década a partir de 2035 (LANCTOT, 2017). Nesse quesito, soluções de detecção de objetos e a análise de cenários em tempo real, podem ser um aliado dos motoristas no seu dia a dia (Federal Highway Administration, 2012). Visto que tais tecnologias embarcadas, podem auxiliar e assistir os condutores durante uma tarefa de condução evitando acidentes e garantindo uma navegação segura (JANAI et al., 2020).

No entanto, para que a tecnologia de VAs avance até o ponto de uma direção autônoma nível 5 SAE (SAE, 2021, p. 28), é necessário enfrentar uma série de desafios técnicos, especialmente em relação à visão computacional e sistemas de detecção de objetos, onde se faz necessário identificar e informar rapidamente sobre novas detecções durante uma tarefa de condução. Avanços em estudos de desempenho de soluções de visão computacional em detecção de objetos enfrentam desafios quanto a altos custos na infraestrutura, logística e legislação associados a testes em veículos no mundo real. Devido a isso, pesquisadores estão cada vez mais recorrendo a simulações virtuais para desenvolver e validar essas soluções em sistemas autônomos (DOSOVITSKIY et al., 2017; AHIRE et al., 2024). Ambientes simulados oferecem um cenário controlado e econômico para

testar algoritmos de condução autônoma em diversas situações, sem os riscos imediatos ou custos com recursos dos testes físicos. Dosovitskiy et al. (2017) destacam que as simulações democratizam o acesso a ferramentas de pesquisa e desenvolvimento para VAs, reduzindo barreiras financeiras e logísticas para a inovação. Em soluções autônomas, especialmente em VAs, a detecção de objetos é fundamental para a interpretação em tempo real do ambiente, permitindo que os veículos reconheçam e classifiquem obstáculos, placas de trânsito, pedestres e outros objetos essenciais ao seu redor (NEUFVILLE, 2022; MOZAFFARI, 2020; KHAN et al., 2022). Algoritmos de visão computacional alcançam isso ao utilizar extensos dados de imagem para interpretar pistas visuais, identificar formas, cores, distâncias e objetos, aspectos essenciais para a segurança e a eficiência da condução autônoma. Esses algoritmos são alimentados usando dados oriundos de sensores, por exemplo: câmeras que capturam dados em tempo real, processados pelos VAs para tomar decisões informadas e assistir o condutor durante uma tarefa de condução (SINGH, 2022, p. 15), (KHAN et al., 2022, p. 6). Para que a tarefa de percepção dos VAs se torne eficaz em ambientes variados, os pesquisadores estão investigando abordagens sofisticadas para simular com precisão a detecção de objetos em tempo real (DOSOVITSKIY et al., 2017, p. 1). As soluções integram, geralmente, conjuntos de ferramentas de aprendizado profundo com dados de sensores em tempo real para melhorar a precisão na identificação de objetos. Esses algoritmos são treinados com extensos dados de sensores coletados de simulações virtuais, onde são expostos a milhões de quilômetros de cenários de estrada virtuais, garantindo que encontrem várias condições ambientais, obstáculos, casos extremos e situações de alto risco que, de outra forma, seriam raros ou inseguros de replicar em testes no mundo real (WACHENFELD; WINNER, 2016). Esse grande volume de dados aumenta a robustez dos modelos de detecção de objetos, garantindo que soluções tomem ações mais precisas, de modo a manter conformidade com a regulação de trânsito, alcançando uma condução segura.

Nos últimos anos, algoritmos baseados em Redes Neurais Convolucionais ou *Convolutional Neural Networks* (CNNs) em inglês e técnicas de aprendizado profundo têm sido amplamente utilizados para tarefas de detecção de objetos em VAs, aproveitando as melhorias no poder computacional (LI; WANG; WANG, 2023a, p. 4). Por exemplo, redes treinadas em conjuntos de dados de placas de trânsito podem ser usadas para detectar e classificar placas de trânsito relacionados à velocidade (KHAN et al., 2022, p. 6), e passar essas informações para a camada de Planejamento ou Sistemas Avançados de Assistência ao Condutor em inglês: *Advanced Driver Assistance Systems* (ADAS). Os quais utilizam essas informações para reagir a obstáculos modificando sua rota e movimento e/ou assistir o condutor com informações de trânsito baseadas nos resultados dessas detecções e classificações (AHIRE et al., 2024, p. 46), (OKPONO et al., 2024), (KHAN et al., 2022, p. 4).

Esses recursos de assistência são amplamente implementados em veículos com níveis

de automação inferiores ao nível 4, conforme a classificação SAE (SAE, 2021). Por esses níveis demandarem atenção constante ou parcial do condutor durante a tarefa de condução, esses sistemas de assistência durante a condução, desempenham um papel crucial no suporte aos motoristas, aprimorando a segurança de todos os usuários (OKPONO et al., 2024).

1.1 Problema

A falha na detecção e interpretação precisa e oportuna de sinais de trânsito, muitas vezes decorrente de limitações humanas, permanece um desafio significativo, contribuindo para acidentes e comprometendo a segurança.

Estudos apontam que erros humanos são responsáveis por mais de 90% dos acidentes de trânsito, evidenciando a necessidade de soluções tecnológicas capazes de mitigar esses riscos (National Highway Traffic Safety Administration, 2018). Embora os sistemas de assistência avançada ao condutor (*ADAS*) representem um avanço, sua eficácia depende da precisão dos algoritmos de detecção de objetos, que nem sempre apresentam resultados satisfatórios (AHIRE et al., 2024), (KHAN et al., 2022).

Além disso, a introdução de tecnologias autônomas, especialmente nos níveis mais altos de automação segundo a classificação da SAE (2021), enfrenta barreiras técnicas e práticas significativas, incluindo desafios relacionados à capacidade de processamento em tempo real, confiabilidade em condições ambientais variadas e integração com outros sistemas do veículo (DOSOVITSKIY et al., 2017), (WACHENFELD; WINNER, 2016). Esses desafios são agravados pelos altos custos associados ao treinamento e teste de algoritmos em ambientes reais, bem como pelas limitações legais e logísticas (DOSOVITSKIY et al., 2017), (AHIRE et al., 2024).

Portanto, é essencial desenvolver e validar soluções que integrem detecção de objetos com precisão elevada e processamento eficiente em tempo real, utilizando plataformas simuladas para contornar as limitações de custo e risco. Tais soluções têm o potencial de aumentar a segurança no trânsito, oferecendo suporte mais robusto a motoristas e possibilitando avanços na implementação de VAs (JANAI et al., 2020), (KHAN et al., 2022).

1.2 Hipótese

Um sistema de assistência à condução em carros autônomos pode oferecer feedback visual de placas de trânsito.

Essa hipótese é fundamentada na viabilidade técnica e nos resultados de trabalhos anteriores, que demonstram a eficácia de soluções baseadas em algoritmos avançados de

detecção de objetos aplicados ao reconhecimento de sinais de trânsito. Nesse quesito, [Dosovitskiy et al. \(2017\)](#), demonstraram a eficácia do uso de simuladores como o CARLA para validar sistemas de condução autônoma, reduzindo custos e riscos associados a testes em ambientes reais. Por meio de simulações, [Juanola \(2019\)](#), alcançaram uma precisão de 92% na detecção de placas de limite de velocidade utilizando o algoritmo YOLO, destacando sua capacidade de processar imagens em tempo real e fornecer informações críticas ao motorista. Esse sistema foi testado com tempos de execução rápidos (8 ms em GPU), mostrando ser adequado para aplicações que demandam baixa latência.

Trabalhos como [Li, Wang e Wang \(2023b\)](#) e [Wang e Yu \(2020\)](#) propuseram melhorias em algoritmos de detecção, como YOLOv7 e YOLOv4, aumentando a precisão na identificação de alvos pequenos, como placas de trânsito. Tais aprimoramentos incluem o uso de métricas específicas, como a distância de Wasserstein e redes de atenção (ACmix), que permitem melhor captação de características visuais em ambientes complexos. Essas abordagens comprovam ser possível aprimorar ainda mais a detecção de placas, mesmo em condições adversas ou em escalas reduzidas.

Além disso, [Andrade \(2022\)](#), demonstraram a integração entre detecção de objetos e cálculos de distância, validando soluções robustas em simulações no CARLA. Essas implementações reforçam a possibilidade de desenvolver sistemas que, além de reconhecerem placas de trânsito, forneçam informações detalhadas, como distâncias relativas, para apoio em tarefas de condução.

Portanto, baseando-se nos trabalhos supracitados, é factível concluir que sistemas de assistência à condução, integrados com algoritmos avançados e testados em simulações virtuais, podem oferecer uma solução prática e eficiente para mitigar falhas humanas na interpretação de sinais de trânsito. Esses sistemas podem aprimorar a segurança e confiabilidade da condução, especialmente em VAs e semi-autônomos, conforme apontam estudos anteriores ([AHIRE et al., 2024](#); [KHAN et al., 2022](#)).

1.3 Objetivos

Neste terceiro ano de pesquisa (09/2024 - 08/2025), nosso objetivo principal foi compreender e aplicar os seguintes conceitos: estimativa e localização de estado, percepção visual e planejamento de movimentos de carros autônomos.

De modo a aplicar esses conceitos: desenvolvemos um sistema de detecção e reconhecimento de placas de trânsito para usuários em um carro autônomo.

1.3.1 Objetivos Específicos

1. Desenvolver uma solução de carro autônomo capaz de percorrer um trajeto de uma área urbana. Essa abordagem é fundamental porque ambientes urbanos incluem elementos como placas de trânsito e obstáculos que permitem validar o desempenho do sistema de detecção;
2. A solução de carro autônomo deverá lidar com objetos pelo trajeto, e reagir a sinalizações de trânsito. Visto que a solução precisa reconhecer corretamente sinais relevantes e tomar decisões apropriadas, como reduzir a velocidade ou parar, assegurando a segurança durante a condução;

1.4 Justificativa

A elevada taxa de acidentes de trânsito causadas por falhas humanas, muitas vezes ocasionados por falhas na percepção e interpretação de sinalizações de trânsito ([National Highway Traffic Safety Administration, 2018](#)), evidencia a necessidade de soluções tecnológicas que auxiliem os motoristas a identificar e reagir rapidamente às condições do ambiente. Sinalizações visuais como placas de trânsito são essenciais para a segurança, mas podem ser mal interpretadas ou ignoradas devido a fatores como visibilidade reduzida, distração ou cansaço do condutor.

Diante da necessidade de compreender e aplicar os conceitos de estimativa e localização de estado, percepção visual e planejamento de movimento para VAs, a implementação de uma solução de assistência baseado em visão computacional oferece um suporte adicional ao motorista, aumentando a segurança, especialmente em situações de alto risco ou baixa visibilidade ([OKPONO et al., 2024](#)). Possibilitando alcançar os objetivos [1.3](#) deste trabalho.

1.5 Método

De modo a alcançar os objetivos propostos na seção [1.3](#), este trabalho adota uma abordagem metodológica baseada no método científico visualizado na Figura [1](#), estruturada em etapas sequenciais e sistemáticas.

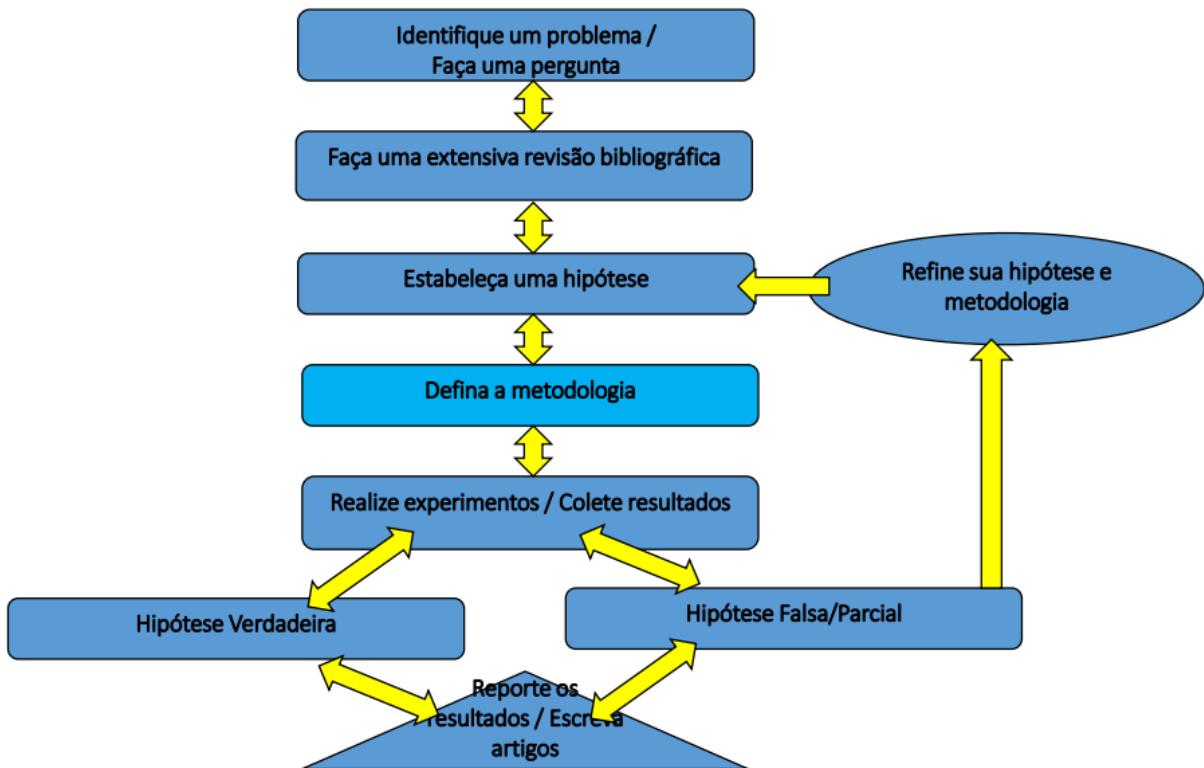


Figura 1 – Método Científico (CRISTINA; WAZLAWICK, 2021, p. 4).

Essas etapas visam validar a hipótese de que um sistema de assistência à condução pode oferecer *feedback* visual de placas de trânsito. A seguir, são descritas as principais etapas do método adotado:

1. Definição do Problema e Planejamento:

- A metodologia inicia com a análise aprofundada do problema identificado: a falha na detecção e interpretação de sinalizações de trânsito, suas implicações na segurança veicular, e a necessidade de sistemas que superem as limitações humanas.
- A partir disso, foram delimitados os objetivos gerais 1.3 e específicos 1.3.1, que orientaram a elaboração de um plano experimental para desenvolver e validar o sistema de detecção de objetos e interação motorista-veículo em simulações.

2. Modelo YOLO e Seleção de Ferramentas:

- Dados para treinamento e validação do modelo de detecção de objetos foram coletados a partir de trabalhos presentes na revisão bibliográfica disponibilizados de maneira pública e dentro da licença MIT. Nesse trabalho usaremos o modelo padrão treinado do YOLO. Para trabalhos futuros, sugere-se conjuntos de dados que foram escolhidos devido à sua diversidade em tipos de placas, condições ambientais e ângulos de captura. Visto que o conjunto de dados contém imagens

do simulador CARLA (2.950 rotuladas e 29.102 não rotuladas) misturando diferentes sessões em diferentes cenários (Cidade01/Cidade02) e condições climáticas (Ensolarado/Chuvoso/Nublado) ([JUANOLA, 2019](#)) e outro conjunto de dados rotulado para detecção de objetos no simulador Carla contendo 1028 imagens ([Daniel, 2023](#)).

- As ferramentas e tecnologias escolhidas para o desenvolvimento incluem:
 - **Simulador CARLA:** um ambiente virtual para simulações de condução autônoma, permitindo a realização de experimentos controlados em cenários urbanos variados ([DOSOVITSKIY et al., 2017](#)).
 - **Algoritmo YOLO (You Only Look Once):** escolhido por sua eficácia na detecção de objetos em tempo real, especialmente em aplicações automotivas ([BOCHKOVSKIY; WANG; LIAO, 2020; WU; CAO, 2022](#)).
 - **Linguagem de programação Python:** escolhido pelo alto número de soluções computacionais encontrados durante a revisão bibliográfica nesta linguagem, e pela maior familiaridade do autor com essa linguagem.
 - **Ambiente de programação:** Visual Studio Code, familiaridade do autor.
 - **Sistema operacional:** Linux e Windows. A edição dos mapas do simulador CARLA, para incluir placas de trânsitos em trajetos, é facilitada a partir de sistema Linux, de mesmo modo o treinamento do Algoritmo YOLO. Devido a isso, essas etapas serão feitas em Linux. O desenvolvimento da solução foi iniciando em Windows e continuará.
 - **Hardware de Processamento:** a execução será realizada preferencialmente em GPU para assegurar baixa latência e maior capacidade computacional, essencial para a avaliação do sistema em condições simuladas.

3. Desenvolvimento da solução:

- Modelo de Detecção:
 - Usaremos o algoritmo YOLO treinado, seguindo a documentação do YOLO Irie ([\(2020\)](#), ([REDMON, 2013–2016](#))).
- Integração no Ambiente Simulado:
 - O modelo treinado será incorporado a solução de carro autônomo, onde será responsável por identificar placas de trânsito em cenários urbanos predefinidos no CARLA. Os cenários predefinidos serão àqueles que contém placas de trânsito que possibilitem a validação da nossa solução.
 - O sistema de interação veículo-motorista será implementado para dar *feedback* visual, informando sobre sinalizações detectadas e suas interpretações.
- Carro autônomo:

- Percepção usando algoritmo YOLO.
- Planejamento de movimento funcional que possa evitar obstáculos estáticos e dinâmicos enquanto rastreia a linha central de uma faixa, ao mesmo tempo, em que lida com placas de trânsito (ex.: placa de parada).
 - * Implementar lógica de planejamento comportamental, bem como verificação de colisão estática, seleção de caminho e geração de perfil de velocidade.
- Controlar o veículo de modo a percorrer o trajeto predeterminado em uma certa velocidade, portanto, será necessário controle longitudinal e lateral.
 - * Implementar Controlador Longitudinal PID (Proporcional-Integral-Derivativo).
 - * Implementar Controle Lateral Geométrico - Controlador de Perseguição Pura.

4. Experimentação e Validação:

- Cenários de Teste:
 - A solução será experimentada em um trajeto predefinido que contenha as placas de trânsitos que possibilitem a validação da solução.
 - As variáveis independentes incluem o tipo de placa.
- Coleta de Dados Experimentais:
 - As variáveis dependentes, como detecção de placas, serão medidas.

5. Análise Crítica dos Resultados:

- Os dados coletados serão analisados de modo a avaliar se as detenções condizem com as suas classificações e 90% de confiança.
- A solução de carro autônomo terá seu desempenho avaliado pela conclusão bem sucedida do trajeto predefinido. O sucesso é dado pela não colisão com obstáculos, identificação de placas de trânsitos e conclusão de trajeto.

6. Conclusão e Propostas Futuras:

- Os resultados experimentais serão consolidados para verificar se os objetivos propostos na seção 1.3 foram atingidos, validando ou refutando a hipótese inicial apresentada na seção 1.2.
- Com base nas conclusões, serão sugeridas melhorias e direções para trabalhos futuros, como a implementação em ambientes reais, integração com outros sensores automotivos, uso de outros algoritmos, entre outros, conforme visto em 8.

A partir das etapas apresentadas podemos identificar as Variáveis e Hipóteses Experimentais. Onde as Variáveis Dependentes:

- Taxa de acurácia na detecção de placas de trânsito.
- Tempo de processamento por imagem (latência).

e Variáveis Independentes:

- Condições climáticas (ex.: ensolarado, chuva, neblina).
- Tipo de sinalização (ex.: limites de velocidade, placa de pare, etc.).

As quais quando combinadas podem alterar os resultados dos experimentos. Desse modo, a Hipótese a ser testada se expande para incluir as etapas experimentais: **algoritmos de visão computacional baseados em YOLO, integrados a uma solução de carro autônomo, podem detectar e classificar placas de trânsito com precisão acima de 90% no simulador CARLA e oferecer feedback (assistência) visual ao condutor.**

Essa metodologia oferece uma estrutura robusta para alcançar os objetivos 1.3 do trabalho, avaliando capacidades e limitações das soluções propostas e contribuindo para avanços no desenvolvimento de sistemas autônomos.

Desse modo, durante o desenvolvimento desta Iniciação Científica. A pesquisa exploratória foi realizada da seguinte forma:

Inicialmente, fizemos pesquisas para o levantamento bibliográfico de modo a agrupar e selecionar os materiais relevantes para este projeto, que foi desenvolvida através dos seguintes meios na internet: Google acadêmico, Google livros, biblioteca virtual, jornais virtuais, site das bibliotecas de universidades, plataforma CAPES, plataforma de cursos, YouTube e outros. A busca nesses bancos de dados contemplaram múltiplos anos. A ideia inicial era contemplar apenas os anos de 2023 a 2024. Entretanto, nos deparamos com um número pequeno de conteúdos de relevância para essa pesquisa, de modo a contornar essa situação expandimos nossas pesquisas para referências de mais anos. Como palavra-chave para as pesquisas iniciais, empregamos os termos: *Autonomous Vehicles Software, Autonomous, Cars, Mobility, Connected Car, AV, TaxiBot, Self-driving cars, Algorithmus, Deep Learning, Computer Vision, etc.* Com o avanço das pesquisas, refinamos os termos de modo a explorar termos como: *localização de estado, percepção visual, e planejamento de movimentos.*

Dessa forma, chegamos nos conteúdos presentes nesse trabalho, que foi elaborado, principalmente, por artigos científicos, e uma Especialização em carros autônomos da

Universidade de Toronto publicado em 2018 na plataforma de cursos online Coursera, visto na Figura 23. A Especialização tem duração de aproximadamente de 3 meses e foi ministrado pelos instrutores Steven Waslander e Jonathan Kelly no idioma inglês, ambos da Universidade de Toronto, e com ampla experiência no setor de autonomia. A minha participação neste curso contribuiu de maneira singular para o desenvolvimento desta pesquisa, a partir das informações e conhecimentos compartilhados conseguiu-se compreender de maneira completa como as soluções de VAs são desenvolvidos desde sua concepção até a aplicação. Para isso, tomamos nota de todas as aulas, lemos todos os artigos e materiais adicionais as aulas ministradas, e concluímos todas as atividades com 100% de aproveitamento. Esse ato aumentou o tempo de duração da Especialização em 4 vezes. O certificado de conclusão pode ser encontrado no capítulo 9.

Sabendo disso, o desenvolvimento da solução foi realizado da seguinte maneira:

Implementamos um controlador (desenvolvido na Iniciação do ano passado Gomes (2024)) para percorrer um trecho de pista da *Town 1* do simulador de sistemas de direção Carla, usando os conhecimentos do controlador PID para o controle longitudinal (desenvolvido na Iniciação do ano passado Gomes (2024)) e do controlador de Perseguição Pura para o controle lateral (desenvolvido na Iniciação do ano passado Gomes (2024)). Ademais, para englobar a percepção visual promovemos uma solução de visão computacional para identificar placas de trânsito no trajeto percorrido. Em seguida, o planejamento de movimento gera perfil de velocidade, e calcula trajetórias eficientes para o veículo. Por fim, para localização de estado, usaremos um conjunto de pontos de referência, apresentados na Iniciação do ano passado Gomes (2024).

De modo a concluir a tarefa de direção, o veículo percorre o trajeto definido por meio de um conjunto de pontos de referência, de modo a concluir o trajetória estipulada.

A solução em questão, foram desenvolvidos usando linguagem de programação Python na sua versão 3.6.0, versão compatível com o simulador Carla modificado disponibilizado pelo curso realizado. Para o desenvolvimento dos códigos necessários para as soluções, usamos a abordagem metodológica Ágil, onde usamos o *framework* Scrum para otimizar o desenvolvimento da solução por meio de processos iterativos e incrementais. Escolhemos a metodologia Ágil para essa etapa devido a sua adaptabilidade, agilidade e eficiência. Nessa abordagem, temos a possibilidade de sempre estar incrementando o trabalho já realizado. De modo a alcançar isso, seguimos as definições do autor Anwer et al. (2017) e as aplicamos a realidade do nosso projeto. Ademais, recorremos à plataforma de versionamento de códigos, GitHub, a qual possibilitou a integração contínua dos códigos desenvolvidos a partir da metodologia Ágil (GitHub, Inc, 2024). Todos os códigos e matérias adicionais, assim como a versão modificada do simulador Carla, desenvolvidos estão disponíveis no capítulo A, onde pode ser encontrado o link de redirecionamento para o nosso repositório no GitHub.

Por fim, elaboramos o relatório final, por meio do sistema de preparação de documentos LaTeX, usando os materiais encontrados e desenvolvidos, e das notas tomadas ao longo de todo processo.

1.6 Organização do Trabalho

O presente trabalho foi estruturado de forma a apresentar de maneira lógica e progressiva as etapas desenvolvidas no estudo e os resultados obtidos. No **Capítulo 1**, são apresentados o tema, a delimitação do problema, a hipótese de pesquisa, os objetivos (geral e específicos), a justificativa e a metodologia adotada para condução do projeto.

O **Capítulo 2** apresenta uma visão geral das etapas inicialmente propostas no plano de trabalho, servindo como um guia para o acompanhamento da execução do projeto ao longo do tempo.

O **Capítulo 3** concentra-se na discussão sobre os fundamentos teóricos e práticos relacionados à *Detecção e Classificação de Objetos*, especialmente no contexto da aplicação de redes neurais convolucionais para reconhecimento visual em ambientes simulados.

No **Capítulo 4**, são abordadas as principais estratégias de *modelagem conceitual* de sistemas de navegação autônoma. Esse capítulo discute as noções de planejamento de movimento (4.1) e percepção visual (4.2), com foco em câmeras (4.2.1) e no modelo *You Only Look Once - YOLO* (4.2.2, 4.2.2.1, 4.2.2.2).

O **Capítulo 5** descreve, com rigor técnico, a implementação prática dos conceitos estudados, incluindo a configuração do simulador CARLA (5.1), a implementação dos controladores de movimento (5.2), e a integração do sistema de detecção YOLO à simulação (5.3).

No **Capítulo 6**, são apresentados os *resultados e discussões* decorrentes da aplicação da solução desenvolvida, com ênfase na análise da performance, limitações e contribuições da proposta.

As **considerações finais**, no **Capítulo 7**, sintetizam os principais aprendizados, dificuldades e perspectivas resultantes do estudo, ao passo que o **Capítulo 8** propõe possibilidades de continuidade e aprofundamento da pesquisa.

O trabalho inclui ainda o **Capítulo 9**, que apresenta as participações em congressos, trabalhos publicados ou submetidos e outras atividades acadêmicas de relevância para a pesquisa desenvolvidas durante o período de vigência.

Por fim, os **Apêndices (A)** e os **Anexos (A)** disponibilizam materiais complementares, incluindo o repositório completo de código-fonte, configurações utilizadas e recursos visuais empregados na simulação. Tais materiais têm por objetivo ampliar a reproduzibilidade da pesquisa e fomentar o aprofundamento técnico por parte dos leitores.

Adicionalmente, uma lista das principais siglas utilizadas neste trabalho encontra-se na Página [8](#).

2 Etapas propostas no Plano de Trabalho

Neste capítulo, apresentamos as etapas (também vistas na ilustração da Figura 2) necessárias para atingir os objetivos delineados no capítulo 1.3, enumeradas da seguinte forma:

1. **Exploração** sobre os conceitos de estimativa e localização de estado, percepção visual e planejamento de movimento para veículos autônomos;
2. **Levantamento** bibliográfico de materiais relacionados a estimativa e localização de estado, percepção visual e planejamento de movimento para veículos autônomos;
3. **Seleção** de conteúdos de relevância encontrados durante o levantamento bibliográfico;
4. Aplicação **prática** em ambiente de simulação de alguns dos conteúdos de relevância selecionados;
5. Elaboração do **Relatório** Final.

As etapas enumeradas acima podem ser compreendidas a partir da ilustração na Figura 2, onde cada etapa nomeada na ilustração refere-se a uma etapa listada anteriormente:

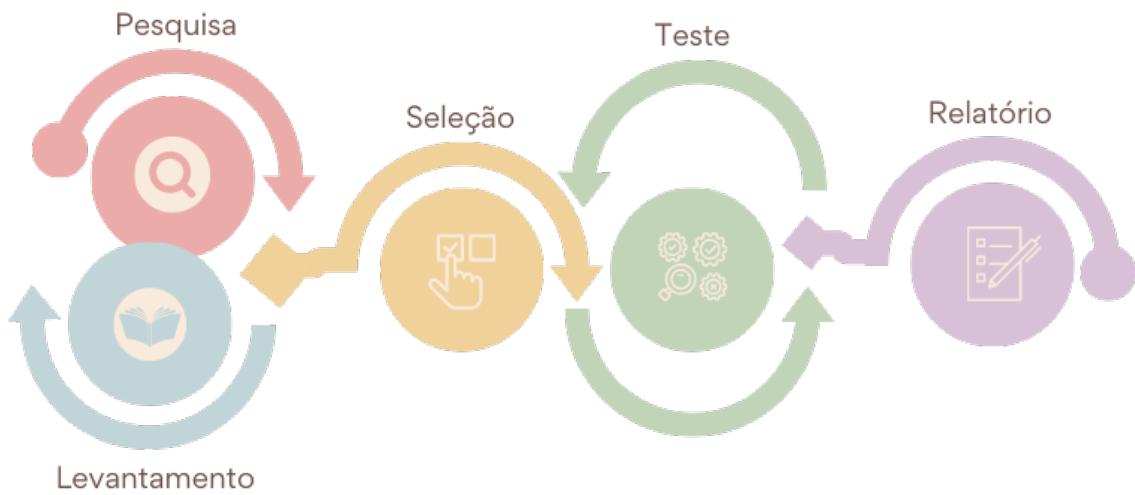


Figura 2 – Etapas do projeto para o 3º ano de pesquisa.

3 Detecção e Classificação de Objetos

A evolução na área de visão computacional trouxe significativos avanços na detecção de objetos, com destaque para a arquitetura You Only Look Once (YOLO), proposta por Redmon et al. (2016). Essa abordagem se diferencia por tratar o processo de detecção e classificação de objetos como um problema único de regressão, unificando a identificação de áreas de interesse e a atribuição de classes. Essa estratégia reduz a complexidade computacional e permite o reconhecimento em tempo real em streaming de vídeo com menos de 25 milissegundos de latência no YOLO (REDMON et al., 2016, p. 1), característica crucial para aplicações como VAs. O YOLO oferece uma solução integrada para detecção, refinamento e classificação que observa a imagem apenas uma vez (You Only Look Once), reduzindo assim a complexidade e os custos computacionais envolvidos no processo (REDMON et al., 2016).



Figura 3 – O Sistema de Detecção YOLO. O processamento de imagens com YOLO é simples e direto. O sistema (1) redimensiona a imagem de entrada para 448×448 , (2) executa uma única rede convolucional na imagem e (3) limita as detecções resultantes pela confiança do modelo. (REDMON et al., 2016, p. 1).

O Funcionamento da Arquitetura YOLO inicia seu processamento redimensionando as imagens para o tamanho esperado pela rede neural convolucional como visto na Figura 3. Em seguida, a imagem é submetida a uma rede que gera predições para as áreas de interesse, as classes correspondentes e as probabilidades associadas. Por fim, os valores redundantes são eliminados por meio de um filtro que aplica a técnica de Non-Maximum Suppression (NMS), mantendo apenas as predições com maior confiança (REDMON et al., 2016).

Ao combinar a identificação de áreas de interesse com a classificação em um único processo, o YOLO se torna um dos algoritmos mais rápidos de sua classe, embora, em algumas circunstâncias, sacrifique um pouco de precisão em relação a métodos mais lentos. A arquitetura do YOLO também se diferencia por considerar os dados globais da imagem, facilitando a distinção entre o plano de fundo e os objetos de interesse, cometendo menos da metade do número de erros comparação ao Fast Region-based Convolutional Networks (Fast R-CNN) (REDMON et al., 2016, p. 2).

Desde sua primeira versão, o YOLO passou por melhorias significativas. O tornando uma referência em detecção de objetos. Entre essas melhorias, destaca-se o suporte para maior número de classes, detecção aprimorada de objetos pequenos e avanços em precisão e eficiência, como observado nas versões YOLOv2, YOLOv3 e YOLOv4 (REDMON; FARHADI, 2016; REDMON; FARHADI, 2018; BOCHKOVSKIY; WANG; LIAO, 2020).

O progresso da série YOLO não se limitou a apenas detecção de objetos. Ao longo do tempo, a metodologia também passou a influenciar áreas correlatas, como segmentação de instâncias, rastreamento de múltiplos objetos, análise comportamental, reconhecimento facial, entre outras. Isso reflete sua importância como um pilar tecnológico para aplicações em direção autônoma, robótica industrial, autenticação de identidade, saúde inteligente e vigilância visual (WANG; LIAO, 2024).

A capacidade do YOLO de se adaptar a diferentes contextos e sua contínua evolução são características fundamentais que sustentam sua popularidade. Até mesmo as versões mais recentes, como o YOLOv7 e o YOLOv8, continuam a superar desafios em cenários complexos, como ambientes com objetos pequenos ou parcialmente ocluídos. Além disso, a versão YOLOv10 representa uma convergência entre precisão e eficiência computacional, mostrando que a pesquisa em torno deste framework está longe de atingir seu limite (WANG; LIAO, 2024). Para questão de comparativo, a Figura 4 visa ilustrar esse ganho de uma versão de YOLO para outra.

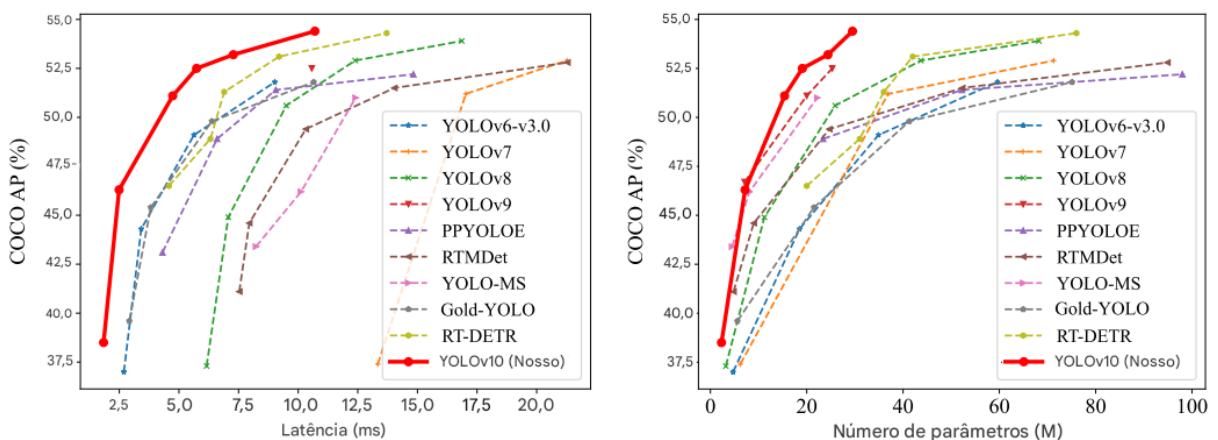


Figura 4 – Comparações com outros em termos de trade-offs de latência-precisão (esquerda) e tamanho-precisão (direita). Medimos a latência de ponta a ponta usando os modelos oficiais pré-treinados. (WANG et al., 2024, p. 1).

4 Modelagem Conceitual para simulação de Carros Autônomos

Neste capítulo, formularemos a modelagem conceitual do artefato que será implementado no Capítulo 5. A fundamentação teórica apresentada na pesquisa de Gomes (2024) oferece a base necessária para a concepção desta arquitetura. Dessa forma, tomamos como referência a Arquitetura da Tarefa de Condução apresentada na IC Gomes (2024) para elaborar uma solução que atende aos objetivos delineados nos Objetivos 1.3, validando, consequentemente, a Hipótese 1.2 proposta neste trabalho.

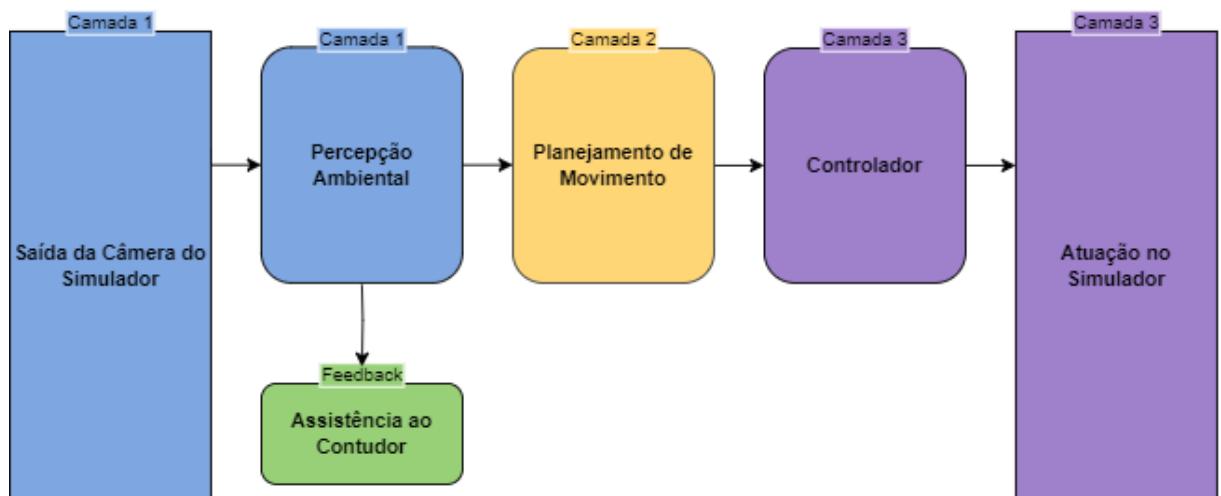


Figura 5 – Arquitetura do Software. Elaborado pelo Autor.

A arquitetura desenvolvida, ilustrada na Figura 5, é composta por múltiplas camadas que refletem os diferentes módulos necessários para a operação de um carro autônomo em um ambiente simulado. O sensor escolhido para a primeira camada é a câmera, devido à sua versatilidade e à riqueza de dados visuais, essenciais para tarefas de percepção ambiental, como elaboraremos na seção 4.2. Essa camada 1 será a responsável por detectar e classificar os objetos e o seu resultado permitirá o feedback ao usuário. Baseado nesses resultados da Camada 1, a Camada 2 (Seção 4.1) ditará os movimentos necessários para a Camada 3 exercer no simulador Gomes (2024). Como descritas abaixo:

- **Camada 1 - Percepção Ambiental:** A primeira camada processa os dados provenientes da câmera do simulador. O módulo realiza a análise da cena para identificar objetos: placas de velocidade e parada. Este módulo utiliza técnicas de visão computacional, a detecção de objetos baseada no algoritmo YOLO, descrito na seção 4.2.2. Além disso, a partir dos resultados dessa camada podemos assistir o condutor via feedbacks em tempo real, conforme visto na Figura 9, ampliando

a segurança e a confiabilidade na direção. O modelo geral da Camada 1 para a percepção do ambiente pode ser visto na Figura 6.

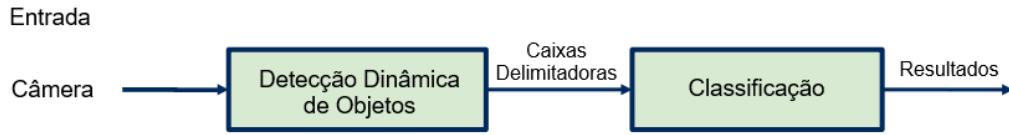


Figura 6 – Arquitetura de Software da Percepção do Ambiente. Baseado em [University of Toronto \(2018a\)](#).

- **Camada 2 - Planejamento de Movimento:** Baseado nas informações fornecidas pela camada de percepção, o módulo de planejamento de movimento gera perfil de velocidade, e calcula trajetórias eficientes para o veículo. Este módulo considera restrições dinâmicas, regras de trânsito e a previsão de movimentos de outros agentes na cena, descrito na seção 4.1. O modelo geral da camada 2 pode ser visto na Figura 7.

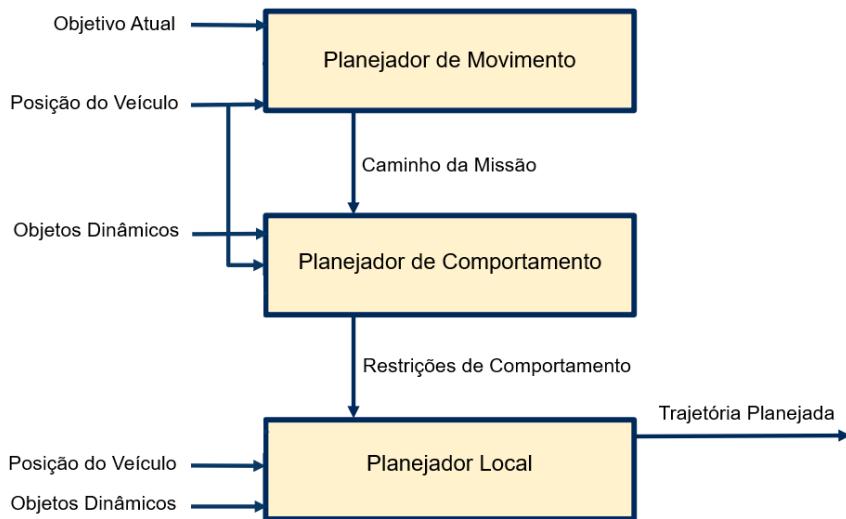


Figura 7 – Arquitetura de Software do Planejador de Movimento. Baseado em [University of Toronto \(2018a\)](#).

- **Camada 3 - Controlador e Atuação:** A terceira camada é responsável por traduzir as trajetórias planejadas em comandos de controle de aceleração e direção, para a nossa solução, que serão aplicados ao simulador. Controladores são utilizados para garantir que o veículo siga a trajetória planejada com precisão, ajustando

continuamente os comandos, como elaborado no trabalho Gomes (2024). O modelo geral da camada 3 pode ser visto na Figura 8.

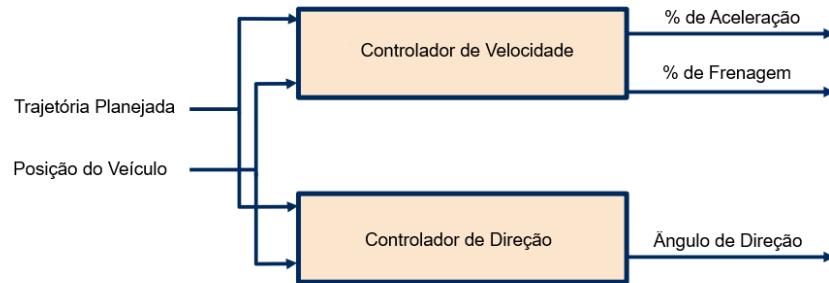


Figura 8 – Arquitetura de Software do Controlador do Veículo. Baseado em University of Toronto (2018a).

Desse modo, o fluxo de dados é iniciado pela captura de informações visuais na saída da câmera do simulador. Em seguida, esses dados percorrem as camadas descritas acima, resultando na atuação final no ambiente simulado. O fluxo também inclui o feedback para assistência ao condutor a partir da camada de percepção, assegurando a atualização contínua das informações percebidas, conforme podemos identificar no fluxograma da Figura 9.

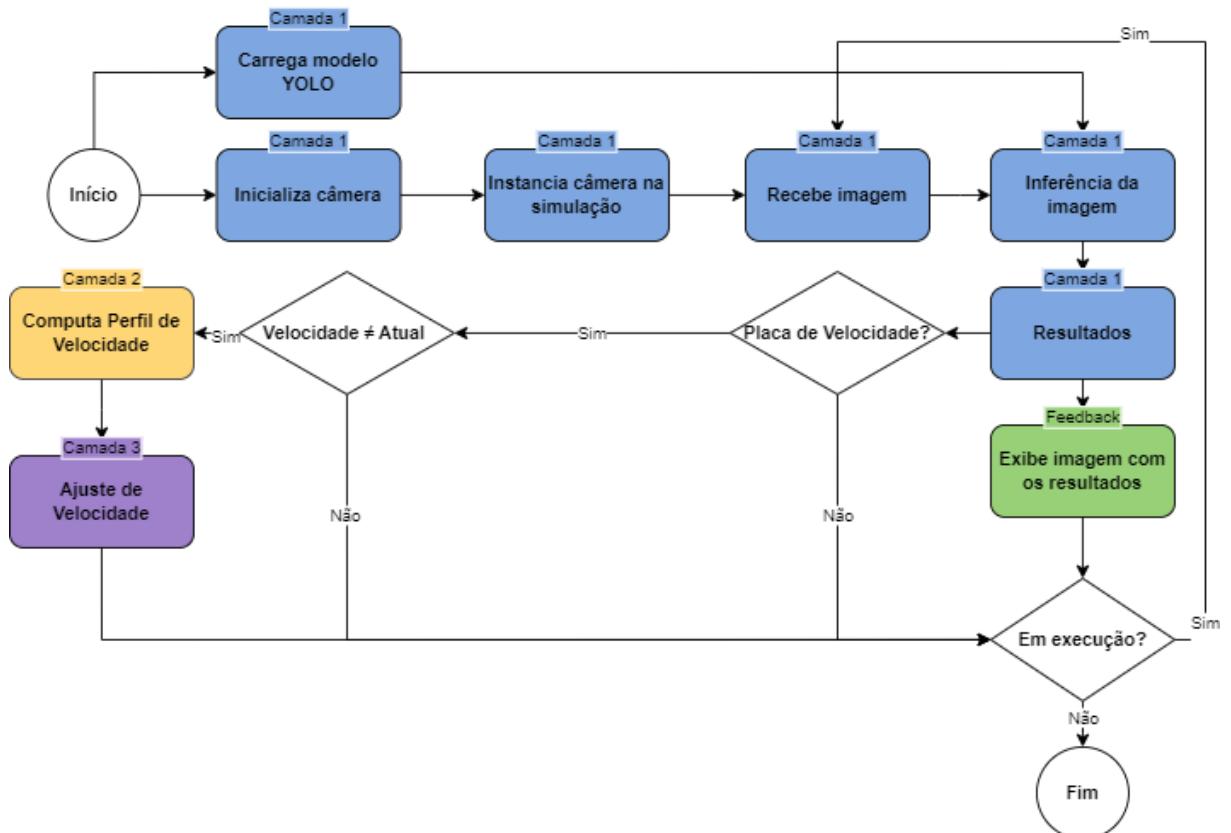


Figura 9 – Fluxo da Execução da Solução. Elaborado pelo Autor.

O fluxograma da Figura 9 apresenta de maneira detalhada o processo completo de percepção, feedback, planejamento e controle, em etapas distribuídas entre as camadas da arquitetura. Como podemos identificar na Figura 9, o processo é iniciado com o carregamento do modelo de detecção YOLO, responsável pela detecção e classificação de objetos na cena. Em seguida, a câmera é inicializada e instanciada na simulação, garantindo a captura contínua de imagens em tempo real. As imagens capturadas são processadas pelo modelo, que realiza inferências para identificar os objetos de interesse. Após a inferência, os resultados são processados e exibidos visualmente, permitindo a validação e interpretação das informações detectadas. Se uma placa de limite de velocidade for identificada, o sistema retorna os dados para o próximo módulo, possibilitando ajustes dinâmicos. Esse ajuste é permitido pela Camada 2, o perfil de velocidade é computado com base nos dados de detecção e nas condições simuladas. Caso a velocidade atual do veículo não corresponda ao limite indicado, uma resposta é enviada para a Camada 2 para a mudança do perfil de velocidade e então o fluxo avança para a Camada 3, onde é realizado o ajuste da velocidade, enviando comandos ao simulador para adequação em tempo real.

Esse fluxo segue iterativamente enquanto a solução permanece em execução. Caso o processo seja encerrado, o sistema interrompe a captura de dados e finaliza as operações.

Desse modo, as seções subsequentes 4.1 e 4.2 desenvolverão como cada camada será projetada.

4.1 Planejamento de Movimento de Carros Autônomos

O planejamento hierárquico é uma abordagem estruturada para resolver o problema de planejamento de movimento em VAs, dividindo o problema em uma sequência de problemas de otimização organizados em níveis de abstração. Essa hierarquia permite lidar com a complexidade do problema de maneira modular e eficiente, facilitando a execução em tempo real.

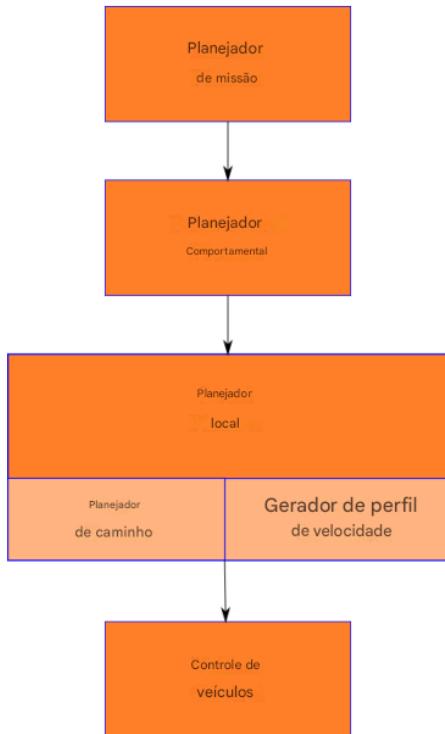


Figura 10 – Planejamento Hierárquico ([University of Toronto, 2018b](#), Module 2 - Lesson 1: Driving Missions, Scenarios, and Behaviour. 10min55s).

No topo da hierarquia da Figura 10 encontra-se o planejamento de missão, responsável por definir a rota global que o veículo deve seguir para navegar do ponto atual ao destino, considerando mapas, conexões viárias e o fluxo de tráfego. Essa etapa abstrai variáveis de baixo nível, como obstáculos e outros agentes, focando na eficiência do trajeto em termos de tempo ou distância.

O segundo nível corresponde ao planejamento de comportamento, que determina as ações do veículo em função do cenário de direção atual. Essas ações incluem manutenção de faixa, mudanças de faixa, curvas em interseções e manobras como retornos, considerando elementos regulatórios, como semáforos e placas de sinalização.

Enquanto, o planejamento local opera em menor escala, gerando trajetórias livres de colisões e perfis de velocidade detalhados para alcançar o estado final desejado. Essa etapa pode ser dividida em duas subetapas: planejamento de caminho, para calcular a trajetória espacial, e geração de perfil de velocidade, para assegurar uma movimentação suave e segura, o qual buscamos para nossa solução quando uma placa de velocidade for identificada pela camada de percepção.

Por fim, chega a camada de Controle de Veículos a qual desenvolvemos no trabalho [Gomes \(2024\)](#) e pode ser visualizado na Figura 8.

O qual recebe instruções do planejamento hierárquico considera diversos cenários de condução, que variam em complexidade e dependem de fatores como a estrutura da via

e a presença de obstáculos estáticos ou dinâmicos (University of Toronto, 2018b).

Quanto a objetos estáticos temos, por exemplo, placas de parada. Neste contexto, as Máquinas de Estados Finitos (FSMs) surgem como uma abordagem robusta para lidar com o problema do planejamento comportamental. As FSMs são baseadas em dois componentes principais: estados e transições. Os estados representam manobras específicas que o VA pode executar, como "acelerar", "frear" ou "parar em um sinal". As transições, por sua vez, determinam como o sistema evolui de um estado para outro, dependendo das entradas recebidas, como a posição de veículos próximos, a mudança de um sinal de trânsito ou a presença de pedestres (University of Toronto, 2018b).

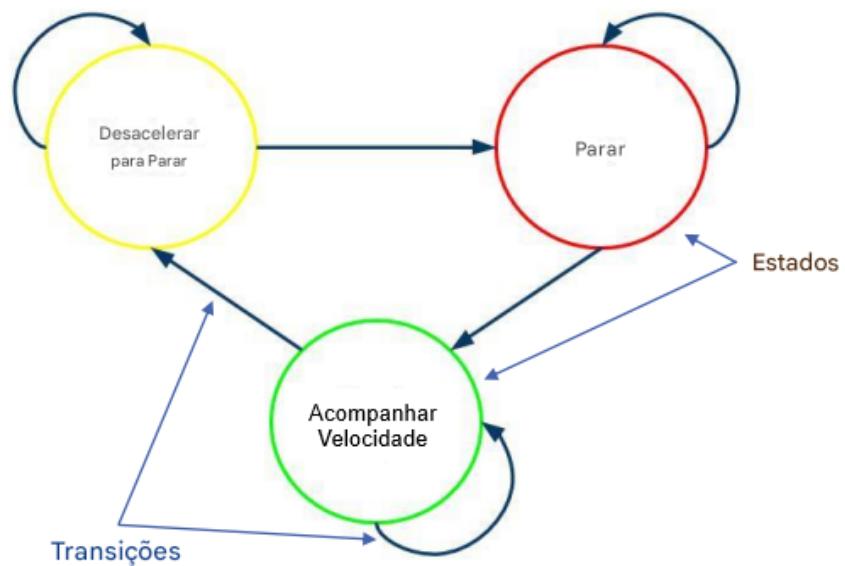


Figura 11 – Máquinas de Estados Finitos (University of Toronto, 2018b, Module 2 - Lesson 4: Hierarchical Motion Planning. 7min54s).

Conforme a Figura 11, ao encontrar um sinal de "Pare", o FSM poderia transitar entre os seguintes estados:

1. **Desacelerar para Parar:** o planejador sinaliza a necessidade de reduzir a velocidade.
2. **Parar:** após atingir a posição correta, o veículo mantém-se parado por um tempo determinado.
3. **Acompanhar Velocidade:** ao liberar o cruzamento, o FSM ordena o prosseguimento seguro.

O planejador ainda precisa considerar o planejamento local, que exige a geração de trajetórias cinemática e dinamicamente viáveis, livres de colisões que garantam o conforto

e segurança dos ocupantes. Nesse cenário, os Lattice planners, visto na Figura 12, têm se destacado como uma abordagem eficiente e robusta para o planejamento de movimento em sistemas autônomos, abordando restrições diferenciais e limitações do espaço de estados. Esses planejadores reduzem a complexidade computacional do problema ao restringir o espaço de busca a um conjunto limitado de ações que o veículo pode executar em qualquer ponto do ambiente. Esse conjunto, denominado conjunto de controle, é emparelhado com uma discretização do espaço de trabalho, formando implicitamente um grafo que pode ser explorado por algoritmos de busca como Dijkstra ou A*. Esse processo permite não apenas a busca eficiente por trajetórias, mas também a verificação de colisões, atribuindo custos infinitos às arestas que cruzam obstáculos (University of Toronto, 2018b).

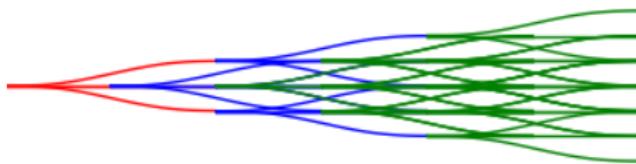


Figura 12 – Lattice Planner (University of Toronto, 2018b, Module 2 - Lesson 4: Hierarchical Motion Planning. 15min30s).

A discretização proposta em um estado lattice é uma representação regular do espaço de estados, onde cada vértice corresponde a um estado alcançável e cada aresta representa um movimento exato e factível. Essa estrutura regular permite a reutilização de um conjunto canônico de arestas, mantendo a propriedade de conectividade exata entre vértices vizinhos. O conjunto de controle, nesse contexto, é definido pelo fator de ramificação do grafo, isto é, o número de arestas conectando cada vértice. Isso resulta em um grafo estruturado que facilita a aplicação de algoritmos de busca para encontrar movimentos viáveis que satisfaçam as restrições ambientais e diferenciais, preservando a regularidade do espaço de busca (PIVTORAIKO; KNEPPER; KELLY, 2009).

Segundo o autor University of Toronto (2018b), uma variação comumente utilizada dos planejadores de lattice é o conformal lattice planner, no qual pontos-alvo são definidos em posições à frente do veículo, lateralmente deslocados em relação à direção da via, conforme podemos identificar na Figura 13.

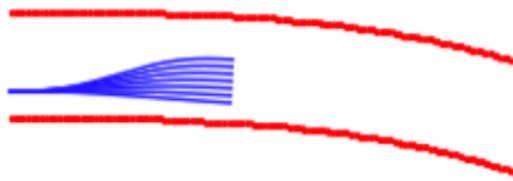


Figura 13 – Conformal Lattice Planner (University of Toronto, 2018b, Module 2 - Lesson 4: Hierarchical Motion Planning. 15min30s).

Para cada ponto, uma trajetória é otimizada, e a trajetória selecionada é aquela que melhor satisfaz os objetivos definidos, como minimização de energia de curvatura, enquanto permanece livre de colisões. Adicionalmente, a geração de perfis de velocidade é tratada como um problema de otimização restrita, integrando objetivos como minimizar variações bruscas de aceleração e respeitar limites de conforto dinâmico. Métodos para aproximações convexas são frequentemente empregados para evitar mínimos locais durante a otimização (University of Toronto, 2018b).

Os planejadores de lattice, ao integrarem restrições diferenciais diretamente na estrutura do grafo, conseguem se destacar por oferecerem soluções determinísticas e eficientes. Eles não apenas abordam desafios relacionados à viabilidade dos movimentos, mas também otimizam trajetórias nos limites impostos pelas dinâmicas do sistema. Isso os torna ferramentas essenciais para o planejamento de movimento em sistemas autônomos, equilibrando desempenho computacional e qualidade das trajetórias (PIVTORAIKO; KNEPPER; KELLY, 2009).

4.2 Percepção Visual de Carros Autônomos

Essa seção aborda os principais aspectos relacionados à percepção visual em carros autônomos, destacando o uso de câmeras e algoritmos avançados de detecção de objetos, como o YOLO (*You Only Look Once*). Inicialmente, exploraremos o papel das câmeras, descritas na subseção 4.2.1, suas características e limitações, bem como sua aplicação no contexto do simulador CARLA. Em seguida, na subseção 4.2.2, apresentaremos o algoritmo YOLO, suas vantagens em relação a outros métodos, e sua arquitetura, detalhada na subseção 4.2.2.1. Por fim, discutiremos o treinamento do YOLO, os conjuntos de dados utilizados e como esses elementos serão integrados à nossa solução para detecção e classificação de objetos em tempo real no ambiente simulado.

4.2.1 Câmeras

As Câmeras são uma das tecnologias mais utilizadas para observar o ambiente, e é o melhor sensor para classificação de objeto (KHAN et al., 2022, p. 6). Uma câmera produz imagens nítidas dos arredores detectando as luzes emitidas de uma superfície fotossensível (plano de imagem) usando uma lente de câmera (colocada na frente do sensor) (IGNATIOUS; HESHAM-EL-SAYED; KHAN, 2022). Os VAs possuem esses sensores de luz visível para fornecer uma visão de 360 graus do ambiente. As câmeras são ótimos na detecção e reconhecimento de objetos, fornecendo detalhes mais ricos e ajudando a entender os objetos sem ou com profundidade, que geralmente não são detectados por outros tipos de sensores. Dentre esses objetos estão: Sinais de trânsito (limite de velocidade, sinais de parada, sinais de ultrapassagem), semáforos, pedestres, animais são alguns exemplos de

tais objetos sem ou com profundidade (KHAN et al., 2022). Esses dados coletados com as câmeras são enviados para os algoritmos baseados em IA para uso posterior, e criação de uma imagem 2D (SINGH, 2022). No entanto, esses sensores são imprecisos em ambientes escuros e geram uma abundância de dados para processar. Outros tipos de câmeras como as infravermelhas, também, são utilizadas para melhor desempenho em condições de baixa visibilidade (PAREKH NISHI PODDAR, 2022). As câmeras são dispostas nos VAs como mostrado no trabalho Gomes (2024), e para detalhes específicos dos diferentes modelos de câmeras.

No cenário de simulação, câmeras e outros sensores podem ser adicionados ao veículo definindo as configurações enviadas pelo cliente. Há quatro sensores de câmera diferentes disponíveis no simulador CARLA: "Scene final", "Depth map", "Semantic segmentation", e "Ray-cast based lidar" (CARLA Simulator, 2024). Sendo a câmera "Scene final" a escolhida para a solução de detecção e classificação de objetos, por trazer imagens mais próximas ao mundo real. A câmera "Scene final" fornece uma visão da cena após aplicar alguns efeitos de pós-processamento para criar uma sensação mais realista como visto na Figura 14.

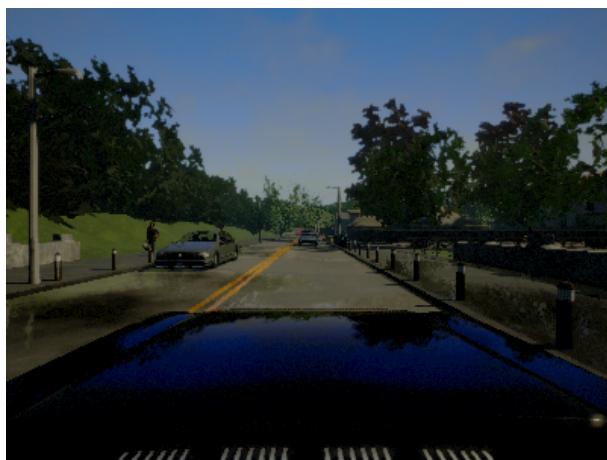


Figura 14 – Imagem a partir da Camera: Scene final (CARLA Simulator, 2024).

No trecho de código 4.1 podemos ver um exemplo dessa câmera adicionado ao cliente do simulador Carla.

```

1 # Cria uma câmera no simulador CARLA com pós-processamento Scene Final
2 camera = carla.sensor.Camera('MyCamera', PostProcessing='SceneFinal')
3 # Define o campo de visão da câmera
4 camera.set(FOV=90.0)
5 # Define o tamanho da imagem gerada pela câmera
6 camera.set_image_size(800, 600)
7 # Configura a posição da câmera em relação ao veículo
8 camera.set_position(x=0.30, y=0, z=1.30)
9 # Define a rotação da câmera
10 camera.set_rotation(pitch=0, yaw=0, roll=0)
11 # Adiciona a câmera configurada ao cenário

```

```
12 carla_settings.add_sensor(camera)
```

Lista de código 4.1 – Exemplo da documentação CARLA para adicionar uma câmera “Scene final” no simulador ([CARLA Simulator, 2024](#)).

4.2.2 You Only Look Once - YOLO

Apresentado na subseção 3 juntamente com trabalhos relacionados. Essa subseção visa apresentar como o YOLO foi projetado e como o usaremos para nossa solução de detecção e classificação de imagens em tempo real no simulador. Assim como a arquitetura unificada do YOLO e sua capacidade de operar em velocidades de tempo real, sem comprometer a precisão média, tendo como um dos principais benefícios a utilização do contexto global da imagem durante as previsões, ao contrário de métodos baseados em classificadores que analisam regiões isoladas. Além disso, o YOLO realiza todas as previsões com uma única avaliação da rede, diferentemente de sistemas como o R-CNN, que requer milhares de avaliações para processar uma única imagem ([REDMON, 2018](#)). Essa característica confere ao YOLO uma velocidade excepcional, características essenciais para lidar com os desafios em sistemas autônomos, sendo mais de 1.000 vezes mais rápido que o R-CNN e 100 vezes mais rápido que o Fast R-CNN, conforme detalhado em trabalhos anteriores sobre o sistema [Redmon e Farhadi \(2018\)](#).

4.2.2.1 Arquitetura

O YOLO adota uma abordagem unificada para a detecção de objetos, integrando componentes tradicionalmente separados em uma única rede neural. Essa rede processa a imagem inteira para prever caixas delimitadoras (*bounding boxes*) e classes simultaneamente, raciocinando globalmente sobre o conteúdo completo da imagem ([REDMON et al., 2016, p. 1](#)).

A entrada da rede é dividida em uma grade $S \times S$, onde cada célula da grade é responsável por detectar objetos cujo centro está contido na célula. Para cada célula, a rede prevê B caixas delimitadoras e uma pontuação de confiança associada, refletindo a probabilidade de presença de um objeto e a precisão da previsão, com base no índice de interseção sobre união (*Intersection over Union - IOU*¹) entre a previsão e o valor real ([REDMON et al., 2016, p. 1](#)).

Cada caixa delimitadora contém cinco valores: x , y , w , h e a confiança. As probabilidades condicionais de classe são multiplicadas pelas pontuações de confiança das caixas. Esse processo pode ser melhor entendido a partir da Figura 15 ([REDMON et al., 2016, p. 1](#)).

¹ IOU é uma métrica de avaliação de detectores de objetos que mede a taxa de sobreposição entre a verdade básica e os limites previstos.

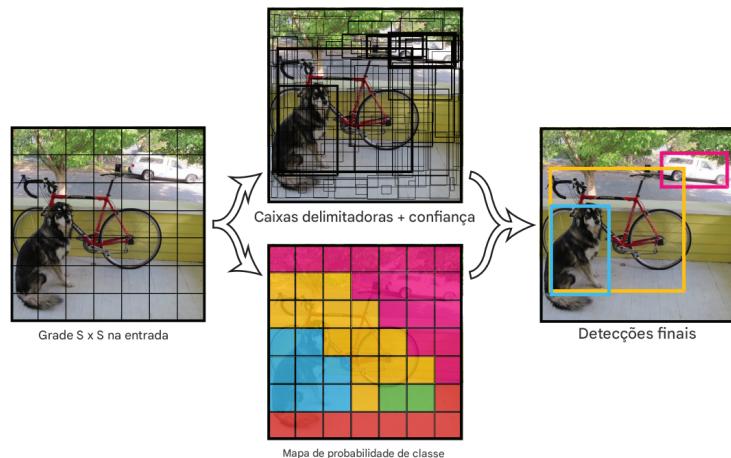


Figura 15 – Modelo. O sistema modela a detecção como um problema de regressão. Dividindo a imagem em uma grade $S \times S$ e para cada célula da grade prevê caixas delimitadoras B , confiança para essas caixas e probabilidades de classe C . Segundo os autores, essas previsões são codificadas como um tensor $S \times S \times (B \times 5 + C)$. [Redmon et al. \(2016, p. 2\)](#).

A arquitetura da rede é baseada no modelo GoogLeNet, combinando 24 camadas convolucionais seguidas por 2 camadas totalmente conectadas. As camadas convolucionais iniciais extraem características da imagem, enquanto as camadas totalmente conectadas preveem as probabilidades de saída e as coordenadas. Reduções 1×1 e convoluções 3×3 alternadas são usadas para reduzir a dimensionalidade e manter a robustez do modelo ([REDMON et al., 2016, p. 2](#)).

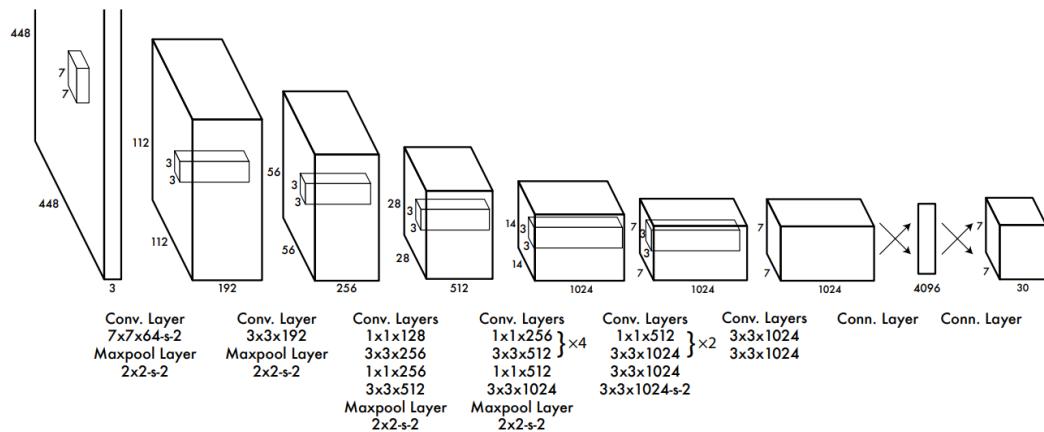


Figura 16 – Arquitetura. A rede de detecção tem 24 camadas convolucionais seguidas por 2 camadas totalmente conectadas. Camadas convolucionais alternadas 1×1 reduzem o espaço de recursos das camadas precedentes. Nós pré-treinamos as camadas convolucionais na tarefa de classificação do ImageNet na metade da resolução (imagem de entrada 224×224) e então dobramos a resolução para detecção ([REDMON et al., 2016, p. 3](#)).

A Figura 16 apresenta o design da arquitetura, que inclui camadas convolucionais

pré-treinadas no conjunto de dados ImageNet para classificação e adaptadas posteriormente para detecção com resolução ampliada.

4.2.2.2 Treinamento

O treinamento do YOLO segue uma abordagem cuidadosamente estruturada para otimizar sua eficiência e precisão. Segundo Redmon et al. (2016), a rede neural é inicialmente pré-treinada em um conjunto de dados amplo, como o ImageNet, que contém 1.000 classes, permitindo à rede aprender características visuais genéricas. Esse pré-treinamento utiliza as primeiras 20 camadas convolucionais, seguidas por uma camada de *pooling* média e uma camada totalmente conectada, alcançando uma precisão *top-5* de 88% na validação do ImageNet. Após essa fase, a rede é adaptada para detecção de objetos, adicionando camadas convolucionais e totalmente conectadas com pesos inicializados aleatoriamente, além de aumentar a resolução de entrada de 224×224 para 448×448 , garantindo maior detalhamento visual, conforme podemos identificar na Figura 16 e 17 que mostra o esquema global desse tipo de sistema para detecção. Uma imagem de entrada capturada com uma câmera é alimentada para o algoritmo de detecção de objetos YOLO, e por meio de uma rede neural convolucional profunda 16 ele detecta objetos e emite sinais de trânsito isolados quando apropriado. No contexto da nossa solução teremos a caixa delimitadora da detecção.

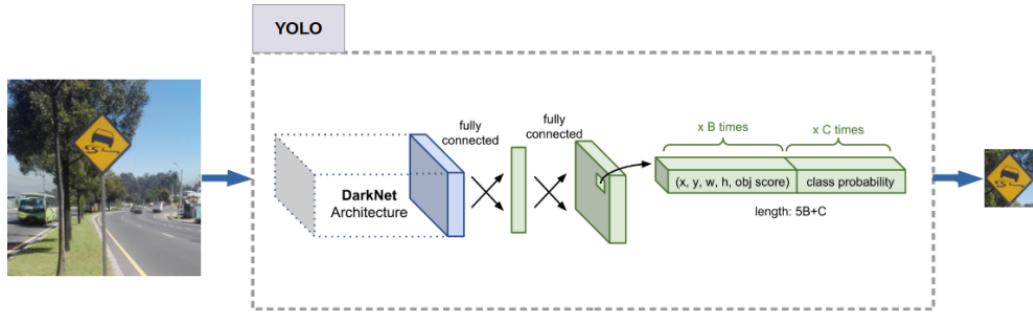


Figura 17 – Esquema geral de um sistema de detecção e reconhecimento de sinais de trânsito usando o algoritmo de detecção de objetos YOLO. Imagem de ([FLORES-CALERO et al., 2024](#), p. 2).

Entrada: Uma imagem é fornecida como entrada (neste caso, uma estrada com uma placa de trânsito).

Arquitetura Darknet: A imagem passa pela rede convolucional Darknet para extração de características.

Camadas totalmente conectadas: As características extraídas são processadas por camadas totalmente conectadas para gerar as previsões.

Predição de caixas delimitadoras: Para cada âncora (B), são calculados:

- x, y : Coordenadas do centro da caixa delimitadora.
- w, h : Largura e altura da caixa.
- obj : Confiança de que a caixa contém um objeto.

Probabilidade de classe: Para cada classe (C), é calculada a probabilidade de que o objeto na caixa pertence a essa classe.

Saída final: A posição da placa de trânsito e sua categoria são destacadas como a previsão final.

Segundo o autor Redmon et al. (2016), a função de perda do YOLO é projetada para equilibrar diferentes aspectos da detecção, como localização, confiança e classificação, empregando pesos distintos para as contribuições individuais ao erro. Durante o treinamento, a perda relacionada às coordenadas das caixas delimitadoras é ampliada para aumentar a precisão de localização, enquanto a perda associada a caixas sem objetos é reduzida, evitando a predominância de gradientes irrelevantes. O uso do índice de IOU, apresentado inicialmente [4.2.2.1](#), orienta a atribuição de responsabilidade às caixas preditoras, promovendo especialização e melhorando a capacidade da rede em prever objetos de tamanhos e proporções variados.

A regularização e a generalização são asseguradas por técnicas como *dropout* e aumento de dados. Aplicado após a primeira camada totalmente conectada, prevenindo sobreajuste/*overfitting* entre camadas.

Nesta trabalho, utilizaremos o conjunto disponibilizado por ([JUANOLA, 2019](#)) e ([Daniel, 2023](#)), que contém imagens extraídas do simulador CARLA. Este conjunto foi mencionado previamente na seção ([1.5](#)) deste relatório. O treinamento será realizado seguindo as etapas da documentação do YOLO ([REDMON, 2018](#)) e teremos como base o procedimento de treinamento realizado pelo autor [Juanola \(2019\)](#). Contudo, neste trabalho usaremos o YOLOv3 já treinado disponibilizado por [Redmon e Farhadi \(2018\)](#).

5 Implementação

Este capítulo apresenta a implementação da solução proposta para a simulação de um sistema de navegação autônoma utilizando o simulador CARLA. O objetivo central deste capítulo é demonstrar como os conceitos teóricos discutidos nos capítulos anteriores — tais como controle longitudinal e lateral, planejamento de trajetória e percepção visual — foram traduzidos em uma aplicação funcional no ambiente simulado.

Inicialmente, apresentamos o processo de instalação e configuração do simulador CARLA, incluindo os requisitos de hardware e software, bem como a estrutura de diretórios adotada no projeto. Em seguida, discutimos a implementação dos módulos responsáveis pelo controle de movimento do veículo, abordando tanto o controle longitudinal por meio de um controlador PID quanto o controle lateral baseado no algoritmo de Perseguição Pura (Pure Pursuit). Estes controladores foram programados em Python e integrados à simulação para permitir a navegação do veículo por uma trajetória previamente definida no mapa.

Além disso, este capítulo detalha os principais arquivos, scripts e configurações utilizados no projeto, incluindo módulos adicionais relacionados ao planejamento de trajetória, detecção de objetos por visão computacional com YOLO, e leitura de dados de sensores simulados. O fluxo completo da aplicação, desde a leitura dos waypoints até a geração dos comandos de direção, aceleração e frenagem, é descrito, de modo a permitir sua replicação.

Ressalvamos, no entanto, que, devido à extensão e à complexidade dos códigos-fonte desenvolvidos, não foi viável incluir todos os trechos de implementação diretamente neste relatório. A inclusão integral de todos os scripts e módulos comprometeria a fluidez e a clareza da exposição textual, além de extrapolar os limites editoriais típicos de trabalhos acadêmicos dessa natureza.

Entretanto, com o intuito de manter a transparência e de proporcionar ao leitor a possibilidade de aprofundamento técnico, todos os códigos, arquivos de configuração, *scripts* auxiliares e dados gerados ao longo da pesquisa foram organizados e disponibilizados em um repositório público, cujo link encontra-se no Capítulo (A) de Anexos. Incentivamos fortemente que interessados em replicar, estender ou adaptar a solução, consultem o material completo acessível via o link: https://github.com/ARRETdaniel/CARLA_simulator_YOLO-openCV_realTime_objectDetection_for_autonomousVehicles

O repositório foi estruturado para refletir a modularidade da implementação apresentada neste relatório. Cada pasta contém a documentação necessária para compreensão e execução autônoma dos módulos, incluindo instruções de uso, dependências e exemplos

de entrada e saída. Adicionalmente, o repositório é atualizado conforme o progresso da pesquisa, permitindo que a comunidade acompanhe o desenvolvimento contínuo do projeto.

Essa disponibilização está alinhada com boas práticas acadêmicas e científicas, promovendo a reproduzibilidade dos experimentos e fomentando a colaboração aberta. Ressaltamos, por fim, que o uso e análise do código disponibilizado serão particularmente úteis para estudantes, pesquisadores e profissionais que atuam nas áreas de veículos autônomos, robótica e visão computacional.

Caso haja dúvidas específicas sobre algum trecho da implementação ou dificuldades na execução dos *scripts*, os interessados poderão entrar em contato com o autor por meio do e-mail institucional indicado nos anexos.

5.1 Configurando o Simulador Carla

Nesta seção, apresentamos a configuração do simulador Carla utilizada para o desenvolvimento da solução proposta. Inicialmente, é necessário atender aos pré-requisitos descritos na documentação oficial do simulador ([University of Toronto, 2018a](#)). Todos os arquivos necessários encontram-se disponíveis no repositório da solução, conforme indicado nos Apêndices A.

Para a execução e testes da solução, foram empregados dois ambientes computacionais distintos:

- **Dell XPS 8960 (Linux):** Este equipamento, disponibilizado pelo Laboratório P5 da Universidade Estadual do Norte Fluminense (UENF), possui as seguintes especificações:
 - **Processador:** Intel Core i7-13700, com 16 núcleos (8 de desempenho e 8 de eficiência) e 24 threads, frequência base de 2.1 GHz e turbo de até 5.2 GHz;
 - **Memória RAM:** 32 GB DDR5-4800 MHz, configurada em dois módulos de 16 GB cada, operando em modo dual-channel;
 - **Armazenamento:** SSD NVMe PCIe de 1 TB;
 - **Placa de Vídeo:** NVIDIA GeForce RTX 4060 com 8 GB de memória GDDR6;
 - **Sistema Operacional:** Distribuição Linux compatível com o simulador Carla.
- **Lenovo Legion 5i 82CF0002BR (Windows):** Este notebook, pertencente ao pesquisador responsável por este trabalho, apresenta as seguintes configurações:
 - **Processador:** Intel Core i7-10750H (10^a geração), com 6 núcleos e 12 threads, frequência base de 2.6 GHz e turbo de até 5.0 GHz;
 - **Memória RAM:** 16 GB DDR4-2933 MHz;

- **Armazenamento:** SSD NVMe PCIe de 512 GB;
- **Placa de Vídeo:** NVIDIA GeForce RTX 2060 com 6 GB de memória GDDR6;
- **Sistema Operacional:** Windows 11 Home.

Pré-requisitos

Especificações de Hardware recomendadas:

- Processador Intel ou AMD quad-core, 2.5 GHz ou mais rápido
- NVIDIA GeForce 470 GTX ou AMD Radeon 6870 HD series ou superior
- 8 GB de RAM
- Aproximadamente 10GB de espaço em disco para a configuração do simulador

Nota: Computadores com especificações inferiores, incluindo sistemas com gráficos integrados, também podem executar o simulador CARLA, embora com desempenho reduzido.

Especificações de Software:

Windows/Ubuntu: CARLA requer Windows 7 64-bit ou Ubuntu 16.04 (ou superior). **Firewall:** Foi necessário habilitar a rede e permitir o acesso do firewall ao carregador do CARLA e, por padrão, às portas 2000, 2001 e 2002 (TCP e UDP).

Drivers da Placa Gráfica: Atualizamos os drivers mais recentes para evitar problemas gráficos. No caso, OpenGL 3.3 ou superior e DirectX 10 como recomendado.

Python:

- Foi instalado Python 3.6.0 com *pip*. O cliente Python do CARLA funciona com Python 3.5.x ou Python 3.6.x.
- Segundo o autor, Python 3.7 atualmente não é compatível.

Preparando o Simulador CARLA

Baixando e extraindo o Simulador CARLA

1. Baixamos o simulador CARLA (`CarlaUE4Windows.zip`), o qual pode ser encontrado no capítulo [A](#) (apêndice) deste trabalho.

2. Extraímos o conteúdo do `CarlaUE4Windows.zip`. A extração criou uma pasta chamada `CarlaSimulator` no diretório de trabalho, que hospeda os arquivos do servidor e cliente CARLA necessários para os projetos. O guia disponibilizado pelo autor ([University of Toronto, 2018a](#)) assume que o simulador é extraído para `C:\Coursera\CarlaSimulator`.

Instalação de Dependências Python para o Cliente As dependências adicionais necessárias para os arquivos do cliente do Simulador CARLA estão detalhadas no arquivo: `C:\Coursera\CarlaSimulator\requirements.txt`.

Foi executado o seguinte comando para instalar as dependências para o usuário atual:

```
> python -m pip install -r C:\Coursera\CarlaSimulator\requirements.txt  
--user [^2^] [2]
```

Para mais detalhes e instruções completas para a instalação, ou se houver problemas ao instalar essas dependências do simulador CARLA, consulte o material do capítulo [A](#). Salientamos que os binários do CARLA utilizados nesta pesquisa são uma versão modificada pelo autor ([University of Toronto, 2018a](#)) do CARLA original, com mapas adicionais incluídos.

5.2 Implementação do Controle de Veículos Autônomos

Nesta subseção, implementamos uma solução na linguagem de programação Python que visou guiar um carro por um trecho no ambiente de simulação Carla, como pode se observado na Figura 18, onde o marcador (1) represente o início da trajetória e o marcador (0) representa o final.



Figura 18 – Trecho do Mapa 1 ([CARLA, 2018](#)).

De modo a alcançarmos esse objetivo, fornecemos uma lista ordenada de pontos de referência igualmente espaçados no trecho da Figura 18 para o controlador, abordamos esses pontos de referência no trabalho de IC do ano passado [Gomes \(2024\)](#). Os pontos de referência incluem suas posições, bem como a velocidade que os veículos devem atingir, e podem ser encontrados no arquivo *waypoints.txt* do projeto. Como resultado, os pontos de referência tornam-se o sinal de referência para o controlador, dessa forma navegar por todos os pontos de referência completa efetivamente o trecho.

Devido a isso, precisamos implementar os controles longitudinais e laterais, ambos desenvolvidos extensivamente no ano de 2023/2024, respectivamente. Visto que a referência do controlador engloba tanto a posição quanto a velocidade do veículo. A saída do nosso controlador consistirá nos comandos de aceleração, freio e ângulo de direção do veículo. Onde a aceleração e o freio serão provenientes do controle longitudinal de velocidade, enquanto a direção será determinada pelo controle lateral, onde seu esquema pode ser visualizado na Figura 8.

```
C:.
├── behavioural_planner.py
├── collision_checker.py
├── controller2d.py
├── cutils.py
├── how_to_run.txt
├── local_planner.py
├── module_7.py
├── options.cfg
├── output_frames.txt
├── parked_vehicle_params.txt
├── parked_vehicle_params_old.txt
├── path_optimizer.py
├── predictions.jpg
├── process_img.py
├── readme.md
├── requirements.txt
├── stop_sign_params.txt
├── velocity_planner.py
└── waypoints.txt
    └── yolo_opencv.py
        └── yolo_utils.py

---yolov3-coco
    ├── coco-labels
    ├── get_model.sh
    ├── yolov3-tiny.cfg
    ├── yolov3-tiny.weights
    ├── yolov3.cfg
    └── yolov3.weights
```

Figura 19 – Estrutura do projeto (Elaborado pelos autores).

Podemos compreender a estrutura do nosso projeto, a partir da Figura 19. Onde os códigos para os controladores foram desenvolvidos no arquivo Python nomeado de *controller2d.py*, apresentado na subseção 5.1. Usamos uma classe (*class Controller2D*) no arquivo *controller2d.py* que engloba todas as informações pertinentes para a implementação do controlador, tais como: o estado do veículo, ponto de referência desejado, velocidade desejada e as saídas do controlador. Esta classe também contém funções que atualizam continuamente o estado do veículo e enviam as saídas do controlador para o simulador, conforme apresentamos na IC Gomes (2024).

Na estrutura do projeto ainda temos os seguintes módulos principais, os quais foram descritos de uma forma geral para o entendimento da funcionalidade de cada um, e disponíveis no apêndices A:

- **behavioural_planner.py** – Implementa estratégias de tomada de decisão para a navegação autônoma, garantindo a escolha adequada de trajetórias e reações a diferentes cenários viários.
- **collision_checker.py** – Responsável por detectar potenciais colisões com base na previsão de trajetórias e na lógica de desvio de obstáculos, garantindo a segurança na navegação.
- **local_planner.py** – Gera trajetórias de curto prazo para a navegação do veículo, complementando as decisões do planejador comportamental.
- **path_optimizer.py** – Otimiza as trajetórias geradas pelo planejador local, assegurando suavidade e viabilidade no movimento do veículo.

- **velocity_planner.py** – Regula a velocidade do veículo com base nas condições do tráfego, nos limites da trajetória e nas diretrizes de segurança.
- **yolo_opencv.py** – Implementa a integração do modelo YOLO para a detecção de objetos em tempo real utilizando OpenCV, possibilitando o reconhecimento de placas de trânsito e outros elementos do ambiente.
- **yolo_utils.py** – Contém funções auxiliares para o carregamento do modelo YOLO, pré-processamento de imagens e inferência dos resultados.
- **process_img.py** – Realiza o pré-processamento das imagens antes de alimentar os quadros no pipeline de percepção, aprimorando a detecção e a segmentação dos objetos de interesse.
- **options.cfg** – Arquivo de configuração contendo hiperparâmetros e ajustes essenciais para o funcionamento dos diferentes módulos do projeto.
- **waypoints.txt** – Contém um conjunto de waypoints pré-definidos que guiam a navegação do veículo autônomo durante a simulação.
- **stop_sign_params.txt** – Arquivos que definem os parâmetros para a detecção de placas de parada, incluindo limiares de reconhecimento e ajustes de classificação.
- **parked_vehicle_params.txt** – Parâmetros para a identificação de veículos estacionados no ambiente simulado, auxiliando na navegação autônoma.
- **readme.md** – Documentação detalhada do projeto, contendo instruções de configuração, uso e descrição dos módulos.
- **how_to_run.txt** – Instruções específicas sobre como executar os diferentes módulos do projeto e realizar os testes na simulação.

O conjunto desses módulos possibilita um *pipeline* completo de percepção e planejamento para VAs, integrando reconhecimento de placas de trânsito baseado em YOLO com planejamento de movimento e controle.

A implementação que realizamos no arquivo *controller2d.py* foi a de um controlador PID. O mesmo que foi discutido na IC [Gomes \(2024\)](#), para o controle longitudinal. Onde seu diagrama pode ser visualizado na Figura 8, e para implementação da solução do controlador PID seguimos o material da IC [Gomes \(2024\)](#). Juntamente à implementação do PID, implementamos um controle de antecipação. Como apresentado na IC [Gomes \(2024\)](#), os VAs requerem comandos constantes de aceleração ou frenagem para manter velocidades ou taxas de desaceleração constantes, os comandos de *feedforward* são extremamente benéficos para melhorar o desempenho da condução autônoma.

De mesmo modo, implementamos o Controlador de Perseguição Pura para o controle Lateral, desenvolvido na IC [Gomes \(2024\)](#). Para chegarmos na nossa solução seguimos o material da IC [Gomes \(2024\)](#), na qual apresentamos sobre os pontos de referência usados para guiar o veículo pela trajetória.

Desse modo, implementamos ambos os controladores seguindo todo material elaborado ao longo do relatório e estudo durante a iniciação científica como a base teórica, as quais foram apresentadas no referencial teórico do relatório de [Gomes \(2024\)](#), onde também se encontra o trecho de código explicado para essa solução de controlador. Adicionalmente, no anexo A disponibilizamos o link para o repositório com a solução completa.

Para executar a implementação inserimos os comandos da Listagem 5.1 no terminal de comando CMD para iniciar o simulador Carla na pista *Town 1*.

```
1 C:  
2 cd \CarlaSimulator  
3 CarlaUE4.exe /Game/Maps/Course4 -windowed -carla-server -  
    ↪ benchmark -fps=30
```

Lista de código 5.1 – Inicializando o simulador Carla na pista Town 1

Em outro terminal, inserimos, também, os comandos da Listagem 5.2, vistos a seguir, para executar a nossa solução.

```
1 C:  
2 cd \Coursera\CarlaSimulator\PythonClient\FinalProject  
3 python module_7.py
```

Lista de código 5.2 – Executando a solução no simulador Carla

Isso resulta na execução da simulação, conforme podemos visualizar na captura de tela da Figura 20.

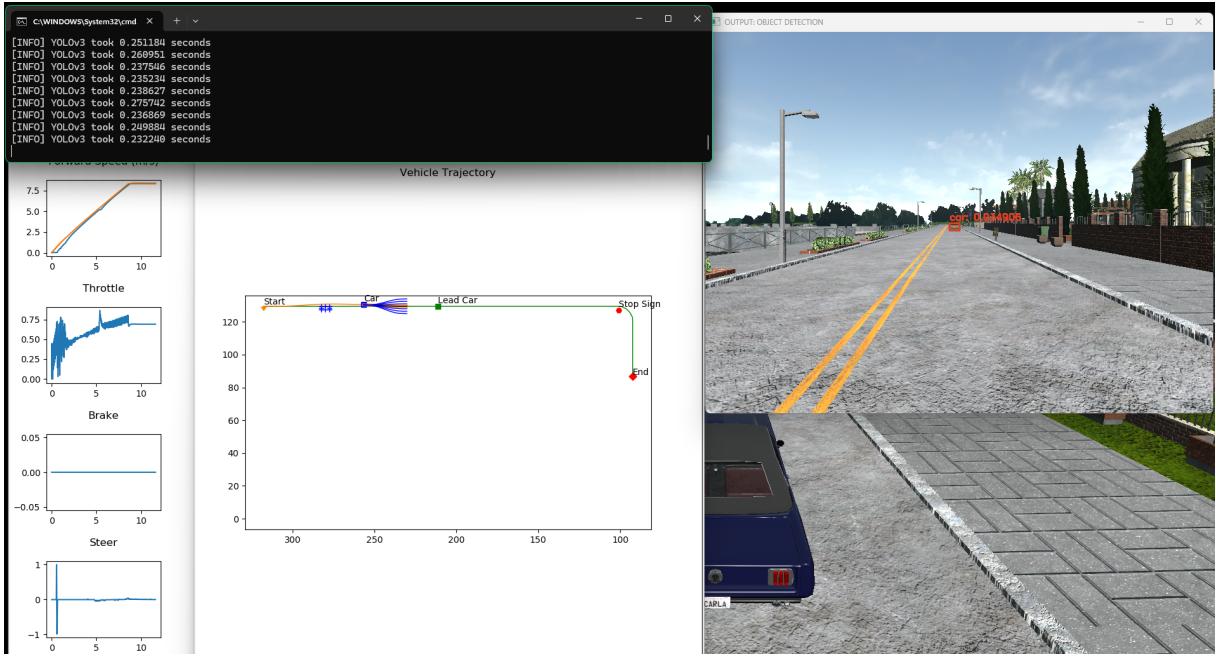


Figura 20 – Captura de Tela da execução da solução no simulador CARLA (Elaborado pelos autores).

Ao finalizar a execução, os arquivos *trajectory.txt* e *collision_count.txt* são gerados, nos dando a trajetória percorrida pela nossa solução e se houve alguma colisão com objetos pelo caminho. A solução completa com todos os trechos de código pode se encontrar no Anexo A.

5.3 Integração da Solução YOLO ao Simulador CARLA

Para realizar a detecção de objetos em tempo real no ambiente simulado pelo CARLA, optamos pela utilização do modelo YOLOv3 (You Only Look Once, versão 3), amplamente reconhecido por seu desempenho eficiente em tarefas de detecção em tempo real. A integração da arquitetura de detecção ao ambiente simulado demanda a correta configuração e carregamento dos arquivos essenciais: o arquivo de configuração da rede, os pesos treinados, e a lista de classes do dataset COCO, conforme apresentado na subseção 4.2.2.

A inicialização do modelo é apresentada no trecho de código 5.3, no qual realizamos a leitura dos arquivos necessários utilizando a API de redes neurais do OpenCV (`cv2.dnn`). Esta etapa é fundamental para preparar a rede para o processamento dos frames provenientes da câmera do simulador.

```

1 # Define os caminhos para os arquivos de configuração, pesos e rótulos
  do modelo YOLOv3
2 model_configuration = './yolov3-coco/yolov3.cfg'      # Arquivo de
  configuração da arquitetura da rede YOLOv3

```

```

3 model_weights = './yolov3-coco/yolov3.weights'           # Arquivo com os
   pesos pré-treinados da rede YOLOv3
4 model_labels = './yolov3-coco/coco-labels'             # Arquivo com os
   nomes das classes (rótulos) do dataset COCO
5 # Carrega os rótulos das classes a partir do arquivo de texto
6 labels = open(model_labels).read().strip().split('\n') # Lê o conteúdo
   do arquivo, remove espaços extras e divide cada linha em uma entrada
   da lista
7 # Inicializa o modelo YOLOv3 com os arquivos de configuração e pesos
   carregados
8 net = cv2.dnn.readNetFromDarknet(model_configuration, model_weights)
9 # Obtém os nomes de todas as camadas da rede
10 layer_names = net.getLayerNames()
11 # Filtra apenas os nomes das camadas de saída (output layers), ou seja,
   as camadas que fornecem as detecções finais
12 layer_names = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

```

Lista de código 5.3 – Configuração e carregamento do modelo YOLOv3 utilizando o OpenCV DNN (Elaborado pelos autores).

Os arquivos utilizados nesta configuração encontram-se organizados conforme a estrutura de diretórios ilustrada na Figura 19. Uma vez carregado e inicializado o modelo, é possível realizar inferências diretamente sobre os frames capturados do ambiente simulado, conforme demonstrado na Listagem do código 5.4.

Neste trecho de código 5.4, observa-se a lógica implementada para a detecção de objetos de forma a otimizar o desempenho computacional, realizando inferências completas a cada 6 quadros. Nos quadros intermediários, reutiliza-se a informação previamente inferida, reduzindo o custo computacional sem comprometer significativamente a responsividade do sistema.

```

1 for frame in range(TOTAL_EPISODE_FRAMES):
2     # Coleta os dados atuais do servidor CARLA, incluindo medições e
   sensores
3     measurement_data, sensor_data = client.read_data()
4
5     # Captura o frame atual da câmera RGB
6     frame_camera = sensor_data['CAMERA']
7     frame_obj_to_detect = np.array(frame_camera.data) # Converte os
   dados da imagem em um array NumPy
8
9     # Executa a inferência de detecção de objetos a cada 6 frames para
   otimizar desempenho
10    if count_obj_detection == 0:
11        # Primeira inferência: executa detecção de objetos normalmente
12        frame_obj_to_detect, boxes, confidences, classids, idxs =

```

```

13     infer_image(
14         net, layer_names,
15         sensor_data['CAMERA'].height, sensor_data['CAMERA'].width,
16         frame_obj_to_detect, colors, labels
17     )
18     count_obj_detection += 1
19 else:
20     # Nos próximos 5 frames, reutiliza as detecções anteriores (sem
21     # nova inferência)
22     frame_obj_to_detect, boxes, confidences, classids, idxs =
23     infer_image(
24         net, layer_names,
25         sensor_data['CAMERA'].height, sensor_data['CAMERA'].width,
26         frame_obj_to_detect, colors, labels,
27         boxes, confidences, classids, idxs,
28         infer=False # Sinaliza para não realizar nova inferência
29     )
30     # Atualiza o contador para reiniciar após 6 frames
31     count_obj_detection = (count_obj_detection + 1) % 6
32
33     # Converte a imagem de RGB para BGR (formato padrão do OpenCV)
34     frame_obj_detected = cv2.cvtColor(frame_obj_to_detect, cv2.
COLOR_RGB2BGR)
35
36     # Exibe a imagem com os objetos detectados em uma janela
37     cv2.imshow('OUTPUT: OBJECT DETECTION', frame_obj_detected)

```

Lista de código 5.4 – Loop de detecção de objetos com uso do modelo YOLOv3 sobre imagens da câmera no CARLA (Elaborado pelos autores).

Dessa forma, a solução proposta encontra-se adequadamente integrada ao ambiente CARLA, permitindo a realização de inferências em tempo real para a detecção e acompanhamento de objetos no cenário urbano simulado.

6 Resultados e Discussão

Os resultados deste terceiro ano de pesquisa são desenvolvidos neste capítulo 6. No qual visa satisfazer o *objetivo* definido no Capítulo 1.3. Desse modo, desenvolveremos o objetivo proposto, elaborando e apresentando os resultados e o que significam em relação ao trabalho realizado neste ano.

A avaliação do desempenho da nossa solução consistiu na verificação da conformidade do veículo com a trajetória, se o veículo seguiu os pontos de referência do trecho do mapa durante a simulação e se não houve colisão com obstáculos durante a trajetória. Para realizar essa verificação, ao final da simulação, gerou-se dois arquivos: *trajectory.txt* e *collision_count.txt*, que contém a trajetória completa do veículo ao longo do experimento, assim como se houve colisão com objetos, o trajeto em questão pode ser visto na Figura 18. O resultado obtido pela nossa solução pode ser visualizado no gráfico da Figura 21.

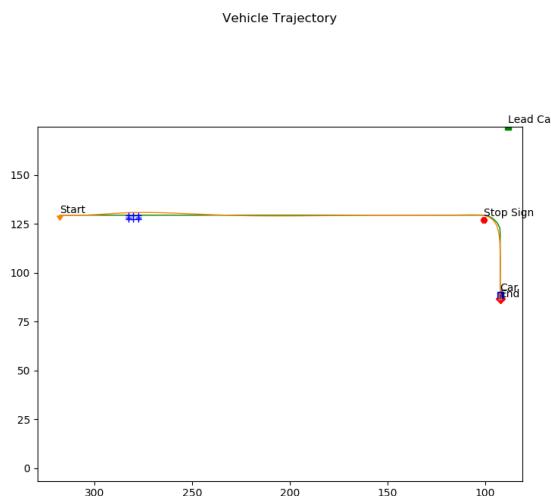


Figura 21 – Trajetória percorrida pela solução (Elaborado pelos autores).

Além da análise da trajetória e verificação de colisões durante a simulação, também foi realizada uma avaliação da acurácia do sistema de percepção visual implementado, utilizando a arquitetura YOLO integrada ao simulador CARLA. A Figura 22 ilustra um dos momentos da simulação em que dois elementos (outro veículo e um sinal de parada) fundamentais do trânsito foram corretamente detectados e classificados pelo modelo.



Figura 22 – Imagem capturada da saída do YOLO durante a simulação.

No exemplo da Figura 22, observa-se a eficácia do modelo YOLOv3 na identificação de objetos críticos para a condução autônoma. O carro à frente foi identificado com uma confiança de 0.99, indicando altíssima certeza da rede quanto à presença e à classe do objeto detectado. Este nível de confiança é desejável em aplicações de tempo real, pois reduz a probabilidade de falsos positivos, o que, em um cenário real, pode implicar em reações inadequadas do veículo (por exemplo, frenagens desnecessárias ou desvios imprecisos).

Adicionalmente, a placa de "Stop/PARE", na mesma Figura 22, foi detectada com uma confiança de 0.97. Este resultado é particularmente relevante para os sistemas de condução autônoma, pois sinalizações de parada obrigatória demandam resposta imediata e precisa por parte do veículo. A alta taxa de confiança obtida sugere que o sistema está apto a reconhecer sinais regulamentares em tempo hábil, condição essencial para o cumprimento da legislação de trânsito e para a segurança dos ocupantes e demais usuários da via.

A análise desses resultados permite concluir que a integração do modelo de percepção com o ambiente simulado foi bem-sucedida. Ressalta-se, no entanto, que o desempenho em ambiente controlado (como o CARLA) não necessariamente se traduz em desempenho equivalente em ambientes reais, dado o aumento da variabilidade e do ruído. Ainda assim, os dados obtidos apontam para um sistema robusto no contexto simulado.

7 Considerações Finais

O objetivo estabelecido para o terceiro ano desta Iniciação Científica (2024–2025), conforme apresentado no capítulo 1.3, foi a exploração e aplicação de conceitos fundamentais voltados à percepção visual, estimativa e localização de estado, bem como ao planejamento de movimento em VAs. Esse novo ciclo representou uma evolução natural em relação às etapas anteriores do projeto, permitindo um aprofundamento técnico e prático em áreas críticas para o desenvolvimento de VAs.

Com base em revisões bibliográficas e referenciais teóricos atualizados, estruturamos as etapas do trabalho com foco na construção de um sistema de assistência à condução. Tal sistema foi projetado para detectar e reconhecer placas de trânsito, dando feedback visual ao condutor e promovendo maior segurança na condução. Para isso, utilizamos o ambiente de simulação CARLA aliado ao modelo de detecção baseado no algoritmo YOLO. Usando dados sintéticos gerados a partir de cenários virtuais no CARLA permitindo não apenas a replicação de diferentes condições ambientais, mas também a realização de experimentos sem os custos e riscos associados aos testes em ambientes reais. Os resultados obtidos demonstraram que a proposta foi bem-sucedida em identificar sinalizações de trânsito com elevado grau de acurácia ($\geq 90\%$), evidenciando o potencial dessas soluções no contexto de VAs.

Adicionalmente, foi possível observar desafios relacionados à interpretabilidade desses modelos, indicando a necessidade de futuras investigações voltadas ao uso de abordagens mais explicáveis. De maneira geral, este terceiro ano de pesquisa proporcionou uma experiência enriquecedora, ampliando significativamente a compreensão sobre os sistemas de percepção e planejamento utilizados pela indústria automotiva na construção de veículos inteligentes.

Com isso, podemos concluir que os objetivos propostos para o ano de 2024–2025 foram plenamente alcançados, consolidando a formação técnica adquirida ao longo dos três anos de Iniciação Científica e preparando terreno para investigações futuras em níveis mais avançados de automação veicular.

8 Perspectiva de continuidade

Conforme os *objetivos* delineados no capítulo 1.3, este projeto de Iniciação Científica 2024–2025 concentrou-se no aprofundamento dos estudos sobre percepção visual, estimativa e localização de estado e planejamento de movimento aplicados a VAs. Tais temas foram explorados por meio de implementações práticas em ambientes de simulação, com ênfase na utilização de algoritmos de detecção baseados em YOLO, para o reconhecimento de placas de trânsito.

No entanto, apesar do desempenho promissor apresentado por esses modelos em tarefas de detecção em tempo real, observou-se que a interpretabilidade e transparência das decisões tomadas por essas soluções ainda constituem uma limitação relevante. Em virtude disso, propõe-se para o próximos ciclos de pesquisa uma continuidade do projeto com ênfase na investigação de soluções mais explicáveis para sistemas de percepção visual. O objetivo será buscar metodologias que combinem acurácia e robustez com capacidade de justificativa de decisão, de forma a alinhar os sistemas embarcados de percepção às exigências de confiabilidade e segurança, fundamentais para o emprego de VAs em ambientes reais.

Pretende-se, portanto, explorar abordagens que possibilitem maior interpretabilidade das soluções. Das quais serão testadas em cenários simulados no simulador CARLA, o que permitirá avaliar o desempenho e a viabilidade prática dessas soluções em contextos variados.

Adicionalmente, surge o interesse em dar continuidade à pesquisa em curso, direcionando-a para a implementação de uma abordagem dinâmica, na qual o módulo de percepção transmite, em tempo real, os dados relativos à localização dos objetos identificados ao módulo de planejamento, permitindo uma tomada de decisão adaptativa. Essa proposta se distingue da abordagem atual, que se baseia na utilização de dados estáticos para a definição da trajetória e a execução do planejamento e controle. Com essa evolução, busca-se integrar de forma mais eficaz os módulos de percepção e planejamento, garantindo uma adaptação contínua e reativa às mudanças no ambiente, o que representa um avanço significativo no processo de navegação autônoma para a solução anterior.

Dessa forma, as próximas etapas desta pesquisa tem como perspectiva a consolidação dos conhecimentos adquiridos até o momento, aliada à ampliação crítica das ferramentas computacionais disponíveis, visando tornar os sistemas autônomos não apenas mais eficientes, mas também mais compreensíveis, confiáveis e relativos para os usuários e desenvolvedores.

9 Participação em congressos e trabalhos publicados ou submetidos e outras atividades acadêmicas e de pesquisa

Os certificados obtidos e/ou comprovantes de participação podem ser encontrados neste capítulo. Na Figura 23 encontra-se o Certificado de conclusão da Especialização em Self-Driving Cars, o qual pode ser verificado através da url: <https://coursera.org/verify/specialization/FWWPSO6451JX>.



Figura 23 – Certificado de conclusão da Especialização em Self-Driving Cars.

Na Figura 24 encontra-se o certificado de participação no XVI Congresso Fluminense de Iniciação Científica e Tecnológica e IX Congresso Fluminense de Pós-Graduação, realizado entre os dias 10/06/2024 e 14/06/2024 na UENF, com carga horária total de 40

horas.



Verifique o código de autenticidade 18377491.86620015.6.8.83774918662001568 em <https://www.even3.com.br/documents>

Certificamos que **Daniel Terra Gomes**, participou do XVI Congresso Fluminense de Iniciação Científica e Tecnológica e do IX Congresso Fluminense de Pós-Graduação realizados de 10/06/2024 a 14/06/2024, na Universidade Estadual Norte Fluminense Darcy Ribeiro em Campos dos Goytacazes, contabilizando carga horária total de 40 horas.

Campos dos Goytacazes, 18 de junho de 2024.

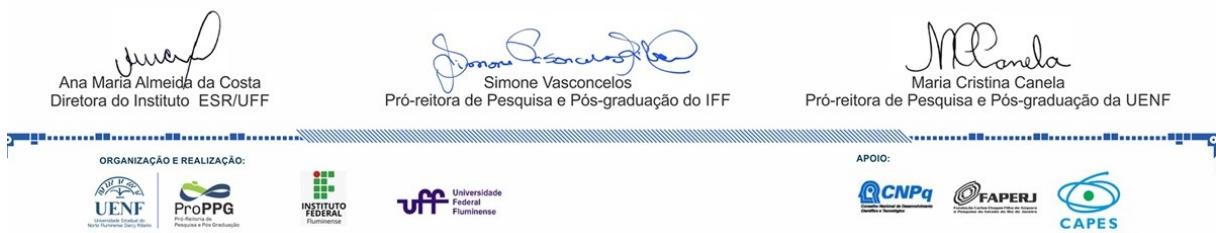


Figura 24 – Certificado de participação no CONFICT 2024.

Na Figura 25 encontra-se o certificado de participação no minicurso "Robótica competitiva, vivências da equipe Goytaborgs do IFF", realizado no dia 10/06/2024 durante o CONFICT 2024, com carga horária de 2 horas.



Certificamos que **Daniel Terra Gomes**, participou do **Minicurso - Robótica competitiva, vivências da equipe Goytaborgs do IFF** realizado em 10/06/2024, durante o XVI Congresso Fluminense de Iniciação Científica e Tecnológica e do IX Congresso Fluminense de Pós-Graduação, contabilizando carga horária total de 2 horas.

Campos dos Goytacazes, 18 de junho de 2024.

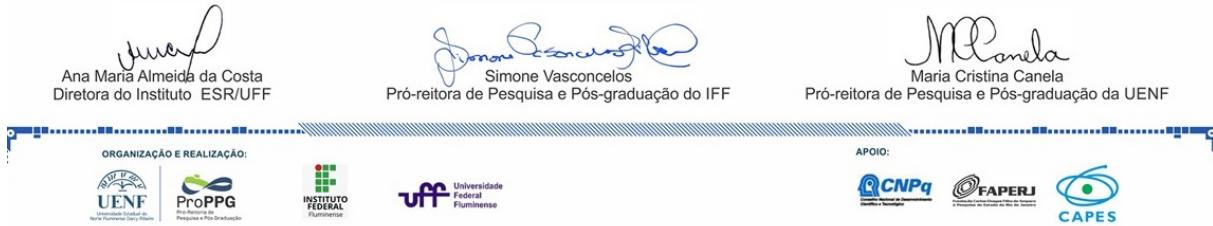
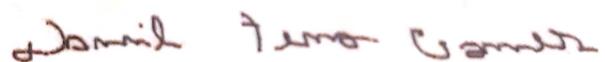


Figura 25 – Certificado de participação no minicurso de Robótica Competitiva durante o CONFICT 2024.

10 Datas e assinaturas

10.1 Data e assinatura do bolsista (assinatura digitalizada)

A digitalized signature in brown ink, appearing to read "Daniel Tavares Gama".

30/04/2025

10.2 Data e assinatura do orientador (assinatura digitalizada)

A digitalized signature in black ink, appearing to read "Annabell D.R. Tamariz".

30/04/2025

Referências

AHIRE, P. et al. Simulating vehicle driving using carla. *Journal of Electrical Systems*, Engineering and Scientific Research Groups, v. 20, n. 10s, p. 44–52, 2024. Citado 4 vezes nas páginas 11, 12, 13 e 14.

ANDRADE, G. G. de. Trabalho de Conclusão de Curso, *Uma Proposta para Detecção de Objetos e Estimação de Distância em um Simulador de Veículos Autônomos*. Brasília, DF: [s.n.], 2022. 72 p., il. (algumas color.); 30 cm. Keywords: Veículos Autônomos, Detecção de Objetos. Disponível em: https://github.com/guilherme1guy/carla_darknet_integration. Citado na página 14.

ANWER, F. et al. Comparative analysis of two popular agile process models: extreme programming and scrum. *International Journal of Computer Science and Telecommunications*, v. 8, n. 2, p. 1–7, 2017. Citado na página 20.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. Disponível em: <https://arxiv.org/abs/2004.10934>. Citado 2 vezes nas páginas 17 e 25.

BRATZEL, S.; CAM. *Die Zukunft der Mobilität - Die Zukunftsrends in den Bereichen Elektromobilität, Connected Car und Mobilitätsdienstleistungen*. 2022. <https://www.bnpparibascardif.de/web/guest>. Acessado 09-03-2024. Citado na página 11.

BUSSEMAKER, K. Sensing requirements for an automated vehicle for highway and rural environments. Delft University of Technology, 2014. Citado na página 67.

CARLA. [S.l.]: CARLA Team, 2018. <https://carla.org/>. Acessado: 26-03-2024. Citado 2 vezes nas páginas 6 e 44.

CARLA Simulator. *CARLA Simulator: Cameras and Sensors*. [S.l.], 2024. Available in the CARLA Simulator documentation, Quick Start section. Acessado: 28-11-2024. Citado 4 vezes nas páginas 5, 7, 34 e 35.

CRISTINA, P. M.; WAZLAWICK, P. R. S. *Metodologia de Pesquisa para Ciência da Computação*. 2021. https://edisciplinas.usp.br/pluginfile.php/4409810/mod_resource/content/2/2-MetodologiaDePesquisa-EstilosDePesquisa_cap2.pdf. Acessado: 26-11-2024. Citado 2 vezes nas páginas 5 e 16.

Daniel. *Carla-Object-Detection-Dataset: Labeled Dataset for Object Detection in Carla Simulator*. 2023. Acessado: 26-11-2024. Disponível em: <https://github.com/DanielHfnr/Carla-Object-Detection-Dataset/tree/master>. Citado 2 vezes nas páginas 17 e 39.

DOSOVITSKIY, A. et al. Carla: An open urban driving simulator. In: PMLR. *Conference on robot learning*. [S.l.], 2017. p. 1–16. Citado 5 vezes nas páginas 11, 12, 13, 14 e 17.

DREIBELBIS, E. *Global fleet of autonomous vehicles may emit more carbon than Argentina*. 2023. <https://www.pcmag.com/news/global-fleet-of-autonomous-vehicles-may-emit-more-carbon-than-argentina>. Accessed: 2023-3-7. Citado na página 11.

European Commission, Directorate-General for Mobility and Transport. *Safer roads for all: EU Road Safety Policy Framework 2021-2030*. 2021. Acessado: 11-07-2024. Disponível em: https://road-safety.transport.ec.europa.eu/system/files/2021-07/safer_roads4all.pdf. Citado na página 11.

Federal Highway Administration. *The Relation Between Speed and Crashes*. 2012. Acessado: 11-07-2024. Disponível em: https://safety.fhwa.dot.gov/speedmgt/ref_mats/fhwasa1304/Resources3/08%20-%20The%20Relation%20Between%20Speed%20and%20Crashes.pdf. Citado na página 11.

FLORES-CALERO, M. et al. Traffic sign detection and recognition using yolo object detection algorithm: A systematic review. *Mathematics*, v. 12, n. 2, 2024. ISSN 2227-7390. Disponível em: <https://www.mdpi.com/2227-7390/12/2/297>. Citado 2 vezes nas páginas 6 e 38.

GitHub, Inc. *Github Docs - About continuous integration*. 2024. <https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration>. Acessado: 05-03-2024. Citado na página 20.

GOMES, D. T. Relatório Técnico, *Dirigindo para o futuro: Softwares e Algoritmos Avançados que Alimentam Veículos Autônomos*: Relatório anual pibic - projeto: Project-driven data science: Aprendendo e mapeando. 2024. Período: Junho 2023 – Março 2024. Orientadora: Prof.^a Dra. Annabell Del Real Tamariz. Fonte Financiadora: PIBIC/CNPq. Disponível em: https://github.com/ARRETdaniel/CARLA_simulator_YOLO-openCV_realTime_objectDetection_for_autonomousVehicles/tree/main/materialComplementar. Citado 9 vezes nas páginas 20, 26, 28, 30, 34, 44, 45, 46 e 47.

IGNATIOUS, H. A.; HESHAM-EL-SAYED; KHAN, M. An overview of sensors in autonomous vehicles. *sciencedirect*, Elsevier, p. 1–6, 2022. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050921025540>. Citado na página 33.

IRIE, K. *Yolov4_darknet: YOLoV4 - Neural Networks for Object Detection (Windows and Linux version of darknet)*. 2020. Citado na página 17.

JACOBSON, B. et al. Vehicle dynamics compendium. *Chalmers University of Technology*, Chalmers, 2020. Disponível em: <https://research.chalmers.se/en/publication/520229>. Citado na página 67.

JANAI, J. et al. Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art. *Foundations and Trends® in Computer Graphics and Vision*, v. 12, n. 1–3, p. 1–308, 2020. ISSN 1572-2740, 1572-2759. Citado 2 vezes nas páginas 11 e 13.

JUANOLA, M. S. *Speed Traffic Sign Detection on the CARLA Simulator Using YOLO*. Dissertaçāo (Master's Thesis) — Master in Intelligent Interactive Systems, 2019. Treball fi de màster. Disponível em: <http://hdl.handle.net/10230/42548>. Citado 3 vezes nas páginas 14, 17 e 39.

KHAN, M. A. et al. Level-5 autonomous driving—are we there yet? a review of research literature. *ACM Computing Surveys (CSUR)*, ACM New York, NY, p. 1–38, 2022. Disponível em: https://www.researchgate.net/publication/358040996_Level-5_Autonomous_Driving_Are_We_There_Yet_A_Review_of_Research_Literature. Citado 5 vezes nas páginas 12, 13, 14, 33 e 34.

- KPMG International. *2020 Autonomous Vehicles Readiness Index*. 2020. <https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2020/07/2020-autonomous-vehicles-readiness-index.pdf>. Accessed: 2023-2-22. Citado na página 11.
- LAGE, C. A. Quatro cenários para os veículos autônomos no mundo ocidental. *UNIVERSIDADE DE BRASÍLIA*, 2019. Citado na página 11.
- LANCTOT, R. *Accelerating the Future: The Economic Impact of the Emerging Passenger Economy*. Strategy Analytics, 2017. Acessado: 11-07-2024. Disponível em: <https://www.intel.com/content/dam/www/public/us/en/documents/pdf/passenger-economy-report-autonomous-driving.pdf>. Citado na página 11.
- LI, S.; WANG, S.; WANG, P. A Small Object Detection Algorithm for Traffic Signs Based on Improved YOLOv7. *Sensors*, MDPI, v. 23, p. 7145, 2023. Citado na página 12.
- LI, S.; WANG, S.; WANG, P. A small object detection algorithm for traffic signs based on improved yolov7. *Sensors*, v. 23, n. 16, 2023. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/23/16/7145>. Citado na página 14.
- MOZAFFARI, O. Y. S. Deep learning-based vehicle behaviour prediction for autonomous driving applications: a review. *elsevier*, v. 2, p. 1–15, 2020. Citado na página 12.
- National Highway Traffic Safety Administration. *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey: DOT HS 812 506 A Brief Statistical Summary*. 2018. Acessado: 11-07-2024. Disponível em: <https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812506>. Citado 3 vezes nas páginas 11, 13 e 15.
- NEUFVILLE, R. Potential of connected fully autonomous vehicles in reducing congestion and associated carbon emissions. *Sustainability*, 2022. Disponível em: <https://www.mdpi.com/2071-1050/14/11/6910>. Citado na página 12.
- OKPONO, J. et al. Advanced driver assistance systems road accident data insights: Uncovering trends and risk factors. *The International Journal of Engineering Research. Review ID-TIJER2409017*, ISSN, p. 2349–9249, 2024. Citado 4 vezes nas páginas 11, 12, 13 e 15.
- OTHMAN, K. Multidimension analysis of autonomous vehicles: The future of mobility. *Civil Engineering Journal*, 2021. ISSN 2676-6957. Disponível em: www.CivileJournal.org. Citado na página 11.
- PAREKH NISHI PODDAR, M. C. D. A review on autonomous vehicles: Progress, methods and challenges. *Electronics*, 2022. ISSN 2525-8761. Citado 2 vezes nas páginas 11 e 34.
- PIVTORAIKO, M.; KNEPPER, R. A.; KELLY, A. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, Wiley Online Library, v. 26, n. 3, p. 308–333, 2009. Citado 2 vezes nas páginas 32 e 33.
- REDMON, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. <http://pjreddie.com/darknet/>. Acessado: 28-11-2024. Citado na página 17.

- REDMON, J. *You Only Look Once (YOLO) - Darknet*. 2018. <https://pjreddie.com/darknet/yolo/>. Acessado: 28-11-2024. Citado 2 vezes nas páginas 35 e 39.
- REDMON, J. et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. Disponível em: <https://arxiv.org/abs/1506.02640>. Citado 7 vezes nas páginas 5, 6, 24, 35, 36, 37 e 38.
- REDMON, J.; FARHADI, A. *YOLO9000: Better, Faster, Stronger*. 2016. Disponível em: <https://arxiv.org/abs/1612.08242>. Citado na página 25.
- REDMON, J.; FARHADI, A. *YOLOv3: An Incremental Improvement*. 2018. Disponível em: <https://arxiv.org/abs/1804.02767>. Citado 3 vezes nas páginas 25, 35 e 39.
- SAE. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE Mobilius*, SAE International, p. 1–41, 2021. Disponível em: https://www.sae.org/standards/content/j3016_202104. Citado 3 vezes nas páginas 11, 13 e 67.
- SEBO, D. Impact of electric vehicle market growth on automotive industry transformation: Trends, potentials, and challenges analysis. In: *Economic and Social Development (Book of Proceedings), 106th International Scientific Conference on Economic and Social*. [S.l.: s.n.], 2024. p. 73. Citado na página 11.
- SINGH, G. B. . K. B. . R. K. Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. Hindawi, 2022. Citado 2 vezes nas páginas 12 e 34.
- SNIDER, J. M. et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009. Citado na página 67.
- University of Toronto. *Introduction to Self-Driving Cars*. Coursera Inc, 2018. Taught by Steven Waslander and Jonathan Kelly, Associate Professors in Aerospace Studies. Part of the Self-Driving Cars Specialization. Advanced level. Includes a project using the CARLA simulation environment. Online; Último acesso em: 1 de Março de 2024. Disponível em: <https://www.coursera.org/learn/intro-self-driving-cars?specialization=self-driving-cars>. Citado 5 vezes nas páginas 5, 27, 28, 41 e 43.
- University of Toronto. *Motion Planning for Self-Driving Cars*. Coursera Inc, 2018. Taught by Steven Waslander and Jonathan Kelly, Associate Professors in Aerospace Studies. Part of the Self-Driving Cars Specialization. Advanced level. Online; Último acesso em: 24 de Novembro de 2024. Disponível em: <https://www.coursera.org/learn/motion-planning-self-driving-cars>. Citado 5 vezes nas páginas 5, 30, 31, 32 e 33.
- WACHENFELD, W.; WINNER, H. The Release of Autonomous Vehicles. In: MAURER, M. et al. (Ed.). *Autonomous Driving: Technical, Legal and Social Aspects*. Berlin, Heidelberg: Springer, 2016. p. 425–449. ISBN 978-3-662-48847-8. Citado 2 vezes nas páginas 12 e 13.
- WANG, A. et al. *YOLOv10: Real-Time End-to-End Object Detection*. 2024. Disponível em: <https://arxiv.org/abs/2405.14458>. Citado 2 vezes nas páginas 5 e 25.

- WANG, C.-Y.; LIAO, H.-Y. M. YOLOv1 to YOLOv10: The Fastest and Most Accurate Real-Time Object Detection Systems. *arXiv:2408.09332v1 [cs.CV]*, ago. 2024. License: arXiv.org perpetual non-exclusive license. Available at: <<https://arxiv.org/html/2408.09332v1>>. Disponível em: <https://arxiv.org/html/2408.09332v1>. Citado na página 25.
- WANG, H.; YU, H. Traffic sign detection algorithm based on improved yolov4. In: *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. [S.l.: s.n.], 2020. v. 9, p. 1946–1950. Citado na página 14.
- WU, X.; CAO, H. Traffic sign detection algorithm based on improved yolov4. *Journal of Physics: Conference Series*, IOP Publishing, v. 2258, n. 1, p. 012009, apr 2022. Disponível em: <https://dx.doi.org/10.1088/1742-6596/2258/1/012009>. Citado na página 17.

Apêndices

APÊNDICE A – Material Completo

Neste apêndice, disponibilizamos o link para o material completo desenvolvido ao longo do presente trabalho. O conteúdo abrange todos os códigos, dados, gráficos e informações detalhadas referentes ao tema em questão. O acesso ao material completo proporciona uma compreensão mais aprofundada do trabalho realizado, permitindo uma análise mais minuciosa dos resultados obtidos.

Para acessar o repositório com toda a solução desenvolvida neste trabalho, clique no seguinte link: https://github.com/ARRETdaniel/CARLA_simulator_YOLO-openCV_realTime_objectDetection_for_autonomousVehicles

Para acessar o material completo sobre a subseção **Configurando o Simulador Carla 5.1**, clique no seguinte link: https://github.com/ARRETdaniel/Self-Driving_Cars_Specialization/tree/main/CARLA%20Installation%20Guide

Recomendamos a exploração deste recurso para uma apreciação abrangente das etapas apresentadas ao longo do trabalho. O material está disponível online para facilitar o acesso e a referência contínua. Se houver problema ao tentar consultar qualquer um desses materiais, não hesite em nos contactar via e-mail: danielterra@pq.uenf.br.

Agradecemos a atenção e interesse na pesquisa apresentada, esperamos que o material disponibilizado enriqueça ainda mais a compreensão sobre o assunto abordado.

Anexos

ANEXO A – Material de Relevância

Neste anexo, fornecemos uma descrição dos materiais relevantes para aprofundamento relacionados a esta iniciação científica, e a sua contribuição.

Iniciamos com o artigo *Vehicle Dynamics COMPENDIUM* de 2020, publicado pela Chalmers University of Technology ([JACOBSON et al., 2020](#)). Este trabalho é essencial para compreender todos os aspectos relacionados à modelagem abordada nesta pesquisa. Ele oferece a base necessária para a compreensão da modelagem lateral, longitudinal e dos subsistemas dos VAs, incluindo aspectos avançados não abordados neste estudo.

Adicionalmente, temos a dissertação de mestrado *Sensing requirements for an automated vehicle for highway and rural environments*, de 2014, que aborda todos os sensores e métricas associados aos VAs, bem como análises desses sensores em diferentes contextos de aplicação ([BUSSEMAKER, 2014](#)).

Além disso, mencionamos o documento SAE *International J3016* de 2021, elaborado para descrever sistemas autônomos ([SAE, 2021](#)). Ele engloba todas as discussões e definições relevantes para caracterizar e definir os níveis de condução autônoma.

Ademais, o artigo *Automatic Steering Methods for Autonomous Automobile Path Tracking* de 2009 contribuiu de maneira significativa, auxiliando-nos na compreensão de algumas das equações presentes nos modelos apresentados neste trabalho de iniciação científica ([SNIDER et al., 2009](#)).

Por fim, desenvolvemos de maneira paralela a este trabalho sobre o **Modelo Cinemático da Bicicleta** os códigos e materiais referentes a implementação podem ser encontrados no seguinte link: https://github.com/ARRETdaniel/Self-Driving_Cars_Specialization/blob/main/Introduction%20to%20Self-Driving%20Cars/week4/module_4/Course_1_Module_4/Kinematic_Bicycle_Model.ipynb De mesma forma, implementamos sobre o **Modelo Longitudinal de Veículo**, acesso à implementação a partir do link:

https://github.com/ARRETdaniel/Self-Driving_Cars_Specialization/blob/main/Introduction%20to%20Self-Driving%20Cars/week4/module_4/Course_1_Module_4/Longitudinal_Vehicle_Model.ipynb