

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA

**UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
DARCY RIBEIRO**

*Centro CCT
Labotatório LCMAT*

Relatório do período: Junho 2023 - Março 2024

Relatório Anual PIBIC

Bolsista: Daniel Terra Gomes

Matrícula: 00119110484

Orientadora: Prof. 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: Dirigindo para o futuro: Softwares e Algoritmos avançados que alimentam Veículos 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

Dirigir um carro é geralmente considerado uma tarefa que os humanos podem realizar com relativa facilidade, devido ao seu domínio das habilidades de direção, conhecimento das regras de trânsito e capacidade de observar e responder prontamente às condições da estrada e situações especiais. Sabendo disso, este relatório de iniciação científica visa desenvolver sobre diferentes controladores veiculares destinados a alcançar a direção autônoma, e implementar um controlador para guiar um carro em ambiente de simulação virtual. O escopo abrange modelos de controle comumente utilizados na prática, assim como alguns conceitos teóricos encontrados de relevância para o entendimento do funcionamento dos controladores de carros autônomos. Visto que, esse trabalho visou a familiarização e compreensão dos principais *software* e algoritmos de controle aplicados em Veículos Autônomos (VAs). De modo a cumprir este objetivo, este trabalho explora a literatura, identificando modelos importantes de rastreamento de trajetória e algoritmos de controle a partir de artigos científicos e cursos participados. Além disso, implementa no simulador Carla os controladores PID e PI, e realiza uma comparação de seus desempenhos no controle longitudinal, enquanto emprega o controlador de Perseguição Pura para o controle lateral. Para alcançar isso, usamos o *framework* Scrum, e GitHub para o desenvolvimento dos códigos Python e versionamento do projeto. Este trabalho complementa a literatura com uma coleção abrangente de ideias importantes sobre controladores de VAs, e como um guia inicial para suas implementações. Este relatório não cataloga todo o trabalho em modelagem e controle de veículos; apenas uma seleção percebida como ideias importantes. De mesmo modo, não implementamos todos os controladores encontrados no procedimento bibliográfico; apenas os que julgamos com maior respaldo literário, facilitando, assim, sua implementação. Este estudo, em última instância, proporcionou uma familiarização e compreensão de diversos controladores aplicados em carros autônomos pela indústria automobilística.

Palavras-chave: Veículos autônomos. Modelagem Veicular. Controle Veicular. Simulação.

Abstract

Driving a car is generally considered a task that humans can perform relatively well, due to their mastery of driving skills, knowledge of traffic rules, and ability to observe and promptly respond to road conditions and special situations. With this in mind, this undergraduate research report aims to address different vehicle controllers aimed at achieving autonomous driving and implements a controller to guide a car in a virtual simulation environment. The scope covers control models commonly used in practice, as well as some theoretical concepts found to be relevant for understanding the functioning of autonomous car controllers. This work sought to familiarize and understand the main software and control algorithms applied to Autonomous Vehicles (AV). This work explores the literature to fulfill this goal, identifying important trajectory-tracking models and control algorithms from scientific articles and courses attended. Additionally, we implement PID and PI controllers in the Carla simulator and compare their performances for longitudinal control, while employing the Pure Pursuit controller for lateral control. To achieve this, we utilized the Scrum framework and GitHub for Python code development and project versioning. This work augments the literature with a comprehensive collection of major ideas about AV controllers and as an initial guide for their implementations. This report does not catalog all work in vehicle modeling and control; just a selection perceived as relevant ideas. Likewise, we did not implement all the controllers found in the bibliographic procedure; only those judged to have notable literary support, thus facilitating their implementation. This study ultimately provided familiarization and understanding of various controllers applied to autonomous cars by the automobile industry.

Keywords: Autonomous vehicles. Vehicle Modeling. Vehicle Control. Simulation.

Lista de ilustrações

Figura 1 – Sistema Massa-Mola-Amortecedor (University of Toronto, 2018, Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 9min00s).	23
Figura 2 – Controlador P (University of Toronto, 2018, Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min13s).	25
Figura 3 – Controlador PD (University of Toronto, 2018, Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min13s).	25
Figura 4 – Controlador PI (University of Toronto, 2018, Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min13s).	26
Figura 5 – Controlador PID (University of Toronto, 2018, Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min50s).	26
Figura 6 – Arquitetura da Estratégia de Controle de Veículos (University of Toronto, 2018, Week 5 - Lesson 2: Longitudinal Speed Control with PID. 2min00s).	27
Figura 7 – Controle de velocidade longitudinal (University of Toronto, 2018, Week 5 - Lesson 2: Longitudinal Speed Control with PID. 4min13s).	28
Figura 8 – Mapa de motor (University of Toronto, 2018, Week 5 - Lesson 2: Longitudinal Speed Control with PID. 7min09s).	30
Figura 9 – Diagrama da Simulação (University of Toronto, 2018, Week 5 - Lesson 2: Longitudinal Speed Control with PID. 7min58s).	31
Figura 10 – Exemplo da Simulação (University of Toronto, 2018, Week 5 - Lesson 2: Longitudinal Speed Control with PID. 7min58s).	31
Figura 11 – Controle Combinado de Feedforward e Feedback (University of Toronto, 2018, Week 5 - Lesson 3: Feedforward Speed Control. 2min00s).	32
Figura 12 – Controle de velocidade do veículo (University of Toronto, 2018, Week 5 - Lesson 3: Feedforward Speed Control. 3min28s).	33
Figura 13 – Resultados de simulação feedforward (University of Toronto, 2018, Week 5 - Lesson 3: Feedforward Speed Control. 6min20s).	35
Figura 14 – Segmentos de linha reta (University of Toronto, 2018, Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 2min52s).	37
Figura 15 – Pontos de referência (University of Toronto, 2018, Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 3min00s).	37
Figura 16 – Curvas parametrizadas (University of Toronto, 2018, Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 3min30s).	37
Figura 17 – Trajetória de referência (University of Toronto, 2018, Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 6min23s).	38
Figura 18 – Erro de trajetória cruzada (University of Toronto, 2018, Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 8min31s).	39

Figura 19 – Rastreamento de caminho geométrico (University of Toronto, 2018, Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 2min27s).	41
Figura 20 – Pure pursuit (University of Toronto, 2018, Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 3min15s).	41
Figura 21 – Pure pursuit ICR (University of Toronto, 2018, Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 3min52s).	42
Figura 22 – Erro Crosstrack (University of Toronto, 2018, Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 3min52s).	43
Figura 23 – Simulador Carla (CARLA, 2018). Diferentes condições ambientais e Mapas (Capturas de tela pelos autores).	46
Figura 24 – Trecho do circuito (CARLA, 2018).	49
Figura 25 – Estrutura do projeto (Elaborado pelos autores).	50
Figura 26 – Limites do Perfil de velocidade (Elaborado pelos autores).	52
Figura 27 – Pontos de referência (Elaborado pelos autores).	52
Figura 28 – Captura de Tela da execução (PID e Pure Pursuit) da simulação e controle do veículo (Elaborado pelos autores).	57
Figura 29 – Perfil de velocidade alcançado usando o PID e Pure Pursuit (Elaborado pelos autores).	58
Figura 30 – Trajetória alcançada usando o PID e Pure Pursuit (Elaborado pelos autores).	58
Figura 31 – Captura de Tela da execução (PI e Pure Pursuit) da simulação e controle do veículo (Elaborado pelos autores).	59
Figura 32 – Perfil de velocidade alcançado usando o PI e Pure Pursuit (Elaborado pelos autores).	59
Figura 33 – Trajetória alcançada usando o PI e Pure Pursuit (Elaborado pelos autores).	60
Figura 34 – Comparação dos sobressaltos do Perfil de Velocidade PID e PI (Elaborado pelos autores).	60
Figura 35 – Certificado de conclusão do curso Introduction to Self-Driving Cars.	63

Lista de tabelas

Tabela 1 – Características dos ganhos P, I e D (University of Toronto, 2018, Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 8min11s). 23

Lista de trechos de código

4.1	class Controller2D: variáveis e funções (Elaborado pelos autores).	50
4.2	Implementação dos controladores PID e Pure Pursuit (Elaborado pelos autores).	53
4.3	Alteração para o Controlador PI (Elaborado pelos autores).	58

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

Introdução	12
1 ETAPAS PROPOSTAS NO PLANO DE TRABALHO	15
2 OBJETIVOS	16
3 PROCEDIMENTOS METODOLÓGICOS	17
4 RESULTADOS E DISCUSSÃO	19
4.1 Introdução ao Controle Longitudinal de Veículos	19
4.1.1 Controle de Velocidade Longitudinal com PID	27
4.1.2 Controle de Velocidade Antecipado	32
4.2 Introdução ao Controle Lateral de Veículos	35
4.2.1 Controle Lateral Geométrico - Controlador de Perseguição Pura	40
4.3 Controle de Veículos Autônomos	44
4.3.1 Ambientes de simulação de carro autônomo	45
4.3.1.1 Configurando o Simulador Carla	47
4.3.2 Implementação Controle de Veículos Autônomos	49
5 CONSIDERAÇÕES FINAIS	61
6 PERSPECTIVA DE CONTINUIDADE	62
7 PARTICIPAÇÃO EM CONGRESSOS E TRABALHOS PUBLICADOS OU SUBMETIDOS E OUTRAS ATIVIDADES ACADÊMICAS E DE PESQUISA	63
8 DATAS E ASSINATURAS	64
8.1 Data e assinatura do bolsista (assinatura digitalizada)	64
8.2 Data e assinatura do orientador (assinatura digitalizada)	64
REFERÊNCIAS	65
APÊNDICES	68
APÊNDICE A – MATERIAL COMPLETO	69

ANEXOS**70****ANEXO A – MATERIAL DE RELEVÂNCIA 71**

Introdução

O mercado de veículos elétricos vem em uma crescente (SEBO, 2024), possibilitando a aplicação de direção autônoma por muitas empresas que trabalhavam apenas com pesquisas em VAs. Muitas dessas empresas ao redor do mundo, como *ZOOX®* e *WAYMO®*, estão investindo milhões de dólares no desenvolvimento de carros autônomos (BRATZEL; CAM, 2022). Em comparação com carros tradicionais controlados exclusivamente por motoristas humanos e carros com assistências básicas passivas, os carros autônomos apresentam as seguintes vantagens: a quantidade de acidentes de carro poderia ser drasticamente reduzida, salvando milhares de vidas (OTHMAN, 2021); os congestionamentos de tráfego podem ser significativamente reduzidos e a eficiência energética pode ser grandemente aprimorada (KPMG International, 2020); as pessoas podem aproveitar seu tempo de deslocamento em atividades mais valiosas (NEUFVILLE, 2022); é mais conveniente para idosos, crianças e pessoas com deficiência utilizarem VAs (LAGE, 2019). Nesse contexto, propomos uma familiarização com esse setor tecnológico promissor. A partir da exploração dos tópicos relacionados a direção autônoma. Para alcançar a direção autônoma é necessário estudar e resolver os problemas relacionados ao controle autônomo do veículo. Nesse sentido, o artigo (TAXONOMY..., 2021) propõem uma definição dos diferentes níveis de direção autônoma e como cada nível contribui para o controle do veículo. Os artigos (ZHENG et al., 2023), (IGNATIOUS; HESHAM-EL-SAYED; KHAN, 2022), (SINGH, 2022), (YAO et al., 2024), (MAO et al., 2023), (REINHOLTZ et al., 2007) e o material (University of Toronto, 2018) apresentam como a percepção do ambiente é realizada, e quais componentes estão envolvidos nessa tarefa, que comina na tomada de decisão do controlador do VA. De modo a processar todas essas informações e tomar decisões assertivas, os VAs necessitam de componentes de *hardwares* poderosos. Para tal, o artigo (MOCHURAD; MAMCHUR, 2023), e o material (University of Toronto, 2018) apresentam os componentes físicos envolvidos nessa dinâmica e quais estratégias são tomadas para facilitar o processamento de um volume massivo de dados oriundo dos sensores. Esses dados são usados para diversas tarefas, como planejamento de movimento, como desdobra os artigos (PADEN et al., 2016), e o material (University of Toronto, 2018). Assim, o planejamento do veículo chega ao componente de controle do veículo. Onde, para compreendermos o seu funcionamento, seguimos os autores (PADEN et al., 2016), (University of Toronto, 2018), (FRANCIS et al., 2016), (JACOBSON et al., 2020), (JACOBSON et al., 2016), (RAJAMANI, 2011), e (SNIDER et al., 2009) que compreendem essa tarefa, como governadas pela Modelagem Cinemática e/ou Dinâmica, das quais englobam o controle longitudinal e lateral do veículo. Finalmente, nesse âmbito dos controladores longitudinais e laterais, o autor (University of Toronto, 2018) fundamentou toda nossa parte matemática e teórica para que pudés-

semos compreender e desenvolver controladores. Sendo esses os autores e materiais que contribuíram para essa iniciação científica. O qual teve como objetivo: **A familiarização e compreensão dos principais softwares e algoritmos de controle de VAs.** Para cumprir esse objetivo, os seguintes aspectos precisaram ser abordados:

1. Investigação da literatura existente, o que foi feito principalmente pela investigação de projetos existentes de pesquisas em VAs. Verificou-se que apenas informações limitadas estão disponíveis. A pesquisa é conduzida principalmente por grandes empresas automotivas e, como o setor automotivo é um mercado altamente competitivo, grande parte das informações são mantidas internamente. Devido a isso, nos direcionamos a conteúdos educacionais disponibilizados por empresas e universidades que trabalham no desenvolvimento de carros autônomos.
2. Identificação de diferentes tipos de *softwares* e algoritmos aplicados no controle de carros autônomos. No desenrolar das investigações do levantamento bibliográfico, foi constatado que os algoritmos implementados nos VAs são oriundos de uma área conhecida como Modelagem Veicular ([JACOBSON et al., 2020](#)). Devido a isso, para que pudéssemos compreender como os algoritmos e software que controlam os VAs funcionam, precisaríamos compreender sobre essa área, e como ela estabelece os algoritmos e *software* que controlam os VAs.
3. Investigação sobre Modelagem Veicular, aplicada em VAs. Para cada modelagem encontrada, apresentamos o seu embasamento teórico de maneira progressiva, a fim de obter uma compreensão mais aprofundada de como determinadas modelagens podem ser aplicadas no controle de carros autônomos.
4. Implementação das modelagens de controle, encontradas, em um *software* de simulação destinado ao teste de carros autônomos. Para tal, implementamos um controlador usando os modelos encontrados, permitindo que o nosso carro precorresse um trecho de uma pista do ambiente de simulação virtual.

Os três primeiros itens apresentados acima, foram investigados como parte da pesquisa exploratória antes da elaboração deste relatório. A maioria destes estudos estão incorporados neste relatório de iniciação científica para fornecer um quadro completo e garantir que todas as informações estejam em um só lugar. Neste trabalho, nos concentraremos principalmente na identificação e compreensão dos diferentes controladores que podem fazer parte de uma estratégia de controle de um carro autônomo. Também implementamos o controle necessário para guiar um carro autônomo em ambiente de simulação virtual, usando a tecnologia atual disponível no mercado. Essas informações e implementação podem ser usadas para selecionar e verificar um conjunto de componentes e modelos, de modo a fazer possíveis comparações de desempenhos específicos no cenário de condução autônoma.

O relatório está estruturado da seguinte forma: o capítulo [1](#) apresenta as etapas propostas no *plano de trabalho*, enquanto o capítulo [2](#) mostra a principal motivação e objetivos deste estudo. O capítulo [3](#) apresenta a metodologia utilizada para o desenvolvimento do capítulo [4](#), no qual apresenta uma introdução aos controles de VAs utilizados para desenvolver a implementação dos conteúdos abordados ao longo do trabalho. O capítulo [5](#) demonstra os cumprimentos do *plano de trabalho* referente a este relatório, enquanto o capítulo [6](#) apresenta a perspectiva de continuidade dos estudos do bolsista, de modo a aprimorar seus conhecimentos e aprofundar em partes essenciais em relação a esse estudo. O capítulo [7](#) apresenta a participação em eventos, cursos e atividades do bolsista de relevância para essa iniciação científica. Em seguida, o capítulo [8](#) apresenta a assinatura do bolsista e da sua queridíssima orientadora. Finalmente, os capítulos ([Apêndice A](#)) e ([Anexo A](#)) disponibilizam conteúdos de relevância e adicionais, referentes ao desenvolvimento desta iniciação científica.

Uma lista das principais siglas usadas ao longo do relatório é fornecida na Página [9](#).

1 Etapas propostas no Plano de Trabalho

As etapas do *Plano de Trabalho* apresentado em 2023 teve em vista direcionar as atividades de pesquisa de modo que a cada passo fosse possível formar uma base sólida de entendimento da área de pesquisa. A fim de alcançar os objetivos do Projeto de Pesquisa as etapas a seguir foram estipuladas:

1. Pesquisa bibliográfica sobre softwares de controle para veículos autônomos;
2. Levantamento bibliográfico sobre algoritmos de controle de Veículos Autônomos;
3. Seleção dos principais algoritmos achados no levantamento bibliográfico;
4. Teste computacional de alguns dos principais algoritmos achados;
5. Elaboração do Relatório Final.

No decorrer desse segundo ano de pesquisas foi possível desenvolver, de forma satisfatória, todas as etapas propostas no plano de trabalho.

2 Objetivos

Neste segundo ano de pesquisa (03/2023 - 03/2024), nosso objetivo principal foi o levantamento bibliográfico, a familiarização e a compreensão dos principais softwares e algoritmos usados para o controle de Veículos Autônomos (VAs), publicados na literatura científica.

3 Procedimentos Metodológicos

Conduzimos um procedimento de levantamento bibliográfico, visando explorar o tema de controladores de VAs. Baseados nisso, aplicamos os conhecimentos obtidos durante a pesquisa exploratória para desenvolver um controlador para guiar um carro em ambiente de simulação virtual, e comparar o desempenho de dois controladores.

Para tal, o levantamento bibliográfico da nossa 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 2022 a 2023. 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 de 5 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: *Modelagem Veicular, Cinemática, Dinâmica, Controladores, Simulação, etc.*

Dessa forma, chegamos nos conteúdos presentes nesse trabalho, que foi elaborado, principalmente, por artigos científicos, e um curso de Especialização em carros autônomos da Universidade de Toronto publicado em 2018 na plataforma de cursos online Coursera, visto na Figura 35. O curso teve duração de aproximadamente 35 horas 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 os controladores 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 às aulas ministradas, e concluímos todas as atividades com 100% de aproveitamento. Esse ato aumentou o tempo aproximado de duração do curso em mais de 10 vezes. O certificado de conclusão pode ser encontrado no capítulo 7.

Sabendo disso, o desenvolvimento do controlador foi realizado da seguinte maneira:

Implementamos um controlador (apresentado na seção 4.3) para percorrer o trecho

sudeste da pista *RaceTrack* do simulador de sistemas de direção Carla, usando os conhecimentos do controlador PID para o controle longitudinal (apresentado na subseção 4.1) e do controlador de Perseguição Pura para o controle lateral (apresentado na subseção 4.2.1). Ademais, realizamos uma comparação de desempenho entre os controladores longitudinais PID e PI. De modo a verificar se ocorre a variação esperada de sobressalto entre as duas implementações, como descrito na literatura e apresentado na subseção 4.1. Optamos por esses controladores devido à sua ampla documentação na literatura, facilitando a implementação.

De modo a concluir a tarefa de direção, o veículo percorre o trajeto definido por meio de um conjunto de pontos de referência (visto na Figura 15) e segue um perfil de velocidade específico (visto na Figura 26). Os controladores 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 os controladores, usamos a abordagem metodológica Ágil, onde usamos o *framework* Scrum para otimizar o desenvolvimento do nosso controlador por meio de processos iterativos e incrementais. Escolhemos a metodologia Ágil para essa etapa devido a sua adaptabilidade, agilidade e eficiência. Onde, 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.

4 Resultados e Discussão

Os resultados deste segundo ano de pesquisa são desenvolvidos nas próximas seções e subseções deste capítulo 4. No qual visa satisfazer o *objetivo* definido no Capítulo 2. Desse modo, desenvolveremos o objetivo proposto, elaborando e apresentando os resultados e o que significam em relação ao trabalho realizado neste ano.

4.1 Introdução ao Controle Longitudinal de Veículos

No âmbito da condução autônoma, o controle efetivo da dinâmica longitudinal de um veículo é crucial. Esta seção explora os fundamentos do controle longitudinal de veículos, com foco específico no controlador PID (Proporcional-Integral-Derivativo). O objetivo é proporcionar uma compreensão abrangente de como os VAs regulam sua velocidade, um aspecto crítico que influencia o desempenho geral do veículo e a segurança nas estradas.

Será apresentado um guia sobre sistemas de controle lineares clássicos e invariantes no tempo, abrangendo conceitos vitais como funções de transferência e análise no domínio de Laplace. O objetivo é garantir um entendimento sólido dos fundamentos teóricos antes de adentrar na aplicação prática dos controladores PID, vistos na seção 4.3. Devido a isso, veremos sobre os conceitos centrais dos fundamentos dos controladores. Isso inclui uma exploração de circuitos de realimentação (*feedback* no inglês) e como os controladores interpretam dados sensoriais para gerar sinais de atuação. A representação de sistemas por meio de funções de transferência será discutida em detalhes, esclarecendo as nuances de sistemas lineares e não lineares. A importância de zeros e pólos em funções de transferência será elucidada para fornecer uma compreensão abrangente do comportamento do sistema.

O foco será então direcionado para o controlador PID, um pilar na teoria de controle. A formulação matemática do controlador PID, compreendendo termos proporcionais, integrais e derivativos, será dissecada. Os papéis dos ganhos proporcional, integral e derivativo (K_p , K_i e K_d) serão explicados, preparando o terreno para uma discussão aprofundada sobre métodos de ajuste e otimização.

Para ilustrar a aplicação prática dos controladores PID, será utilizado um modelo de amortecedor de massa-mola de segunda ordem como estudo de caso, elaborado pelo autor (University of Toronto, 2018). Será realizada uma análise do sistema, incluindo a derivação da função de transferência do sistema. Através da análise da resposta ao degrau, será demonstrada a influência do controle PID no comportamento do sistema, abrangendo parâmetros como tempo de subida, *overshoot* ou “sobressinal ou sobressalto” e tempo de

estabilização.

Em conclusão, esta seção visa proporcionar uma visão abrangente do controle longitudinal de veículos, desde as bases teóricas até aplicações práticas utilizando controladores PID. As subseções subsequentes explorarão ainda mais a aplicação do controle PID no contexto da regulação de velocidade de carros autônomos.

Utilizaremos o referencial teórico incorporado na introdução deste relatório referente a modelagem para relacionar o desenvolvimento de modelos dinâmico e cinemático para um veículo com base no modelo de bicicleta. Esses modelos visam capturar como o sistema dinâmico reage a comandos de entrada do motorista, como direção, aceleração e freio, e como reage a perturbações, como vento, superfície da estrada e diferentes cargas no veículo. Os efeitos dos comandos de entrada e das perturbações sobre os estados, como velocidade e taxa de rotação do veículo, são definidos pelos modelos cinemáticos e dinâmicos. Sabendo disso, o papel do controlador, então, é regular alguns desses estados do veículo, detectando as variáveis de estado atuais e gerando sinais de atuadores para atender aos comandos fornecidos ([University of Toronto, 2018](#), Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control). Neste caso, no controle longitudinal, o controlador detecta a velocidade do veículo e ajusta os comandos do acelerador e do freio para corresponder à velocidade desejada definida pelo sistema autônomo de planejamento de movimento ([University of Toronto, 2018](#), Week 2 - Lesson 3: Software Architecture. 6min24s).

Dessa forma, segundo o autor ([University of Toronto, 2018](#)) em um típico ciclo de controle de *feedback*, o modelo de processo recebe os sinais do atuador como entrada e gera as variáveis de saída ou de estado do sistema. Estas saídas são medidas por sensores, e estimadores são utilizados para fundir as medições em estimativas precisas de saída. As estimativas de saída são então comparadas com as variáveis de saída desejadas ou de referência, e a diferença ou erro é encaminhada para o controlador. O controlador pode ser entendido como um algoritmo matemático que gera sinais para o atuador para minimizar o sinal de erro, aproximando as variáveis de estado do modelo de processo das variáveis de estado desejadas. O autor ([University of Toronto, 2018](#)), salienta que o modelo de processo, seja ele linear ou não linear, pode ser representado de várias maneiras. Duas das formas mais comuns são a forma de espaço de estados, que acompanha a evolução de um estado interno para conectar a entrada à saída, e a forma de função de transferência, onde o sistema deve ser linear e invariante no tempo, que modela diretamente a relação entre entrada e saída. Segundo o autor ([University of Toronto, 2018](#)), a função de transferência é uma expressão que relaciona as entradas (U) e as saídas (Y) de um sistema. Essa relação é definida no domínio de Laplace como uma função de S , uma variável complexa utilizada para transitar do domínio do tempo para o domínio de S . Isso permite uma análise mais simplificada da relação entrada-saída, como demonstrado nas equações 4.1 e 4.2 a seguir:

$$Y(s) = G(s)U(s) \quad (4.1)$$

Onde: $Y(s)$ = A saída no domínio de Laplace;
 $G(s)$ = A função de transferência;
 $U(s)$ = A entrada no domínio de Laplace.

$$s = \sigma + j\omega \quad (4.2)$$

Onde: s = Variável de frequência complexa;
 σ = Parte real da frequência complexa;
 j = Unidade imaginária;
 ω = Frequência angular.

Conforme o autor ([University of Toronto, 2018](#)) apresenta, no contexto da análise de funções de transferência, as raízes do numerador e do denominador oferecem uma compreensão significativa sobre a resposta de um sistema às funções de entrada. Os zeros de um sistema correspondem às raízes do numerador, enquanto os pólos do sistema são as raízes do denominador, e podemos identificar isso na equação 4.3 a seguir:

$$Y(s) = G(s)U(s) = \frac{N(s)}{D(s)}U(s) \quad (4.3)$$

Onde: $Y(s)$ = Resposta de saída no domínio de Laplace;
 $G(s)$ = Função de transferência;
 $U(s)$ = Sinal de entrada no domínio de Laplace;
 $N(s)$ = Numerador da função de transferência, relacionado aos zeros;
 $D(s)$ = Denominador da função de transferência, relacionado aos pólos.

Esses sistemas de resposta para entradas e saídas, segundo o autor ([University of Toronto, 2018](#)) podem apresentar variações, desde abordagens simples, como multiplicação por ganho constante, tabelas de consulta e equações lineares, até métodos mais detalhados fundamentados em funções não lineares e otimização ao longo de horizontes de previsão finitos. Entre os controladores básicos e clássicos, destacam-se os controladores “lead-lag” e o controlador Proporcional Integral Derivativo, conhecidos como PID. O controle PID é expresso matematicamente pela adição de três termos dependentes da função de erro (K_p , K_i e K_d). Um termo proporcional diretamente relacionado ao erro, um termo integral proporcional à integral do erro e um termo derivativo proporcional à derivada do erro, conforme a equação 4.4 do controlador PID que é denotado $u(t)$.

$$u(t) = K_p e(t) + K_I \int_0^t e(t) dt + K_d \dot{e}(t) \quad (4.4)$$

Onde: $u(t)$ = A saída do controlador;
 K_p = Ganho proporcional;
 $e(t)$ = Erro, a diferença entre a entrada e a saída desejada;
 K_i = Ganho integral;
 $\int_0^t e(t)dt$ = Integral do erro ao longo do tempo;
 K_d = Ganho derivativo;
 $\dot{e}(t)$ = Taxa de variação do erro.

Segundo o autor (University of Toronto, 2018), ao realizar a transformada de Laplace do controle PID 4.4, obtém-se a função de transferência $G_c(s)$. A multiplicação por s no domínio de Laplace é equivalente a realizar uma derivada no domínio do tempo, enquanto a divisão por s é equivalente a realizar uma integral. Ao somar esses três termos do controlador PID, obtemos uma única função de transferência para o controle PID, conforme a equação 4.5.

$$U(s) = G_c(s)E(s) = \left(K_p + \frac{K_I}{s} + K_D s \right) E(s) = \frac{(K_D s^2 + K_p s + K_I)}{s} E(s) \quad (4.5)$$

Onde: $U(s)$ = A função de transferência do controlador;
 $G_c(s)$ = Ganho do controlador;
 $E(s)$ = Erro no domínio de Laplace;
 K_p = Ganho proporcional;
 K_I = Ganho integral;
 K_D = Ganho derivativo;
 s = Variável complexa no domínio de Laplace.

Segundo o autor (University of Toronto, 2018), a função de transferência do controlador PID apresenta um pólo simples na origem proveniente do termo integral. Além disso, ela inclui um numerador de segunda ordem com dois zeros que podem ser posicionados em qualquer lugar do plano complexo ao selecionar valores apropriados para os ganhos. Assim, o projeto de controle PID resume-se na escolha das posições dos zeros para alcançar a saída desejada ou desempenho com base no modelo, conforme podemos observar na equação 4.6.

$$G_c(s) = \frac{K_D s^2 + K_p s + K_I}{s} \quad (4.6)$$

Onde: $G_c(s)$ = Função de transferência do controlador;

$K_d s^2$ = Termo proporcional com dois zeros adicionados;

$K_p s$ = Termo integral;

K_i = Termo derivativo;

s = Variável de Laplace com um polo adicionado.

Os efeitos de cada ação Proporcional (P), Integral (I) e Derivativa (D) estão resumidos na Tabela 1.

Resposta Fechada	T. de Subida	Sobressalto	T. de Acomodação	Erro de Estado Estacionário
Aumentar K_p	Diminuir	Aumentar	Peq. mudança	Diminuir
Aumentar K_i	Diminuir	Aumentar	Aumentar	Eliminar
Aumentar K_d	Peq. mudança	Diminuir	Diminuir	Pequena mudança

Tabela 1 – Características dos ganhos P, I e D ([University of Toronto, 2018](#), Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 8min11s).

Em última instância, os ganhos PID devem ser selecionados com conhecimento da interação de seus efeitos, a fim de ajustar a resposta do sistema para obter o desempenho adequado. Os conhecimentos relacionados ao PID serão necessários com o avanço deste relatório.

Dessa forma, elaboraremos um exemplo proposto pelo autor ([University of Toronto, 2018](#)), onde primeiro revisaremos a função de transferência do sistema dinâmico proposto e então projetaremos um controlador PID o modelo de amortecedor de massa de mola, visto na Figura 1.

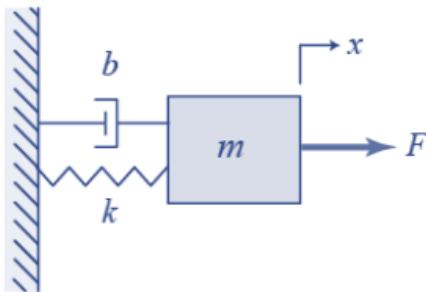


Figura 1 – Sistema Massa-Mola-Amortecedor ([University of Toronto, 2018](#), Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 9min00s).

O sistema é submetido à força de entrada F , e a saída do modelo é o deslocamento do corpo x . A massa M está conectada a uma fundação rígida por uma mola com constante elástica K e um amortecedor com coeficiente de amortecimento b , conforme a equação 4.7.

$$m\ddot{x} + b\dot{x} + kx = F, \quad x(0) = 0 \quad (4.7)$$

Onde: m = Massa do objeto;
 \ddot{x} = Aceleração do objeto;
 b = Coeficiente de amortecimento;
 \dot{x} = velocidade do objeto;
 k = Coeficiente de rigidez;
 x = Deslocamento do objeto;
 F = Força externa aplicada ao sistema;
 $x(0) = 0$ = Condição inicial que indica que no tempo $t=0$, o deslocamento é zero.

Segundo o autor (University of Toronto, 2018) para transformarmos a equação para o domínio S ou domínio de Laplace, precisamos utilizar a transformada de Laplace e expressar a equação de segundo grau da seguinte forma, conforme visto na equação 4.8.

$$ms^2X(s) + bsX(s) + kX(s) = F(s) \quad (4.8)$$

Onde: $ms^2X(s)$ = massa vezes aceleração no domínio s;
 $bsX(s)$ = coeficiente de amortecimento vezes velocidade no domínio s;
 $kX(s)$ = constante da mola vezes deslocamento no domínio s;
 $F(s)$ = força externa no domínio s.

Por fim, a função de transferência é estabelecida, representando a relação entre a saída $x(s)$ e a entrada $F(s)$, sendo definida como a função de transferência da planta, ou “Plant” no inglês, $G(s)$, conforme visto na equação 4.9. Esse conceito de planta é elaborado a partir da Figura 11.

$$G(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k} \quad (4.9)$$

Onde: $G(s)$ = Função de transferência de saída para entrada;
 $X(s)$ = Transformada de Laplace da saída;
 $F(s)$ = Transformada de Laplace da entrada;
 m = Massa ou coeficiente do termo de segunda ordem;
 b = Amortecimento ou coeficiente do termo de primeira ordem;
 k = Constante da mola ou termo de ordem zero.

Segundo o autor (University of Toronto, 2018) precisamos usar uma entrada de degrau unitário para avaliar as características do sistema. Normalmente, esse é o primeiro passo para avaliar as características dinâmicas de uma planta.

Desse modo, analisaremos a resposta ao degrau de alguns controladores PID diferentes, disponibilizados pelo autor (University of Toronto, 2018). A linha horizontal

tracejada representa a referência ou saída desejada, e o objetivo dos controladores é manter a saída real próxima a essa referência.

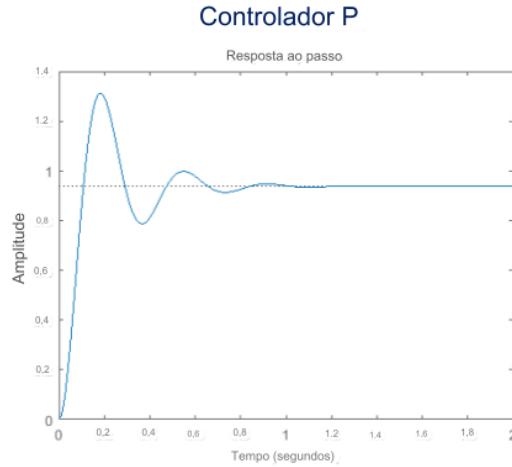


Figura 2 – Controlador P ([University of Toronto, 2018](#), Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min13s).

No primeiro exemplo ($G_P(s) = K_P$), visto no gráfico da Figura 2, são apresentadas as respostas ao degrau para o controle proporcional puro do sistema massa-mola-amortecedor. Na resposta do controlador P, observamos um tempo de subida rápido, um sobressalto significativo e uma oscilação prolongada, resultando em um longo tempo de acomodação.

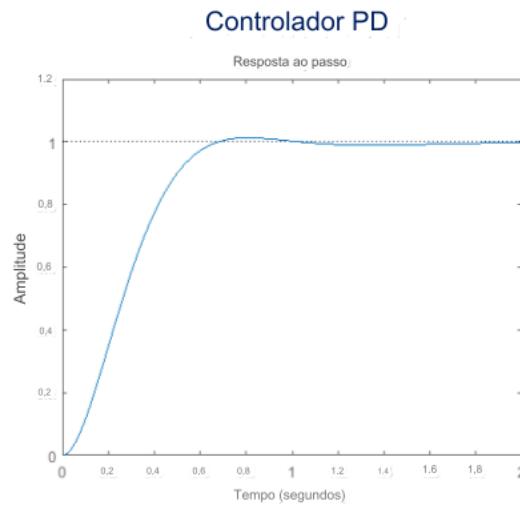


Figura 3 – Controlador PD ([University of Toronto, 2018](#), Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min13s).

A inclusão do controle derivativo ($G_{PD}(s) = K_P + sK_D$), visto no gráfico da Figura 3, melhora a resposta ao degrau em termos de sobressalto e tempo de acomodação, mas diminui o tempo de subida.

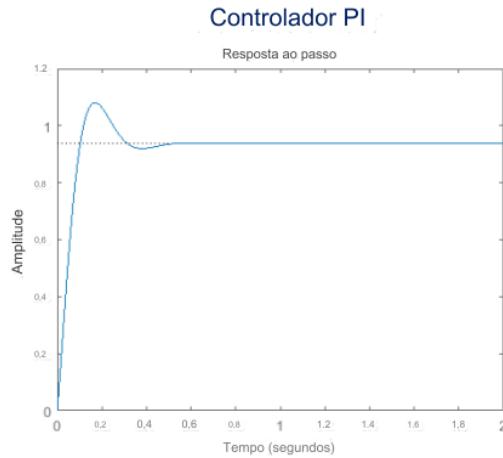


Figura 4 – Controlador PI ([University of Toronto, 2018](#), Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min13s).

A adição do termo integral ($G_{PI}(s) = K_P + \frac{K_I}{s}$), visto no gráfico da Figura 4, por outro lado, mantém um tempo de subida curto e consegue reduzir oscilações e sobressalto, resultando em um tempo de acomodação rápido também.

Incorporar todos os três termos PID no controlador proporciona ainda mais flexibilidade na criação da resposta ao degrau. Através da afinação cuidadosa dos ganhos do controlador, é possível aproveitar os benefícios de todos os três termos para eliminar sobressaltos e ainda manter tempos de subida e de estabilização muito curtos.

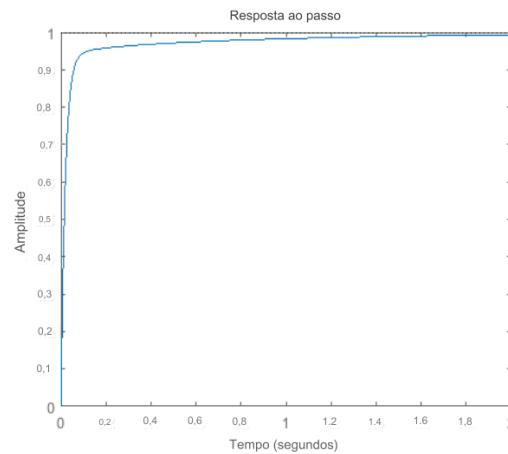


Figura 5 – Controlador PID ([University of Toronto, 2018](#), Week 5 - Lesson 1: Proportional-Integral-Derivative (PID) Control. 12min50s).

Conforme evidenciado no gráfico da Figura 5, o sistema se aproxima rapidamente da referência, sem apresentar sobressalto, quando submetido ao controle PID, dado pela equação do controle 4.10 e o sistema de circuito fechado torna-se 4.11

$$G_{PID}(s) = \left(K_p + K_d s + \frac{K_I}{s} \right) \quad (4.10)$$

$$G_{CL}(s) = \frac{K_D s^2 + K_P s + K_I}{s^3 + (10 + K_D)s^2 + (20 + K_P)s + K_I} \quad (4.11)$$

4.1.1 Controle de Velocidade Longitudinal com PID

Esta subseção aborda a aplicação crucial do controle PID e mapas do motor para a regulamentação da velocidade de veículos em sistemas autônomos, com foco específico no controle de cruzeiro. Iniciando com a arquitetura de controle do veículo, explorando suas seções interconectadas: percepção, geração de trajetória e perfil de velocidade, controladores e atuadores. Definindo os pontos de ajuste para aceleração e desaceleração, estabelecendo as bases para o rastreamento preciso de velocidade e trajetória. Apresentaremos, também, sistemas de controle de cruzeiro, desde a manutenção básica de velocidade até funcionalidades avançadas como controle de cruzeiro adaptativo, ilustrando o contínuo aprimoramento na tecnologia de direção autônoma.

A exploração então se volta para controladores de alto nível, enfatizando seu papel na determinação da aceleração necessária com base no erro de velocidade por meio do controle PID. Considerações práticas na implementação de software, incluindo discretização e tratamento de termos derivativos, são componentes cruciais.

Por fim, apresentamos uma simulação do autor ([University of Toronto, 2018](#)) que ilustra as implicações práticas do controle PID e da utilização de mapas do motor.

Dessa forma, analisaremos mais de perto a arquitetura de controle do veículo e como ela se insere no conjunto geral de software de autonomia dos VAs.

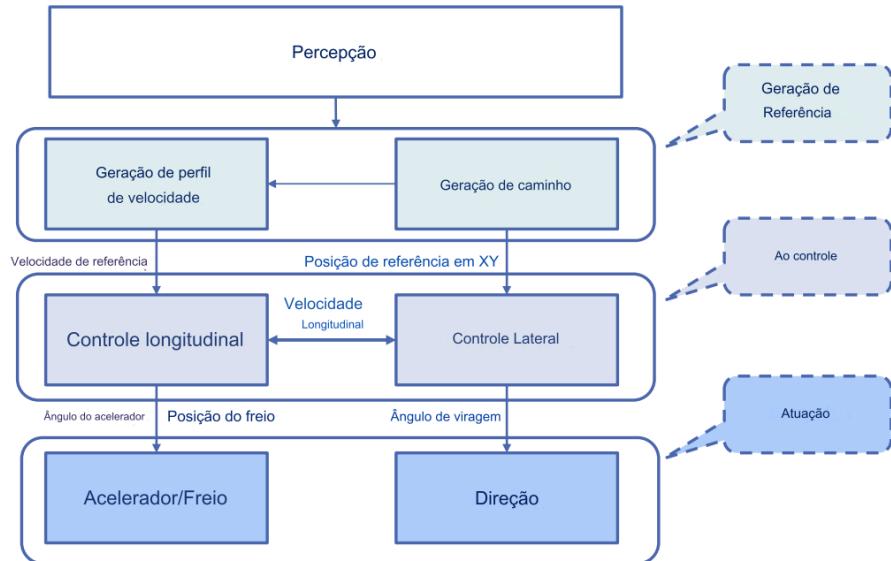


Figura 6 – Arquitetura da Estratégia de Controle de Veículos ([University of Toronto, 2018](#), Week 5 - Lesson 2: Longitudinal Speed Control with PID. 2min00s).

Segundo o autor ([University of Toronto, 2018](#)), a estrutura do sistema pode ser

dividida em quatro seções interconectadas, e vistas da Figura 6. A primeira seção aborda a percepção da via e do ambiente, sendo essa percepção capturada por sensores que geram referências de entrada para o sistema (IGNATIOUS; HESHAM-EL-SAYED; KHAN, 2022, p. 737). Na segunda camada, encontra-se a geração do trajeto e do perfil de velocidade, conhecido no âmbito automotivo como ciclo de direção. Esses perfis são criados por meio do processo de planejamento de movimento. Tanto para o controle longitudinal quanto para o lateral de um veículo autônomo, a principal tarefa consiste então em seguir o plano de maneira precisa, minimizando assim o erro entre a trajetória e velocidade reais e de referência. Os controladores, por fim, geram comandos de entrada ou sinais de atuadores para o veículo, incluindo a direção para o controle lateral, e comandos de aceleração e frenagem para o controle longitudinal, conforme discutido pelo autor (JACOBSON et al., 2020).

Baseados nisso, analisaremos um exemplo de controle longitudinal de veículos. Segundo o autor (University of Toronto, 2018), uma das aplicações de controle mais conhecidas e amplamente disponíveis no contexto do controle automotivo é o sistema de controle de cruzeiro operando no controle de velocidades na estrada. O sistema de controle de cruzeiro desempenha a função de manter uma velocidade de referência fixa por meio de comandos do acelerador, acelerando ou desacelerando para uma nova velocidade de referência conforme solicitado pelo motorista. Quando o veículo é submetido a diferentes cargas e resistências, o ângulo do acelerador será ajustado pelo controlador de cruzeiro de modo a manter a velocidade desejada. Nesse cenário, o sistema automotivo que se destaca é o controle de cruzeiro adaptativo, capaz de ajustar a velocidade de referência com base nas medições de um veículo à frente (BUSSEMAKER, 2014, p. 8), e os sistemas semi autônomos, por exemplo, o assistente de congestionamento de tráfego. Sendo esse capaz de operar em toda a faixa de velocidade do veículo e criar espaços para a entrada de veículos à frente na mesma faixa da lista. No entanto, o autor (University of Toronto, 2018) salienta que para tais exemplos é demandada a implementação de controladores adicionais, projetados para lidar com uma gama mais ampla de situações operacionais.

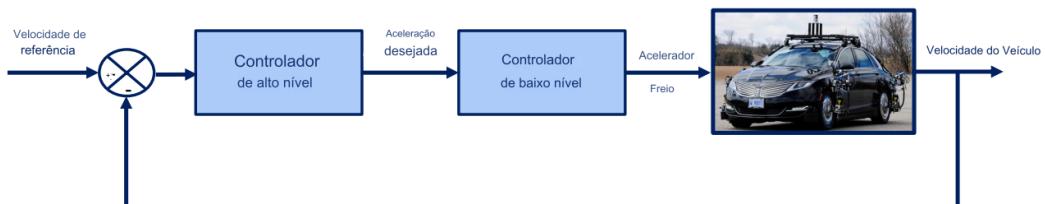


Figura 7 – Controle de velocidade longitudinal (University of Toronto, 2018, Week 5 - Lesson 2: Longitudinal Speed Control with PID. 4min13s).

O diagrama de blocos da Figura 7 apresenta o controlador de cruzeiro e o modelo

de veículo como um sistema de laço fechado projetado para manter a velocidade do veículo próxima à velocidade de referência. Conforme o autor (University of Toronto, 2018), o sistema de controle de cruzeiro e o modelo veicular da planta formam um sistema em laço fechado projetado para manter a velocidade do veículo próxima à velocidade de referência. O controlador pode ser dividido em dois níveis: um controlador de alto nível e um controlador de baixo nível. O controlador de alto nível calcula a diferença entre a velocidade de referência e a velocidade real do veículo, gerando a aceleração desejada para reduzir essa diferença. Por outro lado, o controlador de baixo nível recebe a aceleração do veículo e gera uma atuação no acelerador ou freio para seguir a aceleração de referência. Na prática, essa abordagem de dois estágios permite ir além do controle PID, impondo limites ou perfis diretamente nas acelerações solicitadas ao veículo para manter a velocidade. Além disso, possibilita separar o uso dos mapas do motor para gerar um torque desejado, dado o estado do motor, da resposta de entrada do controle de cruzeiro.

Portanto, examinaremos o controlador de alto nível, encontrado na Figura 7. Segundo o autor (University of Toronto, 2018), a análise do controlador de alto nível é essencial. Pois é ele que determina a quantidade necessária de aceleração a cada passo de tempo com base no erro de velocidade. Onde a entrada para o controlador de alto nível é o erro de velocidade, e a saída corresponde à aceleração desejada (\ddot{x}_{des}) do veículo, conforme a equação 4.12.

$$\ddot{x}_{des} = K_p(\dot{x}_{ref} - \dot{x}) + K_i \int_0^t (\dot{x}_{ref} - \dot{x}) dt + K_d \frac{d(\dot{x}_{ref} - \dot{x})}{dt} \quad (4.12)$$

Onde: \ddot{x}_{des} = Aceleração Desejada;

K_p = Ganho proporcional no controlador PID;

\dot{x}_{ref} = Velocidade de Referência;

\dot{x} = Velocidade do Veículo;

K_i = Ganho integral do controlador PID;

t = Variável de tempo;

0 = Tempo inicial em que a integração começa;

K_d = Ganho derivativo do controlador PID.

Segundo o autor (University of Toronto, 2018) para implementar o controlador 4.12 em software, precisamos discretizar a lei do controlador, alterando a integral para uma soma ao longo de intervalos de tempo fixos. Salienta que o termo derivativo pode ser aproximado com a diferença finita ao longo de um intervalo de tempo fixo se tanto a aceleração de referência quanto a aceleração estimada do veículo não estiverem disponíveis.

Do outro lado, o autor (University of Toronto, 2018) introduz que temos os controladores de baixo nível que buscam gerar a aceleração desejada a partir do controlador de alto nível, ajustando o torque produzido pelo motor para aumentar ou diminuir. Esse ajuste é realizado por meio do ângulo do acelerador, porém é regido pela dinâmica da

transmissão de potência e pelo mapa do motor, tornando-se um problema não linear que pode representar um desafio para métodos clássicos de controle. Em vez disso, a aceleração desejada é convertida em uma demanda de torque, e essa demanda de torque é então convertida em um comando para o ângulo do acelerador. Com o propósito de descrever a aceleração do veículo, o autor ([University of Toronto, 2018](#)) destaca a necessidade de considerar a diferença entre o torque do motor e o torque de carga. Para efetuar essa análise, é imprescindível fazer referência à equação do modelo do trem de força desenvolvida pelo autor ([University of Toronto, 2018](#), Week 4 - Lesson 4: Longitudinal Vehicle Modeling), e reorganizá-la de modo a calcular o torque desejado do motor. Tal cálculo considera os torques de carga conhecidos e a aceleração desejada do veículo. Ao realizar essas operações, chegamos à seguinte equação [4.13](#).

$$T_{\text{Engine}} = \frac{J_e}{r_{\text{eff}}(GR)} \dot{x}_{\text{des}} + T_{\text{Load}} \quad (4.13)$$

Onde: T_{Engine} = Torque do motor;

- J_e = Soma das inéncias dos componentes do veículo;
- r_{eff} = Raio efetivo do pneu;
- GR = Razões de engrenagem combinadas;
- \dot{x}_{des} = Velocidade desejada;
- T_{Load} = Torque de carga.

O autor ([University of Toronto, 2018](#)) emprega o mapa de motor em estado estacionário para determinar o ângulo de aceleração necessário visando alcançar a quantidade requerida de torque, conforme o gráfico da Figura 8.

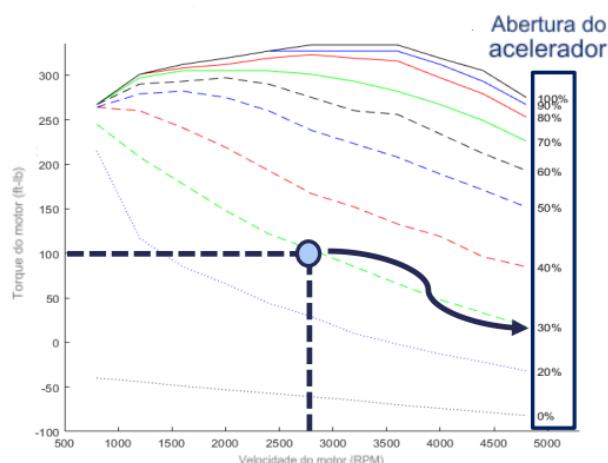


Figura 8 – Mapa de motor ([University of Toronto, 2018](#), Week 5 - Lesson 2: Longitudinal Speed Control with PID. 7min09s).

O gráfico da Figura 8 é gerado durante os testes do motor em diferentes pontos de operação. Nos mapas padrão, a posição do acelerador necessária é determinada pela

interseção entre o torque desejado do motor e a rotação atual do motor, permitindo interpolação quando necessário.

Dessa forma, podemos entender a integração dos componentes do nosso controlador de veículo, a partir da simulação feita pelo autor ([University of Toronto, 2018](#)) que simula a resposta de controle a uma mudança de passo na velocidade desejada dos modelos dinâmicos de veículo com controladores PID.

Os ganhos PID são ajustados por tentativa e erro pelo autor, de modo que as velocidades do veículo sigam a velocidade de referência de 30 metros por segundo, conforme visto no diagrama da Figura 9.

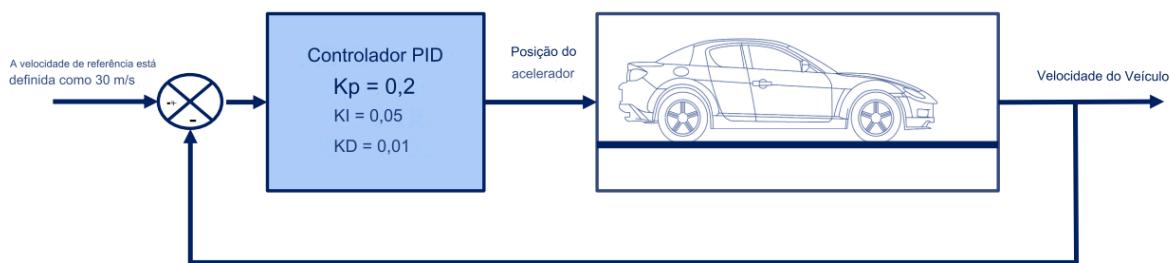


Figura 9 – Diagrama da Simulação ([University of Toronto, 2018](#), Week 5 - Lesson 2: Longitudinal Speed Control with PID. 7min58s).

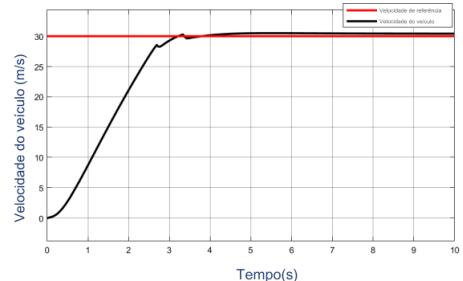
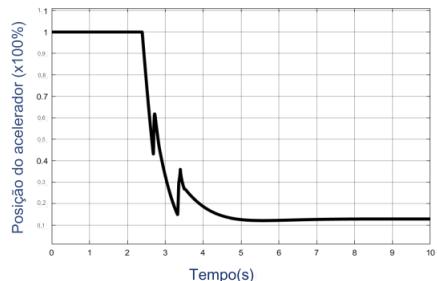


Figura 10 – Exemplo da Simulação ([University of Toronto, 2018](#), Week 5 - Lesson 2: Longitudinal Speed Control with PID. 7min58s).

Esses ajustes resultam no gráfico da Figura 10, onde podemos observar, no gráfico à esquerda, a abertura do acelerador em porcentagem. Agora à direita da Figura 10, podemos identificar a evolução da velocidade real ao longo do tempo, que atinge a velocidade de referência após um tempo de estabilização. O autor ainda salienta que devido a não linearidade do mapa do motor, podemos observar alguns artefatos interessantes na resposta do veículo à medida que se aproxima da velocidade de referência, observado no gráfico à direita.

4.1.2 Controle de Velocidade Antecipado

Nesta subseção desenvolveremos sobre o controle longitudinal para carros autônomos, o foco está na integração de controladores de realimentação (“*feedback*” no inglês) e de antecipação (“*feedforward*” no inglês) para otimizar o desempenho do veículo. A subseção 4.1.1 anterior detalhou a construção de um controlador de retroalimentação, utilizando controle PID para gerar comandos de aceleração. Nesta subseção, a abordagem evolui para incorporar comandos de antecipação (*feedforward*), visando aprimorar o desempenho de rastreamento, especialmente durante manobras dinâmicas.

Desse modo, a investigação sobre a integração do controlador de realimentação (*feedback*) e antecipação (*feedforward*) se faz necessário, devido as suas implementações visando melhorar desempenhos. Segundo o autor (University of Toronto, 2018), essa integração é utilizada para controlar sistemas, onde os controladores de *feedforward* fornecem uma resposta preditiva, uma vez que produzem uma saída de referência para alcançar uma resposta específica de rastreamento, especialmente quando as entradas necessárias são não nulas. Por outro lado, os controladores de *feedback* fornecem uma resposta reativa, que elimina erros de controle devido a perturbações à medida que ocorrem. A combinação de controle de *feedback* e *feedforward* é amplamente utilizada devido a essa relação complementar. Como os VAs requerem comandos de direção não nulos para manter uma curva de raio constante e um comando constante de aceleração ou frenagem para manter velocidades ou taxas de desaceleração constantes, os comandos de *feedforward* são extremamente benéficos para melhorar o desempenho de rastreamento na condução autônoma.

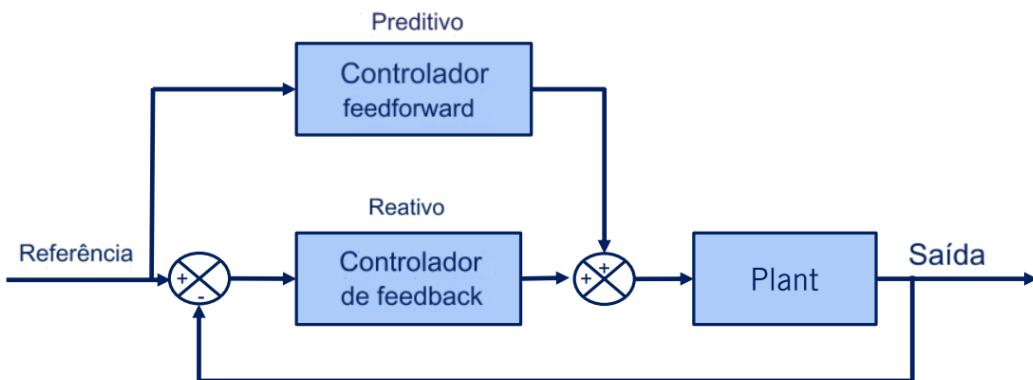


Figura 11 – Controle Combinado de Feedforward e Feedback (University of Toronto, 2018, Week 5 - Lesson 3: Feedforward Speed Control. 2min00s).

Desta maneira, o diagrama de blocos apresentado na Figura 11 ilustra o funcionamento de uma estrutura de controle típica, que engloba tanto a realimentação (*feedback*) quanto a antecipação (*feedforward*). O autor (University of Toronto, 2018) sugere que podemos compreender o controle de *feedforward* como o fornecimento das entradas necessárias para manter a planta, ou “*Plant*” no inglês, seguindo o sinal de referência. Enquanto

isso, o controlador de *feedback* corrige os erros resultantes de perturbações ou imprecisões no modelo da planta utilizado pelo controlador de *feedforward*. A entrada para a planta consiste na adição das entradas de *feedforward* e *feedback*.

Essa mesma estrutura da Figura 11 pode ser usada para gerar atuação do veículo para controle de velocidade longitudinal. Mas dessa vez, o parâmetro de velocidade de referência ou ciclo de condução é definido por um planejador de alto nível (Visto na Figura 7). O autor ([University of Toronto, 2018](#)) destaca que é desejável que o veículo siga precisamente a velocidade de referência. Nesse caso, a velocidade de referência é a entrada para o bloco de avanço de fase, e o erro de velocidade é a entrada para o bloco de controle de *feedback* ou PID, conforme identificados no diagrama de blocos da Figura 12.

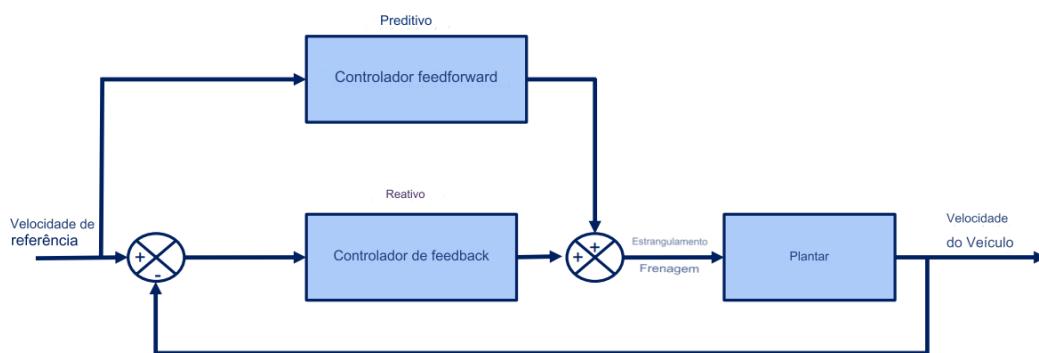


Figura 12 – Controle de velocidade do veículo ([University of Toronto, 2018](#), Week 5 - Lesson 3: Feedforward Speed Control. 3min28s).

Ambos os controladores produzem dois sinais de atuação do veículo: os comandos de aceleração e frenagem. Observa-se que não há controlador de baixo nível incluído no diagrama de blocos da Figura 12, como tínhamos no controle de *feedback* PID da subseção anterior, visto na Figura 7. O papel do controlador de baixo nível, que busca a aceleração desejada por meio de um mapeamento de acelerações para comandos do motor, será agora tratado pelo bloco de *feedforward*. Onde recebe apenas o sinal de referência como entrada, e seu objetivo principal é configurar com precisão as entradas do plano. Visando atingir esse propósito, o autor ([University of Toronto, 2018](#)) sugere a conversão do modelo de dinâmica longitudinal completa em uma tabela de pesquisa fixa ou mapa de referência. Esse mapa associa a velocidade de referência aos sinais correspondentes dos atuadores, considerando que o veículo esteja em estado estacionário. Nesse contexto, a abordagem de antecipação é mais eficaz, ao desconsiderar as dinâmicas internas do trem de força do veículo.

Desse modo, as etapas necessárias para desenvolver os comandos do atuador a partir de uma tabela de pesquisa da antecipação, segundo o autor ([University of Toronto, 2018](#)) são as seguintes:

- 1. Velocidade Angular da Roda:** baseado na relação cinemática entre a velocidade

do veículo e a Velocidade Angular da Roda, podemos calcular a velocidade angular da roda necessária, da seguinte forma na equação 4.14:

$$V_{ref} = r_{eff}\omega_w \rightarrow \omega_w = \frac{V_{ref}}{r_{eff}} \quad (4.14)$$

Onde: V_{ref} = Velocidade de referência;

r_{eff} = Raio efetivo da roda;

ω_w = Velocidade angular da roda.

2. **Velocidade Angular do Motor:** a velocidade angular da roda está relacionada à velocidade angular do motor, ou rotação por minuto (RPM) do motor, por meio das relações de engrenagem da transmissão, diferencial e unidade final. Dessa forma, podemos calcular a RPM do motor correspondente à velocidade angular desejada da roda por meio das relações cinemáticas definidas no módulo de modelagem, conforme a equação 4.15.

$$\omega_w = GR\omega_e \rightarrow \omega_e = \frac{\omega_w}{GR} \quad (4.15)$$

Onde: GR = Relação de transmissão ou razões de engrenagem;

ω_e = Velocidade angular do motor;

ω_w = Velocidade angular da roda.

3. **Torque do Motor:** considerando os requisitos de operação em estado estacionário, a dinâmica do conjunto motriz afirma que o torque do motor deve ser igual ao torque total da carga atuando no veículo. A origem do torque de carga provém da resistência aerodinâmica, resistência de rolamento e resistência gravitacional do veículo. Resultando na equação 4.16.

$$T_{engine} \rightarrow T_{load} \quad (4.16)$$

Onde: T_{engine} = Torque do motor;

T_{load} = Torque total de carga atuando no veículo.

4. **Ângulo do Acelerador:** a partir do torque de motor necessário, podemos combinar isso com a velocidade atual de operação do motor em RPM para definir a posição do acelerador necessária para gerar o torque requerido, conforme a equação 4.17.

$$\begin{array}{ccc} \omega_e & \rightarrow & \theta_{throttle} \\ T_{engine} & & \end{array} \quad (4.17)$$

Onde: ω_e = Velocidade angular do motor;

$\theta_{throttle}$ = Ângulo do acelerador necessário para gerar o torque requerido.

Mais uma vez, um mapa do motor pode ser definido para valores discretos de torque do motor e RPM em estado estacionário, e pode ser interpolado conforme necessário, com base no ponto de operação atual do veículo, de maneira semelhante ao mapa da Figura 8.

Por fim, podemos comparar os dois métodos, o apresentado na subseção anterior 4.1.1, e este que combina o *feedforward* e *feedback*. Para essa comparação, o autor (University of Toronto, 2018) usou as mesmas métricas da simulação da subseção 4.1.1.

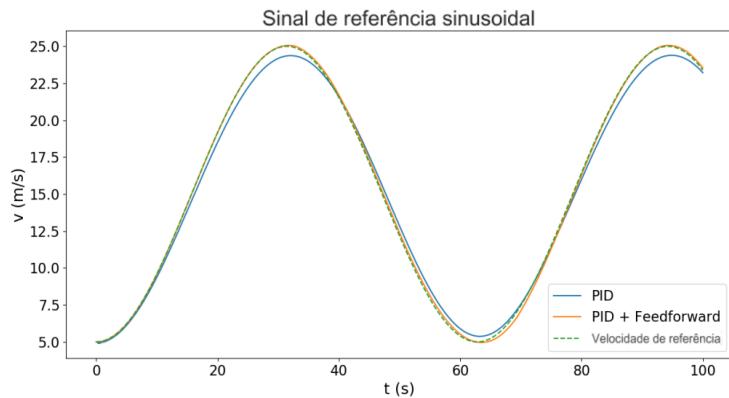


Figura 13 – Resultados de simulação feedforward (University of Toronto, 2018, Week 5 - Lesson 3: Feedforward Speed Control. 6min20s).

A partir do gráfico apresentado na Figura 13, é possível identificar a diferença fundamental entre as duas respostas, especialmente à medida que a velocidade de referência varia. Enquanto o controlador PID requer a presença de erros antes de corrigi-los, sua resposta fica aquém da abordagem *feedforward*, a qual aplica imediatamente os valores de referência de entrada relevantes.

No entanto, o autor (University of Toronto, 2018) ressalta que o rastreamento *feedforward* não é totalmente perfeito, uma vez que a resposta do veículo é, em última instância, influenciada por sua inércia. A abordagem *feedforward* da Figura 13 baseia-se na modelagem em estado estacionário do veículo. À medida que o modelo *feedforward* se torna mais preciso, os componentes de *feedback* podem concentrar-se exclusivamente na rejeição de perturbações, permitindo um rastreamento preciso e consistente do perfil de velocidade.

4.2 Introdução ao Controle Lateral de Veículos

Esta seção aborda os aspectos essenciais do controle lateral para o desenvolvimento de VAs. Na seção 4.1 anterior, foram explorados conceitos fundamentais de controle longitudinal no contexto do desenvolvimento de carros autônomos. A partir dessa base, o foco agora se volta para o controle lateral, um elemento crucial para garantir um seguimento preciso do caminho por parte dos VAs.

As discussões a seguir, visam fornecer uma compreensão abrangente do controle lateral. A exploração começa com uma visão geral do controle lateral do veículo em um exemplo proposto pelo autor ([University of Toronto, 2018](#)). Nesse sentido estudamos sobre estratégias de controle geométrico de seguimento de caminho, baseadas na suposição de aderência perfeita do modelo cinematográfico ([JACOBSON et al., 2020](#)), e apresentamos sobre o Controlador de Perseguição Pura que usaremos no desenvolvimento da nossa implementação da seção [4.3](#). Por último, estudamos sobre o Controlador Stanley e o Controlador Preditivo de Modelo, ou MPC (do inglês: *Model Predictive Controller*), uma estratégia avançada frequentemente empregada em VAs, segundo o autor ([University of Toronto, 2018](#)).

Sabendo disso, o principal objetivo de um controlador lateral é garantir que o veículo consiga seguir precisamente uma trajetória predefinida, executando o plano de movimento ([ZHENG et al., 2023](#), p. 2). Selecionando o ângulo de direção necessário para corrigir quaisquer erros que se acumulem e acompanhar as mudanças na direção da trajetória conforme surgem. Segundo o autor ([University of Toronto, 2018](#)), de modo a projetar um controlador lateral, é necessário definir os seguintes requisitos:

1. O erro entre a posição do veículo e as coordenadas desejadas adequadas da trajetória;
2. Selecionar uma estratégia de projeto de controle que leve os erros a zero, enquanto ainda atende as limites de ângulo de direção;
3. Considerar as limitações dinâmicas do veículo e as características desejadas do percurso, como aceleração lateral máxima e mínimo solavanco.

No geral, o comando de controle deve estar ciente das forças disponíveis nos pneus e não exceder as capacidades do veículo ao corrigir erros de trajetória.

O autor ([University of Toronto, 2018](#)), evidencia que a trajetória do veículo seguirá um caminho de referência ou seguimento de trajetória com o sistema de planejamento no controlador lateral e pode ser definido das seguintes maneiras:

1. **Segmentos de linha reta:** sendo essa a abordagem mais simples, sendo definida por um segmento de linha reta, exigindo uma sequência de vértices conectados linearmente, conforme identificável na Figura [14](#). É salientando que essa definição de caminho pode ser muito compacta e fácil de construir, supondo que os pontos estejam espaçados adequadamente e se o ambiente permita principalmente movimento em linha reta, como em uma grade de vias urbanas de Manhattan. No entanto, o caminho inclui descontinuidades de direção, tornando o rastreamento preciso de uma trajetória desafiadora.

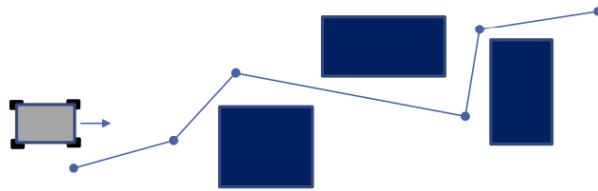


Figura 14 – Segmentos de linha reta ([University of Toronto, 2018](#), Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 2min52s).

2. Pontos de referência: sendo esse uma versão melhorada do segmento de linha reta, onde o espaçamento entre os pontos é geralmente fixado em termos de distância ou tempo de deslocamento, conforme podemos identificar na Figura 15. Neste caso, as posições relativas dos pontos de passagem podem ser reduzidas para atender a uma restrição de curvatura. Segundo o autor ([University of Toronto, 2018](#)), os caminhos pontilhados são bastante úteis devido à sua simplicidade de uso e à possibilidade de serem elaborados diretamente a partir de estimativas de estado ou pontos de passagem de GPS obtidos em operações anteriores de uma rota específica.

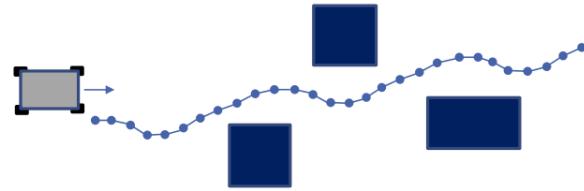


Figura 15 – Pontos de referência ([University of Toronto, 2018](#), Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 3min00s).

3. Curvas parametrizadas: definido por um caminho utilizando uma sequência de curvas parametrizadas contínuas, as quais podem ser desenhadas a partir de um conjunto fixo de primitivas de movimento ou podem ser identificadas por meio de otimização durante o planejamento, podemos entender esse processo a partir da Figura 16. Onde oferecem a vantagem de um movimento continuamente variável e podem ser construídas de modo a possuir derivadas suaves, contribuindo para a consistência nos cálculos de erro e taxa de erro.

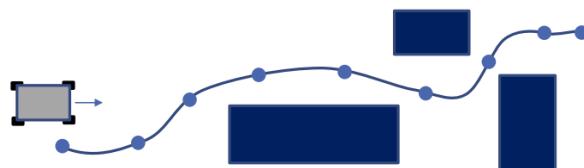


Figura 16 – Curvas parametrizadas ([University of Toronto, 2018](#), Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 3min30s).

O autor saliente ([University of Toronto, 2018](#)) que em todos esses casos de seguimento de trajetória, o controlador busca eliminar o desvio do veículo em relação à trajetória desejada e alinhar a orientação do veículo com a orientação da trajetória.

Desse modo, iniciaremos o entendimento de como se é dada a trajetória desejada no projeto de controle lateral, a qual, segundo o autor (University of Toronto, 2018), é dividida em duas categorias principais. A primeira categoria de controladores são os controladores geométricos, que dependem da geometria e das coordenadas do caminho desejado, bem como dos modelos cinemáticos do veículo. Sendo esses, o controlador de perseguição pura “perseguidor de cenoura” e o controlador Stanley. A outra categoria de controladores é denominada controladores dinâmicos. O controlador avançado mais popular e comumente utilizado nessa categoria é o MPC, que realiza uma otimização de horizonte finito para identificar o comando de controle a ser aplicado (University of Toronto, 2018).

De modo a compreendermos o funcionamento dos controladores laterais usaremos o modelo cinemático de bicicleta, como sugerido pelo autor (University of Toronto, 2018), como as bases para a elaboração desta discussão, para isso nos baseamos nos autores (University of Toronto, 2018; RAJAMANI, 2011; FRANCIS et al., 2016; SNIDER et al., 2009). Desse modo, para esta discussão, é utilizado um segmento de reta como nossa trajetória de referência, representada por uma linha preta contínua, vista na Figura 17.

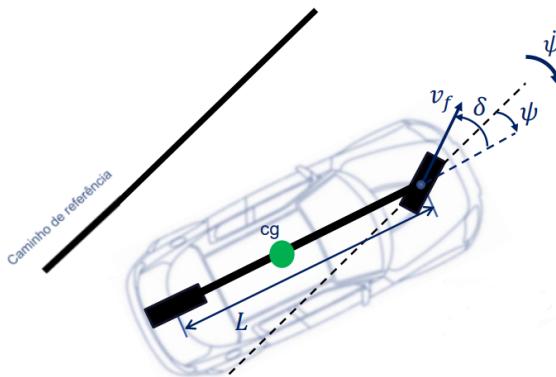


Figura 17 – Trajetória de referência (University of Toronto, 2018, Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 6min23s).

Nesse cenário, podemos ter 2 tipos de erros: erro de rumo e erro de trajetória cruzada, segundo o autor (University of Toronto, 2018). O erro de rumo é igual à diferença entre a orientação da trajetória e a orientação do veículo no ponto de referência ao longo do caminho. Trata-se de uma medida fundamental que avalia quão bem o veículo está alinhado e movendo-se na direção do caminho desejado. A taxa de erro de rumo ($\dot{\psi}$) auxilia na compreensão de como o erro de rumo evolui ao longo do tempo e pode ser calculada a partir das equações do modelo cinemático de bicicleta (SNIDER et al., 2009, p. 18), conforme visto na equação 4.18 resolvida em relação ao eixo dianteiro.

$$\dot{\psi}_{\text{des}}(t) - \dot{\psi}(t) = \frac{v_f(t) \sin \delta(t)}{L} \quad (4.18)$$

Onde: $\dot{\psi}_{\text{des}}(t)$ = Taxa de erro de rumo no tempo t
 $\dot{\psi}(t)$ = Rumo atual no tempo t
 $v_f(t)$ = Velocidade para frente no tempo t
 $\sin \delta(t)$ = Seno do ângulo de direção no tempo t
 L = Comprimento da distância entre eixos.

Para segmentos de reta, a taxa desejada de mudança de direção é zero, e pode ser desconsiderada da equação 4.18. Segundo o autor (University of Toronto, 2018), isso ocorre porque a direção de referência não varia com o tempo para uma reta, uma vez que redefinimos nossa direção em relação à direção atual do caminho, resultando na equação 4.19.

$$\dot{\psi}(t) = \frac{-v_f(t) \sin \delta(t)}{L} \quad (4.19)$$

Como mencionado, o outro tipo de erro é um erro de desvio chamado de erro de trajetória cruzada, também conhecido como “*Crosstrack Error*” no inglês. O autor (University of Toronto, 2018), frisa que o erro de trajetória cruzada é a distância entre o ponto de referência no veículo e o ponto mais próximo na trajetória de caminho desejada. A qual é a principal medida de quão próxima à posição do veículo está da posição desejada ao longo do caminho de referência. Obs.: Tanto o erro de rumo quanto o erro de trajetória cruzada devem convergir para zero para o veículo rastrear adequadamente o caminho desejado (University of Toronto, 2018).

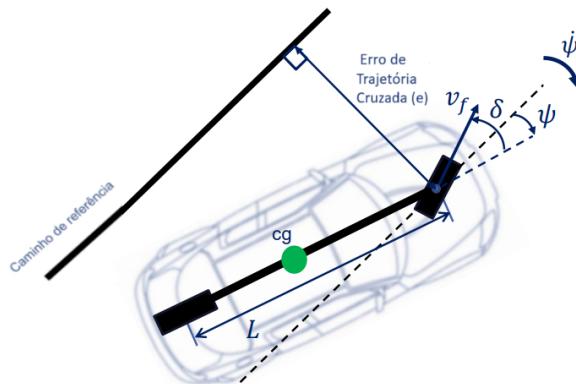


Figura 18 – Erro de trajetória cruzada (University of Toronto, 2018, Week 6 - Lesson 1: Introduction to Lateral Vehicle Control. 8min31s).

Desse modo, a taxa de variação do erro de trajetória cruzada, visto na Figura 18, pode ser calculada extraíndo o componente lateral da velocidade de avanço, conforme apresentado na equação 4.20.

$$\dot{e}(t) = v_f(t) \sin(\psi(t) - \delta(t)) \quad (4.20)$$

Onde:	$\dot{e}(t)$	= Taxa de variação do erro de trajetória cruzada no tempo t
	$v_f(t)$	= Velocidade para frente no tempo t
	$\sin(\psi(t) - \delta(t))$	= Seno da diferença entre o ângulo do curso e o ângulo de direção no tempo t
	$\psi(t)$	= ângulo do curso no tempo t
	$\delta(t)$	= ângulo de direção no tempo t

A equação 4.20 revela que, à medida que a velocidade aumenta, o erro de trajetória cruzada varia mais rapidamente. Conforme indicado pelo autor ([University of Toronto, 2018](#)), esse comportamento implica na necessidade de aplicar ângulos de direção menores para corrigir erros de trajetória cruzada de magnitude semelhante. No entanto, não nos aprofundaremos nesta correção neste relatório, uma vez que já estudamos detalhadamente os temas relevantes para os Objetivos deste relatório, como caminho de referência, controlador MPC e Stanley, erro de rumo, e erro de trajetória cruzada.

4.2.1 Controle Lateral Geométrico - Controlador de Perseguição Pura

A seção 4.2 anterior proporcionou uma definição dos conceitos essenciais para o controle lateral de veículos. Nesta subseção, o foco inicial será na introdução do conceito de um controlador geométrico de trajetória, que utiliza o modelo cinemático do veículo para selecionar comandos de direção. Em seguida, será desenvolvido um controlador de perseguição pura (no inglês: *Pure Pursuit*) para VAs, que visa seguir uma trajetória de referência no ambiente.

Segundo o autor ([University of Toronto, 2018](#)), o controlador geométrico de trajetória é, genericamente, qualquer controlador que acompanha uma trajetória de referência usando apenas a geometria da cinemática do veículo e da trajetória de referência. Especificamente para VAs, esse controlador é uma forma de controle lateral que desconsidera as forças dinâmicas no veículo, assumindo que a condição sem deslizamento se mantém nos pneus. Este controlador se baseia em um modelo cinemático de bicicleta e em medidas de erro definidas na seção 4.2 para construir uma regra de comando de direção que alcance a perseguição de trajetória.

Em seguida, apresentaremos o controlador de perseguição pura, onde um ponto de referência é colocado em uma trajetória a uma distância fixa à frente do veículo, e os comandos de direção necessários para interceptar esse ponto usando um ângulo de direção constante são calculados. Este método utiliza o ponto central do eixo traseiro como ponto de referência no veículo e define a linha que conecta o centro do eixo traseiro ao ponto de referência alvo como uma linha de distância fixa.

Com base nesse contexto, começaremos apresentando os controladores geométricos de seguimento de trajetória, os quais se fundamentam em um ponto de referência ao longo

da trajetória desejada. Esse ponto pode coincidir com o utilizado para calcular os erros de orientação e desvio lateral, ou, conforme discutiremos nesta subseção, pode ser um ponto de antecipação posicionado a uma determinada distância à frente do veículo ao longo da trajetória, como ilustrado na Figura 19 como um ponto vermelho.

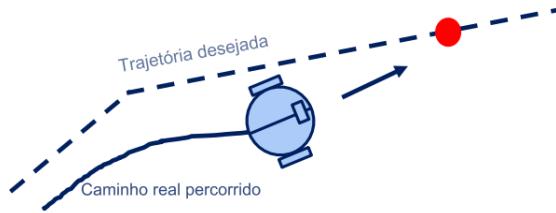


Figura 19 – Rastreamento de caminho geométrico ([University of Toronto, 2018](#), Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 2min27s).

No método de perseguição pura usaremos esse ponto vermelho da Figura 19, onde a sua ideia central, segundo os autores ([University of Toronto, 2018](#)), e ([SNIDER et al., 2009](#), p. 9): é que um ponto de referência pode ser colocado na trajetória a uma distância fixa à frente do veículo, e os comandos de direção necessários podem ser calculados para interseccionar esse ponto usando um ângulo de direção constante. Desse modo, à medida que o veículo vira em direção à trajetória para seguir essa curva, o ponto continua a se movimentar para frente, reduzindo o ângulo de direção e levando suavemente o veículo em direção à trajetória.

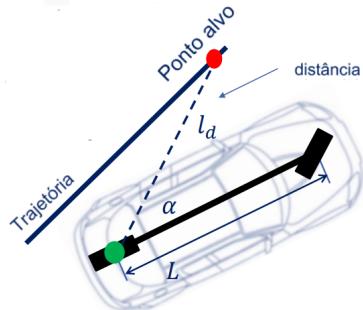


Figura 20 – Pure pursuit ([University of Toronto, 2018](#), Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 3min15s).

O ponto de referência no veículo corresponde ao centro do eixo traseiro, conforme destacado na Figura 20. Além disso, observamos haver uma linha que conecta o centro do eixo traseiro ao ponto de referência desejado, estabelecendo uma distância fixa l_d conhecida como distância de antecipação. Essa distância é representada pela linha tracejada que se estende até o ponto vermelho. O ângulo formado entre a direção da carroceria do veículo ao ponto vermelho é denominado α .

Segundo o autor ([University of Toronto, 2018](#)), para elaborar a lei do controlador de perseguição pura, precisamos dos conceitos sobre centro instantâneo de rotação. Nesse

caso, o ponto de destino na trajetória (ponto vermelho), o centro do eixo traseiro e o centro instantâneo de rotação formam um triângulo com dois lados de comprimento R e um de comprimento l_d , conforme podemos identificar na Figura 21.

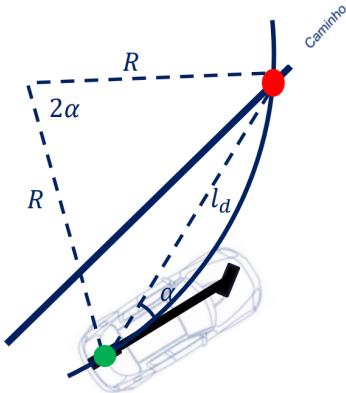


Figura 21 – Pure pursuit ICR ([University of Toronto, 2018](#), Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 3min52s).

Segundo o autor ([University of Toronto, 2018](#)), precisamos definir o arco que leva o ponto de referência do veículo ao ponto de destino na trajetória. Este arco é a parte do círculo do ICR que abrange o ângulo de 2α e pode ser derivado usando identidades trigonométricas padrão, conforme a equação 4.21 escrita baseada na lei dos senos ([SNIDER et al., 2009](#), p. 9):

$$\frac{L_d}{\sin 2\alpha} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)} \quad (4.21)$$

Então, usando mais algumas identidades trigonométricas, podemos chegar na equação 4.22 simplificada.

$$\frac{l_d}{2 \sin \alpha \cos \alpha} = \frac{R}{\cos(\alpha)} \quad (4.22)$$

O que leva à expressão compacta da equação 4.23.

$$\frac{l_d}{\sin \alpha} = 2R \quad (4.23)$$

Finalmente, a curvatura κ , a qual é a inversa do raio de arco R , pode ser expressa conforme na equação 4.24.

$$\kappa = \frac{1}{R} = \frac{2 \sin \alpha}{l_d} \quad (4.24)$$

Para calcular o ângulo de direção precisamos rastrear esse arco, e isso é alcançado a partir do modelo da bicicleta, conforme podemos identificar na Figura 21. Nesse cenário, temos que o ângulo de direção define o raio da curva ([SNIDER et al., 2009](#), p. 10), e podemos estabelecer a equação 4.25.

$$\delta = \tan^{-1} kL \quad (4.25)$$

Segundo o autor ([University of Toronto, 2018](#)), combinando esta expressão (δ) da equação 4.25 com a equação 4.24 (κ), podemos expressar o ângulo de direção necessário para seguir a curva, conforme a equação 4.26.

$$\delta = \tan^{-1} \left(\frac{2L \sin \alpha}{l_d} \right) \quad (4.26)$$

De modo a compreender essa implementação para direção, precisamos nos aprofundar em como os valores de erro evoluem em circuito fechado, para o controlador de perseguição pura que estamos trabalhando nesta subseção.

Desse modo, podemos definir o erro de trajetória cruzada (e) como a distância entre o vetor de orientação e o ponto alvo (círculo em vermelho), conforme podemos identificar na Figura 22, e a expressão do erro na equação 4.27.

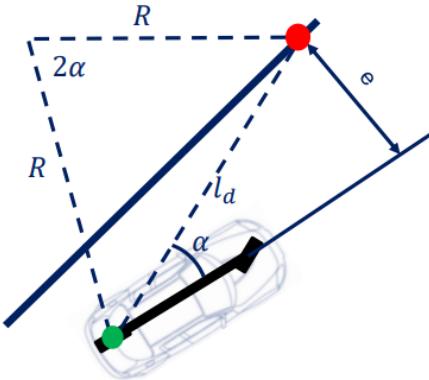


Figura 22 – Erro Crosstrack ([University of Toronto, 2018](#), Week 6 - Lesson 2: Geometric Lateral Control - Pure Pursuit. 3min52s).

$$\sin \alpha = \frac{e}{l_d} \quad (4.27)$$

Ao combinar a equação 4.27 com a expressão da equação 4.24, o autor ([University of Toronto, 2018](#)) salienta que podemos observar que a curvatura da trajetória gerada pelo controlador de perseguição pura é proporcional ao erro transversal no ponto de referência de busca à frente (ponto vermelho). O resultado dessa combinação pode ser observado na equação 4.28.

$$\kappa = \frac{2}{l_d^2} e \quad (4.28)$$

O autor ([University of Toronto, 2018](#)) reforça que a medida que o erro aumenta, a curvatura também se intensifica, reposicionando o veículo mais assertivamente na trajetória. A equação 4.28 evidencia que o controlador de perseguição pura opera de

maneira semelhante ao controle proporcional, visto em subseções anteriores, para corrigir o erro de desvio lateral, utilizando a curvatura da trajetória como saída do controlador.

Entretanto, será necessário incluir uma modificação no controle de curvatura para considerar diferentes valores de velocidade do veículo. Desse modo, podemos ajustar a distância de antecipação (l_d) com base na velocidade do veículo ($K_{dd}v_f$).

Substituindo este ajuste nas equações 4.26 e 4.28, chegamos ao controlador completo de perseguição pura (SNIDER et al., 2009, p. 10), conforme podemos identificar na equação 4.29.

$$\delta = \tan^{-1} \left(\frac{2L \sin \alpha}{K_{dd}v_f} \right) \quad (4.29)$$

Onde: δ = Ângulo de direção;

$2L \sin \alpha$ = Duas vezes o produto do comprimento entre os eixos do veículo e o seno do ângulo do alvo;

$K_{dd}v_f$ = Produto do ganho proporcional pela velocidade para frente.

O controlador seleciona o ângulo de direção que formará um arco até o ponto de referência antecipado e ajusta este ponto de referência para ficar mais distante à medida que o veículo está se deslocando mais rapidamente. Este projeto resulta em comandos de direção e taxas de curva que são alcançáveis dadas as forças disponíveis nos pneus, embora seja necessário ajustá-lo para atingir esse objetivo, segundo o autor (University of Toronto, 2018). Sendo esses ajustes tratados a fundo pelo autor (SNIDER et al., 2009, p. 11).

Dessa forma, com o que foi desenvolvido nesta subseção, temos o conhecimento necessário para construir controladores laterais geométricos de perseguição pura para VAs.

4.3 Controle de Veículos Autônomos

Esta seção aborda o aspecto prático do desenvolvimento de carros autônomos por meio da utilização de simulações, fundamentais no processo iterativo e de mitigação de riscos na criação de VA. Ademais, reunimos todos os conceitos discutidos ao longo das seções e subseções anteriores, e aplicamos em um ambiente de simulação. Onde obtivemos um ambiente controlado e dinâmico no qual conceitos, algoritmos e sistemas podem ser rigorosamente testados sem a necessidade de implementação no mundo real. Podemos, então, executar cenários na simulação envolvendo vários veículos e pedestres controlados por inteligência artificial (IA). Onde podemos realizar variações nesses cenários, centenas ou até milhares de vezes, para garantir que nosso veículo tome consistentemente a decisão correta. Este método garante uma avaliação minuciosa das capacidades do veículo, identificando desafios potenciais e aprimorando soluções antes de sua implementação no domínio físico

(DOSOVITSKIY et al., 2017, p. 1). Desse modo, essa seção visa cumprir o objetivo proposto no capítulo 2 deste trabalho.

4.3.1 Ambientes de simulação de carro autônomo

Existe uma ampla variedade de simuladores disponíveis, desenvolvidos por equipes tanto da indústria quanto da academia. Segundo o autor (DOSOVITSKIY et al., 2017, p. 1), esses simuladores são usados desde os primeiros dias de pesquisa em direção autônoma. Onde configurações personalizadas de simulações são utilizadas para treinar e avaliar os sistemas de percepção do veículo. Um exemplo notável em relação ao uso de simuladores, é o uso *ad-hoc*¹ de jogos digitais para extrair dados de alta fidelidade destinados ao treinamento e avaliação de sistemas de percepção visual. Embora haja um uso *ad-hoc* significativo de simuladores na pesquisa sobre direção autônoma, alguns desses ambientes ou plataformas de simulação existentes apresentam limitações. Por exemplo, segundo o autor (DOSOVITSKIY et al., 2017, p. 1), simuladores de corrida de código aberto, como TORCS, não apresentam complexidade da condução urbana: os simuladores de jogos de corrida existentes apresentam limitações significativas quando se trata de replicar a complexidade da condução urbana. Esses simuladores carecem geralmente de elementos essenciais, como pedestres, cruzamentos, tráfego cruzado, regras de trânsito e outras complicações que distinguem a condução urbana das corridas em pistas. Além disso, jogos digitais que simulam ambientes urbanos com alta fidelidade, como o *Grand Theft Auto V*, não oferecem suporte adequado para a avaliação detalhada de políticas de trânsito. Esses jogos têm pouca personalização e controle sobre o ambiente, uma variação de sensores severamente limitada, falta de *feedback* detalhado ao violar regras de trânsito, entre outras limitações devido à sua natureza comercial de código fechado e objetivos fundamentalmente diferentes do seu desenvolvimento.

Devido a isso, neste trabalho utilizaremos o simulador chamado Carla (CARLA, 2018). Segundo os autores (University of Toronto, 2018) e (DOSOVITSKIY et al., 2017), Carla é um simulador de código aberto desenvolvido por uma equipe composta por membros do *Computer Vision Center* da Universidade Autônoma de Barcelona, *Intel* e do *Toyota Research Institute*, utilizando o *Unreal Engine 4*². O simulador em questão apresenta ambientes virtuaismeticulosamente elaborados, desenvolvidos integralmente por uma equipe especializada de artistas digitais contratados para esse propósito. Esses ambientes abrangem desde cenários urbanos até uma variedade de modelos de veículos,

¹ O termo “ad hoc” tem sua origem na expressão latina que se traduz literalmente como “para isso” ou “para esta finalidade”. Frequentemente utilizado para descrever uma solução elaborada de forma específica para resolver um problema ou atender a um objetivo preciso, sendo, portanto, não passível de generalização ou aplicação em outros contextos (ADHOC..., 2024).

² Engine e/ou Game Engine um software e/ou um conjunto de bibliotecas projetado para simplificar e abstrair o processo de desenvolvimento de jogos eletrônicos ou outras aplicações que envolvam gráficos em tempo real, destinadas a videogames e/ou computadores (LEWIS; JACOBSON, 2002).

edifícios, pedestres, placas de rua, entre outros elementos. Diferenciando-se de outros simuladores discutidos, o simulador Carla possibilita a configuração flexível de conjuntos de sensores e fornece sinais que podem ser utilizados para o treinamento de estratégias de direção, tais como coordenadas GPS, velocidade, aceleração e dados detalhados acerca de colisões e outras infrações.

Além disso, destaca-se a capacidade deste simulador em permitir uma ampla gama de condições ambientais, incluindo variações climáticas e diferentes horários do dia. Algumas dessas condições ambientais são exemplificadas na Figura 23, assim como os mapas disponíveis no simulador: *Town1*, *Town2*, *RaceTrack* e *FlatEarth* (“fun-mode”) podem ser vistos, respectivamente.



Figura 23 – Simulador Carla ([CARLA, 2018](#)). Diferentes condições ambientais e Mapas (Capturas de tela pelos autores).

Toda a simulação pode ser controlada por um cliente externo, que permite enviar comandos ao veículo, registrar dados e executar automaticamente cenários para avaliar o desempenho do carro.

4.3.1.1 Configurando o Simulador Carla

Nesta subseção, apresentaremos como o simulador Carla foi configurado para desenvolver nossa implementação do controlador.

Primeiramente, precisamos dos seguintes pré-requisitos para usarmos o simulador ([University of Toronto, 2018](#)):

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, 2018](#)) 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 os seguintes comandos para instalar essas dependências para o usuário atual:

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

De modo a solucionar os erros que ocorreram durante essa instalação, recorremos ao material do capítulo [A](#).

Agora com tudo propriamente instalado, fizemos *download* do projeto nomeado de *Project*, disponibilizado pelo autor ([University of Toronto, 2018](#)), na pasta *PythonClient* da seguinte forma: *PythonClient\Project*. Esse mesmo *Project* pode ser encontrado no material complementar do capítulo [A](#). A partir disso, iniciamos as nossas implementações do controlador no arquivo *controller2d.py*.

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 aqui são uma versão modificada pelo autor ([University of Toronto, 2018](#)) do CARLA original, com mapas adicionais incluídos.

4.3.2 Implementação Controle de Veículos Autônomos

Nesta subseção, implementamos os conceitos que desenvolvemos durante este relatório, e para isso implementamos um controlador na linguagem de programação Python que visou guiar um carro por um trecho de um circuito no ambiente de simulação Carla. Podemos observar o trecho do circuito percorrido na Figura 24.



Figura 24 – Trecho do circuito ([CARLA, 2018](#)).

De modo a alcançarmos esse objetivo, fornecemos uma lista ordenada de pontos de referência igualmente espaçados no circuito da Figura 24 para o controlador, abordamos esses pontos de referência na subseção 4.2. Os pontos de referência incluem suas posições, bem como a velocidade que os veículos devem atingir, e podem ser encontrados no arquivo *racetrack_waypoints.txt* do projeto. Como resultado, os pontos de referência torna-se o sinal de referência para o controlador, dessa forma navegar por todos os pontos de referência completa efetivamente o circuito. Devido a isso, precisamos implementar os controles longitudinais e laterais, ambos desenvolvidos nas seções 4.1 e 4.2, 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.

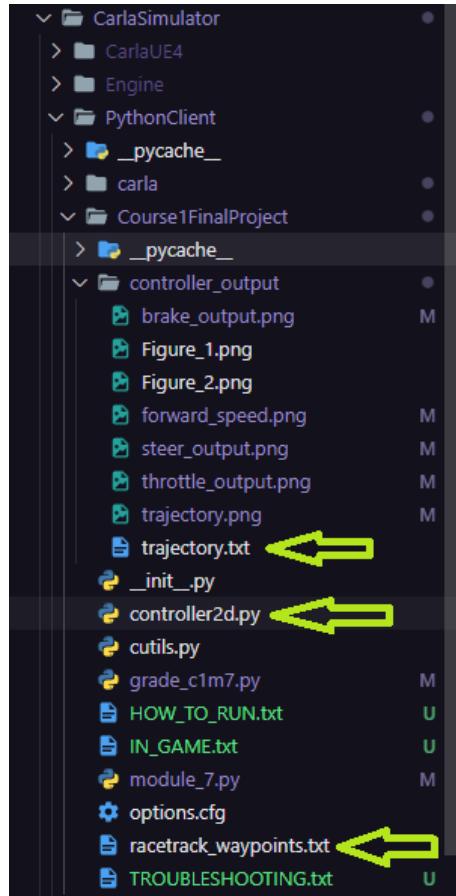


Figura 25 – Estrutura do projeto (Elaborado pelos autores).

Podemos compreender a estrutura do nosso projeto, a partir da Figura 25. Onde os códigos para os controladores foram desenvolvidos no arquivo Python nomeado de *controller2d.py*, apresentado na subseção 4.3.1.1 anterior. 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 podemos identificar pelo trecho de código 4.1. Usamos o caractere (#), de modo a explicar o que está sendo usado e feito durante o trecho de código 4.1.

```

1 class Controller2D(object):
2     def __init__(self, waypoints):
3         # Instância de uma classe utilitária
4         self.vars = cutils.CUtils()
5         # Variáveis atuais do estado do veículo
6         self._current_x = 0
7         self._current_y = 0
8         self._current_yaw = 0
9         self._current_speed = 0
10        self._desired_speed = 0
11        self._current_frame = 0

```

```

12         self._current_timestamp = 0
13         self._start_control_loop = False
14         # Entradas de controle
15         self._set_throttle = 0
16         self._set_brake = 0
17         self._set_steer = 0
18         # Pontos de referência
19         self._waypoints = waypoints
20         self._conv_rad_to_steer = 180.0 / 70.0 / np.pi
21         self._pi = np.pi
22         self._2pi = 2.0 * np.pi
23         # Atualizar variáveis atuais do estado do veículo
24     >def update_values(self, x, y, yaw, speed, timestamp, frame):
25         # Atualizar a velocidade desejada
26     >def update_desired_speed(self):
27         # Atualizar os referência
28     >def update_waypoints(self, new_waypoints):
29         # Pega os comandos de controle
30     >def get_commands(self):
31         # Defini a entrada de aceleração
32     >def set_throttle(self, input_throttle):
33         # Defini a entrada de direção
34     >def set_steer(self, input_steer_in_rad):
35         # Defini a entrada de freio
36     >def set_brake(self, input_brake):
37         # Atualiza os controladores long (PID/PI) e lat (Pursuit)
38     >def update_controls(self):

```

Lista de código 4.1 – class Controller2D: variáveis e funções (Elaborado pelos autores).

A avaliação do desempenho do nosso controlador consistiu na verificação da conformidade do veículo com um perfil de velocidade estabelecido e se o veículo seguiu os pontos de referência do circuito durante a simulação. Para realizar essa verificação, ao final da simulação, gerou-se um arquivo de texto nomeado *trajectory.txt*, que contém a trajetória completa do veículo ao longo do experimento, assim como seu perfil de velocidade, conforme visto na Figura 25. Com base nessas informações, foram elaborados dois gráficos com o intuito de analisar se o nosso controlador conseguiu manter o veículo no perfil de velocidade e trajetória desejada, conforme especificado pelo autor (University of Toronto, 2018). O gráfico que representa o perfil de velocidade que o veículo deve manter pode ser visto na Figura 26, enquanto o gráfico contendo os pontos de referência a serem seguidos pelo veículo pode ser visualizado na Figura 27.

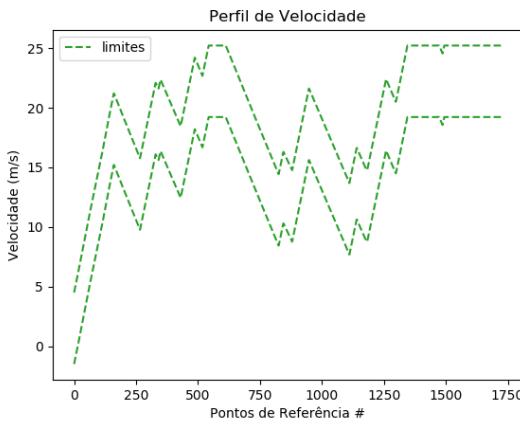


Figura 26 – Limites do Perfil de velocidade (Elaborado pelos autores).

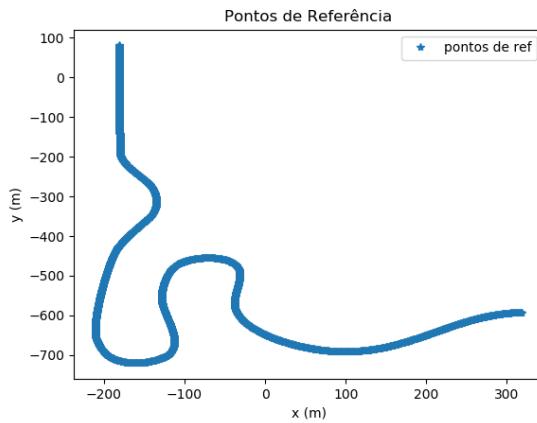


Figura 27 – Pontos de referência (Elaborado pelos autores).

Sabendo disso, a implementação que realizamos no arquivo *controller2d.py* foi a de um controlador PID. O mesmo que foi discutido na seção 4.1, para o controle longitudinal. Onde seu diagrama pode ser visualizado na Figura 7, e para implementação da solução do controlador PID seguimos o material da subseção 4.12. Juntamente à implementação do PID, implementamos um controle de antecipação. Como apresentado na subseção 4.1.2, 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 subseção 4.2.1. Para chegarmos na nossa solução seguimos o material da seção 4.2, na qual apresentamos sobre os pontos de referência usados para guiar o veículo pela trajetória. Sabendo disso, implementamos a equação da subseção 4.2.1 referente a implementação do controlador lateral.

Desse modo, implementamos ambos os controladores no trecho de código 4.2, seguindo todo material elaborado ao longo deste relatório e estudo durante a iniciação

científica como a base teórica, as quais foram apresentadas no referencial teórico deste relatório. Usamos o caractere (#), de modo a explicar o que está sendo usado e feito durante os trechos de códigos 4.2 e 4.3.

```

1 def update_controls(self):
2     ##### RECUPERAR O FEEDBACK DO SIMULADOR #####
3
4     x          = self._current_x
5     y          = self._current_y
6     yaw        = self._current_yaw
7     v          = self._current_speed
8
9     min_index = self.update_desired_speed()
10    v_desired   = self._desired_speed
11    t           = self._current_timestamp
12    waypoints   = self._waypoints
13    throttle_output = 0
14    steer_output   = 0
15    brake_output   = 0
16
17    ##### DECLARAÇÃO DAS VARIÁVEIS DE USO #####
18
19    # Acelerador para torque do motor
20    ##### Relação de engrenagem, raio efetivo, massa + inércia #####
21
22    a_0 = 400
23    a_1 = 0.1
24    a_2 = -0.0002
25
26
27    # Coeficientes aerodinâmicos e de atrito
28    GR = 0.35
29    r_e = 0.3
30    J_e = 10
31    m = 2000
32    g = 9.81
33
34    # Ganhos PID. Tabela 1
35    k_p = 1 / 1.13
36    k_i = k_p / 10
37    k_d = k_p * 0.1 # Ajuste conforme necessário
38
39    """
40
41
42
43

```

```
44     Use-se 'self.vars.create_var(<nome da variável>, <valor padrão>)
45     ,
46     para criar uma variável persistente (não destruída a cada iteração).
47     Isso significa que o valor pode ser armazenado para uso na próxima
48     iteração do loop de controle.
49
50     Exemplo: Criação de 'v_previous', valor padrão para ser 0
51     self.vars.create_var('v_previous', 0.0)
52
53     Exemplo: Definindo 'v_previous' para ser 1.0
54     self.vars.v_previous = 1.0
55
56     Exemplo: Acessando o valor de 'v_previous' para ser usado
57     throttle_output = 0.5 * self.vars.v_previous
58
59     """
60
61     self.vars.create_var('sum_integral', 0.0)
62     self.vars.create_var('steer_output_old', 0.0)
63     self.vars.create_var('v_previous', 0.0)
64     self.vars.create_var('v_total_error', 0.0)
65     self.vars.create_var('v_previous_error', 0.0)
66     self.vars.create_var('t_previous', 0.0)
67
68     # Pula o primeiro quadro para armazenar os valores anteriores
69     # corretamente
70     if self._start_control_loop:
71         #####
72         ##### # IMPLEMENTAÇÃO DO CONTROLADOR LONGITUDINAL
73         ##### # PID / PI (customização necessária)
74         #####
75         ##### # =====
76         ##### # controlador feedforward
77         ##### # Controle de Velocidade Antecipado
78         ##### # =====
79         ##### # calcular F_load e T_e respectivamente
80         ##### # Cálculos de F_load
81         f_aero = c_a * (v_desired ** 2)
82         r_x = c_r1 * v_desired
83         f_g = m * g * np.sin(0)
84         f_load = f_aero + r_x + f_g
85
86         # Cálculo de T_e (assumindo t_e = t_load)
```

```

87     t_e = GR * r_e * f_load
88
89     # calcular a velocidade do motor w_e
90     w_e = v_desired / (GR * r_e)
91
92     # agora atualize o acelerador de acordo com a velocidade do
93     # motor atualizada w_e
94     throttle_forward = t_e / (a_0 + (a_1 * w_e) + (a_2 * w_e ** 2))
95
96     # =====
97     # controlador de feedback
98     # Implementação Controle PID
99     # =====
100
101    dt = t - self.vars.t_previous
102    #Implementação da equação
103    v_current_error = v_desired - v
104    v_total_error = self.vars.v_total_error + v_current_error * dt
105    v_error_rate = (v_current_error - self.vars.v_previous_error) /
106        dt
107
108    throttle_feedback = (k_p * v_current_error) + (k_i *
109        v_total_error) + (k_d * v_error_rate)
110    # Saídas para o controlador
111    throttle_output = throttle_feedback + throttle_forward
112    brake_output = 0
113
114    ######
115    # IMPLEMENTAÇÃO DO CONTROLADOR LATERAL
116    # Pure Pursuit
117    # Controle Lateral Geométrico - Controlador de Perseguição Pura
118    ######
119    # variáveis necessárias
120    Kp_ld = 0.8
121    min_ld = 10
122    L = 3
123    # Implementação da equação
124    x_rear = x - L * cos(yaw) / 2
125    y_rear = y - L * sin(yaw) / 2
126    lookahead_distance = max(min_ld, Kp_ld * v)
127
128    for wp in waypoints:
129        dist = sqrt((wp[0] - x_rear)**2 + (wp[1] - y)**2)
130        if dist > lookahead_distance:

```

```

131         carrot = wp
132         break
133     else:
134         carrot = waypoints[0]
135     alpha = atan2(carrot[1] - y_rear, carrot[0] - x_rear) - yaw
136
137     # Altere a saída de direção com o controlador lateral.
138     # equação 4.86
139     steer_output = atan2(2 * L * sin(alpha), lookahead_distance)
140
141 ######
142 # DEFINI AS SAÍDA DE CONTROLES
143 #####
144 self.set_throttle(throttle_output) # em percentual (0 a 1)
145 self.set_steer(steer_output) # em rad (-1.22 a 1.22)
146 self.set_brake(brake_output) # em percentual (0 a 1)
147
148     print ("yaw: ",yaw," lookahead_distance: ",lookahead_distance,"
steer_output: ",steer_output," throttle_output:",throttle_output,
brake_output:, brake_output)
149
150 #####
151 #####
152 # ARMAZENA VALORES ANTIGOS
153 #####
154 #####
155 """
156     Use-se este bloco para armazenar valores antigos (por exemplo,
podemos armazenar os valores atuais de x, y e yaw aqui usando variáveis persistentes para uso na próxima iteração)
157 """
158 # Armazene a velocidade para frente para ser usada no próximo passo
159 self.vars.v_previous = v
160 self.vars.v_total_error = v_total_error # Da implementação PID
161 self.vars.v_previous_error = v_current_error # Da implementação PID
162 self.vars.t_previous = t
163 self.vars.steer_output_old = steer_output

```

Lista de código 4.2 – Implementação dos controladores PID e Pure Pursuit (Elaborado pelos autores).

Para rodar a implementação inserimos os comandos 4.3.2 no terminal de comando CMD para iniciar o simulador Carla na pista *RaceTrack*.

```

\> C:
\> cd \CarlaSimulator
\> CarlaUE4.exe /Game/Maps/RaceTrack -windowed -carla-server -benchmark -fps=30

```

Em outro terminal, inserimos, também, os comandos 4.3.2, vistos a seguir, para executar os nossos controladores.

```
\> C:
```

```
\> cd \Coursera\CarlaSimulator\PythonClient\Course1FinalProject
\> python module_7.py
```

Isso resulta na execução da simulação, conforme podemos visualizar na captura de tela da Figura 28.

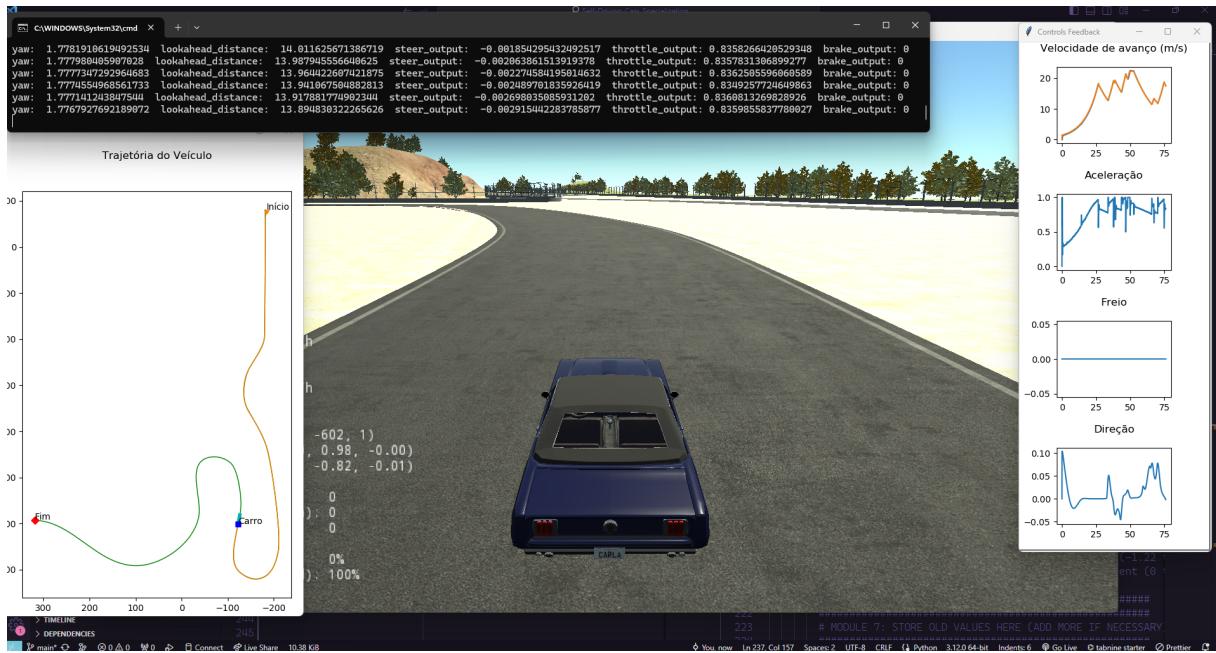


Figura 28 – Captura de Tela da execução (PID e Pure Pursuit) da simulação e controle do veículo (Elaborado pelos autores).

Ao finalizar a execução, o arquivo *trajectory.txt* é gerado, nos dando o perfil de velocidade, conforme visto na Figura 29, e trajetória, conforme visto na Figura 30, obtidos durante a simulação.

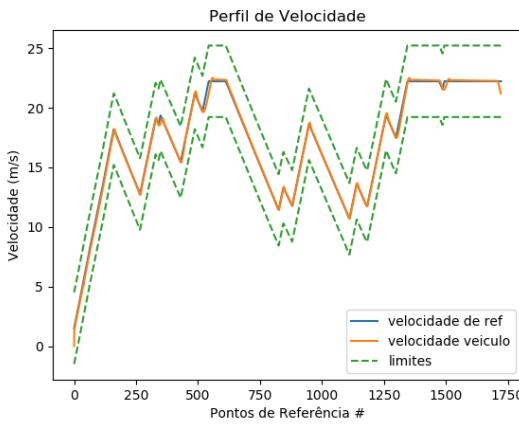


Figura 29 – Perfil de velocidade alcançado usando o PID e Pure Pursuit (Elaborado pelos autores).

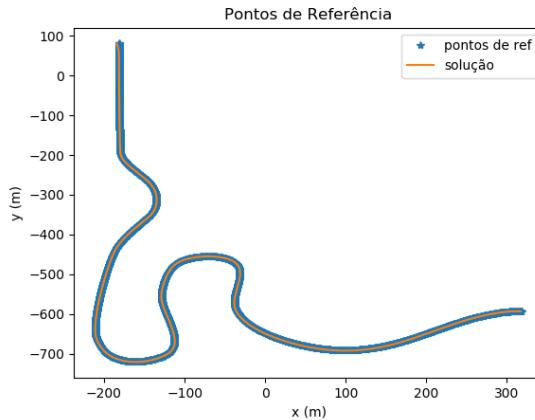


Figura 30 – Trajetória alcançada usando o PID e Pure Pursuit (Elaborado pelos autores).

A partir de agora estamos aptos a fazer uma comparação de desempenho entre os controladores PID e PI. De modo a verificar os sobressaltos estipulados na literatura para controlador PI, visto na Figura 4, e para o controlador PID, visto na Figura 5. Para isso, alteramos a implementação do trecho de código 4.2, de modo a retirar o termo D (derivativo) do controlador longitudinal. A alteração feita no controlador longitudinal, pode ser vista no trecho de código 4.3.

```

1      """ O restante do código pode se manter o mesmo """
2      # retiramos (k_d * v_error_rate). Se tornando apenas PI
3      throttle_feedback = ( k_p * v_current_error ) + ( k_i *
4          v_total_error )
      """ O restante do código pode se manter o mesmo """

```

Lista de código 4.3 – Alteração para o Controlador PI (Elaborado pelos autores).

Após retirar o termo (D), executamos a implementação novamente, conforme visto na captura de tela da Figura 31.



Figura 31 – Captura de Tela da execução (PI e Pure Pursuit) da simulação e controle do veículo (Elaborado pelos autores).

Novamente, geramos os gráficos de Perfil de Velocidade (visto na Figura 32) e de trajetória (visto na Figura 33) referentes a simulação para o controlador PI.

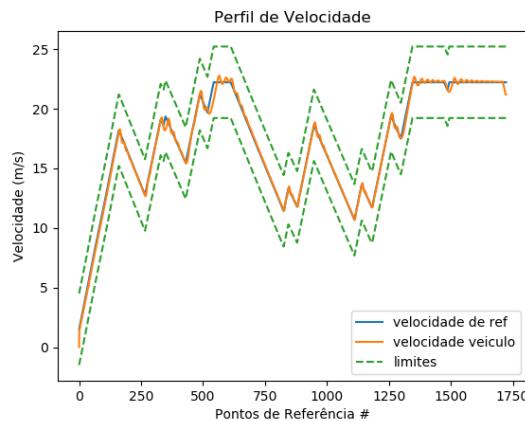


Figura 32 – Perfil de velocidade alcançado usando o PI e Pure Pursuit (Elaborado pelos autores).

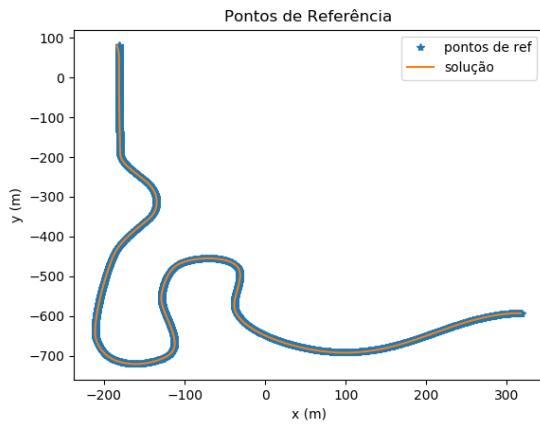


Figura 33 – Trajetória alcançada usando o PI e Pure Pursuit (Elaborado pelos autores).

Desse modo, a partir dos gráficos do perfil de velocidade PID e PI podemos identificar uma diferença em relação aos sobressaltos durante a simulação, conforme estipulado pela literatura discutida na seção 4.1. Desse modo, se comparamos os dois gráficos, podemos identificar que o perfil de velocidade do gráfico da Figura 29, referente ao PID, segue de maneira mais uniforme, comparado com o gráfico da Figura 32, referente ao PI, que apresenta um perfil de velocidade com mais sobressaltos. Adicionamos os dois gráficos lado a lado na Figura 34 para facilitar a visualização das diferenças no perfil de velocidade das implementações.

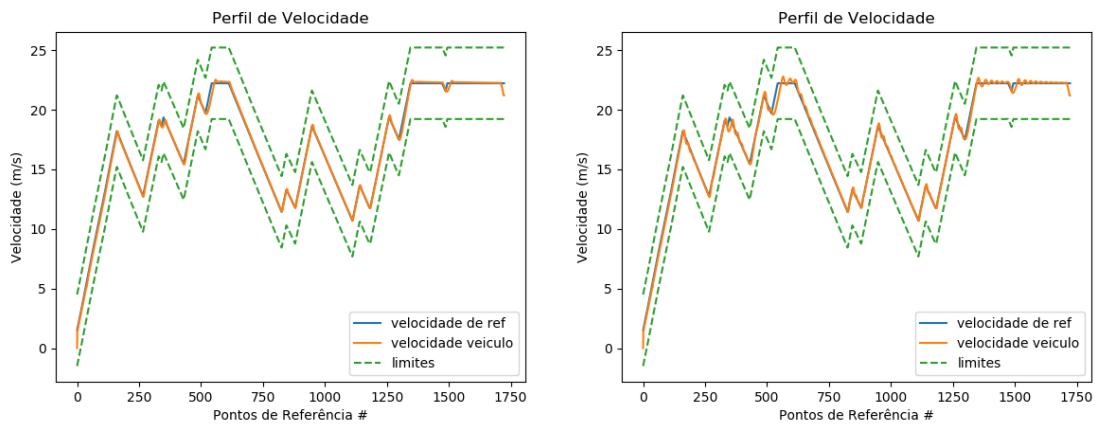


Figura 34 – Comparaçao dos sobressaltos do Perfil de Velocidade PID e PI (Elaborado pelos autores).

5 Considerações Finais

O objetivo (apresentado no capítulo 2) desse segundo ano de iniciação científica (2023-2024) foi a familiarização e compreensão dos principais *softwares* e algoritmos de controle de VAs. De modo a cumprir esse objetivo, conduzimos um levantamento bibliográfico que nos proporcionou uma base teórica, onde exploramos temas relacionados a VAs e seus componentes de controle. Nesse processo, encontramos alguns controladores longitudinais e laterais que julgamos pertinentes para desenvolvê-los neste trabalho. Desse modo, apresentamos um visão geral sobre controladores de VAs e suas modelagens. De modo a dar continuidade às seções, subsequentes, amparados dos materiais encontrados, estudamos sobre modelagem de veículos. Visto que, o estudo e compreensão da modelagem veicular nos proporciona uma compreensão abrangente dos princípios fundamentais que governam o movimento e controle de veículos. Isso nos proporcionou uma compreensão e familiarização com todos os controladores abordados e citados neste relatório. Por fim, implementamos um controlador Lateral Geométrico - Controlador de Perseguição Pura, e dois controladores longitudinais, PID e PI, e realizamos uma comparação de desempenho entre as duas implementações longitudinais para guiar um carro em ambiente de simulação virtual. Para tal, usamos todos os conhecimentos e informações obtidas no desenvolver dos capítulos e seções deste presente relatório. Proporcionando, assim, uma experiência de aprendizado singular. Onde conseguimos colocar em prática todos os conceitos que cobrimos na exploração sobre controladores de VAs em uma aplicação prática em ambiente de simulação virtual.

Sabendo disso, podemos concluir que alcançamos o objetivo desta iniciação científica de familiarização e compreensão dos principais *softwares* e algoritmos de controle de VAs.

6 Perspectiva de continuidade

Conforme os *objetivos* (vistos no capítulo 2) propostos no projeto deste ano 2023-2024, propomos para o próximo ano (2024-2025) dar continuidade e aprofundamento aos conhecimentos que desenvolvemos neste relatório. Visto que, esse ano trabalhamos com a introdução a modelagem veicular e implementação de controladores longitudinais e laterais. Sabendo disso, propomos ampliar os conhecimentos sobre estimativa e localização de estado, percepção visual e planejamento de movimento. Pois esses tópicos foram estudados de maneira introdutória neste ano de iniciação científica com o intuito de entender como sua função é valiosa para a tarefa de direção de um VA. Diante disso, o estudo, a exploração e aplicação sobre esses tópicos se faz essencial para a ampliação dos conhecimentos relacionados à tarefa de direção de VAs, e para darmos continuidade a esta iniciação científica.

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

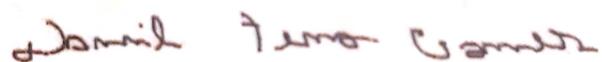
Os certificados obtidos e/ou comprovantes de participação podem ser encontrados neste capítulo. Na Figura 35 encontra-se o Certificado de conclusão do curso Introduction to Self-Driving Cars, o qual pode ser verificado através da url: <<https://coursera.org/verify/QBFUULQCBPQG>> .



Figura 35 – Certificado de conclusão do curso Introduction to Self-Driving Cars.

8 Datas e assinaturas

8.1 Data e assinatura do bolsista (assinatura digitalizada)

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

30/03/2024

8.2 Data e assinatura do orientador (assinatura digitalizada)

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

30/03/2024

Referências

- ADHOC - Quick search results. 2024. <<https://www.oed.com/search/dictionary/?scope=Entries&q=adhoc>>. Acessado: 05-03-2024. Citado na página 45.
- 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 18.
- BRATZEL, S.; CAM. *Die Zukunft der Mobilität - Die Zukunftstrends in den Bereichen Elektromobilität, Connected Car und Mobilitätsdienstleistungen*. 2022. <<https://www.bnpparibascardif.de/web/guest>>. Acessado 09-03-2024. Citado na página 12.
- BUSSEMAKER, K. Sensing requirements for an automated vehicle for highway and rural environments. Delft University of Technology, 2014. Citado 2 vezes nas páginas 28 e 71.
- CARLA. [S.I.]: CARLA Team, 2018. <<https://carla.org/>>. Acessado: 26-03-2024. Citado 4 vezes nas páginas 6, 45, 46 e 49.
- DOSOVITSKIY, A. et al. Carla: An open urban driving simulator. In: PMLR. *Conference on robot learning*. [S.I.], 2017. p. 1–16. Citado na página 45.
- FRANCIS, B. A. et al. Models of mobile robots in the plane. *Flocking and rendezvous in distributed robotics*, Springer, p. 7–23, 2016. Citado 2 vezes nas páginas 12 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 18.
- 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 2 vezes nas páginas 12 e 28.
- JACOBSON, B. et al. Vehicle dynamics. *Chalmers University of Technology*, Chalmers, 2016. Citado na página 12.
- JACOBSON, B. et al. Vehicle dynamics compendium. *Chalmers University of Technology*, Chalmers, 2020. Disponível em: <<https://research.chalmers.se/en/publication/520229>>. Citado 5 vezes nas páginas 12, 13, 28, 36 e 71.
- 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 12.
- LAGE, C. A. Quatro cenários para os veículos autônomos no mundo ocidental. *UNIVERSIDADE DE BRASÍLIA*, 2019. Citado na página 12.

- LEWIS, M.; JACOBSON, J. Game engines. *Communications of the ACM*, v. 45, n. 1, p. 27, 2002. Citado na página 45.
- MAO, J. et al. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, Springer, p. 1–55, 2023. Citado na página 12.
- MOCHURAD, L.; MAMCHUR, M. Parallel and distributed computing technologies for autonomous vehicle navigation. *Radio Electronics, Computer Science, Control*, n. 4, p. 111–111, 2023. Citado na página 12.
- 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.
- 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 12.
- PADEN, B. et al. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, IEEE, v. 1, n. 1, p. 33–55, 2016. Citado na página 12.
- RAJAMANI, R. *Vehicle dynamics and control*. [S.l.]: Springer Science & Business Media, 2011. Citado 2 vezes nas páginas 12 e 38.
- REINHOLTZ, C. et al. Darpa urban challenge technical paper. Citeseer, 2007. Citado na página 12.
- 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 12.
- SINGH, G. B. . K. B. . R. K. Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. Hindawi, 2022. Citado na página 12.
- 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 6 vezes nas páginas 12, 38, 41, 42, 44 e 71.
- 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 2 vezes nas páginas 12 e 71.
- University of Toronto. *Introduction to Self-Driving Cars*. Coursera Inc, 2018. 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 33 vezes nas páginas 5, 6, 7, 12, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48 e 51.
- YAO, S. et al. Radar-camera fusion for object detection and semantic segmentation in autonomous driving: A comprehensive review. *IEEE Transactions on Intelligent Vehicles*, v. 9, n. 1, p. 2094–2128, 2024. Citado na página 12.

ZHENG, S. et al. Simultaneous localization and mapping (slam) for autonomous driving: Concept and analysis. *Remote Sensing*, v. 15, n. 4, 2023. ISSN 2072-4292. Disponível em: <<https://www.mdpi.com/2072-4292/15/4/1156>>. Citado 2 vezes nas páginas 12 e 36.

Apêndices

APÊNDICE A – Material Completo

Neste apêndice, disponibilizamos o link para o material completo desenvolvido ao longo do presente relatório. 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 material completo sobre a subseção **Configurando o Simulador Carla 4.3.1.1**, clique no seguinte link: <https://github.com/ARRETdaniel/Self-Driving_Cars_Specialization/tree/main/CARLA%20Installation%20Guide>

Para acessar o material completo sobre a subseção **Implementação Controle de Veículos Autônomos 4.3.2**, clique no seguinte link:
<https://github.com/ARRETdaniel/Self-Driving_Cars_Specialization/tree/main/CarlaSimulator/PythonClient/Course1FinalProject>

Recomenda-se 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 ([TAXONOMY..., 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 relatório 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>