LUAN FERNANDO ALVES

DESENVOLVIMENTO DE UM CONTROLE REFERENCIADO EM VISÃO APLICADO EM UM CARRO AUTÔNOMO UTILIZANDO ROS E GAZEBO

LAVRAS – MG

2022

LUAN FERNANDO ALVES

DESENVOLVIMENTO DE UM CONTROLE REFERENCIADO EM VISÃO APLICADO EM UM CARRO AUTÔNOMO UTILIZANDO ROS E GAZEBO

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para obtenção do título de Bacharel.

Prof. Dr. Danilo Alves de Lima

Orientador

LAVRAS – MG

2022

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Alves, Luan Fernando
Desenvolvimento de um controle referenciado em visão aplicado em um carro autônomo utilizando ROS e Gazebo / Luan Fernando Alves. – Lavras : UFLA, 2022.
36 p. : il.

TCC (graduação)–Universidade Federal de Lavras, 2022.
Orientador: Prof. Dr. Danilo Alves de Lima.
Bibliografia.

1. ROS. 2. Gazebo. 3. Controle referenciado em visão. I. Lima, Danilo Alves de. II. Título.

RESUMO

Estudos em mobilidade autônoma são uma possibilidade de reduzir o elevado número de acidentes nas vias brasileiras, dado que a maioria desses acidentes são causados por fatores humanos. Contudo, grande parte desses estudos são realizados em veículos ou em plataformas de testes de alto valor agregado, sendo um empecilho para pesquisadores em estágio inicial que não têm grandes auxílios financeiros. Uma forma mais acessível para efetuar tais teste é por meio de simuladores. Entretanto, para validar as técnicas de controle é necessário implementá-las em veículos reais. Nesse contexto, o uso de uma arquitetura de *software* que facilite a troca entre uma interface com o simulador e a que comanda um veículo real é fundamental. O objetivo deste trabalho consiste em implementar um sistema de controle baseado em visão computacional para manter um veículo em um caminho pré-determinado, utilizando o ambiente Gazebo. O Gazebo é um *software* de simulação de robôs, onde, para comandar um robô, é empregado o *framework* ROS (do inglês *Robot Operating System*), que implementa diversos recursos para aplicações em robótica. Resultados mostram que a aplicação da técnica no Gazebo é adequada, conservando o veículo na pista mesmo em curvas fechadas, comprovando que o ambiente simulado pode ser utilizado para o desenvolvimento e testes de veículos autônomos.

Palavras-chave: ROS; Gazebo; Controle referenciado em visão.

ABSTRACT

Studies in autonomous mobility are a possibility to reduce the high number of accidents on Brazilian roads, given that most of these accidents are caused by human factors. However, most of these studies are carried out in vehicles or on high-cost test platforms, which is an obstacle for early-stage researchers who do not have large financial aids. The use of simulators presents itself as a more accessible way to test autonomous vehicles. However, addressing control techniques in real vehicles is necessary to validate them. In this context, the use of a software architecture that provides ways to exchange the interface between a simulator and a real vehicle is fundamental. The objective of this work is to implement a control system based on computer vision to keep a vehicle on a predetermined path, using the Gazebo environment. Gazebo is a software system where it is possible to simulate robots, where, for the robot control, the ROS (Robot Operating System) framework is employed, which implements various features for robotics applications. Results show that the application of the technique in the Gazebo is adequate, keeping the vehicle on the track even in sharp curves, proving that the simulated environment can be used for the development and testing of autonomous vehicles.

Keywords: ROS; Gazebo; Visual-Servoing Controller.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O cenário da mobilidade brasileira apresenta uma elevada taxa de acidentes. De acordo com o Instituto de Pesquisa Econômica Aplicada (IPEA), em 2019, 1,3 milhão de pessoas perderam suas vidas e 50 milhões de pessoas ficaram feridas nas estradas brasileiras. O estudo levanta uma estimativa de 130 bilhões de reais gastos anualmente decorrentes de acidentes de trânsitos (PAULA et al., 2021). O IPEA também estimou que 90% dos acidentes ocorreram por falha humana, sendo as principais causas o excesso de velocidade e a falta de atenção.

Visando reduzir o número de acidentes, diversas ações públicas foram adotadas, dentre elas a campanha do Pacto Nacional pela Redução de Acidentes, com meta de reduzir as taxas de mortalidade em 50% até 2020 promovendo ações de conscientização sobre segurança no trânsito. Mesmo com um relativo sucesso, tendo algumas capitais alcançado reduções de 40%, essas políticas vêm se mostrando pouco eficientes, pois tais reduções podem estar atreladas à menor circulação de veículos nesse período (2019-2020), em função da crise econômica ocorrida no país.

Por outro lado, mas com o mesmo objetivo de reduzir os acidentes, diversos estudos estão em desenvolvimento para criação de sistemas avançados de assistência ao condutor (ADAS)[1] como em Espiau, Chaumette e Rives (1992) e Lima e Victorino (2016) que utilizam câmeras digitais como sensores, e, com o processamento das imagens, obtiveram parâmetros necessários para manter o veículo seguindo um determinado caminho. Apesar do avanço desses trabalhos, devido à complexidade e da diversidade de ambientes que os veículos transitam, ainda é necessário que mais testes e pesquisas sejam realizados. Efetuar tais testes utilizando um veículo requer recursos e infraestrutura apropriados, não sendo viáveis em alguns casos. Porém, é possível simular veículos e diversos cenários e validar diversas metodologias de controle. Assim, um ambiente virtual permite avançar nas tecnologias sem pôr em risco as pessoas durante determinadas fases de testes.

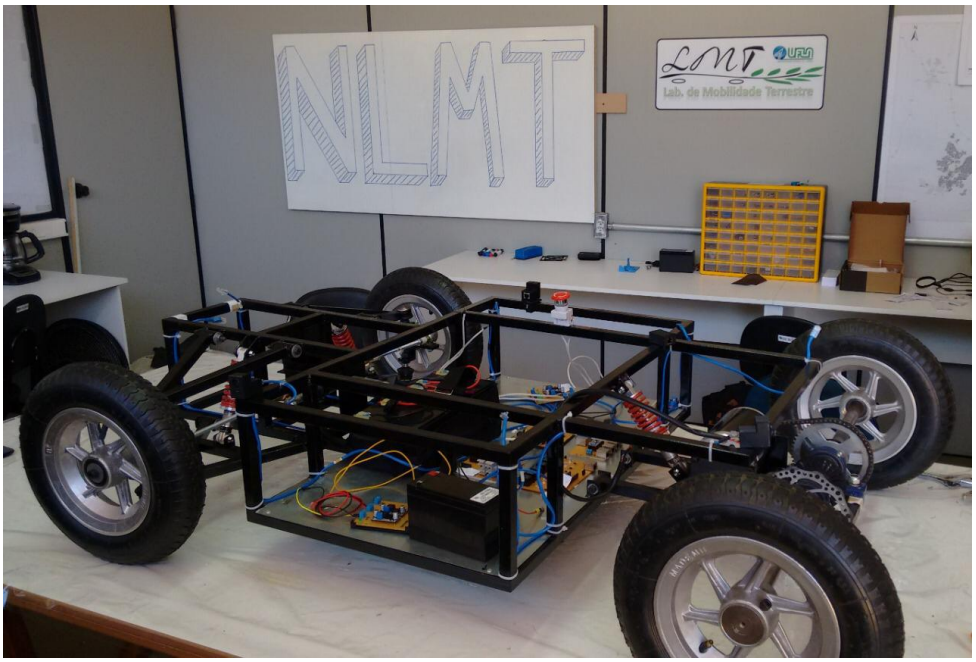
No entanto, mesmo com o avanço dos algoritmos em ambiente simulado, devido às incertezas e aleatoriedades que ocorrem em um ambiente real e que são inviáveis de serem implementadas em simuladores, testes em ambientes reais controlados são importantes. Assim, projetos como o Veículo Inteligente de Desenvolvimento Aplicado (VIDA), são utilizados para realização de tais testes.

[1] Do inglês *Advanced Driver Assistence Systems.*

O VIDA é um projeto em desenvolvimento pelo Laboratório de Mobilidade Terrestre (LMT), cujo objetivo é a criação de uma plataforma que seja capaz de se locomover de forma autônoma e segura (LMT, 2022b). O LMT é uma entidade de ensino, pesquisa, inovação e extensão sediada na Universidade Federal de Lavras (UFLA) (LMT, 2022a). Com o objetivo de atuar na fronteira do conhecimento no que diz respeito a mobilidade e a cidades inteligentes, o LMT proporciona soluções nas áreas de engenharias, computação, administração, economia e direito.

A plataforma VIDA (Figura 1.1) será totalmente aberta, para que qualquer grupo de pesquisa, que tenha interesse, possa desenvolver seu próprio veículo a um custo reduzido. Nesse contexto, ferramentas como o ROS[2] e o Gazebo são amplamente utilizadas.

Figura 1.1 – Plataforma de desenvolvimento VIDA.



Fonte: (LMT, 2022b)

O ROS é um *framework* de desenvolvimento de aplicações robóticas. Ele possibilita a implementação dos algoritmos necessários para o controle de um veículo, testá-los em um ambiente virtual, e quando possível, substituir a interface de comandos para controlar um veículo real. O Gazebo, por sua vez, é um ambiente de simulação facilmente integrado com o ROS para o controle de robôs.

Este trabalho tem como objetivo implementar uma técnica para controlar um veículo em um ambiente de simulação utilizando ROS e Gazebo como base para o teste do

[2] Do inglês *Robot Operating System.*

veículo VIDA em ambiente simulado. Nesse sentido, a implementação será flexível para que a aplicação no VIDA possa ser feita apenas com a alteração da interface de controle.

1.1 Objetivos

O presente trabalho tem como propósito aplicar uma técnica de controle referenciado em visão em um veículo situado em ambiente de simulação. Para tanto, têm-se os seguintes objetivos específicos:

- Desenvolvimento de um ambiente no Gazebo, dispondo de um veículo e um caminho a ser percorrido por tal;
- Implementação de uma interface para o controle do veículo e de uma câmera para a percepção do ambiente;
- Aplicação de uma técnica de controle referenciado em visão, utilizando informações provenientes da câmera, para manter o veículo seguindo o percurso definido.

1.2 Justificativa

O uso de simuladores é uma das formas mais acessíveis para realizar testes de veículos autônomos e, por intermédio dessas simulações, torna-se possível analisar o comportamento do veículo nas mais diversas situações. O projeto VIDA carece de um ambiente simulado para o desenvolvimento de seus algoritmos de controle, fundamental para o teste e validação antes de colocá-lo no mundo real. Além disso, dado a natureza do projeto ser de uma plataforma de código aberto para futuros interessados no estudo e desenvolvimento de veículos autônomos, formas de se facilitar a difusão dos trabalhos desenvolvidos no LMT aumentam a visibilidade do mesmo. Portanto, o ambiente criado neste trabalho servirá como base para trabalhos futuros, como aplicações com pistas mais detalhadas ou simulações de condições variadas.

1.3 Estrutura

Este trabalho foi dividido em 6 capítulos. O Capítulo 2 apresenta uma revisão da literatura, comentando algumas ferramentas de simulação utilizadas para robótica móvel. Também será citado técnicas clássicas utilizadas para controle de veículos, como

controlador proporcional, integral e derivativo (PID) para manter o veículo em velocidade constante ou para reduzir o erro no ângulo de esterçamento. No Capítulo 3, são apresentadas a ferramenta de simulação, a arquitetura e a técnica de controle utilizadas. A metodologia do trabalho é demonstrada no Capítulo 4, definindo o modelo cinemático do veículo simulado, explicando o controle utilizado e apresentando a implementação em ROS e Gazebo. Em seguida, no Capítulo 5, os resultados obtidos são analisados. Por fim,uma breve discussão é realizada no Capítulo 6 juntamente com propostas de trabalhos futuros.

2 REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão dos principais métodos de controle usados na operação de veículos inteligentes, como técnicas que requerem elevados recursos computacionais para processamento e técnicas que devem ter o mínimo de tempo de execução. Uma ênfase é dada a métodos que utilizam câmeras digitais como sensores, abordando algumas formas de converter imagens em sinais mais simples. Também ilustra alguns dos simuladores utilizados no desenvolvimento de sistemas para veículos inteligentes.

2.1 Técnicas de controle de veículos

O controle clássico lida com comportamento dinâmico realizando comparações entre o valor de saída e uma referência com o objetivo de minimizar a diferença entre esses valores. A utilização consiste em representar o sistema por funções de transferências relacionando entradas e saídas. A principal aplicação em veículos é para o controle direto dos atuadores como em Alonso et al. (2013) que usou um controlador Proporcional-Integral-Derivativo (PID) para regular a distância entre veículos, controlando sua velocidade linear. No entanto, devido à dependência de modelos matemáticos, esse método somente é viável com a definição de tal modelo.

Utilizando inteligência computacional, o controle inteligente tem como objetivo simular os comportamentos de um humano. Nesse contexto, modelos matemáticos não são necessários, porém uma base de dados com representações das situações às quais o veículo será submetido é crucial para o funcionamento. Das técnicas de controle inteligente, se destacam (TAVARES, 2017):

- **Lógica *fuzzy***: uma forma robusta de obter um controle contínuo e suave mesmo na presença de erros de medição.
- **Algoritmos genéticos**: que são algoritmos de evolução e seleção natural com objetivo de otimização, podendo ser combinados com outras técnicas para ajuste de parâmetros a fim de funcionar corretamente em diferentes cenários.
- **Rede neurais**: normalmente relacionadas com tarefas de classificação, reconhecimento de padrões e mapeamento e, por possuir uma estrutura simples, sua operação requer poucos recursos computacionais, no entanto, pode demandar um tempo maior para realizar seu treinamento.

Por outro lado, o controle moderno utiliza modelos em espaço de estados, representados em sistemas matriciais, onde é possível analisar múltiplas variáveis de entrada e controlar diversas saídas. Das técnicas aplicadas em veículos as mais relevantes são (TAVARES, 2017):

- **Controle preditivo**: que por meio de modelos matemáticos estima a evolução do sistema, sendo uma alternativa para solução de problemas não-lineares e com elevada dinâmica, mas devido ao seu alto custo computacional, aplicações em tempo real podem requerer hardwares de alta performance para o controle estimar os estados em tempo hábil.

- **Controle robusto**: abordando explicitamente as incertezas envolvidas, sua função é manter a estabilidade de suas múltiplas variáveis mesmo na presença de ruídos e distúrbios.

A Tabela 2.1 representa de forma sucinta em qual sistema cada método de controle é utilizado em um veículo inteligente. Nessa tabela também é apresentado o controle referenciado em sensores, que será abortado na próxima seção.

Tabela 2.1 – Técnicas de controle utilizadas em cada sistema necessário em um veículo inteligentes.

	Clássico	**Inteligente**	**Moderno**	**Referenciado em sensores**
Controle reativo		X		
Controle deliberativo	X	X	X	X
Controle longitudinal	X	X	X	
Controle lateral	X	X	X	X
Navegação local		X		X
Navegação global	X	X	X	
Comportamento humano		X		
Desvio de obstáculos		X	X	
Baixo custo computacional	X			X

Fonte: adapdação de Tavares (2017)

2.2 Controle referenciado em visão

Controle referenciado em sensores é uma estratégia em que a regulação do erro é realizada no referencial de um sensor (LIMA; VICTORINO, 2016). Nesse contexto,

em sistemas de visão, a regulação do erro é realizada no plano da imagem adquirida, conhecida como controle referenciado em visão (VS)[1].

Em sistemas de navegação, o controle referenciado em visão é uma maneira robusta de se orientar um veículo em vias urbanas, já que não é necessário sistema de localização para sua operação. Há diversas formas de abordar o controle referenciado em visão, sendo as principais, referenciado em posição (PBVS)[2] e referenciado em imagem (IBVS)[3] (CHAUMETTE; HUTCHINSON, 2007).

O PBVS consiste em usar câmeras, monovisão ou estereovisão, para estimar informações em um espaço cartesiano tridimensional onde o veículo está contido. Sendo assim, o erro é tratado no espaço tridimensional da câmera. Esse método converge para resultados apropriados, mesmo em situações que apresentam grandes incertezas nas estimativas dos recursos (SILVEIRA, 2003). Já no IBVS, as informações são tratadas de acordo com as medições no plano da imagem, tornando-se mais robusto a erros de calibração da câmera (SILVEIRA, 2003). Aplicando uma lei de controle às velocidades do veículo e combinando a um rastreamento de características na imagem, o controle referenciado em visão é uma opção robusta para navegação local e orientação do veículo (LIMA; VICTORINO, 2016).

Controles referenciados em visão também são utilizados em outros sistemas. Como em Silveira, Malis e Rives (2008), que usa o IBVS para mapeamento e localização simultâneos (SLAM)[4], e mesmos com relativas variações de iluminação, alcança uma boa precisão.

2.3 Ferramentas de simulação para robótica móvel

Existe diversas ferramentas de simulação para auxiliar o desenvolvimento de veículos inteligentes. Esta seção realiza uma breve comparação entre as mais utilizadas, comentando sobre seus funcionamentos. Dentre os *softwares* de simulação se destacam o *Matlab Robotics Toolbox*, *CARLA*, *Stage*, *Drive sim* e o Gazebo.

O *Matlab Robotics System Toolbox* fornece ferramentas e algoritmos para projetar, simular, testar e implantar aplicativos para robôs móveis (MATHWORKS, 2022).

[1] do inglês *Visual Servoing.*
[2] do inglês *Position-Based Visual Servoing.*
[3] do inglês *Image-Based Visual Servoing.*
[4] do inglês *Simultaneous Localization and Mapping.*

Sendo parte dos produtos comerciais da *MathWorks*, ele é totalmente integrado ao *MatLab*, explorando recursos nativos do *software*, como os algoritmos de álgebra linear, de portabilidade e de criação de gráficos. Por ser um produto, sua licença de uso implica em custos, contudo, há soluções similares e de código aberto, como a *Robotics Toolbox for Python* (HAVILAND; CORKE, 2022), que tem a mesma função, porém implementada para ser utilizada em Python.

O *CARLA*[5] é um simulador de código aberto para pesquisa de direção autônoma (DOSOVITSKIY et al., 2017). Desenvolvido para treinamento e validação de veículos autônomos, o simulador suporta uma ampla variedade de sensores e condições ambientais. Por usar a *Unreal Engine* 4 (UE4), utilizada para desenvolvimento de jogos, como ferramenta de renderização e física, o CARLA permite a criação de simulações com alto grau de realismo. Entretanto, devido a esse realismo, o simulador requer um *hardware* com configurações robustas.

Stage é um simulador que permite o desenvolvimento rápido de controladores, que eventualmente conduzirão robôs reais (GERKEY; VAUGHAN; HOWARD, 2003). Projetado para apoiar a pesquisa em sistemas multiagentes, o *Stage* permite experimentos com grandes populações de robôs com uma boa fidelidade tornando-o adequado para testar diversas situações simultaneamente.

Drive Sim é uma plataforma de simulação desenvolvido pela *NVIDIA* (NVIDIA CORPORATION, 2022). Utilizando o *hardware* dedicado à aceleração de *ray-tracing*, é capaz de simular sensores com alta precisão e em tempo real. No entanto, esse simulador é dependente de um *hardware* específico (placa gráfica RTX 3090), que é excessivamente caro.

O Gazebo é uma ferramenta de código aberto utilizada para o desenvolvimento de simulações robóticas (OPEN ROBOTICS, 2022a). Com ele, é possível simular uma variedade de cenários sem a necessidade de utilizar equipamentos custosos. Com mecanismo de física e renderização, o Gazebo permite simulações realistas e precisas. Por conseguir realizar a simulação de maneira distribuída (uso de vários computadores simultaneamente), consegue um alto desempenho, conseguindo executar simulações complexas em tempo real.

[5] Carro Aprendendo a Agir, do inglês *Car Learning to Act*.

Além dos simuladores, há *softwares* que auxiliam na implementação de aplicações para a robótica em geral, como o *Player*, o *PACPUS*[6], e o ROS. Tais *softwares*, quando usados para o desenvolvimento de simulações simplificam a substituição da interface simulada para a utilização em um robô físico.

O *Player* é um servidor que oferece uma interface TCP que permite o controle de atuadores robóticos (GERKEY; VAUGHAN; HOWARD, 2003). Com a simplicidade de transmitir mensagens por protocolo TCP, o *Player* possui bibliotecas clientes implementadas em uma diversidade de linguagens (C, C++, Python, Java, entre outras), e também pode ser executado em qualquer sistema operacional atual. A abstração de protocolo torna o *Player* independente da localização dos dispositivos físicos, podendo exercer controle sobre qualquer dispositivo conectado à rede. Entretanto, devido ao uso do protocolo TCP, o sistema pode sofrer de baixa latência e sobrecarga quando experimentam uma alta demanda.

PACPUS é uma plataforma de pesquisa do Heudiasyc (HEUDIASYC, 2022). Desenvolvida para realizar experimentos com veículos inteligentes e robôs, o objetivo era projetar uma estrutura para integração de sistemas. O *PACPUS* foi inicializado em 2006, antes da existência dos principais *softwares* de integração que são usados atualmente, tornando obsoleto frente aos algoritmos mais otimizados dos sistemas atuais.

O ROS é um conjunto de bibliotecas de *software* e ferramentas que ajudam a construir aplicações robóticas (OPEN ROBOTICS, 2022c). Usar o ROS como base é uma maneira simples de desenvolver sistemas robóticos complexos. Com um código aberto, e uma comunidade ativa de usuários que contribuem para o melhoramento do *framework*, o ROS dispõe de inúmeros algoritmos implementados e otimizados para diversas aplicações robóticas. O ROS é constantemente atualizado, entretanto, essas atualizações vem em formas de versões novas, que normalmente , apesar de apresentarem relativamente poucas alterações (a nível do usuário), são incompatíveis com outras versões.

Apesar dos simuladores e *frameworks* existentes, há ferramentas para desenvolvimento de simuladores para aplicações próprias e específicas. Como Canteri (2011), que usando ferramentas de desenvolvimento de jogos, desenvolveu um ambiente urbano para controle de veículo autônomo.

[6] Percepção e Assistência para uma Condução Mais Segura, do francês *Perception et Assistance pour une Conduite Plus Sûre*.

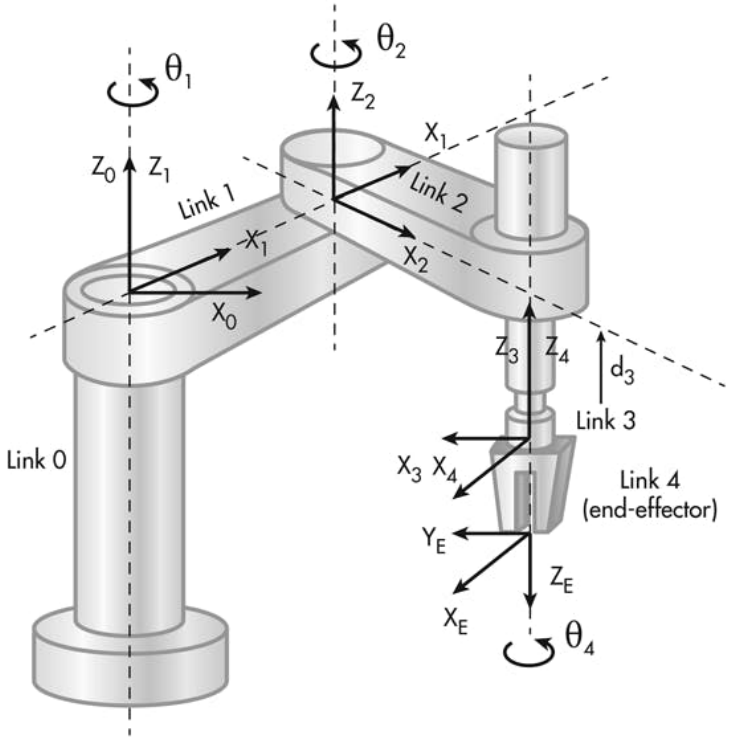
3 CONCEITOS GERAIS

Este capítulo aborda os *frameworks* e *softwares* utilizados no desenvolvimento deste trabalho, apresentando a metodologia para implementar um ambiente de simulação no Gazebo, detalhando seus componentes e características. É descrito, também, os elementos utilizados pelo ROS de modo que possa interagir com o Gazebo.

3.1 Gazebo

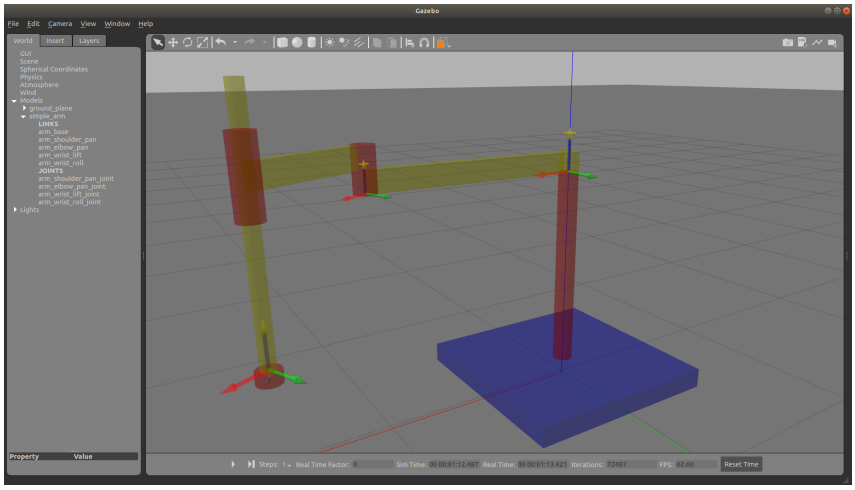
Como citado na Seção 2.3, o Gazebo é uma ferramenta utilizada para o desenvolvimento de simulações robóticas. Para simular um robô no Gazebo, deve-se criar um modelo descritivo, enumerando seus componentes construtivos, seus sensores e seus controladores. Seus componentes construtivos, assim como no estudo teórico de robótica, são definidos por *links* e juntas, como na Figura 3.1 que mostra uma representação de um manipulador $SCARA$. Na Figura 3.2 é mostrado o mesmo tipo de manipulador no Gazebo, e a Figura 3.3 apresenta um resumo do código fonte que o descreve.

Figura 3.1 – Manipulador tipo $SCARA$, caracterizado por possuir um corpo composto por duas juntas rotativas paralelas (θ_1 e θ_2) e uma prismática (d_3) que controla o movimento do elemento final ($Link_4$).



Fonte: New Equipment Digest (2016)

Figura 3.2 – Manipulador tipo *SCARA*, simulado no Gazebo.



Fonte: Do autor (2022)

Figura 3.3 – Código fonte que descreve um manipulador *SCARA* no Gazebo, destacando seus *links* e juntas.

```
<?xml version='1.0'?>
<sdf version='1.6'>
  <model name='simple_arm'>
    <link name='arm_base'>…
    </link>
    <link name='arm_elbow_pan'>…
    </link>
    <link name='arm_shoulder_pan'>…
    </link>
    <link name='arm_wrist_lift'>…
    </link>
    <link name='arm_wrist_roll'>…
    </link>
    <joint name='arm_elbow_pan_joint' type='revolute'>…
    </joint>
    <joint name='arm_shoulder_pan_joint' type='revolute'>…
    </joint>
    <joint name='arm_wrist_lift_joint' type='prismatic'>…
    </joint>
    <joint name='arm_wrist_roll_joint' type='revolute'>…
    </joint>
  </model>
</sdf>
```

Fonte: Do autor (2022)

No Gazebo, os sensores são classificados em 3 tipos:

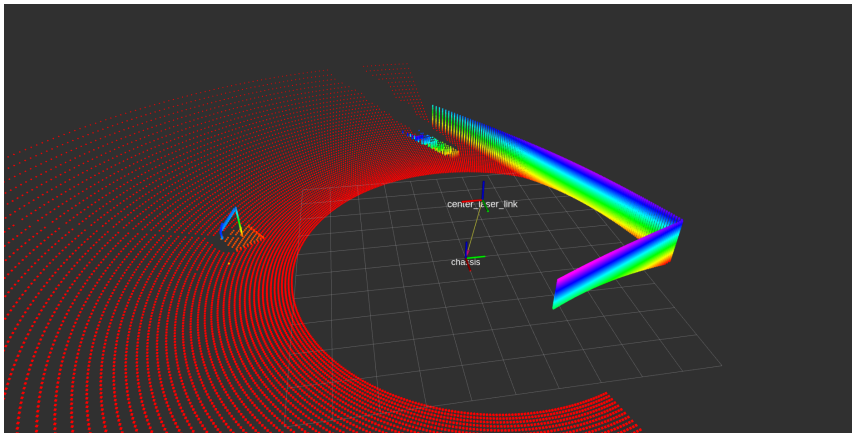
1. ***Ray***

   São feixes que imitam o funcionamento de sensores ultrassônicos ou a *lazer*, também conhecidos como *LIDARs*[1]. Baseado em medir o tempo entre a emissão e retorno a um receptor de um pulso sonoro ou luminoso. Com esse tempo é possível

[1] Do inglês *Light Detection and Ranging.*

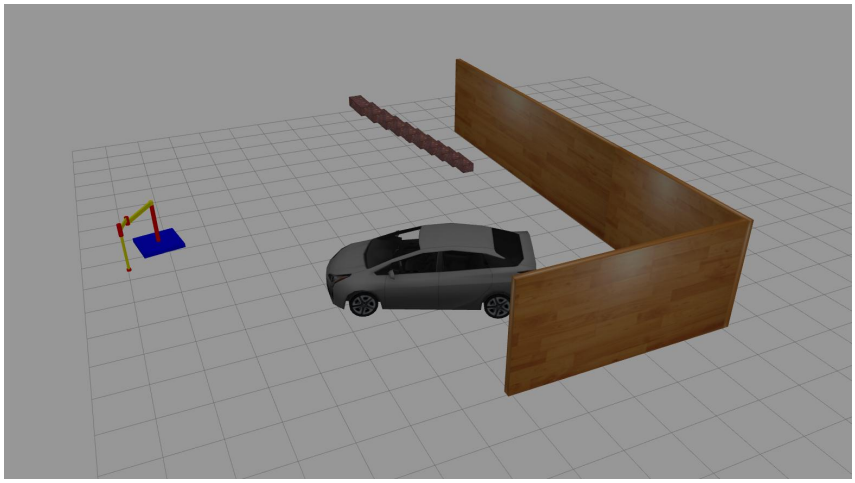
determinar a distância entre sensor e a superfície que reflete o pulso. A distinção dos sensores é definida por suas características construtivas, como alcance, campo de visão, tempo de aquisição, resolução e ruído. A Figura 3.4 mostra uma visualização das distâncias medidas por sensores a laser e na Figura 3.5 há a demonstração dos objetos medidos juntamente com um veículo no Gazebo.

Figura 3.4 – Visualização da nuvem de pontos obtidos por um sensor a laser no Gazebo. Cada ponto é a representação de uma medida de distância e as suas cores apresentam os diferentes planos de altura dos obstáculos.



Fonte: Do autor (2022)

Figura 3.5 – Obstáculos usados para demonstração do sensor a laser mostrado na Figura 3.4.



Fonte: Do autor (2022)

2. ***Camera***

No Gazebo também é possível simular sensores que captam imagens do ambiente. E, como os sensores a laser, é possível definir seus parâmetros de medição. Além dos mesmos parâmetros do laser, as câmeras precisam definir as dimensões da

imagem. Outra funcionalidade importante no uso de câmeras é a capacidade de gerir duas câmeras em sincronia, ou seja câmeras estéreo, e isso também pode ser simulado no Gazebo.

3. ***IMU***

   Os sensores inerciais (IMU)[2] são usados para a aquisição de acelerações lineares e velocidades angulares, simulando acelerômetros e girômetros.

Outro tipo de sensor, usado na robótica móvel, é o de posicionamento global do tipo GNSS[3], no entanto, não há esse sensor no Gazebo. Porém, por meio de *plugins*, é possível obter a posição do robô em relação ao eixo de origem do Gazebo, e obviamente, adicionar ruído nesses dados para gerar um sinal um pouco mais realístico. Com *plugins*, também é possível controlar os atuadores controlando forças e torques aplicados nas juntas do robô, causando seu deslocamento. Tais sinais de controle podem ser enviados pelos *plugins* por meio de *topics* do ROS.

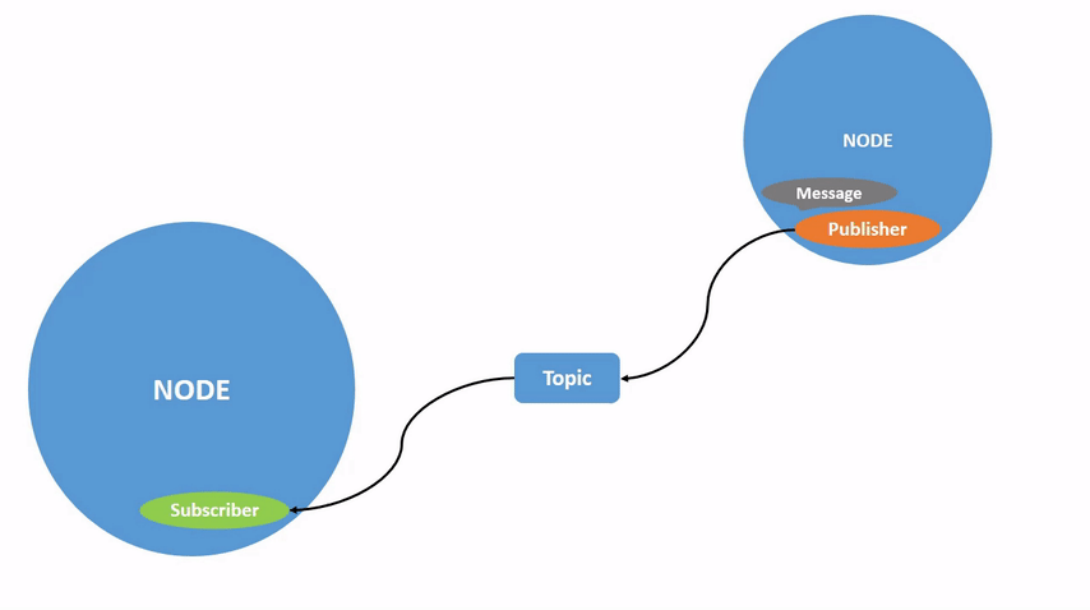
3.2 ROS

O ROS é um conjunto de bibliotecas de código aberto utilizado para o desenvolvimento de aplicações robóticas (OPEN ROBOTICS, 2022c). Permitindo a integração de diferentes pacotes ao sistema, o ROS torna as aplicações modulares.

A arquitetura do ROS é baseada no ROS *graph*, que é uma rede de *nodes*. *Node* é um elemento capaz de se comunicar com outros *nodes*, sendo que podem fazer parte de um mesmo processo, processos diferentes ou até em máquinas diferentes (OPEN ROBOTICS, 2022b). A forma mais comum dos *nodes* se comunicarem é utilizando *topics*, com eles os dados são transmitidos por *messages*. Para enviar uma *message* o *node* deve conter um *publisher*, e os *nodes* que a recebem devem conter um *subscriber*. Evidentemente, o formato da *message* de um *publisher* e o respectivo *subscriber* devem ser o mesmo. A Figura 3.6 ilustra a forma que um *topic* transmite a *message* entre um *publisher* e um *subscriber*.

[2] Do inglês *Inertial Measurement Units.*
[3] Do inglês *Global Navigation Satellite System.*

Figura 3.6 – Transmissão de dados entre *nodes* utilizando *topic*. O *publisher* altera a *message* do *topic* notificando o *subscriber* sobre a alteração realizada.

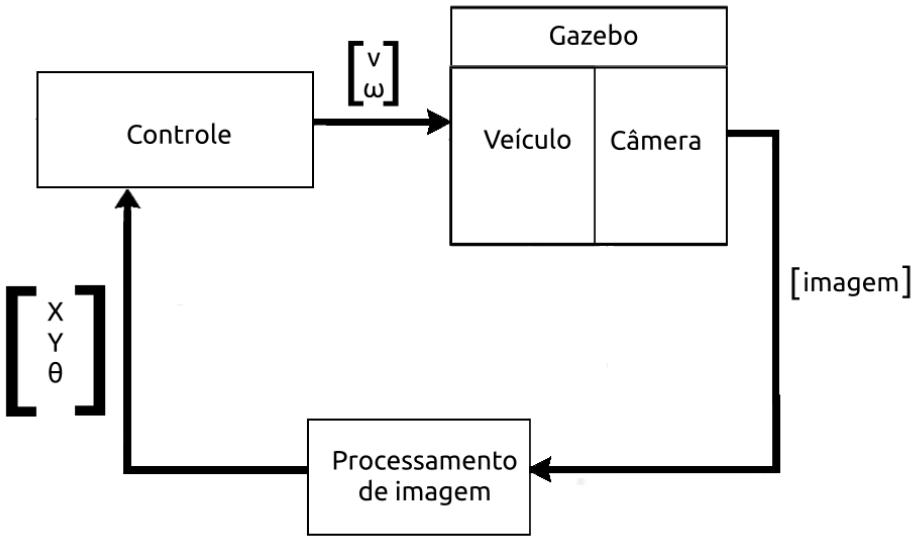


Fonte:Open Robotics (2022d)

4 METODOLOGIA

Este capítulo explora os procedimentos realizados para aplicar um controle baseado em visão em um veículo simulado no ambiente Gazebo utilizando o ROS. Nele é descrito o modelo matemático para o deslocamento do veículo, o algoritmo de processamento de imagens para obtenção de informação e o controle para realização da tarefa proposta. A Figura 4.1 sintetiza o sistema utilizado, o qual está melhor detalhado a seguir.

Figura 4.1 – Diagrama de blocos do sistema proposto, onde cada bloco corresponde a um *node* ROS.



Fonte: Do autor (2022)

4.1 Modelagem

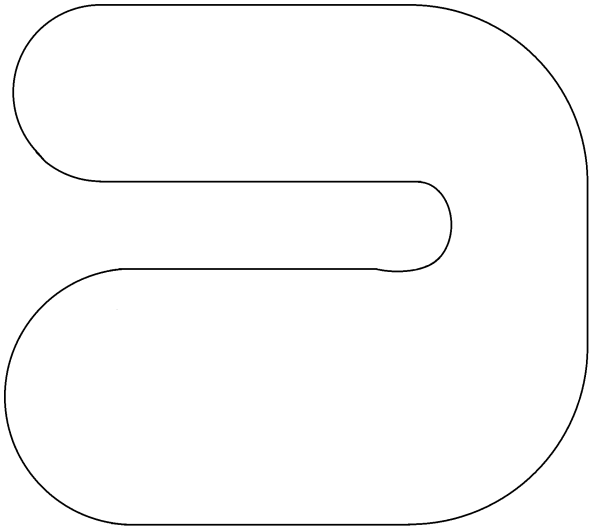
Na Seção 3.1 foi mostrado que um objeto no Gazebo deve ser descrito em relação aos seus componentes físicos, por elementos visuais, elementos de colisão e sensores integrados. Também foi descrito que as interações utilizando o ROS são feitas utilizando *plugins*, os quais são usados como *nodes* do ROS. Assim, nesta seção, é mostrado o planejamento e montagem do ambiente no Gazebo e definido um modelo matemático para o controle do veículo bem como a forma de aplicá-lo na simulação.

4.1.1 Ambiente e modelo físico

O mundo no Gazebo é composto por dois elementos principais, o modelo de veículo e a pista de teste (ambiente). Para simplificar o processamento de imagem, a pista de teste

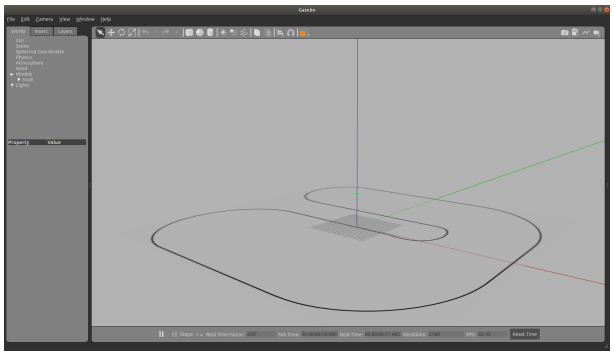
foi definida como uma linha contínua, em um circuito fechado e com uma cor em destaque em relação ao resto do ambiente. Uma exemplificação de imagem para criação de uma pista e sua representação no ambiente de simulação são, respectivamente, apresentadas nas Figuras 4.2 e 4.3.

Figura 4.2 – Exemplo de imagem usada para criação da pista de teste.



Fonte: Do autor (2022)

Figura 4.3 – Exemplo da pista de teste no ambiente Gazebo.

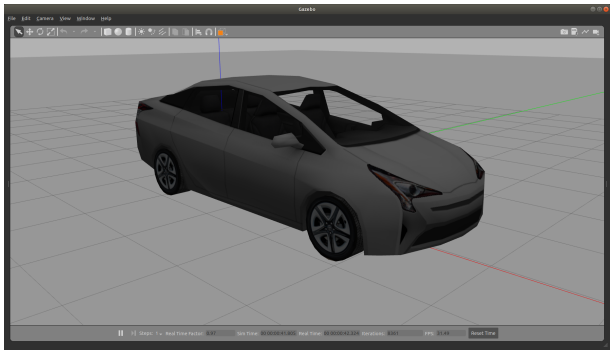


Fonte: Do autor (2022)

O modelo físico utilizado é um veículo convencional com quatro rodas, com tração e esterçamento no eixo dianteiro. Ele tem como base o chassi de um Toyota Prius, disponível na biblioteca Gazebo, mostrado na Figura 4.4.

Para perceber o ambiente em que o veículo está inserido, uma câmera é instalada no para-brisa do veículo sobre o seu eixo de simetria. A câmera possui também uma ligeira inclinação em relação ao plano da superfície, requisito necessário para a técnica de controle referenciado no plano da imagem. A Figura 4.5 mostra um exemplo de imagem

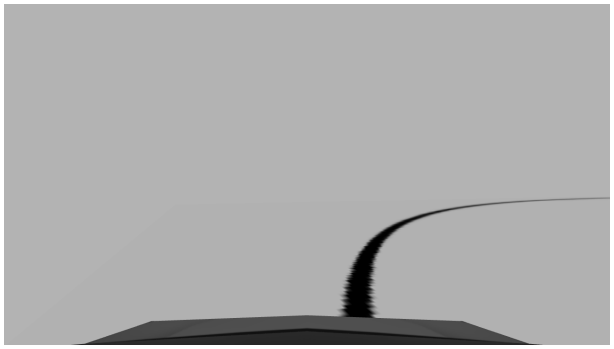
Figura 4.4 – Veículo de testes baseado no chassi de um Toyota Prius, disponibilizado na biblioteca do Gazebo.



Fonte: Do autor (2022)

adquirida pela câmera e na Figura 4.6 sua pose (posição e orientação) é representada pelo referencial $\{\mathcal{C}\}$.

Figura 4.5 – Imagem capturada pela câmera mostrando a visão da pista apresentada na Figura 4.3.



Fonte: Do autor (2022)

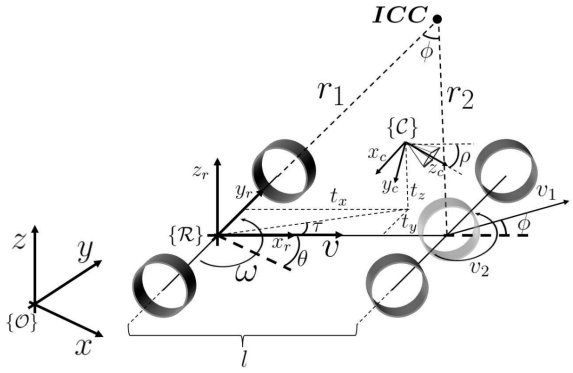
4.1.2 Modelo matemático

Para representar o movimento do veículo foi considerado um modelo cinemático, o qual despreza a ação das forças no veículo. Ele é adequado para baixas velocidades apenas e, para o problema em questão, serve como uma validação inicial dos métodos propostos. No entanto, vale ressaltar que o Gazebo considera elementos dinâmicos para a representação de sua simulação, como resistência do ar e atrito entre o solo e os pneus.

O modelo escolhido é o de Ackermann, semelhante ao usado por Lima e Victorino (2016). A Figura 4.6 mostra o modelo e os principais referenciais utilizados. O movimento do veículo é definido pela Equação (4.1), que apresenta como entradas a velocidade linear

das rodas dianteiras do veículo (v_1) e a velocidade angular do esterçamento da roda virtual central (v_2).

Figura 4.6 – Diagrama do modelo cinemático do veículo referenciado em $\{\mathcal{R}\}$. O referencial $\{\mathcal{C}\}$ representa a posição da câmera usada para aquisição de imagens.



Fonte: Lima e Victorino (2016)

$$\begin{aligned} \dot{x} &= v_1 \; cos\theta \; cos\phi, \\ \dot{y} &= v_1 \; sen\theta \; cos\phi, \\ \dot{\theta} &= \frac{v_1}{l} \; sen\phi, \\ \dot{\phi} &= v_2, \end{aligned} \tag{4.1}$$

onde os estados x, y e θ são a pose do referencial do veículo, $\{\mathcal{R}\}$, relativo a um referencial global, $\{\mathcal{O}\}$, e o estado ϕ é o ângulo de esterçamento da roda virtual central. A constante l é a distância entre os eixos do veículo.

Como $\dot{\theta}$ é a velocidade angular (ω), é possível obter o ângulo de esterçamento da roda virtual central usando:

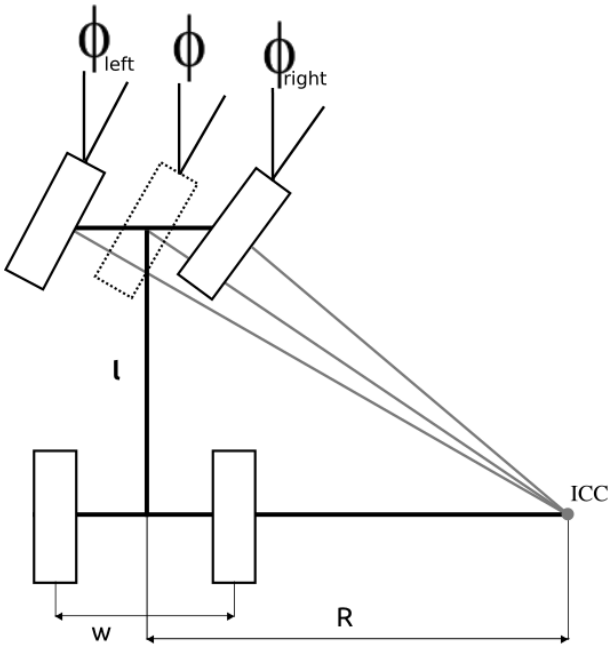
$$\phi = arcsen\frac{\omega l}{v_1} \tag{4.2}$$

Com ϕ, aplicando-o na geometria de Ackermann, obtêm-se os distintos ângulos para cada roda (Figura 4.7), que são definidos por:

$$\begin{aligned} \phi_{right} &= tg^{-1}\frac{2 \; l \; sen\phi}{2 \; l \; cos\phi + w \; sen\phi}, \\ \phi_{left} &= tg^{-1}\frac{2 \; l \; sen\phi}{2 \; l \; cos\phi - w \; sen\phi}, \end{aligned} \tag{4.3}$$

onde w é o tamanho do eixo.

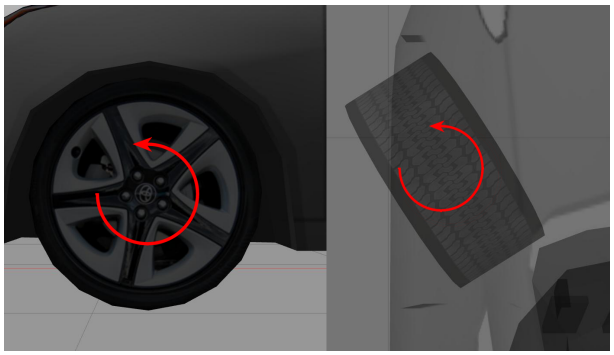
Figura 4.7 – Vista superior da geometria de Ackermann utilizada para definir os ângulos de esterçamento das rodas (ϕ_{right} e ϕ_{left}), utilizando um único ângulo de controle (ϕ). ICC é o centro instantâneo de curvatura apresentando um raio R.



Fonte: Adaptação de PNGWING (2022)

Para aplicar o modelo matemático no ambiente de simulação, é necessário fazer algumas conversões, tendo em vista que no Gazebo usa-se aplicação de torques nas rodas para movimentação do veículo. A Figura 4.8 mostra os torques aplicados para realização dos diferentes movimentos das rodas dianteiras.

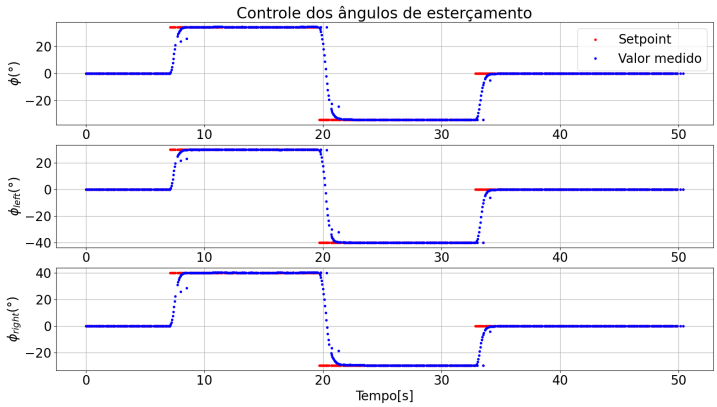
Figura 4.8 – Torques utilizados para movimentar o veículo. O torque representado na vista lateral à esquerda realiza o movimento linear do veículo e o da direita com vista superior para realização do esterçamento.



Fonte: Do autor (2022)

Com os ângulos ϕ_{right} e ϕ_{left}, um controlador PID é implementado para o controle individual dos torques, necessários para deslocar a respectiva roda para a posição desejada. A resposta desse controlador é apresentada na Figura 4.9.

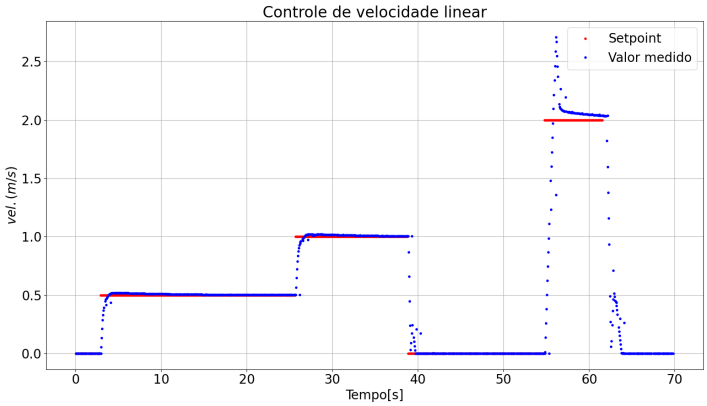
Figura 4.9 – Resposta do controle de esterçamento do veículo para cada roda da geometria de Ackermann.



Fonte: Do autor (2022)

Para o controle de v_1, por sua vez, é implementado um controlador PID com a saída deste sendo a aceleração linear nas rodas dianteiras, que é convertida em um torque proporcional a ser aplicado no eixo de tração. Na Figura 4.10 é apresentada a resposta desse controlador.

Figura 4.10 – Resposta do controle de velocidade linear do veículo.



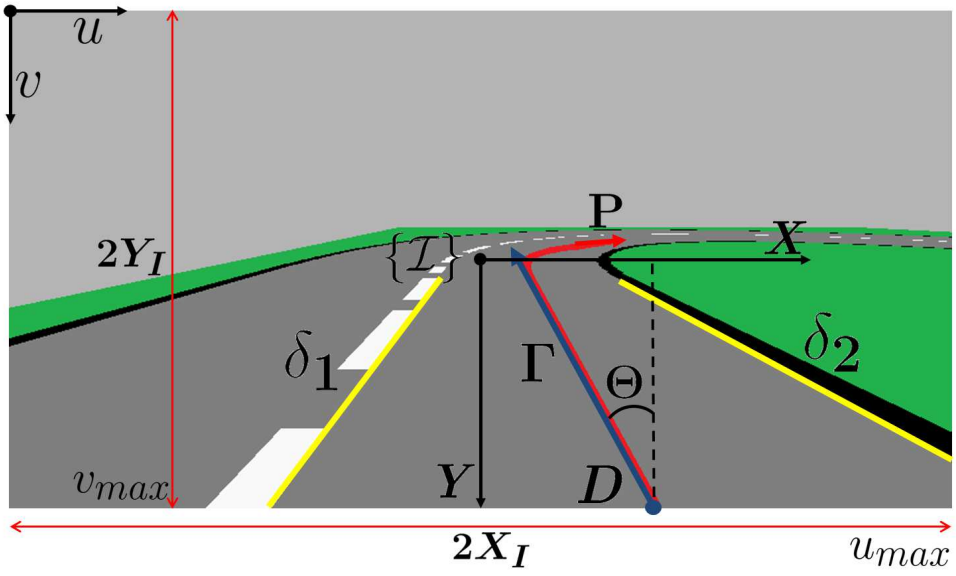
Fonte: Do autor (2022)

Pela resposta obtida, ambos os controladores foram considerados adequados para aplicação neste trabalho.

4.2 Extração de características

O processamento de imagens não é objetivo deste trabalho, mas é necessário citar como as características para o controle referenciado em visão são extraídas da imagem. Segundo Lima e Victorino (2016), as características necessárias são o ponto de referência D e o ângulo Θ, que é o ângulo da reta tangente à trajetória P no ponto D em relação ao eixo -Y do plano da imagem nesse ponto. A Figura 4.11 mostra como Lima e Victorino (2016) definiram essas características.

Figura 4.11 – Imagem de extração de características usada por Lima e Victorino (2016), referenciadas no plano da imagem $\{\mathcal{I}\}$, sua trajetória P (em vermelho) sendo o centro da faixa de rolagem, com extremidades em, δ_1 e δ_2 (em amarelo), a reta Γ (em azul) é a reta tangente a P no ponto de análise D, seu ângulo entre o eixo $-Y$ é definido como Θ.



Fonte: Lima e Victorino (2016)

Com a pista de teste planejada para facilitar o processo de extração das características, as imagens adquiridas pela câmera já apresentam a trajetória, e consequentemente o ponto P, em destaque. Dessa forma, a rotina do *node* começa com a aquisição da imagem usando um *subscription* no *topic* da câmera, e com um pré-processamento para reduzir as dimensões da imagem. Após a redução das dimensões da imagem, define-se uma região de interesse e reduz-se a sua escala.

Um algoritmo de *skeletonize*, para encontrar o esqueleto de uma região em destaque em uma imagem é usado para que a largura da trajetória tenha apenas um pixel. Assim, a trajetória P pode ser representada na forma de uma função:

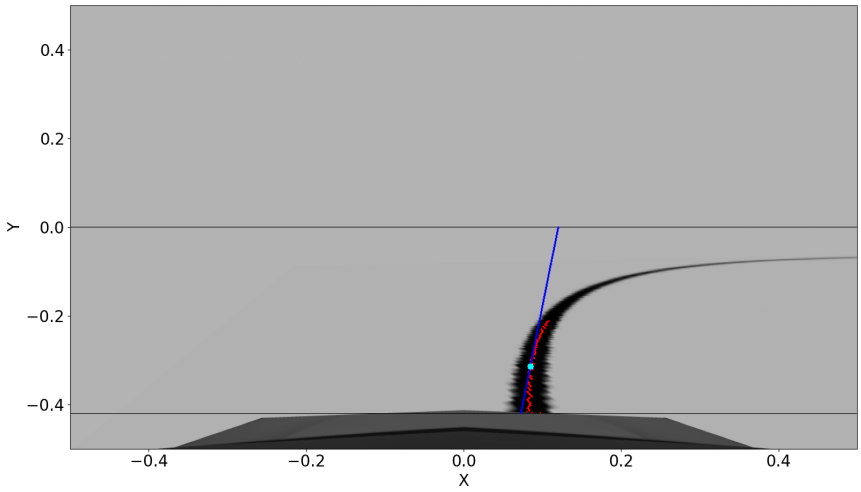
$$X = F(Y). \tag{4.4}$$

Então o valor de Y do ponto D será o maior valor em que houver um valor para X na imagem. O ângulo tangente ao caminho Θ é definido de forma que, em uma pequena região nas proximidades do ponto D, a função da trajetória seja:

$$X = \Theta Y + B. \tag{4.5}$$

Logo, a Figura 4.12 mostra os resultados da extração de características aplicados na imagem da Figura 4.5.

Figura 4.12 – Resultado da extração de características, onde o trajeto P são os pontos vermelhos, o ponto de análise D destacado em ciano e sua tangente à trajetória em azul, cujo inclinação define o ângulo Θ.



Fonte: Do autor (2022)

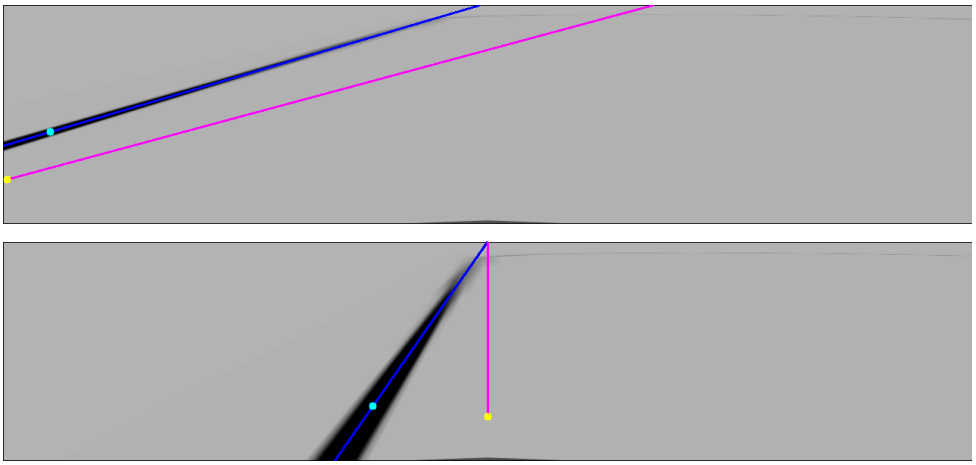
4.3 Controle referenciado em visão

O objetivo do controle referenciado em visão é, com base nas informações originadas na imagem, gerar sinais de controle para o veículo seguir o caminho na pista, corrigindo o erro das características visuais e levando-as para um *setpoint* específico. No caso, o *setpoint* escolhido é a linha preta, que deve ser alinhada sobre o eixo de simetria do veículo. As formulações desse controlador, replicadas neste trabalho, estão apresentadas em Lima e Victorino (2016). Considerando que o controle de velocidade linear,

mencionado anteriormente, garante uma velocidade constante ao veículo, o controlador referenciado em visão é responsável por definir o *setpoint* da velocidade angular.

A função do controle referenciado em visão é manter o ponto de referência D centralizado na parte inferior da imagem (Figura 4.13), e o ângulo tangente Θ alinhado com a vertical. Para realizar essa tarefa, o controle deve minimizar o erro entre o valores lidos $s = [X \quad Y \quad \Theta]^T$ e os valores de referência $s^* = [X^* \quad Y^* \quad \Theta^*]^T$. O valores de s^* apresentam-se de duas formas distintas, dependentes dos valores de s. Caso s_Y não esteja no inferior da imagem, s^* apresentará valores s^*_Y, em um controlador Coluna, caso contrário, a função será centralizar o valor de s_X na imagem, por meio do controlador Linha. Na Figura 4.13, os erros e valores de referência são evidenciados.

Figura 4.13 – Valores de referências de cada controlador (Coluna e Linha), nas cores amarelo e rosa, comparado aos valores medidos, em ciano e azul. A imagem superior apresenta a referência do controlador Coluna e a inferior a referência do controlador Linha.



Fonte: Do autor (2022)

Dessa forma, s^* pode assumir os seguintes valores de *setpoint*:

$$s^*_X = \begin{bmatrix} 0 \\ Y_I \\ 0 \end{bmatrix}, \tag{4.6}$$

$$s^*_Y = \begin{bmatrix} X_I \\ Y_I \\ \Theta_I \end{bmatrix}, \tag{4.7}$$

com (4.6) o setpoint do controlador Linha e (4.7) o do controle Coluna.

Os valores de s estão referenciados no plano da imagem, logo, para definir a velocidade angular é necessário um tratamento para que esses valores sejam representados no referencial do veículo. A relação entre o referencial da imagem e o da câmera, descrita por:

$$\dot{s} = L_s u_C, \tag{4.8}$$

é usada para transformar os dados no plano da imagem para o referencial da câmera, onde u_C é o vetor de velocidades da câmera e $\dot{s}$ as velocidades das características na imagem. Nessa equação, L_s é a matriz de interação entre $\dot{s}$ e u_C definida por:

$$L_s = \begin{bmatrix} L_X \\ L_Y \\ L_\Theta \end{bmatrix} = \begin{bmatrix} \frac{1}{z_c} & 0 & \frac{X}{z_c} & XY & -1-X^2 & Y \\ 0 & \frac{-1}{z_c} & \frac{Y}{z_c} & 1+Y^2 & -XY & -X \\ \frac{C\rho C^2\Theta}{t_z} & \frac{C\rho C\Theta S\Theta}{t_z} & \frac{-\zeta C\rho C\Theta}{t_z} & -\zeta C\Theta & -\zeta S\Theta & -1 \end{bmatrix}, \tag{4.9}$$

onde z_c é a distância entre o centro focal da câmera e o plano da pista, t_z é a distância entre o referencial do veículo e o referencial da câmera no eixo z do veiculo, ρ é o ângulo de inclinação câmera em relação ao eixo x do veiculo, $C\Theta = cos(\Theta)$, $S\Theta = sen(\Theta)$, $C\rho = cos(\rho)$ e $\zeta = Ysen(\Theta) + Xcos(\Theta)$.

Dessa forma, as velocidade u_C podem ser expressas no referencial do veículo $\mathcal{R}$ pela relação:

$$u_C = {}^C T_R u_R, \tag{4.10}$$

onde ${}^C T_R$ é a transformada:

$${}^C T_R = \begin{bmatrix} T_v & T_\omega \end{bmatrix} = \begin{bmatrix} 0 & -t_x cos(\tau) \\ -sen(\rho) & t_y cos(\rho) \\ cos(\rho) & -t_y sen(rho) \\ 0 & 0 \\ 0 & -cos(\tau)cos(\rho) \\ 0 & -cos(\tau)sen(\rho) \end{bmatrix}, \tag{4.11}$$

com τ é o ângulo de inclinação da câmera em relação ao seu eixo y do veículo.

Assim, o controlador Linha de $\dot{s}$ pode ser relacionado com as velocidades do veículo utilizando:

$$\dot{s} = \begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_v v + \begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_\omega \omega. \tag{4.12}$$

A lei de controle para obtenção da velocidade angular do veículo e corrigir o erro ($e = s - s^*$) das características da imagem é então definida como:

$$\omega = -B_r^+(\lambda e + A_r v). \tag{4.13}$$

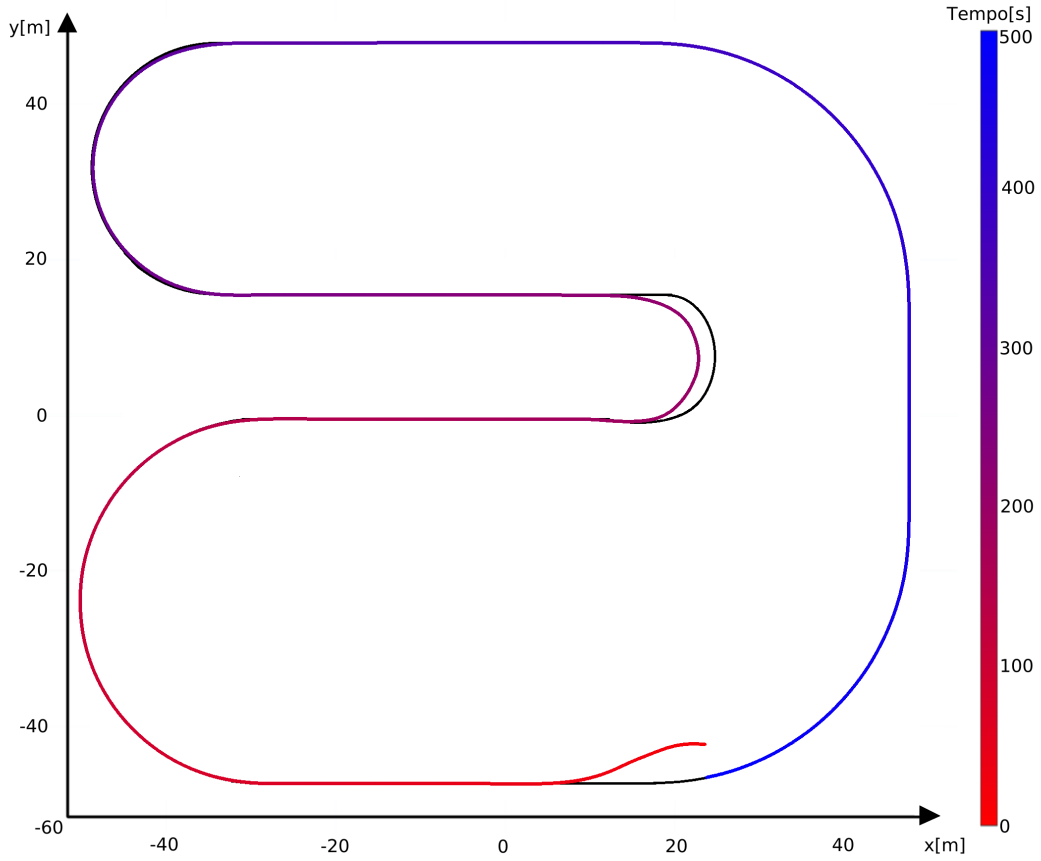
Sendo λ uma matriz de ganhos positivos, $A_r = \begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_v$, $B_r = \begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_\omega$ e B_r^+ a pseudo-inversa de Moore-Penrose de B_r. De forma equivalente, o controlador Coluna pode ser definido substituindo L_X por L_Y em (4.12).

Com ω definido, é possível controlar o veículo usando um *publisher* do ROS. Ele registra o comando de velocidades (v,ω) que serão então lidos pelo *node* de controle dos atuadores do veículo.

5 RESULTADOS

O controle proposto foi testado na pista de testes da Figura 4.2, começando de uma posição arbitrária à direita da pista. Na Figura 5.1 são mostradas as posições $[x, y]$ do veículo, obtidas pelo Gazebo, em função do tempo durante a execução de uma volta.

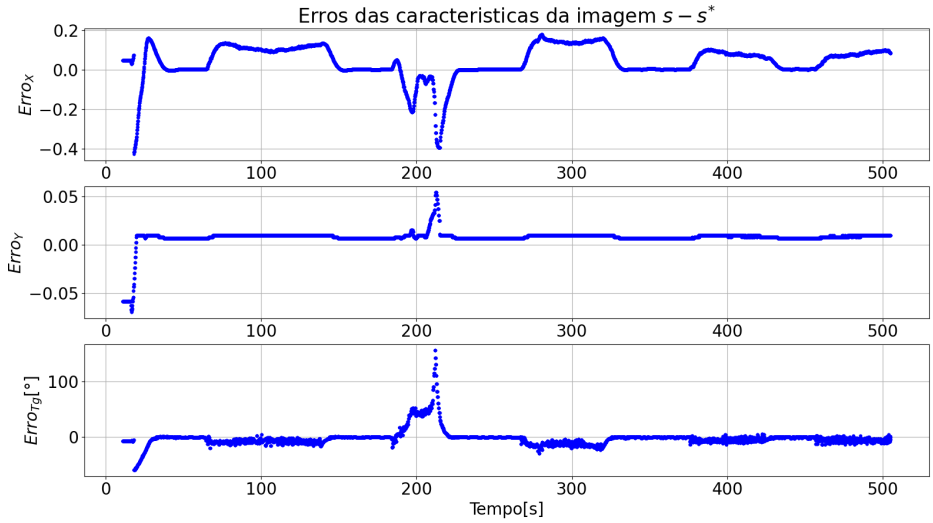
Figura 5.1 – Evolução da posição do referencial do veículo no tempo. As cores próximas do vermelho representam as posições iniciais do veículo e as azuis para as posições ao final da volta. A linha preta representa o mapa da pista utilizado.



Fonte: Do autor (2022)

Nota-se que o controle converge corretamente nas seções retas e nas curvas abertas. Na curva mais acentuada, o referencial apresenta uma leve diferença da pista, isso ocorre para que a câmera na frente do veículo não perca a referência da pista. Os erros obtidos são mostrados na Figura 5.2. É possível perceber picos proeminentes próximos a 200 segundos, onde, analisando o mesmo período de tempo na Figura 5.1, compreende-se que o comportamento é devido à curva mais fechada da pista. Comparando os tempos de oscilações de $Erro_X$ e comparando também com a Figura 5.1, essas oscilações ocorrem principalmente nas curvas.

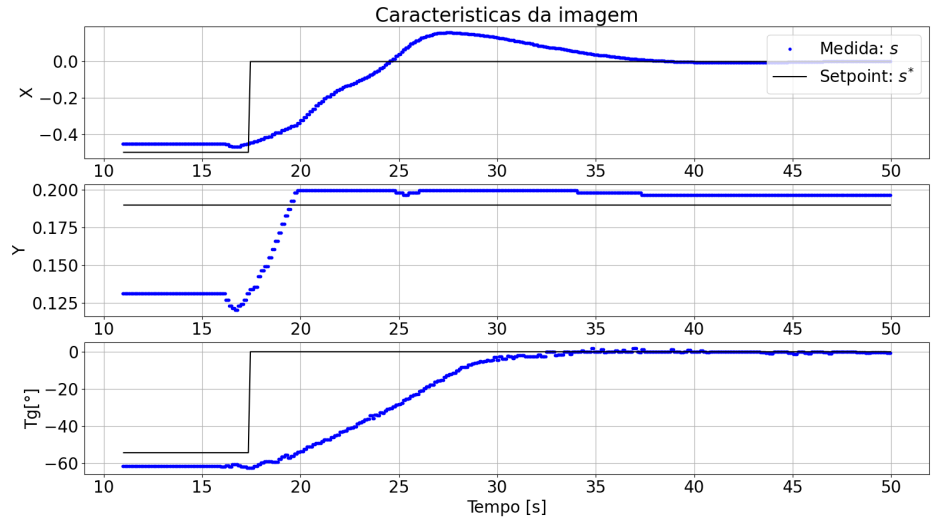
Figura 5.2 – Erros de cada característica, sendo notável picos onde ocorrem as curvas.



Fonte: Do autor (2022)

Os valores iniciais das características são apresentados com mais detalhes na Figura 5.3, juntamente com os respectivos *setpoints*. A mudança dos *setpoints* representa a troca entre o controlador Coluna para o controlador Linha, e, devido as configurações da pista, esse controle permanece até o fim da volta.

Figura 5.3 – Valor medido e *setpoint* de cada característica nos instantes iniciais para mostrar a troca de controlador Coluna para controlador Linha.

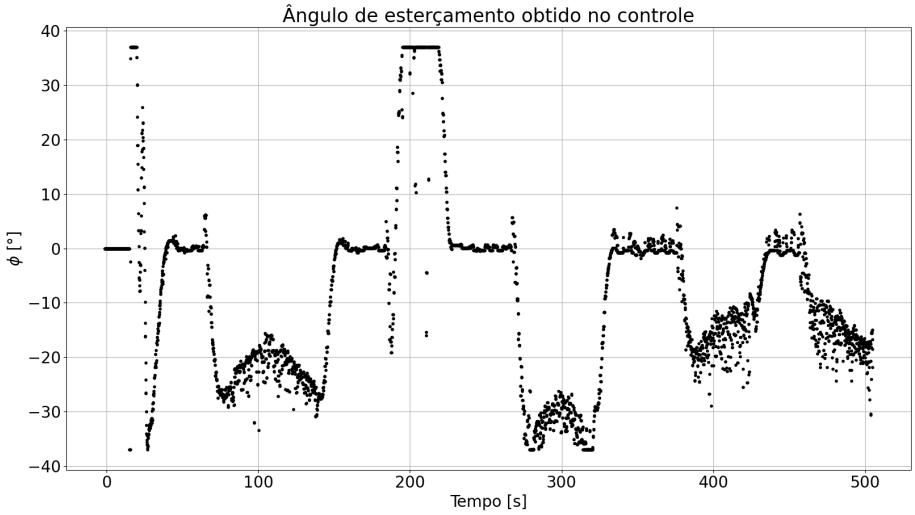


Fonte: Do autor (2022)

A Figura 5.4 mostra o ângulo de controle aplicado. Assim como nas figuras anteriores, o pico próximo a 200 segundos mostra o esforço para a curva mais fechada, chegando

a saturar o atuador. Os esforços para realização das curvas menos acentuadas também são visíveis.

Figura 5.4 – Ângulo de esterçamento aplicado pelo controle referenciado em visão.



Fonte: Do autor (2022)

6 CONSIDERAÇÕES FINAIS

Esse trabalho apresentou o desenvolvimento de um sistema de controle referenciado em visão para um veículo autônomo em um ambiente de simulação utilizando ROS e Gazebo. A técnica de controle referenciado em visão aplicada apresentou-se adequada, realizando o controle adequado para manter o veículo na trajetória definida. Entretanto, na curva mais acentuada, nota-se que o veículo saiu levemente da trajetória, devido à diferença de posição das características visuais na câmera e do veículo, assim para a câmera não perder a trajetória, o controle aumenta o esforço do atuador, chegando até saturar o ângulo de esterçamento. Uma forma de corrigir esse problema seria reduzir a velocidade do veículo nas curvas e aumentar a distância de projeção do ponto D da imagem no mundo. Isso tornaria o tempo de resposta do controle menor permitindo o controle reagir mais rápido. Ademais, o ambiente implementado neste trabalho contribuirá com as pesquisas do LMT, servindo como base para testes do veículo VIDA em ambiente simulado.

Como forma de dar continuidade ao que foi iniciado por este trabalho, há a possibilidade de detalhar mais o ambiente, para abordar situações mais diversas e adaptar o modelo do veículo para que corresponda ao chassi do VIDA.

REFERÊNCIAS

ALONSO, L. et al. Self-tuning PID controller for autonomous car tracking in urban traffic. **International Conference System Theory, Control and Computing**, ICSTCC, v. 17, p. 15–10, 2013.

CANTERI, R. dos P. **Solução para controle de veículos autônomos em um jogo de carro em ambiente urbano**. Ponta Grossa, 2011.

CHAUMETTE, F.; HUTCHINSON, S. Visual Servo Control - part II: Advanced Approaches. **IEEE Robotics and Automation Magazine**, IEEE, v. 14, n. 1, p. 109–118, 2007.

DOSOVITSKIY, A. et al. CARLA: An open urban driving simulator. In: **Proceedings of the 1st Annual Conference on Robot Learning**. [S.l.: s.n.], 2017. p. 1–16.

ESPIAU, B.; CHAUMETTE, F.; RIVES, P. A New Approach to Visual Servoing in Robotics. **IEEE Transactions on Robotics an Automation**, IEEE, v. 8, n. 3, p. 313–326, 1992.

GERKEY, B. P.; VAUGHAN, R. T.; HOWARD, A. The player/stage project: Tools for multi-robot and distributed sensor systems. In: **In Proceedings of the 11th International Conference on Advanced Robotics**. [S.l.: s.n.], 2003. p. 317–323.

HAVILAND, J.; CORKE, P. Robotics toolbox for python. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://petercorke.github.io/robotics-toolbox-python/index.html>.

HEUDIASYC. Pacpus. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://pacpus.hds.utc.fr/>.

LIMA, D. A. de; VICTORINO, A. C. A Hybrid Controller for Vision-Based Navigation of Autonomous Vehicles in Urban Environments. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 17, n. 8, p. 2310–2323, 2016.

LMT. Laboratório de Mobilidade Terrestre. 2022. Acesso em: 15 set. 2022. Disponível em: <http://www.lmt.ufla.br/>.

LMT. Vida, Veículo Inteligente de Desenvolvimento Aplicado. 2022. Acesso em: 15 set. 2022. Disponível em: <http://www.lmt.ufla.br/nlmt/vida-veiculo-autonomo-mobilidade/>.

MATHWORKS. Robotics system toolbox. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://www.mathworks.com/products/robotics.html>.

NEW EQUIPMENT DIGEST. What's the difference between industrial robots? 2016. Acesso em: 08 jul. 2022. Disponível em: <https://www.newequipment.com/industry-trends/article/22059066/whats-the-difference-between-industrial-robots>.

NVIDIA CORPORATION. Nvidia drive sim. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://developer.nvidia.com/drive/drive-sim>.

OPEN ROBOTICS. Gazebo features and benefits. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://gazebosim.org/features>.

OPEN ROBOTICS. Ros 2 concepts. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://docs.ros.org/en/foxy/Concepts.html>.

OPEN ROBOTICS. Ros 2 documentation. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://docs.ros.org/en/foxy/index.html>.

OPEN ROBOTICS. Ros 2 understanding topics. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://docs.ros.org/en/foxy/Concepts.html>.

PAULA, A. F. S. de et al. Por uma agência nacional de prevenção e investicação de acidentes de transportes. **Instituto de Pesquisa Econômica Aplicada**, 2021. Acesso em: 22 mar. 2022. Disponível em: <https://www.ipea.gov.br/portal/images/stories/PDFs/nota_tecnica/210527_nt_diset_n_81.pdf>.

PNGWING. Carro ackermann geometria da direção Ângulo do veículo. 2022. Acesso em: 08 jul. 2022. Disponível em: <https://www.pngwing.com/pt/free-png-dykhk>.

SILVEIRA, G. Comparative Results of 3D, 2D and $2\frac{1}{2}$D visual servoing. **VI Simpósio Brasileiro de Automação Inteligente**, p. 769–774, 2003.

SILVEIRA, G.; MALIS, E.; RIVES, P. An Efficient Direct Approach to Visual SLAM. **IEEE TRANSACTIONS ON ROBOTICS**, IEEE, v. 24, p. 969–979, 2008.

TAVARES, J. P. de O. **TÉCNICAS DE CONTROLE DE MOVIMENTO EM VEÍCULOS INTELIGENTES: UMA COMPARAÇÃO**. Lavras, 2017.