

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Iuri Everton Reis de Sousa

**Planejamento de sistema de direção
autônoma em tempo real em um veículo
com o auxílio de câmera**

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Iuri Everton Reis de Sousa

**Planejamento de sistema de direção autônoma em
tempo real em um veículo com o auxílio de câmera**

Trabalho de conclusão de curso apresentado
à Faculdade de Engenharia Mecânica da Uni-
versidade Federal de Uberlândia, Minas Ge-
rais, como requisito parcial exigido à obten-
ção do grau de Bacharel em Engenharia Me-
catrônica.

Orientador: Werley Rocherter Borges Ferreira

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Mecânica

Bacharelado em Engenharia Mecatrônica

Uberlândia, Brasil

2024

Iuri Everton Reis de Sousa

Planejamento de sistema de direção autônoma em tempo real em um veículo com o auxílio de câmera

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia Mecânica da Universidade Federal de Uberlândia, Minas Gerais, como requisito parcial exigido à obtenção do grau de Bacharel em Engenharia Mecatrônica.

Trabalho aprovado. Uberlândia, Brasil, 25 de Novembro de 2024:

Werley Rocherter Borges Ferreira
Orientador

Edsonei Pereira Parreira

Roberto de Souza Martins

Uberlândia, Brasil
2024

Aos meus pais, ao meu irmão e à minha querida família

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado a graça de sorrir e desfrutar de ótimos momentos durante os anos dessa graduação, que sempre foi o meu sonho. Agradeço ao meu pai, Ronaldo, à minha mãe, Cida, e ao meu irmão, Daniel, que, mesmo com todas as dificuldades, acreditaram em mim e sempre estiveram presentes, apesar da distância.

Agradeço à minha família, especialmente à minha madrinha e às minhas tias, por todo o apoio, e aos meus irmãos da igreja, que me acolheram tão bem nesta cidade.

Gostaria de agradecer à minha namorada, Laura, por todo o apoio e momentos compartilhados. Ela que, a cada dia, me inspira a ser um homem melhor. Por toda a motivação, todo o cuidado e todo o carinho, muito obrigado.

Agradeço ao meu orientador, o professor Werley, o qual se dispôs a enfrentar comigo esse desafio e me auxiliou durante toda a jornada. A ele, só tenho como agradecer e desejar muito sucesso.

Agradeço às minhas *irmãs-de-coração* Anna Karolinn e Alessandra por todos os conselhos, conversas e momentos juntos durante esses anos. Agradeço à turma 29 de engenharia mecatrônica, em especial aos meus queridos amigos, pelas muitas horas de estudo, dificuldades, confraternizações e alegrias. Aos "Paragons da Justiça e Moral", detentores de um poder excepcional de companheirismo, alegria e cumplicidade, agradeço por se tornarem pessoas tão especiais para mim.

Agradeço à EDROM, que, mais do que uma equipe de competição, é uma família. Sou grato por todos os amigos, todo o conhecimento e todas as noites viradas e semanas intensas em períodos pré-competição. Espero ser capaz de, um dia, retribuir tudo que a EDROM fez e representa para mim. *"A verdadeira EDROM são os amigos que fazemos no caminho"*.

Agradeço aos meus colegas de trabalho, em especial ao Marcos, Alysson, Pedro e Felipe, por serem grandes referências na área de controle e por toda a ajuda durante o desenvolvimento do projeto.

Finalmente, deixo o meu muito obrigado a todos que contribuíram, direta ou indiretamente, para que este momento fosse possível.

Resumo

Este trabalho apresenta o desenvolvimento de um veículo autônomo capaz de reconhecer um percurso com o auxílio de uma câmera, planejar a melhor trajetória e executar a navegação adequada. O projeto utiliza o ROS (*Robot Operating System*) como biblioteca para desenvolvimento de códigos e o Gazebo como ambiente de simulação, onde foi criada uma pista virtual. Foram implementados algoritmos para identificação da pista, bem como a conversão dos pontos encontrados em coordenadas do mundo real. O controle do veículo é realizado com base na abordagem do Controle Preditivo Baseado em Modelo (MPC), implementando o modelo cinemático de um robô diferencial de duas rodas. Os resultados esperados incluem métricas extraídas da odometria do veículo, como velocidade média, velocidade máxima, tempo total e o trajeto desenvolvido. O trabalho contribui como base para desenvolvimento de veículos para pesquisas voltadas à navegação autônoma.

Palavras-chave: Robótica, Controle, Direção Autônoma, Visão Computacional, Controle Preditivo.

Abstract

This work presents the development of a vehicle capable of recognizing a path with the aid of a camera, planning the best trajectory, and navigating autonomously. The project utilizes ROS (Robot Operating System) as a framework for code development and Gazebo as a simulation environment, where a virtual track was created. Algorithms were implemented for track identification as well as the conversion of detected points into real-world coordinates. The vehicle control is based on the Model Predictive Control (MPC) approach, using the kinematic model of a two-wheeled differential robot. The expected results include metrics extracted from the vehicle's odometry, such as average speed, maximum speed, total time, and the developed trajectory. This work serves as a foundation for the development of vehicles aimed at research in autonomous navigation.

Palavras-chave: *Robotics, Control, Autonomous driving, Computer Vision, Model Predictive Control.*

Lista de figuras

Figura 1 – Imagem do modelo de uma câmera pinhole. fonte: (HARTLEY; ZIS-SERMAN, 2003) (Adaptado.)	19
Figura 2 – Imagem do modelo de uma câmera pinhole. fonte: (OPENCV, 2024a) (Adaptado.)	19
Figura 3 – Tipos de rodas: Roda fixa, Roda direcionável e Roda Livre. fonte: (SICILIANO et al., 2010) (Adaptado)	22
Figura 4 – Esquema de um robô diferencial com duas rodas motrizes e duas rodas livres. fonte: (SICILIANO et al., 2010) (Adaptado)	23
Figura 5 – Modelo 3D do veículo	30
Figura 6 – Circuito de Interlagos utilizado na Simulação do Gazebo	31
Figura 7 – Imagem capturada do tabuleiro de referência para a calibração da câmera	33
Figura 8 – Suporte da câmera com ajuste de inclinação	34
Figura 9 – Imagem do robô frente a grade de calibração	35
Figura 10 – Imagem do robô visualizando a grade de calibração. (a) Imagem sem tratamento, (b) Imagem com distorção corrigida	35
Figura 11 – Projeção de um ponto com a câmera inclinada	36
Figura 12 – Demarcação da região de interesse sobre a imagem	37
Figura 13 – Operação de Fatiamento das imagens	38
Figura 14 – Operações realizadas no processamento da imagem	39
Figura 15 – Resultado do processamento de imagens	40
Figura 16 – Posição dos pontos detectados	41
Figura 17 – Trajetória de Interpolação Cúbica	42
Figura 18 – Posição dos pontos detectados e Trajetória de Referência	42
Figura 19 – Trajetória de Spline Suavizada	43
Figura 20 – Previsão de Trajetória e Atuação de Controle	45
Figura 21 – Previsão de Trajetória e Atuação de Controle com restrições físicas do sistema	46
Figura 22 – Previsão de Trajetória e Atuação de Controle com restrições físicas do sistema	47
Figura 23 – Captura de pontos na grade de calibração	48
Figura 24 – Mapa dos pontos coletados	49
Figura 25 – Grupos de Pontos com Linhas de Tendência agrupadas pelo eixo X	49
Figura 26 – Grupos de Pontos com Linhas de Tendência agrupadas pelo eixo Z	50
Figura 27 – Veículo sobre uma reta	52
Figura 28 – Veículo orientado à esquerda da pista	52
Figura 29 – Veículo orientado à direita da pista	53

Figura 30 – Veículo em um momento anterior a curva	53
Figura 31 – Veículo em ambiente de simulação.	54
Figura 32 – Previsão de trajetória sobre o percurso.	55
Figura 33 – Odometria coletada do veículo sobre o percurso.	55

Lista de tabelas

Tabela 1	–	Parâmetros de Distorção da câmera	34
Tabela 2	–	Tabela de Média, Desvio Padrão e Intervalo de Confiança para os Grupos do eixo X	50
Tabela 3	–	Tabela de Média, Desvio Padrão e Intervalo de Confiança para os Grupos do eixo Z	51
Tabela 4	–	Estatísticas de Odometria:	56

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
CAD	<i>Computer Aided design</i>
DMC	<i>Dynamic Matrix Control</i>
IPOPT	<i>Interior Point OPTimizer</i>
MPC	<i>Model Predictive Control</i>
ROS	<i>Robot Operating System</i>
SBC	<i>Single Board Computer</i>
SSH	<i>Secure Shell</i>
URDF	<i>Universal Robot Description Format</i>
XML	<i>Extensible Markup Language</i>

Lista de símbolos

C	Centro da imagem
c_x, c_y	Coordenadas do ponto principal (ou centro óptico) da imagem;
d	Distância entre rodas;
f_x, f_y	Distância focal da lente da câmera
k_1, k_2, k_3	Coeficientes de distorção radial simétrica;
n_fatias	Número de divisão da imagem para processamento;
N	Horizonte de Predição;
$P_w(X_w, Y_w, Z_w)$	Ponto tridimensional do ambiente real;
p_1, p_2	Coeficientes de distorção tangencial;
L	Distância entre as rodas;
Q	Penalização por erro de rastreamento;
r_p	Distância entre o ponto desejado e o centro da imagem.
R	Penalização por esforço de controle;
v	Velocidade linear;
v_r	Velocidade angular roda direita;
v_l	Velocidade angular roda esquerda;
w	Velocidade angular em torno do eixo vertical;

Sumário

1	INTRODUÇÃO	14
2	OBJETIVOS	16
3	FUNDAMENTAÇÃO TEÓRICA	17
3.1	Calibração da Câmera	17
3.2	Modelo de câmera pinhole e transformação de pontos	18
3.3	Modelo de um robô diferencial de duas rodas	21
3.4	Controle Preditivo Baseado em Modelo	24
4	DESENVOLVIMENTO	26
4.1	Arquitetura do projeto	26
4.2	Modelagem do Veículo	29
4.3	Implementação da Simulação	30
4.3.1	Descrição do Veículo	31
4.3.2	Confecção da pista	31
4.3.3	Simulação do Movimentação do Veículo	31
4.3.4	Adição da câmera na simulação	32
4.4	Visão Computacional	32
4.4.1	Calibração da câmera	32
4.4.2	Identificação da distância dos píxels da câmera	34
4.4.3	Identificação da pista	37
4.5	Controle Preditivo de Trajetória	40
4.5.1	Estabelecendo a Trajetória de Referência	40
4.5.2	Descrição do modelo para Controle Preditivo	43
4.5.3	Função Custo para otimização	44
4.5.4	Previsão de Trajetória	45
5	RESULTADOS	48
5.1	Mapeamento da distância dos píxels da câmera	48
5.2	Sistema de Identificação da Pista	51
5.3	Sistema de Direção Autônoma	53
6	CONCLUSÕES E SUGESTÕES DE TRABALHOS FUTUROS	57
	REFERÊNCIAS	58

APÊNDICES	59
APÊNDICE A – CÓDIGO DESENVOLVIDO	60

1 Introdução

A robótica tem profundas raízes na cultura da humanidade. Ao longo dos séculos existem diversos relatos da busca da humanidade sobre a automatização de tarefas. Na mitologia grega é possível ver lendas sobre seres artificiais e mecanismos inteligentes, como é o caso do autômato Talos, um gigante de bronze, criado para proteger a cidade de Creta. Do ponto de vista científico, um robô pode ser visto como uma máquina que, independentemente da sua aparência exterior, é capaz de modificar o ambiente em que opera. Isso acontece com a realização de ações que seguem regras de comportamento estabelecidas, bem como por meio de dados coletados sobre o seu estado e sobre o ambiente. Dessa forma, é possível definir robótica como o estudo sobre a conexão inteligente entre percepção e ação (SICILIANO et al., 2010, p 1-2). Nesse contexto, a robótica móvel se destaca por ser uma área que tem experimentado avanços significativos nos últimos anos, uma vez que, a busca por soluções em veículos autônomos capazes de navegar em ambientes dinâmicos e complexos de forma eficiente tem sido alvo de muitas pesquisas.

O objetivo deste trabalho é o desenvolvimento de um sistema de direção autônoma em tempo real em um veículo equipado com uma câmera, capaz de perceber o ambiente e navegar de forma inteligente. A primeira etapa consistiu na concepção do veículo e na seleção dos componentes necessários para garantir uma percepção adequada do ambiente. O design do veículo foi realizado no software Fusion360, com as estruturas manufaturadas por meio da impressão 3D. A escolha dos componentes eletrônicos priorizou aspectos como a autonomia, operabilidade, custo e a capacidade de processamento de dados coletados. Além disso, o projeto exigiu o desenvolvimento de algoritmos de comunicação e de controle que permitissem um sistema de navegação autônoma robusta e eficiente..

O processo de percepção do ambiente ocorre através da captura de imagens e da identificação da pista. Nessa etapa é fundamental conhecer os parâmetros intrínsecos e extrínsecos da câmera, como a posição e orientação em que a câmera se encontra, parâmetros como distância focal e eixo de projeção, além dos parâmetros de distorção associadas ao processo de captura de imagens. Para assegurar a extração de informações precisas das imagens, foi empregada uma metodologia de calibração baseada no trabalho de Zhang (ZHANG, 2000), com o suporte da biblioteca OpenCV (OPENCV, 2024b), que oferece ferramentas para processamento de imagens. além disso, os pontos identificados foram convertidos para distância no mundo real e utilizados nos processos de controle.

O trabalho faz uso do Robot Operating System (ROS), conjunto de bibliotecas focadas no desenvolvimento de robótica, que fornece uma plataforma robusta e flexível com um bom sistema de ferramentas que possibilitam uma alta modularidade nos códigos

desenvolvidos. A integração do ROS com o simulador Gazebo permitiu a criação de um ambiente virtual que utiliza os mesmos algoritmos de controle e visão computacional implementados no veículo real, de forma que, os testes iniciais e ajustes puderam ser realizados virtualmente antes de serem embarcados no robô real.

A abordagem utilizada para o desenvolvimento do algoritmo de controle foi o Controle Preditivo Baseado em Modelo (MPC). O MPC, de forma geral, consiste na previsão dos estados futuros do sistema com base em um modelo matemático (Rossiter JA, 2003). Essa abordagem foi adotada devido a sua capacidade de otimizar a trajetória do robô, levando em consideração as restrições físicas e parâmetros do veículo, garantindo eficiência e robustez na predição de trajetórias.

Por fim, deseja-se posicionar o robô sobre uma pista e testar a eficácia do algoritmo. Essa análise leva em consideração a odometria coletada durante o percurso, verificando os valores de velocidade e tempos obtidos, assim como a análise da trajetória desenvolvida pelo veículo.

2 Objetivos

O objetivo geral deste projeto é a construção de um veículo capaz de navegar de forma autônoma, utilizar uma câmera para reconhecer a pista, planejar a melhor trajetória a ser desenvolvida e controlar a sua movimentação em tempo real.

Objetivos específicos:

- Reconhecimento da pista: Implementar algoritmos de visão computacional para detectar a pista, seus limites e distâncias.
- Planejamento de trajetória: Desenvolver um algoritmo de controle capaz de definir a melhor rota com base nos dados capturados pela câmera.
- Simulação e validação: Construir um ambiente de simulação capaz de validar de forma virtual os algoritmos antes da implementação no robô real.
- Padrão de desenvolvimento: Utilizar de tecnologias e padrões de desenvolvimento de software que garantam a consistência e integridade da solução

3 Fundamentação Teórica

3.1 Calibração da Câmera

Com o intuito de utilizar uma câmera monocular acessível para obtenção das imagens da pista em que o veículo estará operando, é necessário, além de identificar os limites da pista, compreender a distância em que os pontos estão do robô. Essa tarefa leva em consideração diversos métodos e parâmetros desenvolvidos no campo da visão computacional ao longo dos anos, sendo necessário a descrição de um modelo matemático para a câmera, levantamento de parâmetros intrínsecos, como foco e centro de coordenadas, e extrínsecos, como posição da câmera no mundo real, para converter um ponto p de uma imagem capturada em um ponto P do mundo real.

Em 1971, Brown (BROWN, 1971) descreve em seu artigo o que é o processo de calibração de uma câmera e os parâmetros necessários para efetuar a correção de distorção em imagens, posteriormente, o trabalho de Zhang (ZHANG, 2000) descreve uma metodologia capaz de possibilitar a calibração de uma câmera de uma forma acessível e relativamente simples identificando, dentre vários parâmetros, a distorção radial, a distorção tangencial e a matriz intrínseca da câmera. O método consiste em utilizar o fato de que as lentes das câmeras introduzem distorções nas imagens capturadas, tornando possível calcular os parâmetros que definem as distorções utilizando um objeto conhecido como referência e encontrar como a câmera distorce a imagem observada ao movimentar o item em diferentes orientações e perspectivas. O processo descrito foi implementado na biblioteca OpenCV como parte do módulo de calibração de câmera e reconstrução tridimensional. Levando em consideração que a câmera utilizada possui apenas uma lente, foi realizado o procedimento conhecido como calibração monocular.

O artigo de (MAZON; ZACCHI; MARTINS, 2011) cita os parâmetros importantes para a calibração de uma câmera como:

- (f) : Distância focal da lente da câmera
- (c_x, c_y) : Coordenadas do ponto principal (ou centro óptico) da imagem;
- (k_1, k_2, k_3) : Coeficientes de distorção radial simétrica;
- (p_1, p_2) : Coeficientes de distorção tangencial;

A distorção radial ocorre nas imagens devido ao fato de que os raios de luz que passam nas bordas da lente são curvados de maneira diferente dos que passam pelo centro.

Dessa forma, o seu efeito consiste em transformar linhas retas em curvas durante a captura das imagens. Esse tipo de distorção tende a aumentar quanto mais longe o objeto estiver do centro da câmera. Existem dois tipos de distorções radiais: a distorção em barril e a distorção em almofada. Com base em (OPENCV, 2024a), a distorção radial pode ser representada como:

$$x_{distorted} = x(1 + k_1 r_p^2 + k_2 r_p^4 + k_3 r_p^6) \quad (3.1)$$

$$y_{distorted} = y(1 + k_1 r_p^2 + k_2 r_p^4 + k_3 r_p^6) \quad (3.2)$$

Onde r_p é a distância entre o ponto desejado e o centro da imagem.

No processo da imagem também existe a distorção tangencial, causada pela falta de alinhamento entre a lente da câmera e o sensor de captura da imagem, fazendo com que os pontos capturados apareçam com um deslocamento tangencial em relação a onde deveriam estar. Essa distorção pode ser descrita por:

$$x_{distorted} = x + [2p_1 xy + p_2(r_p^2 + 2x^2)] \quad (3.3)$$

$$y_{distorted} = y + [p_1(r_p^2 + 2y^2) + 2p_2 xy] \quad (3.4)$$

A metodologia de (ZHANG, 2000) também define uma matriz de parâmetros conhecida como parâmetros intrínsecos da câmera, comumente denotada como A ou K . Essa matriz contém informações sobre as distâncias focais e coordenadas do ponto principal da imagem, denotada por:

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.5)$$

Os procedimentos necessários para a construção da matriz A podem ser encontrados através do passo-a-passo de calibração da biblioteca OpenCV (OPENCV, 2024b).

3.2 Modelo de câmera pinhole e transformação de pontos

Para relacionar os pontos obtidos na imagem com pontos do mundo real é interessante definir um modelo matemático que represente a câmera. O modelo de câmera estenopeica ou *câmera pinhole*, é um modelo simples que define o processo de geração da imagem. Este modelo considera um centro de projeção e um plano focal, a imagem formada pela câmera é resultado da projeção de pontos do espaço no plano focal. O centro de projeção é definido como a origem de um sistema de coordenadas euclidianas e

o plano focal se encontra na posição $Z = f$ (HARTLEY; ZISSERMAN, 2003). A figura 1 ilustra o modelo, onde C é o centro da câmera, a linha que vai do centro da câmera perpendicularmente ao plano da imagem é chamada de eixo principal, e p é o ponto principal, marcando o encontro do eixo com o plano. Na representação à direita, é possível perceber a semelhança de triângulos entre a altura de um ponto com a distância focal e a coordenada da imagem.

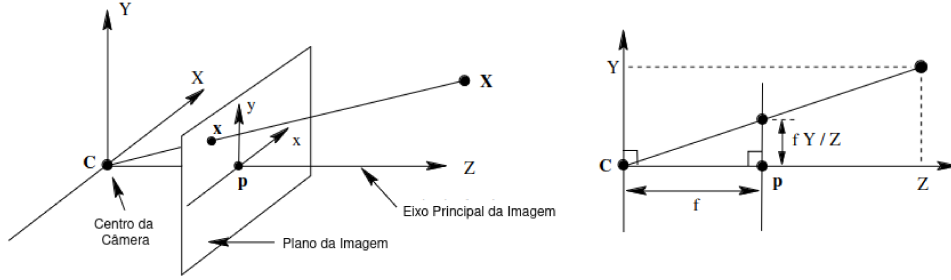


Figura 1 – Imagem do modelo de uma câmera pinhole. fonte: (HARTLEY; ZISSERMAN, 2003) (Adaptado.)

Dessa forma, é possível relacionar matematicamente coordenadas de um ponto tridimensional $P_w(X_w, Y_w, Z_w)$ do ambiente real com a sua projeção em um plano da câmera em um ponto $p(u, v)$, como ilustrado na Figura 2.

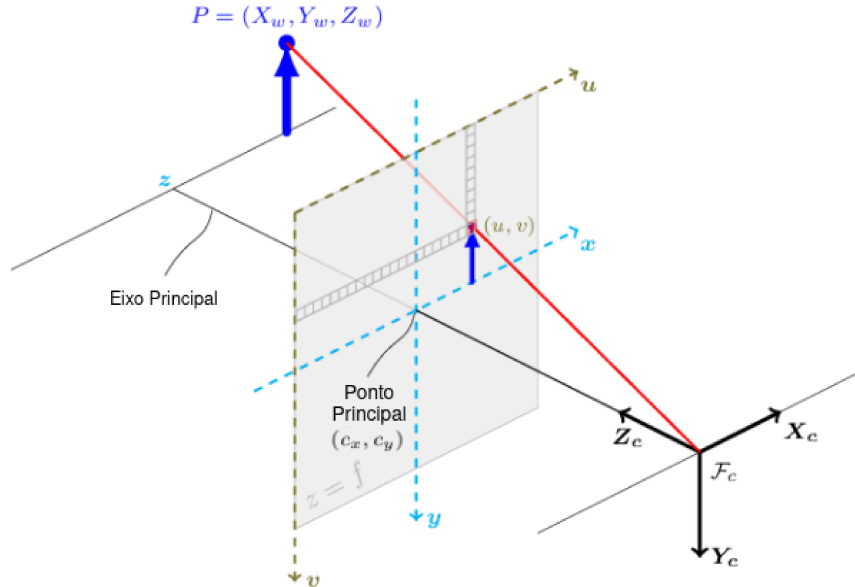


Figura 2 – Imagem do modelo de uma câmera pinhole. fonte: (OPENCV, 2024a) (Adaptado.)

Nesse contexto, pode-se utilizar da seguinte relação para tratar os pontos:

$$sp = A[R|t]P_w \quad (3.6)$$

Onde A é a matriz intrínseca da câmera e $[R|t]$ é uma matriz de rotação e translação que representa a orientação e posição da câmera com relação ao eixo de coordenadas. O termo s define um vetor de escala. É importante ressaltar que no projeto em questão a câmera se mantém em uma posição fixa em relação ao centro de coordenadas do robô.

A matriz intrínseca A é capaz de projetar um ponto tridimensional no sistema de coordenadas da câmera em um sistema de píxels bidimensional da imagem. De forma que:

$$p = AP_c \quad (3.7)$$

Onde p é um ponto no sistema de píxels da câmera e P_c é um ponto do sistema de coordenadas do mundo real transformado para o sistema de coordenadas da câmera.

Portanto,

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \quad (3.8)$$

O P_c é o ponto no sistema de coordenadas da câmera, a relação deste ponto com um ponto P_w do mundo real é através da transformação da matriz $[R|t]$, onde:

$$P_c = \begin{bmatrix} R & t \end{bmatrix} P_w \quad (3.9)$$

Escrevendo de forma completa tem-se:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3.10)$$

Analisando mais uma vez o ponto P_c é possível normalizar esse vetor com relação a coordenada Z_c , fazendo $x' = X_c/Z_c$ e $y' = Y_c/Z_c$, de forma que:

$$Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

Esta operação se torna útil, pois é possível manter a relação com 3.9

$$Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Se retormar a equação 3.6, escrevendo-a de forma completa tem-se:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3.11)$$

Considerando

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \quad (3.12)$$

Assumindo novamente $x' = X_c/Z_c$ e $y' = Y_c/Z_c$:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad (3.13)$$

Finalmente, considerando $Z_c \neq 0$, $s = Z_c$ e dividindo ambos os lados por Z_c , tem-se:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x X_c/Z_c + c_x \\ f_y Y_c/Z_c + c_y \end{bmatrix} \quad (3.14)$$

A equação 3.14 relaciona a transformação de um ponto (u, v) em pontos do sistema de coordenadas da câmera (X_c, Y_c, Z_c) .

3.3 Modelo de um robô diferencial de duas rodas

O robô a ser utilizado nesse projeto se encontra na categoria de robótica móvel. Os robôs móveis são aqueles onde existe a presença de uma base móvel que permite ao robô se locomover no ambiente. De forma geral, essa categoria pode ser dividida em Robôs móveis com rodas e Robôs móveis com pernas.

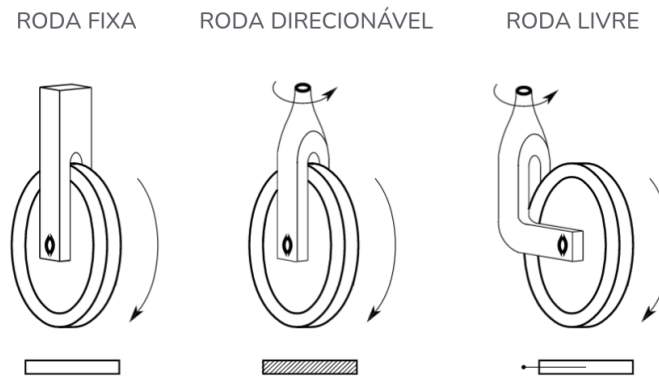


Figura 3 – Tipos de rodas: Roda fixa, Roda direcionável e Roda Livre. fonte: (SICILIANO et al., 2010) (Adaptado)

Os robôs móveis com roda são os mais comuns de serem encontrados e, normalmente, possuem um chassi e um sistema de rodas capazes de gerar movimento. As rodas desses robôs podem ser classificadas em três grandes categorias, que podem ser observadas na figura 3.

- Roda Fixa (*Fixed Wheel*): A roda é acoplada no chassi do robô de forma que o movimento de rotação ocorre sobre o seu centro e na direção ortogonal ao plano da roda;
- Roda Direcionável (*Steerable wheel*): Possui dois eixos de rotação, o primeiro é igual ao da roda fixa e o segundo é vertical, permitindo uma mudança de orientação em relação ao chassi;
- Roda Livre (*Caster Wheel*): Possui também dois eixos de rotação, no entanto, o eixo vertical não passa sobre o centro da roda, permitindo que a roda acompanhe o movimento do chassi, oferecendo baixa resistência à mudança de orientação.

O projeto desenvolvido neste trabalho é constituído de duas rodas fixas acopladas em motores e duas rodas livres de apoio. Esse modelo é conhecido como robô diferencial de duas rodas. As rodas estão paralelas, alinhadas sobre o mesmo eixo, e separadas a uma distância d . Cada uma das rodas pode ser acionada de forma individual, dessa forma, a trajetória desenvolvida pelo robô é determinada pela velocidade angular de cada uma delas.

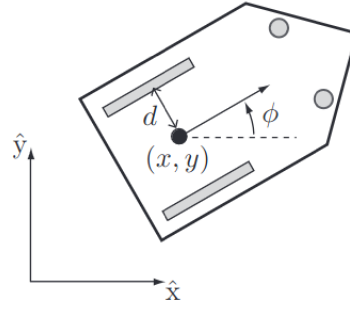


Figura 4 – Esquema de um robô diferencial com duas rodas motrizes e duas rodas livres.
fonte: (SICILIANO et al., 2010) (Adaptado)

A figura 4 mostra o modelo do robô diferencial considerando o centro do robô sobre as coordenadas (x, y) , além disso, ϕ representa o ângulo de orientação do robô em relação ao eixo horizontal.

Devido ao fato de que as rodas são controladas de forma independente elas determinam a velocidade linear e a velocidade angular do robô. (LYNCH; PARK, 2017)

$$v = \frac{v_r + v_l}{2} \quad (3.15)$$

$$w = \frac{v_r - v_l}{L} \quad (3.16)$$

onde:

- v : Velocidade linear;
- w : Velocidade angular em torno do eixo vertical;
- v_r : Velocidade angular roda direita;
- v_l : Velocidade angular roda esquerda;
- L : Distância entre as rodas;

O deslocamento do centro do robô pode ser escrito através da velocidade linear do robô com o ângulo de orientação, dessa forma as equações de movimento são obtidas através da derivada das posições, resultando em:

$$\dot{x} = v \cos(\phi) \quad (3.17)$$

$$\dot{y} = v \sin(\phi) \quad (3.18)$$

$$\dot{\phi} = w \quad (3.19)$$

Como é interessante controlar as rodas de forma individual nesse processo, é conveniente unir as equações que descrevem a velocidade linear com o deslocamento do centro do robô. De forma que:

$$\dot{x} = \frac{v_r + v_l}{2} \cos(\phi) \quad (3.20)$$

$$\dot{y} = \frac{v_r + v_l}{2} \sin(\phi) \quad (3.21)$$

$$\dot{\phi} = \frac{v_r - v_l}{L} \quad (3.22)$$

3.4 Controle Preditivo Baseado em Modelo

O Controle Preditivo Baseado em Modelo é uma abordagem avançada de controle de sistemas. Essa técnica tem como base a utilização do modelo do sistema para previsão das próximas saídas ao longo de um período denominado horizonte de predição. O algoritmo de otimização ajusta as saídas de controle a fim de minimizar uma função de custo predefinida. O MPC se destaca pela capacidade de calcular as saídas de controle levando em consideração as restrições das variáveis de entrada e saída. Por esse motivo, é capaz de garantir um desempenho estável em sistemas complexos.

O MPC tem as suas origens nos anos 1960 e 1970 na busca da indústria de explorar formas mais sofisticadas em relação as alternativas convencionais da época. Durante a década de 1970, o conceito do MPC passa a ganhar força dentro da indústria petrolífera no processo de controle em refinarias de petróleo. O trabalho de (CUTLER; RAMAKER, 1980) introduz o conceito de DMC (Dynamic Matrix Control), uma das primeiras formas de MPC utilizadas na Indústria de petróleo, que lida com previsões de múltiplos passos a frente para otimizar o desempenho do sistema. No início dos anos 2000, o MPC passou a ser utilizado com maior frequência devido aos avanços computacionais. Estes avanços permitiram a implementação do MPC em tempo real com maior facilidade, possibilitando a sua implementação em diversos tipos de indústrias, como aeronáutica, automotiva e robótica.

Segundo (Rossiter JA, 2003), o conceito do controle preditivo se baseia em práticas intuitivas da experiência humana. Por exemplo, ao dirigir um carro, o motorista visualiza a estrada e faz uma previsão do comportamento do carro, ajustando a direção e a velocidade para garantir manobras seguras. Essa ideia é a base do MPC, que consiste na previsão dos estados futuros do sistema com base em um modelo matemático. A partir dessa previsão, são calculadas uma sequência de ações de controles levando em consideração as restrições do sistemas. As ações de controles são calculadas utilizando técnicas de otimização para minimizar uma função custo. A cada passo do controle, é feita uma predição para o

horizonte determinado e a primeira ação de controle é aplicada na planta. No próximo passo, o horizonte se desloca para frente e o processo é repetido.

O MPC possui uma série de vantagens, como o fato da otimização ser feita levando em consideração as restrições impostas e apenas saídas condizentes com o sistema serem utilizadas como resultado. Além disso, o MPC é particularmente eficaz em lidar com sistemas multivariáveis, o que é ideal para robôs que exigem o controle simultâneo de velocidade e direção. Em contrapartida, é necessário uma capacidade computacional capaz de solucionar as equações em um intervalo de tempo que garanta uma performance robusta e estável ao longo da pista.

4 Desenvolvimento

Este capítulo trata de descrever as etapas e conteúdos relacionados ao desenvolvimento do projeto, o qual busca obter um veículo capaz de identificar uma pista com o auxílio de uma câmera, planejar, executar e seguir uma trajetória em tempo real.

4.1 Arquitetura do projeto

O desenvolvimento de um veículo capaz de atender aos requisitos estabelecidos é uma tarefa complexa e exige a utilização de diversos componentes. No caso em questão, é necessário a definição de atuadores, sensores, microcontroladores, processadores e recursos de alimentação para a escolha dos componentes a serem utilizados.

No robô desenvolvido tem-se uma câmera como sensor para identificação da pista, como atuadores, dois motores de corrente contínua com encoders, como controlador principal uma SBC (*Single Board Computer*) capaz de executar os algoritmos de visão computacional e controle e um microcontrolador com o objetivo de realizar o controle dos atuadores. Além destes componentes, também tem-se os periféricos necessários, como os reguladores de tensão e bateria.

Desta forma, a seguinte lista descreve os componentes utilizados no projeto:

- Orange PI 3 LTS:

A Orange PI 3 LTS é uma SBC que oferece uma plataforma compacta e capaz de rodar sistemas operacionais. Dessa forma, é uma boa escolha para embarcar os algoritmos e processamentos no robô. A placa possui um processador ARM Cortex-A53 quad-core e 2 GB de memória RAM, oferecendo uma boa qualidade no processamento e execução dos algoritmos. Neste projeto, foi realizada a instalação do sistema operacional Ubuntu com o objetivo de executar e configurar as dependências necessárias do projeto.

Devido ao fato da Orange PI possuir antena Wi-fi e módulo Ethernet, foi possível habilitar o acesso remoto ao hardware utilizando o SSH (*Secure Shell*), tecnologia que permite enviar comandos para terminal de um outro computador, facilitando a etapa de configuração. Além disso, as portas USBs foram utilizadas para a conexão com a Webcam e com o microcontrolador.

- Webcam Full HD 1080p 30fps:

Com o objetivo de capturar as imagens do ambiente em tempo real, foi utilizado uma câmera capaz de comunicar com a Orange PI através de uma porta USB. A

Câmera utilizada é de um modelo simples, genérico e acessível, que fornece imagens em uma resolução de até 1920 x 1080 pixels e 30 frames por segundo.

Os parâmetros intrínsecos da câmera foram levantados e foi possível realizar o processo de calibração permitindo que o sistema de visão computacional reconheça a pista.

- **Arduino Nano:**

Visando o controle dos motores foi utilizado um Arduino Nano, microcontrolador compacto baseado no ATmega328. O Arduino Nano possui uma frequência de operação de 16 Mhz, 14 pinos digitais de entrada e saída e 8 entradas analógicas. Além disso, possui a capacidade de enviar sinais com modulação por largura de pulso, simulando saídas analógicas, o que é útil para o controle de velocidade dos motores junto com a utilização de uma ponte H. Neste projeto o arduino é conectado à Orange PI e recebe as informações de velocidade desejadas através da comunicação UART, utilizando uma porta USB.

O Arduino conta com o código adaptado da biblioteca `ros_arduino_bridge` ((GOEBEL, 2012)), que permite a sua integração com o ROS, permitindo a comunicação com os algoritmos utilizados no robô através da comunicação serial. Esse pacote implementa também uma malha para controle de velocidade dos robôs, o que é útil para utilização em conjunto com a leitura dos encoders.

- **Ponte H L298N:**

Para efetuar a alimentação e controle de direção e velocidade dos motores de corrente contínua foi escolhido o módulo Ponte H L298N. A L298N suporta a alimentação de até 2 A por canal, além disso permite o controle individual sobre cada um dos motores.

- **2x Motores N20 200 rpm com encoder:**

Os motores utilizados no projeto são de corrente contínua e possuem uma rotação nominal de 200 rpm. O modelo utilizado é o N20, que possui um tamanho compacto e também bastante potente. Além disso, o motor possui um kit de encoder permitindo a medição da rotação dos eixos e possibilitando a implementação de uma malha de controle de velocidade e posição. Esta função é muito importante para a eficiência do projeto.

- **Bateria de LiPo 3s 1300mAh 11.1 V:**

As baterias de polímero de lítio (LiPo) são bastante utilizadas no ambiente de robótica pela sua grande capacidade de carga e leveza. A escolha do projeto foi a utilização de uma bateria 3s (11.1 V) com a função de fornecer a tensão e correntes necessárias para alimentação dos motores e controladores. Além disso, conta com

uma capacidade de carga de 1300 mAh, trazendo grande autonomia de energia para o veículo.

- Regulador de Tensão LM2560:

O regulador de tensão LM2560 é utilizado para assegurar que os módulos do Arduino e Orange PI sejam alimentados com a tensão adequada e estável. Isso é importante pois esses controladores são extremamente sensíveis a mudanças de tensão durante a sua operação.

Além dos materiais utilizados, a seguir tem-se a lista das principais tecnologias e softwares utilizados para o desenvolvimento do projeto:

- Fusion360:

O Fusion360 é um software de modelagem 3D e design assistido por computador (CAD) desenvolvido pela Autodesk. É uma ferramenta muito completa para o desenvolvimento de projetos e possui um plano gratuito para estudantes. O software foi utilizado para a confecção do modelo tridimensional do robô. As peças foram exportadas para impressão 3D e também para montar os modelos necessários para a simulação.

- ROS:

O ROS (Robot Operating System) é um framework de código aberto para o desenvolvimento de robótica. É um conjunto de códigos e bibliotecas que ajudam desenvolvedores na criação de sistemas modulares e eficientes. O ROS permite a divisão do código em componentes menores conhecidos como nós. Cada nó tem como responsabilidade a execução de uma tarefa específica e se comunica com os outros nós através de um sistema de mensagens. A comunicação do ROS se dá através de tópicos e serviços. Os nós podem se inscreverem ou publicarem em tópicos, comunicando uns com os outros. Cada nó pode ser escrito em linguagens de programação diferentes e mesmo assim manter o seu funcionamento.

Além disso, o ROS ainda possui uma grande quantidade de ferramentas compatíveis com ferramentas de processamento e visualização de dados e simuladores de ambientes robóticos, facilitando o desenvolvimento e teste de sistemas.

- Gazebo:

O Gazebo é uma coleção de bibliotecas de código aberto projetado para simplificar o desenvolvimento de aplicações de alta performance (ROBOTICS, 2002, Gazebo). Tem o seu foco voltado para a comunidade de robótica e fornece um ambiente 2D e 3D para simulações dinâmicas, levando em consideração as propriedades dos materiais, a física dos ambientes, além das suas geometrias e conexões. Possui uma

camada de comunicação baseada em mensagens, interagindo fortemente com o ROS. Além disso, tem uma grande quantidade de plugins que podem ser utilizados como sensores, câmeras, lasers, giroscópios entre outros.

No trabalho, o gazebo funciona como ambiente de simulação virtual onde o veículo é capaz de perceber o mundo criado. Dentro do gazebo foram configuradas câmeras e um sistema de direção, além disso, confeccionada a pista que serve como referência para o controle.

- OpenCV:

OpenCV (Open Source Computer Vision Library) é uma biblioteca de código aberto especializada em processamento de imagens e visão computacional. A ferramenta possui diversas funções no tratamento de imagens, filtros, detecção de bordas e transformações geométricas.

Durante o desenvolvimento do projeto foi uma ferramenta muito importante para o tratamento de imagens e reconhecimento da pista. Além disso, facilitou o processo de calibração da câmera e levantamento dos parâmetros específicos da câmera utilizada. Foi muito útil também na utilização em conjunto com o ROS, publicando a saída das imagens e disponibilizando em tópicos específicos.

- IPOPT:

O IPOPT (Interior Point OPTimizer) é um solver de código aberto projetado para resolver problemas de otimização não linear de grande escala (Wächter; Biegler., 2006). O algoritmo é capaz de lidar com funções objetivo e restrições não lineares, iterando na busca de soluções viáveis dentro de intervalos condizentes com as restrições impostas.

O IPOPT é integrado com frameworks de modelagem matemática como Pyomo, por esse motivo, é amplamente utilizado em controle preditivo e otimização em engenharia. Dessa forma, torna-se uma ferramenta essencial para o presente trabalho, atuando para resolver os desafios do controle e gerando saídas para o planejamento de trajetória.

4.2 Modelagem do Veículo

A modelagem do veículo foi feita utilizando o software Fusion 360 da Autodesk. Nesta etapa foi possível confeccionar o veículo no formato de um robô diferencial com duas rodas motoras e duas rodas com movimento livre para apoio. O modelo também levou em consideração a localização e o posicionamento de todos os materiais necessários para o funcionamento, provendo um chassi com suporte para os componentes elétricos tais como

a bateria e interruptores, além disso, um sistema de pavimentos expansíveis. Na figura abaixo, tem-se o design do modelo construído com os seus componentes acoplados.

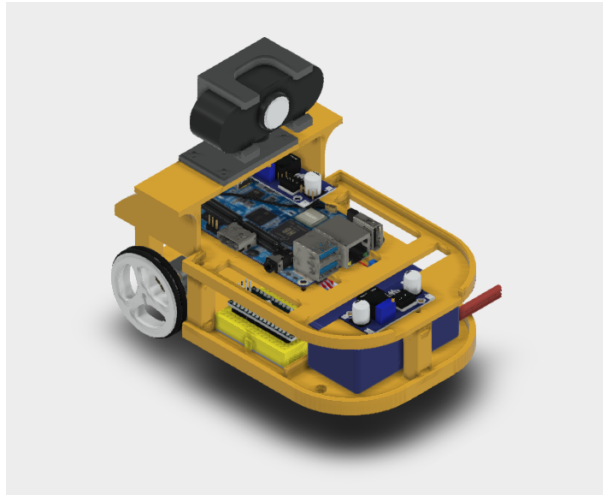


Figura 5 – Modelo 3D do veículo

Para a manufatura das peças que compõem a estrutura e elementos do veículo foi adotada a impressão 3D com a utilização do material PLA, dessa forma, as bases, pilares, rodas e suportes foram projetados de forma a viabilizar a sua construção.

É também interessante destacar que alguns recursos foram utilizados para atender as necessidades do projeto, como por exemplo: as rodas foram desenhadas para conexão direta com os motores, e possuem o diâmetro de 44mm, suficiente para o acoplamento de anéis de vedação de borracha (*popularmente também conhecidos como O-rings e bastante utilizados na vedação de fluídos*) ao seu redor, funcionando como pneus, proporcionando uma boa aderência ao solo com o movimento dos motores. Os suportes das rodas de apoio foram também projetados para acomodar, com certa liberdade, esferas internas de rolamentos reaproveitados, fornecendo a estabilidade necessária para o movimento do robô, oferecendo baixíssima resistência ao movimento. Além desses, tem-se os pilares, itens essenciais para estrutura do projeto, acomodam porcas que são montadas com interferência, a montagem se dá utilizando o baixo ponto de fusão do material aquecendo as porcas com o auxílio de um ferro de solda e posicionando-as no lugar desejado.

4.3 Implementação da Simulação

Quando se trata de uma aplicação complexa, é importante estabelecer um ambiente capaz de validar as hipóteses e testar os possíveis cenários de utilização. Uma vez que o projeto utiliza o ROS como base para todo o código executado, para estabelecimento do ambiente virtual foi escolhido o simulador Gazebo.

4.3.1 Descrição do Veículo

O primeiro passo necessário para descrever o veículo e representá-lo no simulador é a construção do *URDF* - *Universal Robot Description Format* correspondente ao modelo desenvolvido. O URDF é um arquivo XML que possui as propriedades necessárias para descrever a cinemática, propriedades de inércia e geometria de robôs. As suas especificações definem as propriedades gerais dos robôs como a suas dimensões, materiais, cores, sensores, como câmeras, lasers e ultrassons, a estrutura física como as juntas e conexões, os acoplamentos e transmissões relacionados a cada parte da estrutura e as características de simulação como coeficiente de atrito e momento de inércia.

De forma geral o URDF é um arquivo muito importante para o estabelecimento da etapa de simulação do projeto. Com o auxílio do Fusion360 o chassi do robô foi exportado em formato STL e montado no URDF com as suas dimensões e características dinâmicas.

4.3.2 Confecção da pista

A pista virtual utilizada teve como base a pista brasileira de automobilismo do autódromo José Carlos Pace, popularmente conhecido como Autódromo de Interlagos. O circuito foi redimensionado e montado no Gazebo para servir como referência para a direção do veículo.

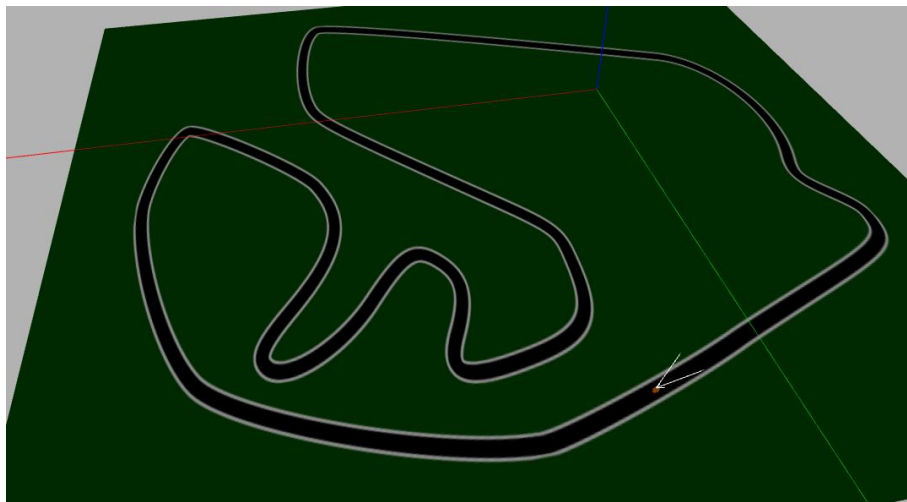


Figura 6 – Circuito de Interlagos utilizado na Simulação do Gazebo

4.3.3 Simulação do Movimentação do Veículo

Para efetuar o controle do veículo na simulação, fez-se o uso da biblioteca *Ros2_control* (*ROS2_CONTROL*, 2024). A biblioteca fornece uma forte integração ao Gazebo e possui a implementação e suporte para integração com diversos hardwares.

Além disso, possui uma extensão da sua API com a implementação do modelo de robô diferencial de duas rodas. Dessa forma, são criados tópicos para o envio de parâmetros relacionados com a velocidade e tópicos de resposta de odometria do modelo, representando uma estimativa da posição do robô através do modelo.

A sua configuração leva em conta parâmetros como o raio das rodas, distância entre elas, velocidade máxima, aceleração máxima e torque dos motores. Assim o sistema consegue efetuar a sua movimentação no simulador Gazebo. É também possível implementar uma camada para comunicação com o Arduino e utilizar da mesma interface de mensagens para que os robôs virtuais e reais tenham um comportamento o mais próximo possível.

4.3.4 Adição da câmera na simulação

Tendo em vista a captura das imagens dentro do simulador para a execução dos algoritmos de visão computacional foi utilizado o plugin de câmera do Gazebo. Esse plugin está ligado ao modelo descrito do robô e recebe também os parâmetros levantados na calibração da câmera, reproduzindo dentro do ambiente uma câmera o mais próximo possível da real.

As imagens são publicadas nos tópicos de câmera, além disso, foi implementado uma camada para compressão de imagens, o que torna o transporte e processamento das imagens mais rápido.

4.4 Visão Computacional

A utilização de visão computacional neste projeto tem o objetivo de utilizar uma câmera para identificar os limites da pista e a distância a ser percorrida pelo veículo. Por esse motivo pode-se dividir o presente tópico em: calibração da câmera; identificação da distância dos pixels da câmera e identificação da pista.

4.4.1 Calibração da câmera

O projeto utiliza como câmera uma webcam genérica capaz de transmitir imagens 1080p através de uma porta USB 2.0. Como o fornecedor da câmera não traz informações detalhadas a respeito das características relacionadas à distorção de imagem, ponto focal, ângulo de visão e outros parâmetros da câmera, é interessante estabelecer um método para levantamento os parâmetros intrínsecos da câmera para determinar variáveis importantes na extração de dados através da visão computacional.

Seguindo a fundamentação teórica descrita em 3.1, foi utilizada a metodologia de calibração monocular descrita em (OPENCV, 2024a) visando o levantamento dos parâ-

metros intrínsecos da câmera.

Inicialmente, foi impresso em uma folha de papel um tabuleiro quadriculado de 10x8 quadrados de dimensão de 14mm. Com a câmera acoplada ao robô e o *ROS* operando, foi executado um algoritmo para disponibilizar a imagem da câmera no tópico */camera_raw*, então, foi utilizado o pacote *camera_calibration* para captura das imagens e parâmetros da câmera. Com o objetivo de tornar o processamento mais rápido também foi escolhido a resolução de 640x480px para captura de imagens em todos os algoritmos de visão computacional do veículo.

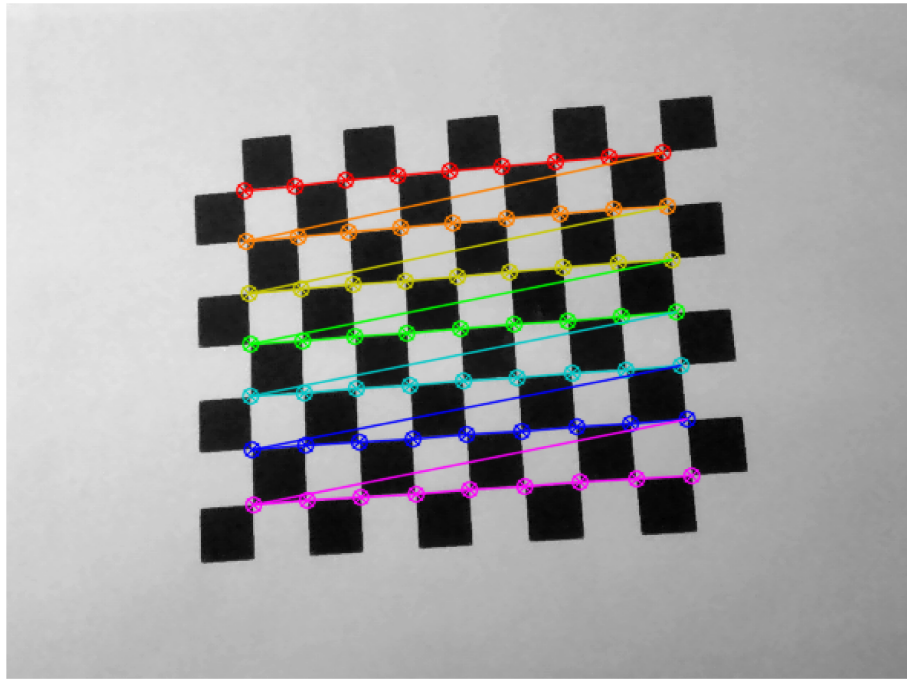


Figura 7 – Imagem capturada do tabuleiro de referência para a calibração da câmera

A figura acima mostra o algoritmo executando o reconhecimento dos vértices do tabuleiro, uma vez que foram fornecidas medidas reais do tamanho de cada um dos quadrados e capturadas uma sequência de 60 imagens em diferentes perspectivas, é possível resolver o sistema de equações de calibração da câmera (3.1) e obter os parâmetros intrínsecos da câmera (4.1) e os valores dos parâmetros de distorção.

A matriz intrínseca de parâmetros da câmera A encontrada foi:

$$A = \begin{bmatrix} 747.477148 & 0.000000 & 281.589364 \\ 0.000000 & 747.474690 & 226.732107 \\ 0.000000 & 0.000000 & 1.000000 \end{bmatrix} \quad (4.1)$$

O vetor de distorção da câmera após a calibração é:

$$d = [0.107080 \quad -0.325658 \quad 0.001122 \quad -0.016761 \quad 0.000000] \quad (4.2)$$

A tabela 1, a seguir, apresenta os parâmetros de distorção da câmera e os parâmetros da matriz intrínseca:

Tabela 1 – Parâmetros de Distorção da câmera

Parâmetro	Valor
f_x	747.477148
f_y	747.474690
c_x	281.589364
c_y	226.732107
k_1	0.107080
k_2	-0.325658
p_1	0.001122
p_2	-0.016761
k_3	0.000000

4.4.2 Identificação da distância dos píxels da câmera

Para fixação da câmera no robô foi desenvolvido um suporte para a webcam com ajuste de inclinação. Isso foi feito pensando na ideia de encontrar o melhor ângulo de visão para o veículo, uma vez que, inclinar a câmera para baixo pode permitir observar pontos mais próximos do robô. A metodologia utilizada para definir a inclinação da câmera foi observar o ângulo onde era possível obter a melhor visão da pista, observando os pontos da pista mais próximos, sem a interferência do chassi do veículo. A medida foi feita no robô real e refletida para o robô virtual. A imagem 8 mostra com detalhe a vista lateral com destaque ao suporte desenvolvido.

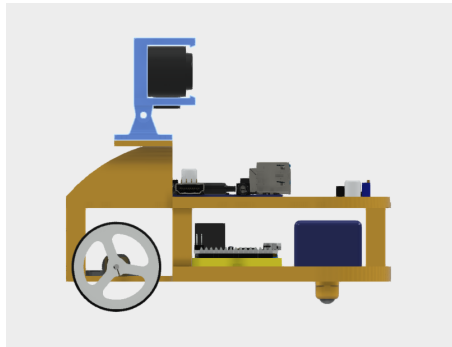


Figura 8 – Suporte da câmera com ajuste de inclinação

Ainda nessa etapa, foi confeccionado, sobre uma folha de papel paraná uma grade de calibração com marcações de 5 cm, o robô foi posicionado no início da grade e o ajuste

da inclinação foi utilizando como referência a observação da imagem da câmera. A figura 7 mostra o posicionamento do robô diante do papel paraná. A linha feita de fita isolante preta marca os primeiros pixels observados pelo robô pela câmera, a uma distância de 8,2 cm do início da grade.

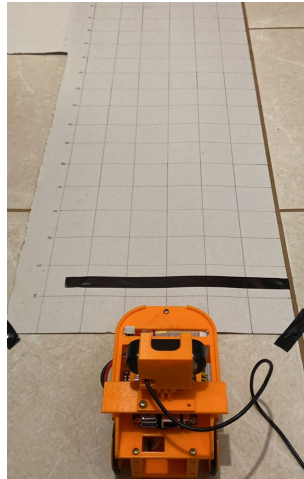


Figura 9 – Imagem do robô frente a grade de calibração

Com o auxílio de um inclinômetro, foi medido que a inclinação encontrada pela metodologia foi de 15 graus da câmera, além disso a lente da câmera se encontra a 12 cm do solo e a 10.25 cm atrás da frente do robô e a 4 cm do eixo das rodas. Essas medidas são importantes para definir a orientação do robô e aplicar a metodologia necessária para a identificação do ponto real. A imagem 10 mostra um frame capturado da imagem observada pelo veículo.



Figura 10 – Imagem do robô visualizando a grade de calibração. (a) Imagem sem tratamento, (b) Imagem com distorção corrigida

Nessa etapa é possível iniciar o desenvolvimento do algoritmo para coletar pontos da imagem e encontrar as distâncias das coordenadas no mundo real. Considerando que o robô irá operar em um terreno plano e que a câmera está inclinada em um ângulo θ e uma altura h do solo é possível, através do modelo de câmera pinhole 3.1, encontrar

a distância Z_c do ponto com equações trigonométricas e, X_c através da equação 3.14. A figura 11 mostra a geração de imagem com a câmera inclinada.

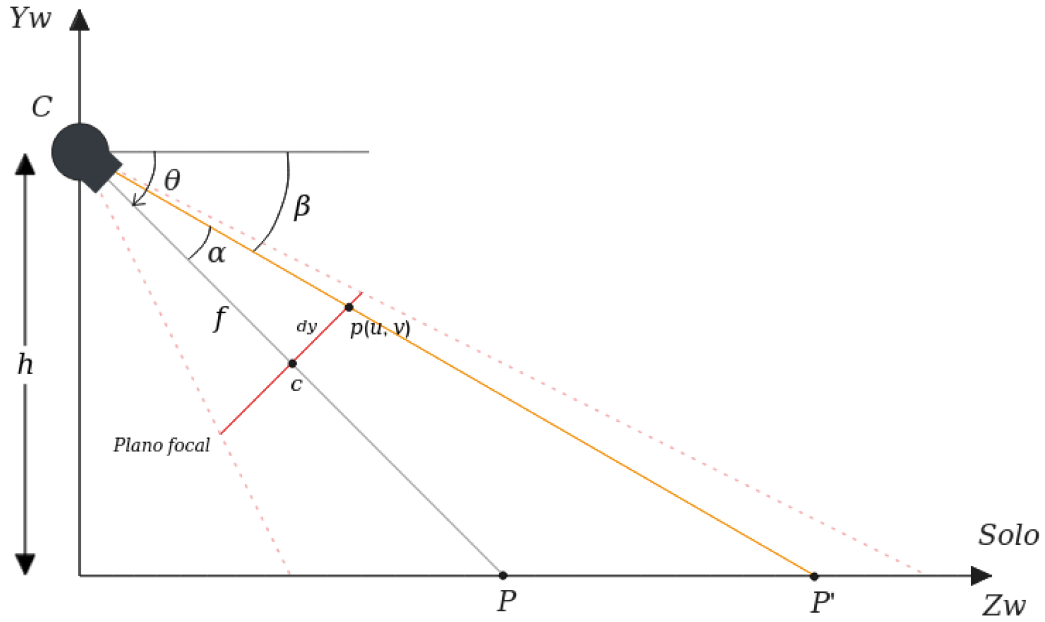


Figura 11 – Projeção de um ponto com a câmera inclinada

Onde, C é o centro de coordenadas da câmera que está rotacionado em um ângulo θ no sentido horário, P é o ponto no mundo que está projetado sobre o centro focal c . A distância focal f define a distância do plano focal, o ângulo α define a relação de d_y com o eixo focal. Por fim, o ângulo β é um ângulo auxiliar para facilitar na resolução trigonométrica. De forma que:

$$\beta = \theta - \alpha \quad (4.3)$$

Um ponto $p(u, v)$ capturado pelas imagens está sujeito a deformações provenientes das características construtivas da câmera, por esse motivo é feito uma transformada pela função *undistortPoints* da biblioteca OpenCV. Essa função implementa as equações de distorção da seção 3.1 que utiliza da matriz intrínseca K e vetor de distorção D para correção e obtenção do valor das coordenadas sem distorção. Após isso, é gerado um novo par de coordenadas $u_{corrigido}, v_{corrigido}$. Os valores obtidos são normalizados com relação ao foco, para encontrar um valor capaz de obter o α , tem-se:

$$dy = v_{corrigido} * f_y \quad (4.4)$$

$$dx = u_{corrigido} * f_x \quad (4.5)$$

Desse modo é possível encontrar α :

$$\tan(\alpha) = dy/f_y \quad (4.6)$$

Portanto, o valor de P'_z é dado por:

$$P'_z = h/\tan(\beta) \quad (4.7)$$

Utilizando a equação 3.14, é possível encontrar as coordenadas P'_x , e P'_y :

$$X_c = Z_c * dx/f_x \quad (4.8)$$

$$Y_c = Z_c * dy/f_y \quad (4.9)$$

Dessa forma, esse método oferece uma interface para realizar a conversão entre um ponto capturado pela câmera e sua posição correspondente no mundo real, bem como a transformação inversa.

4.4.3 Identificação da pista

O processo de identificação da pista com auxílio de uma câmera faz uso de uma sequência de operações sobre imagens. O objetivo desta etapa é obter uma série de pontos que possam servir como trajetória de referência para os algoritmos de controle. Nesse momento é importante ressaltar que as pistas desenvolvidas ao longo do projeto apresentam uma grande distinção entre os limites do percurso desejado. Por esse motivo foi utilizada uma superfície clara com fitas escuras demarcando o trajeto a ser executado. Dessa forma é possível trabalhar com técnicas de processamento de imagens e filtros de maneira simplificada. O algoritmo desenvolvido conta com as seguintes etapas: pré-processamento das imagens, divisão de regiões de interesse, processamento das regiões, identificação dos contornos e tratamento dos resultados.

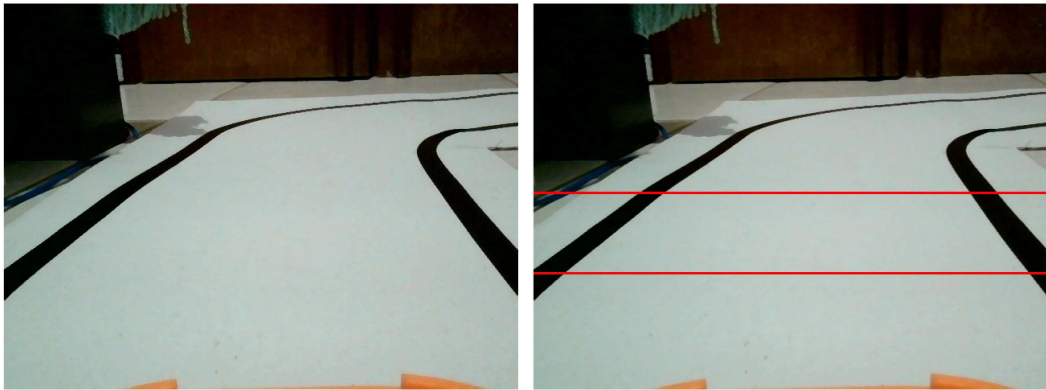


Figura 12 – Demarcação da região de interesse sobre a imagem

O Pré-processamento das imagens recebe como entrada a imagem capturada pela câmera. A primeira operação consiste em definir uma região de interesse a ser utilizada pelo algoritmo de processamento. Como a imagem da câmera é capaz de capturar uma grande parte do ambiente, apenas uma parcela da imagem corresponde a informações que podem ser utilizadas no processo de identificação da pista. Essa parcela é obtida durante o processo de calibração da câmera, analisando a imagem observada e demarcando limites superiores e inferiores que delimitam a região de interesse. A definição dessa região diminui significativamente a quantidade de pontos a serem processados, aumentando a performance geral do algoritmo. A figura 12 mostra a demarcação da região de interesse sobre uma imagem capturada pela câmera.

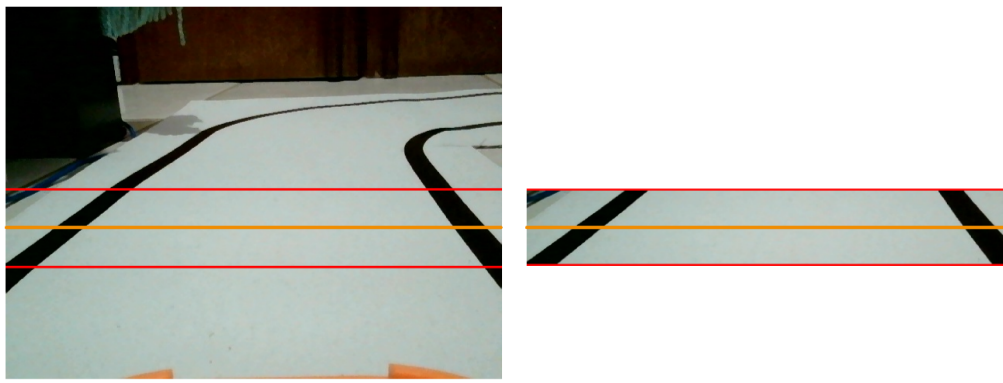


Figura 13 – Operação de Fatiamento das imagens

Após a definição da região de interesse da imagem, a próxima operação consiste em realizar um fatiamento dos dados, sendo possível dividir a região de interesse em um número de fatias (n_fatias) definido pelo autor. Essa divisão é feita com o intuito de separar a imagem em pedaços suficientes para facilitar o processamento, além disso, o número de fatias vai estar relacionado com o número de pontos finais obtidos pelo algoritmo. Cada uma dessas fatias são processadas de forma individual otimizando a extração dos dados. A figura 13 mostra a operação de fatiamento para um $n_fatias = 2$.



Figura 14 – Operações realizadas no processamento da imagem

A figura 14 mostra as etapas relacionadas ao processamento da imagem. A etapa de processamento se inicia com a coleta de cada uma das fatias extraídas pelo pré-processamento (14.a). A imagem é inicialmente convertida para a escala de cinza com o objetivo de simplificar o conjunto de dados (14.b). Após a conversão faz-se o uso de uma técnica de binarização da imagem (14.c), onde um determinado valor limite é estabelecido e cada píxel é analisado. Se o píxel analisado possuir um valor acima do limite o valor dele é definido como 1, caso esteja abaixo, recebe o valor 0. Essa operação transforma todo o campo de operação em um valor binário. É importante ressaltar que, o veículo consegue identificar tanto linhas claras sobre superfícies escuras quanto o contrário, a depender da situação basta inverter o valor do resultado da binarização.

A imagem binarizada no contexto da análise e processamento das imagens traz informações claras sobre a pista observada. Desta forma, é possível utilizar de recursos da biblioteca OpenCV para realizar a extração das informações desejadas. A biblioteca OpenCV possui a função *findContours*, capaz de receber uma imagem, analisar os gradientes e limites, e encontrar polígonos que representam contornos (14.d). Os contornos obtidos são analisados, uma vez que, um dos contornos observados podem identificar a pista. Nesta etapa é preciso definir o contorno principal (14.e). Esta operação é feita levando em consideração o tamanho do contorno e a posição relativa entre o centro do contorno e os contornos anteriores. Assim, é possível identificar e demarcar a região da pista (14.f).

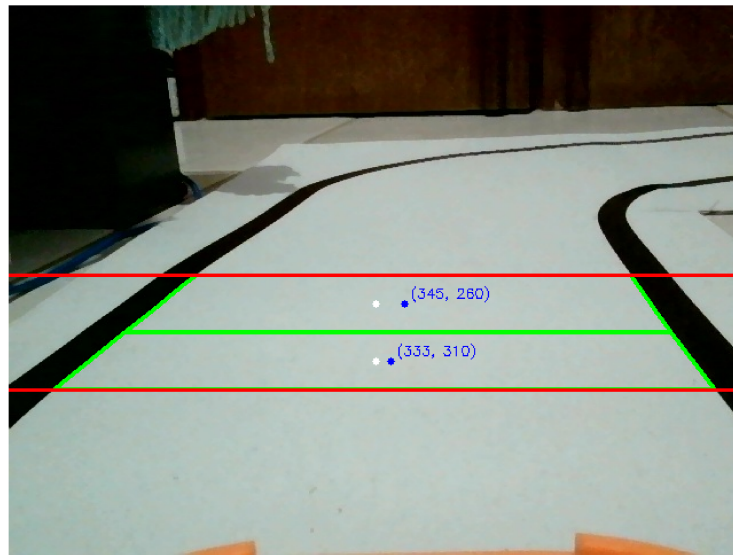


Figura 15 – Resultado do processamento de imagens

Dessa forma, tem-se o contorno principal definido para cada uma das fatias. Sendo possível identificar as coordenadas relativas ao centro dos contornos e utilização do algoritmo de conversão dos píxels da câmera em coordenadas do mundo real (4.4.2). As distâncias reais são então publicadas no tópico `/detected_points` e estão disponíveis para serem utilizadas pelos algoritmos de controle.

4.5 Controle Preditivo de Trajetória

Durante o desenvolvimento do projeto, optou-se pela utilização da técnica de Controle Preditivo Baseado em Modelo (MPC) para o planejamento de trajetórias. O MPC, mais do que um algoritmo específico, representa uma abordagem abrangente de controle de sistemas, oferecendo diversas vantagens. Entre elas, destacam-se a facilidade de compreensão do seu funcionamento; a possibilidade de impor restrições e limites às variáveis de controle e de saída; e a capacidade de lidar com sistemas que possuem múltiplas entradas e saídas, garantindo um desempenho robusto.

4.5.1 Estabelecendo a Trajetória de Referência

O sistema desenvolvido utiliza o algoritmo de identificação da pista, conforme descrito em 4.4.3, como método de percepção do ambiente. Esse algoritmo gera uma sequência de coordenadas relativas ao centro da pista identificada, as quais são publicadas via ROS em um tópico nomeado `/detected_points`. Para o controle da trajetória do veículo, foi implementado um algoritmo que opera como um nó na rede do ROS. Esse nó se subscreve ao tópico dos pontos identificados e funciona de maneira cíclica, recebendo as

informações sobre os pontos para realizar os cálculos necessários à geração da trajetória do veículo.

A detecção dos pontos da pista depende da posição da câmera e dos parâmetros intrínsecos previamente calibrados. Isso implica que os pontos identificados estão a uma distância significativa do veículo. A figura 16 exhibe as coordenadas referentes aos pontos coletados em um instante específico desse processo.

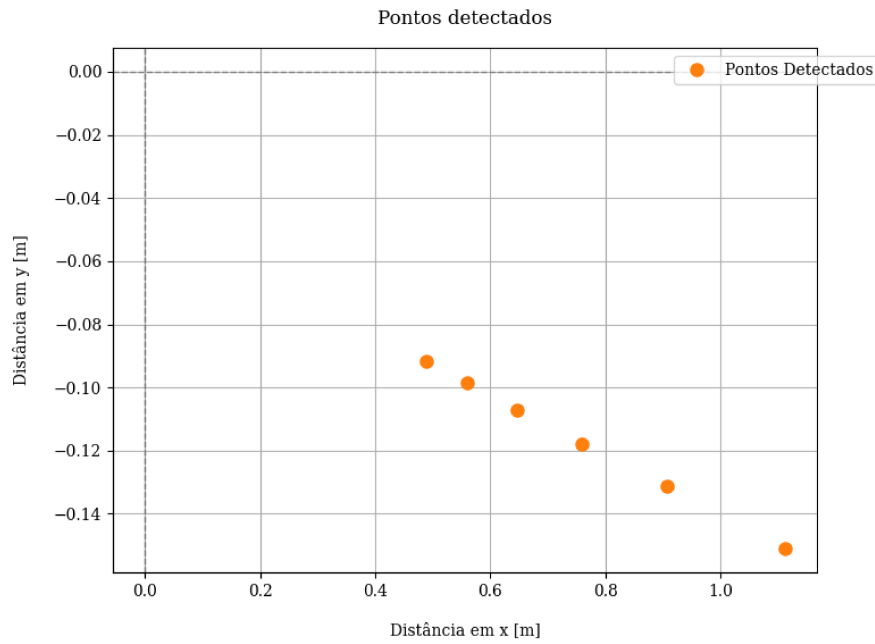


Figura 16 – Posição dos pontos detectados

As coordenadas dos pontos obtidos são utilizadas como base para a criação de uma trajetória de referência para o veículo. Essa trajetória é gerada por meio de uma interpolação cúbica sobre os pontos identificados, garantindo a continuidade do percurso e evitando descontinuidades que poderiam comprometer o desempenho do controle. Além disso, a interpolação assegura que os pontos da trajetória estejam mais próximos da origem do robô, o que vai ser útil durante o processo de otimização. A figura 17 mostra a trajetória de referência sobre os pontos obtidos da câmera.

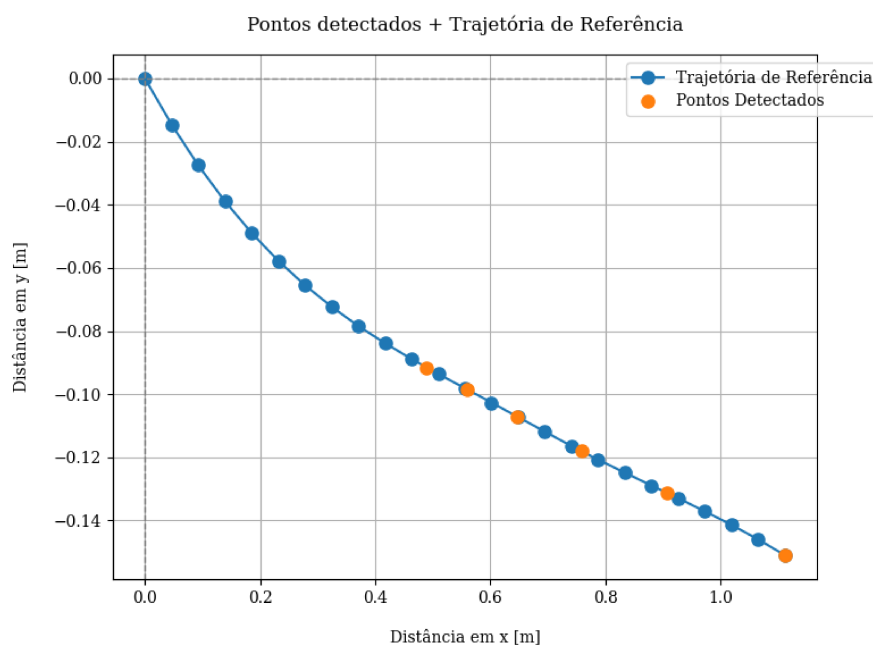


Figura 17 – Trajetória de Interpolação Cúbica

Para efeitos de simulação e teste da robustez da técnica utilizada, foi adicionado um ruído gaussiano no valor de até 10cm, sobre os pontos detectados, de forma que é possível observar o comportamento da interpolação cúbica sobre este novo conjunto de dados, como apresentado na figura 18.

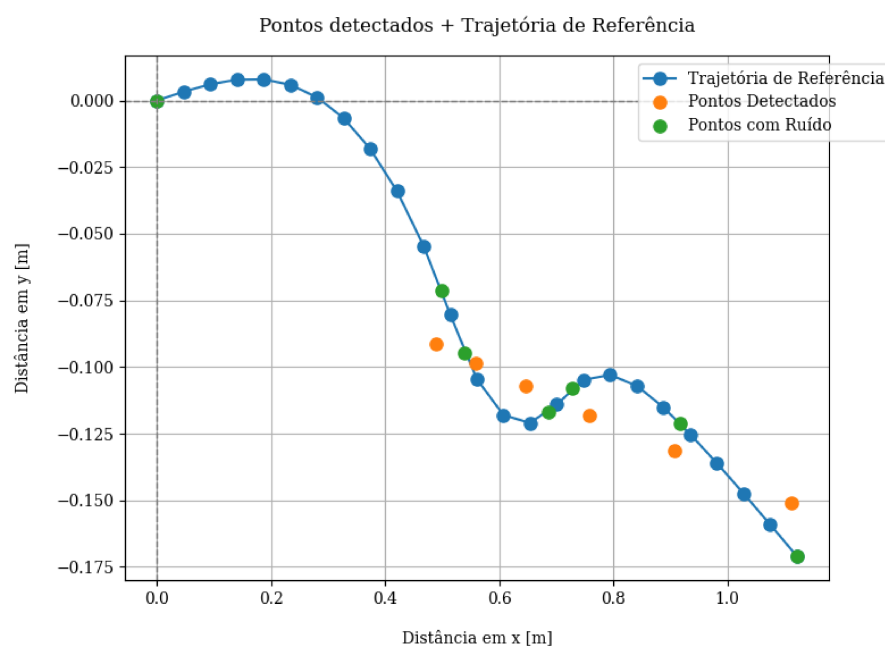


Figura 18 – Posição dos pontos detectados e Trajetória de Referência

Nota-se que a spline passar por todos os pontos pode acarretar em trajetórias cujo início se afasta grandemente do resultado esperado, por esse motivo foi adicionado um parâmetro de suavização sobre a trajetória interpolada, esse método utilizou da função *UnivariateSpline* da biblioteca *scipy*. Esse método consiste em aplicar uma função spline, que é uma combinação de polinômios em trechos entre os pontos de dados com um parâmetro de suavização que controla o quão próximo os pontos devem estar da curva. Essa técnica permite uma representação mais suave e menos suscetível a oscilações bruscas. A figura 19 mostra o resultado dessa trajetória aplicada.

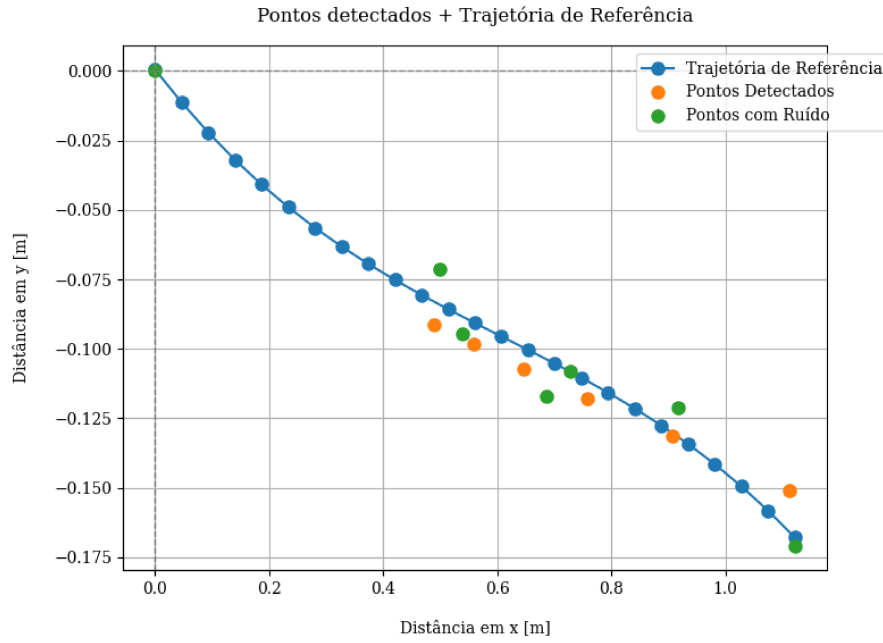


Figura 19 – Trajetória de Spline Suavizada

4.5.2 Descrição do modelo para Controle Preditivo

Com base no modelo descrito em 3.3, foi implementada uma classe para representar um veículo diferencial sobre duas rodas. A classe é inicializada com parâmetros como o raio das rodas (R), distância entre elas (d) e o passo de simulação (Δt). A cada iteração, ao receber um par de velocidades angulares das rodas, a classe calcula e retorna a nova posição (x, y) do robô. Essa funcionalidade é essencial para a aplicação da otimização no algoritmo do MPC. A equação 4.10 ilustra o modelo desenvolvido.

$$\dot{x} = f(x, u) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \cos(\theta) \\ v \sin(\theta) \\ \frac{v_r - v_l}{d} \end{bmatrix} \quad (4.10)$$

onde:

- x representa o vetor de estado $[x, y, \theta]$, com x e y sendo as coordenadas do robô e θ a orientação;
- u é o vetor de entrada $[v_r, v_l]$, velocidades de cada roda;
- d é a distância entre as rodas.

A discretização do modelo foi realizada utilizando o método de Euler para obter um modelo discreto:

$$v = (v_R + v_L) * \frac{R}{2} \quad (4.11)$$

$$\omega = (v_R - v_L) * \frac{R}{L} \quad (4.12)$$

$$\theta_{k+1} = \theta_k + \omega * \Delta t \quad (4.13)$$

$$x_{k+1} = x_k + v * \cos(\theta) * \Delta t \quad (4.14)$$

$$y_{k+1} = y_k + v * \sin(\theta) * \Delta t \quad (4.15)$$

Essa formulação permitiu simular o comportamento do sistema em intervalos discretos de tempo Δt , facilitando a implementação do controlador MPC. A escolha do intervalo de tempo deve estar combinada com a distância entre os pontos interpolados.

É importante ressaltar que para resolver a otimização foi utilizado o algoritmo IPOPT (Interior Point OPTimizer) porque ele é capaz de lidar eficientemente com problemas de otimização em sistemas não lineares. O IPOPT permite a adição de restrições ao problema, essa escolha se alinha com a necessidade do problema de controle preditivo e permite a utilização do sistema não linear.

4.5.3 Função Custo para otimização

Com o auxílio da biblioteca Pyomo, que é uma ferramenta robusta para modelagem de otimização matemática em Python, foi desenvolvido um modelo para a simulação do sistema. O objetivo da otimização é minimizar uma função custo que quantifica o desempenho do sistema em relação à trajetória desejada. Tendo esse objetivo em vista é possível definir uma equação que leve em consideração o erro entre as posições encontradas e as posições de referência. Além disso, avaliar grandes variações indesejadas entre as saídas de controle.

Os erros são multiplicados por constantes definidas e atuam como forma de penalizar as trajetórias que se afastem da referência. Por exemplo, se o veículo se afasta significativamente da trajetória de referência, o erro correspondente aumenta, resultando em um custo mais alto. O objetivo da otimização é minimizar esse custo total para assegurar que o veículo siga uma trajetória próxima à referência. A função custo utilizada no modelo é descrita em 4.16:

$$J = \sum_{t=0}^{N-1} \left[Q \left((x_t - x_{ref,t})^2 + (y_t - y_{ref,t})^2 \right) + R \left((v_{R,t} - v_{L,t})^2 \right) \right] \quad (4.16)$$

onde:

- Q : Importância do erro de rastreamento em relação às coordenadas de referência;
- R : Importância da penalização por variação brusca na velocidade;
- N : Horizonte de predição.

A próxima etapa do algoritmo consiste em definir um horizonte de predição a ser utilizado no MPC. O horizonte de predição especifica quantos passos à frente são considerados pelo controlador para a otimização. Um horizonte de predição maior permite uma previsão mais precisa, porém aumenta a complexidade computacional, enquanto um horizonte menor reduz a carga computacional, mas pode comprometer a eficácia do controle. No desenvolvimento do presente capítulo utiliza-se de um horizonte de predição de 15 pontos facilitando a confecção dos gráficos para explicação do método aplicado.

4.5.4 Previsão de Trajetória

Com a definição do modelo e da trajetória de referência, os pontos do horizonte de predição são enviados para o método que calcula a otimização. O algoritmo está contemplado com as restrições cinemáticas e relações do modelo do veículo. Com a origem sempre sobre o eixo do robô, é executado a previsão das próximas atuações de controle a fim de obter uma saída que se aproxime da trajetória de referência. Na figura 20 é possível observar a previsão de trajetória para os primeiros pontos do horizonte de predição e também atuação do controle com a velocidade de todas as rodas.

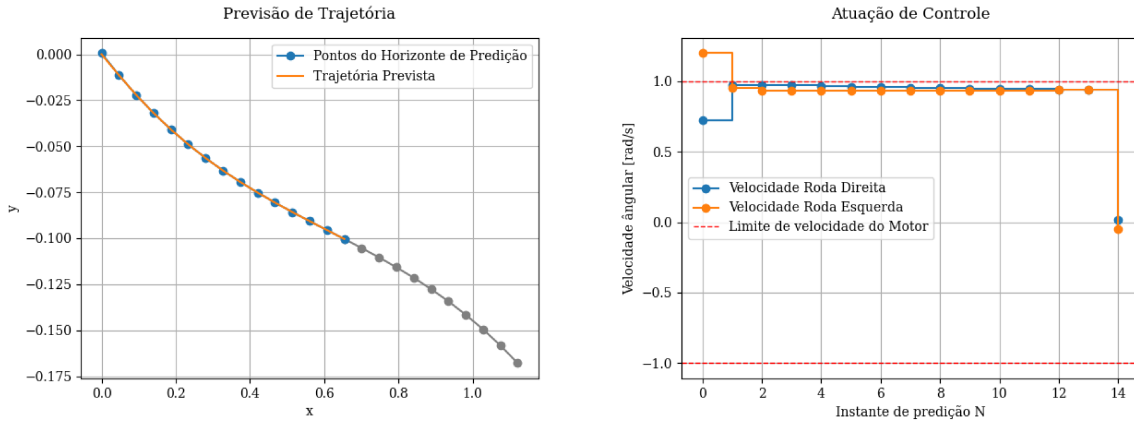


Figura 20 – Previsão de Trajetória e Atuação de Controle

Observando a figura 20 é possível perceber que a ação de controle ultrapassou a velocidade de 1.0 rad/s que é o limite de velocidade do motor que o carro está utilizando, por esse motivo, foi adicionado uma restrição de velocidade dentro do algoritmo de otimização, restringindo a velocidade máxima a ser utilizada pelo motor. A restrição é da forma $v_{\min} \leq v \leq v_{\max}$. Na figura 21 é possível verificar o resultado encontrado pela otimização respeitando os limites físicos do sistema.

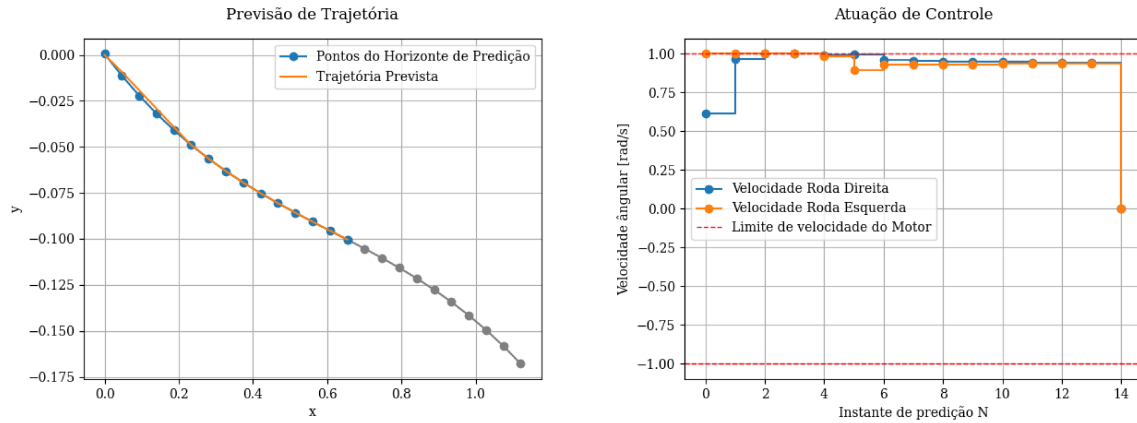


Figura 21 – Previsão de Trajetória e Atuação de Controle com restrições físicas do sistema

O algoritmo dá o seu prosseguimento aplicando a primeira ação de controle sobre o veículo. Com isso o carro se movimenta e o fluxo continua, desta vez, os pontos a frente são passados como novo horizonte de predição para o controle. Por esse motivo o horizonte de predição é também conhecido como horizonte recendente, a medida que o controle se desenvolve novos pontos são otimizados. Para efeitos de simulação foi adicionado um leve descasamento de modelo sobre as saídas aplicadas dos motores, dessa forma é possível acompanhar a evolução do sistema.

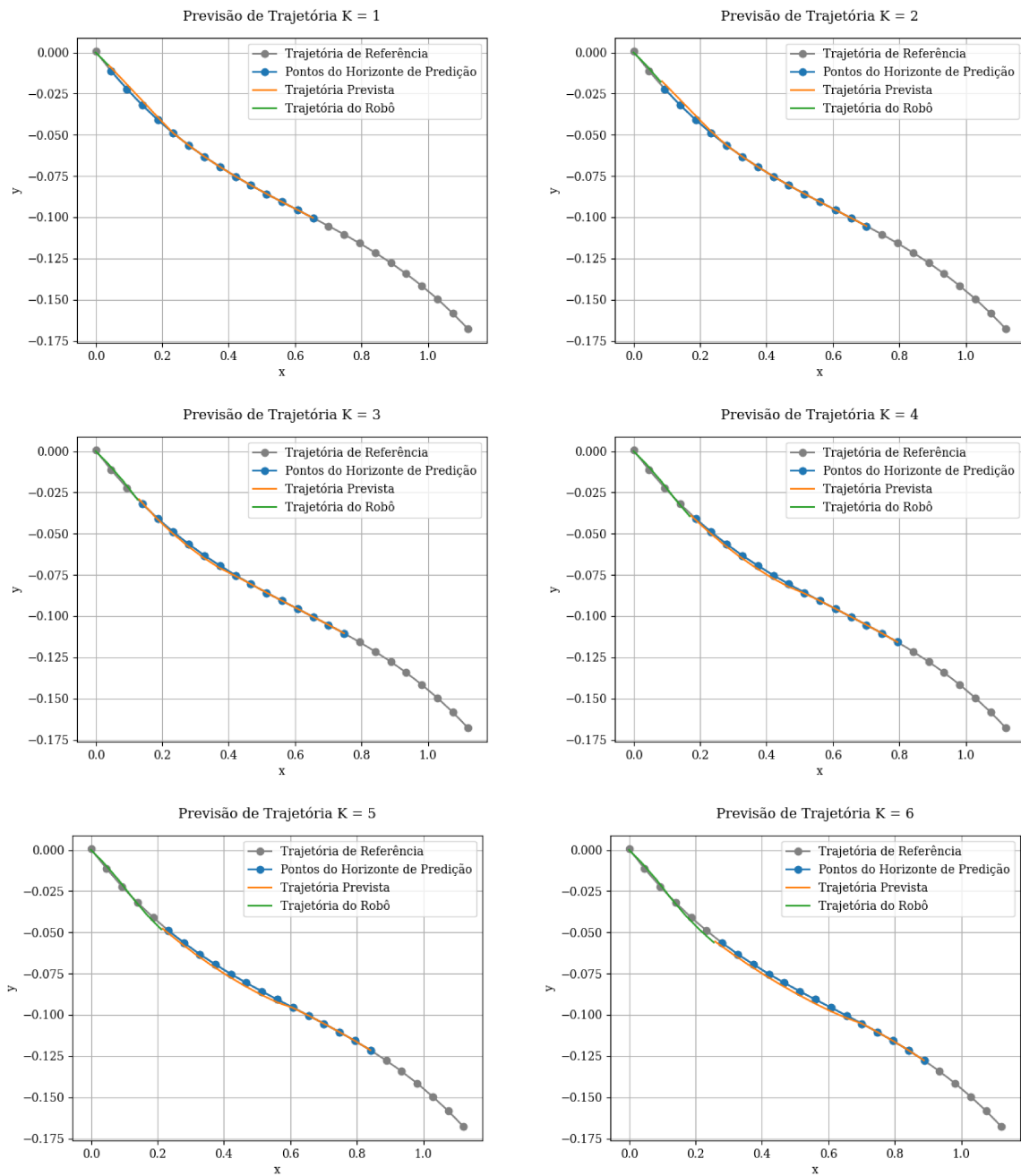


Figura 22 – Previsão de Trajetória e Atuação de Controle com restrições físicas do sistema

5 Resultados

5.1 Mapeamento da distância dos píxels da câmera

O processo de determinação da posição real dos píxels da câmera, descrito no tópico 4.4.2 é essencial para o funcionamento do veículo. Com o objetivo de coletar dados para avaliar o seu desempenho, foi implementado um algoritmo que projeta a imagem da câmera e permite a captura de pontos através de cliques do mouse. Para isso, foi posicionado em frente ao veículo uma grade de calibração com quadrados de 5cm x 5cm, de forma que foram coletados os pontos referentes aos vértices dos quadrados. A figura 23 ilustra o processo mencionado.

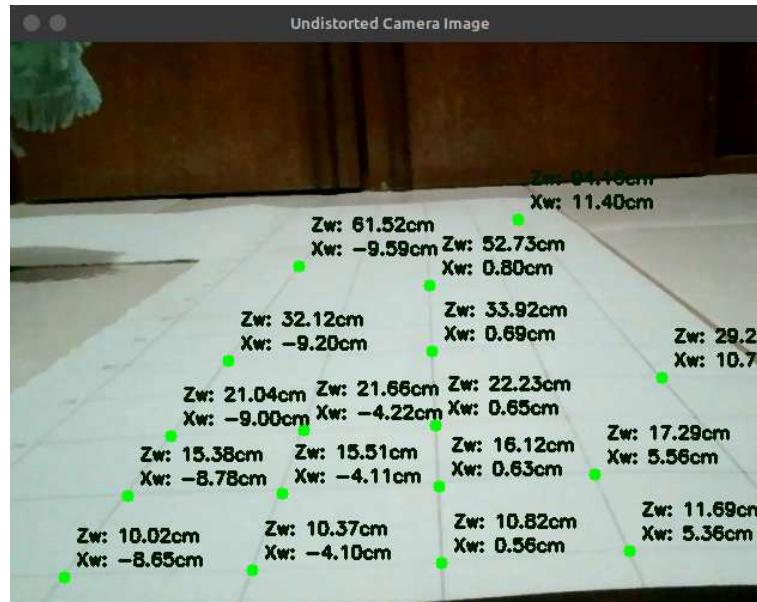


Figura 23 – Captura de pontos na grade de calibração

Nesta etapa foram coletados 172 pontos a partir dos vértices visíveis, cada ponto foi convertido em sua respectiva distância, na figura 24 é possível observar o mapa das coordenadas encontradas.

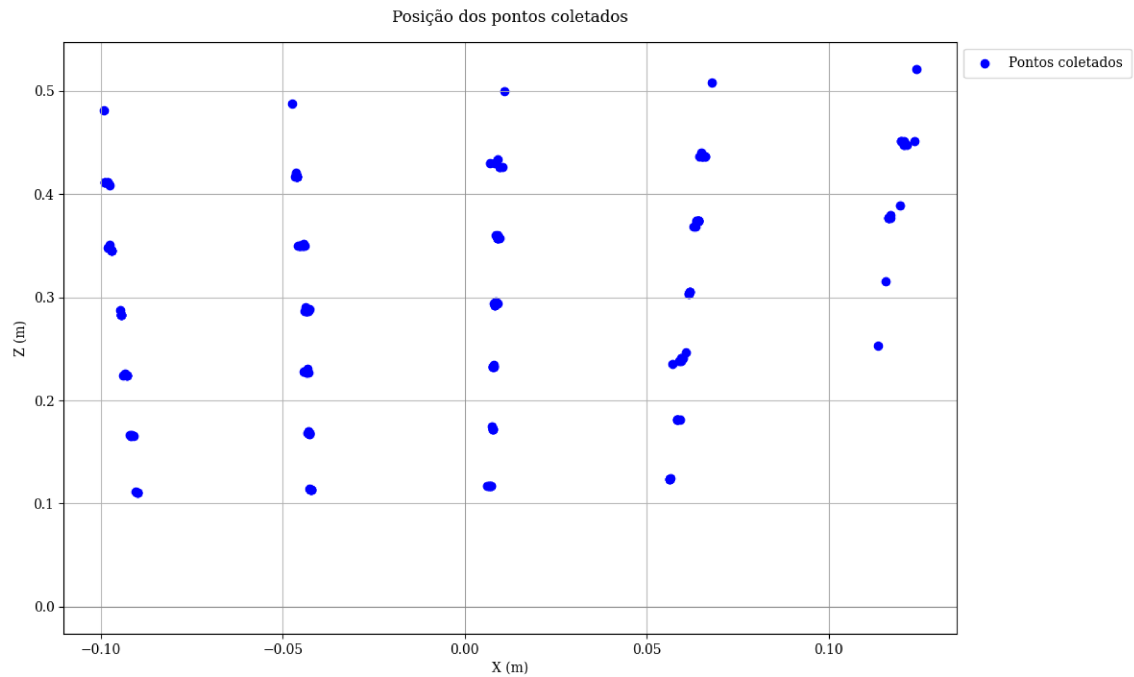


Figura 24 – Mapa dos pontos coletados

Nesse momento, visando avaliar a dispersão dos pontos em relação aos seus eixos, foi aplicada uma técnica de clusterização para agrupar os pontos com características semelhantes. Além disso, foi traçada uma linha de tendência para cada um dos grupos formados. A figura 25 mostra os pontos encontrados pelo agrupamento sobre o eixo X.

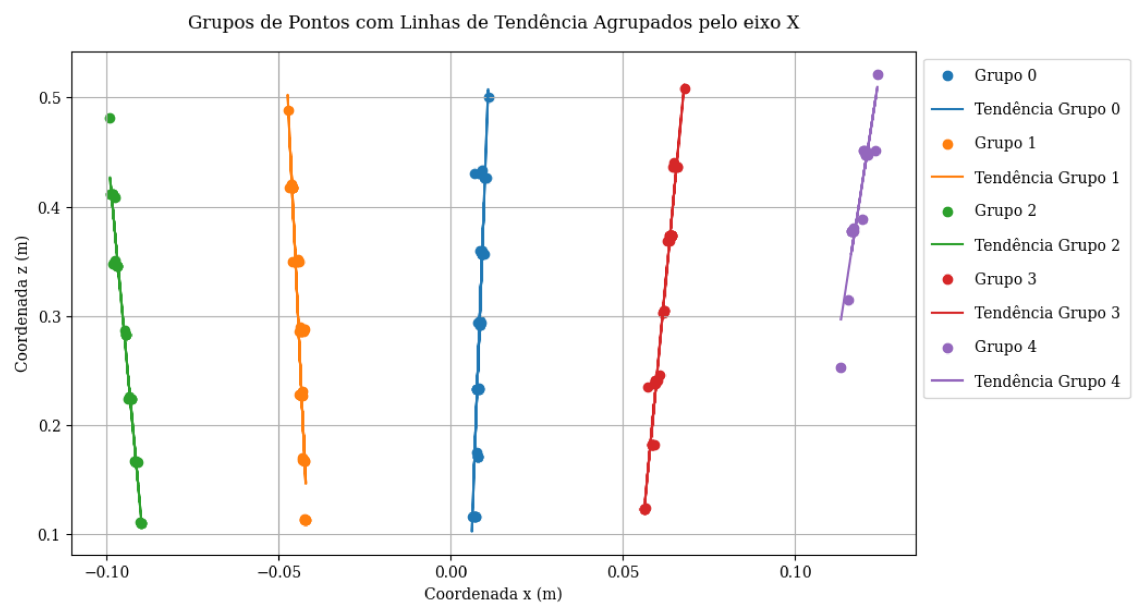


Figura 25 – Grupos de Pontos com Linhas de Tendência agrupadas pelo eixo X

A tabela 2 traz as informações de média, desvio padrão e confiança para os pontos agrupados. Para auxílio da análise foi relacionado um valor de referência que indica a

posição dos vértices utilizados. Os resultados mostram um desvio padrão máximo de 0.00310 metros, o que, para uso na aplicação torna uma ferramenta bastante precisa. Além disso, é importante ressaltar que os erros em relação a referência podem ser advindos do posicionamento e inclinação do veículo frente a grade de calibração.

Tabela 2 – Tabela de Média, Desvio Padrão e Intervalo de Confiança para os Grupos do eixo X

Grupo	Referência	Média	Desvio Padrão	Confiança
0	0.00	0.00831	0.00099	(0.00799, 0.00861)
1	-0.05	-0.04407	0.00151	(-0.04454, -0.04359)
2	-0.10	-0.09480	0.00310	(-0.09588, -0.09372)
3	0.05	0.06150	0.00327	(0.06043, 0.06256)
4	0.10	0.11888	0.00299	(0.11729, 0.12048)

De maneira análoga, foi efetuado o agrupamento dos pontos com relação ao eixo z , o que pode ser observado na figura 26.

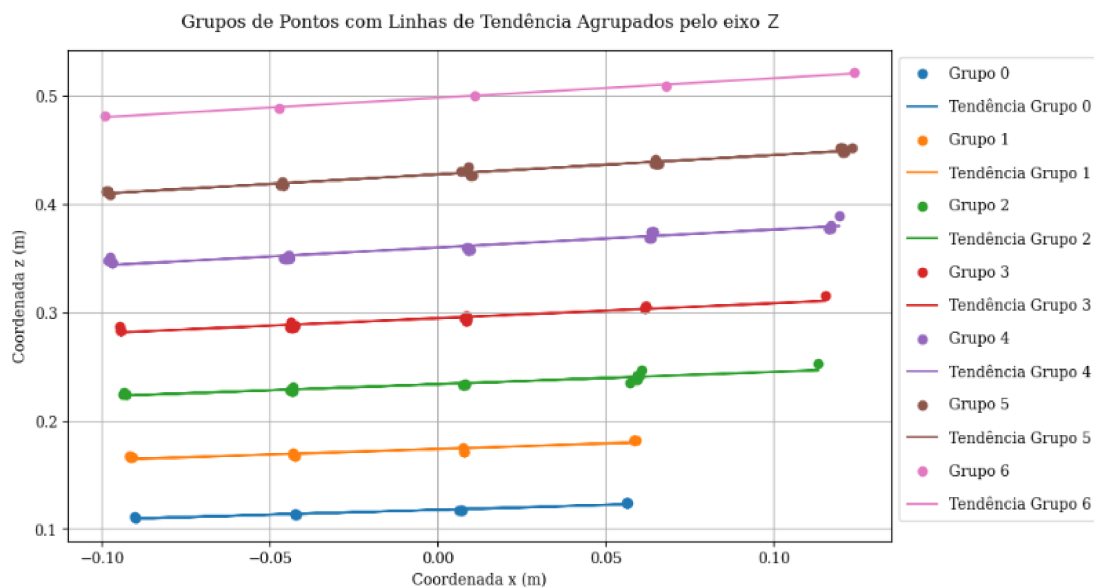


Figura 26 – Grupos de Pontos com Linhas de Tendência agrupadas pelo eixo Z

As linhas de tendência para os grupos em z apresentam uma boa aderência aos dados agrupados, a tabela 3 traz as informações referentes aos grupos observados. Nesse eixo é possível perceber que o desvio padrão atingiu o valor mais alto de 0.01605 metros para para os pontos mais distantes do robô. Além disso, apresentou também uma notável falta de acurácia durante o processo de captura dos pontos, apresentando divergências entre os valores de referências estabelecidas.

É possível que esses erros estejam associados a uma variação no ângulo medido pelo inclinômetro. Além disso, ao se efetuar uma análise sobre linhas de tendência, nota-se que houve uma falta de alinhamento durante a coleta dos pontos, trazendo o aumento do desvio padrão observado.

Tabela 3 – Tabela de Média, Desvio Padrão e Intervalo de Confiança para os Grupos do eixo Z

Grupo	Referência	Média	Desvio Padrão	Confiança
0	0.10	0.11632	0.00492	(0.11402, 0.11863)
1	0.15	0.17189	0.00612	(0.16911, 0.17467)
2	0.20	0.23309	0.00743	(0.23003, 0.23616)
3	0.25	0.29299	0.00840	(0.28967, 0.29632)
4	0.30	0.36095	0.01252	(0.35690, 0.36501)
5	0.35	0.42862	0.01430	(0.42370, 0.43353)
6	0.40	0.49977	0.01605	(0.47984, 0.51970)

O método de utilização da câmera monocular para obtenção das posições dos pontos do mundo real pode introduzir imprecisão nos resultados obtidos. No entanto, nota-se uma precisão satisfatória na captura das coordenadas no eixo x . Por outro lado, a estimativa da distância dos pontos em relação ao robô tende a apresentar um aumento de erro conforme os pontos se distanciam. Ainda assim, os pontos analisados exibem boas linhas de tendência, indicando consistência nos resultados.

De forma geral, o algoritmo oferece apresenta uma boa captura de coordenadas. Ao se analisar os agrupamentos coletados, é possível visualizar um melhor desempenho com relação aos pontos e suas coordenadas sobre o eixo x . Além disso, a região próxima ao robô apresenta um nível maior de confiança, oferecendo uma referência satisfatória para o algoritmo de controle.

5.2 Sistema de Identificação da Pista

O algoritmo descrito na seção 4.4.3 foi implementado em um código utilizando Ros, funcionando como um nó que opera para detecção das pistas. Esse nó recebe a imagem da câmera e publica em um tópico as distâncias correspondentes dos pontos encontrados. A partir desse momento, é gerada uma trajetória de referência, alguns momentos da pista foram coletados para mostrar a eficácia do algoritmo e as operações em algumas situações específicas.

Na figura 27 é possível observar que o veículo está sobre um trecho majoritariamente reto, dessa forma, o algoritmo verifica uma pequena variação no eixo y, que pode ser visualizada no gráfico, e cria uma trajetória suave.

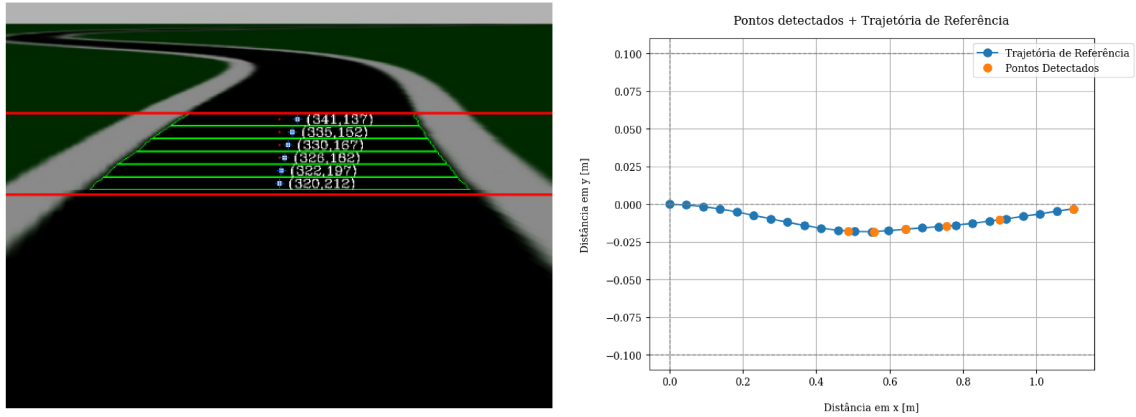


Figura 27 – Veículo sobre uma reta

Em uma situação onde o veículo está direcionado para fora da pista, ainda sim, é possível identificar o traçado de objetivo. Nesse caso, é importante mencionar que uma cor diferente entre o ambiente e a pista torna o processo de identificação da pista mais eficiente. A figura 28 mostra o robô direcionado à esquerda e a sua respectiva trajetória de referência, enquanto a figura 29 mostra o robô em um cenário à direita.

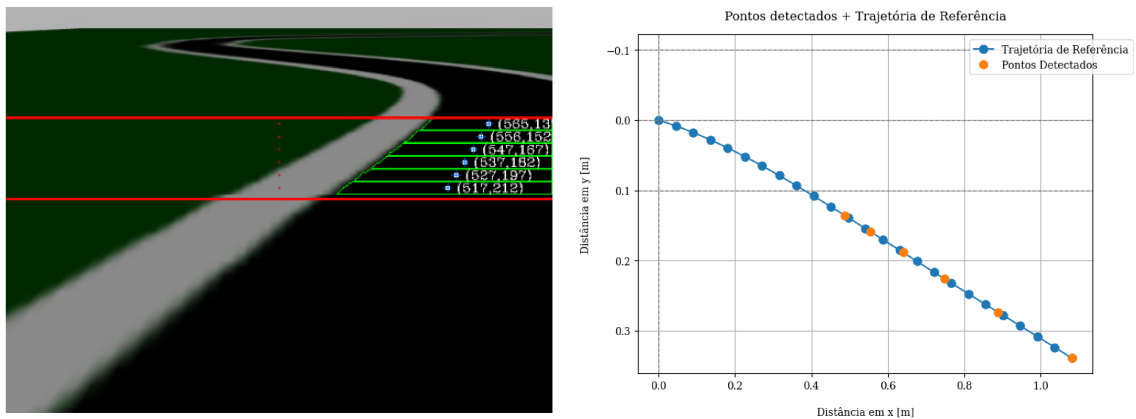


Figura 28 – Veículo orientado à esquerda da pista

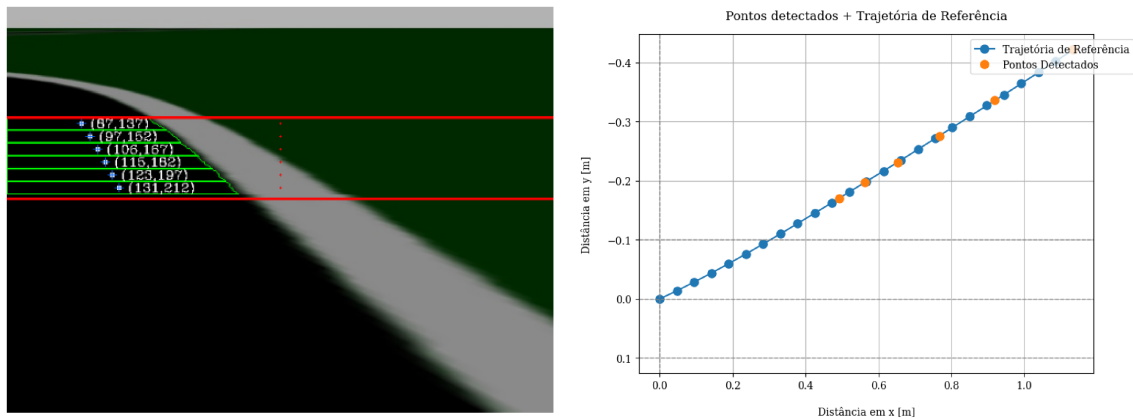


Figura 29 – Veículo orientado à direita da pista

Ainda em momentos em que o robô se encontra sobre um trecho curvo o algoritmo consegue manter o rastreamento da pista. Na figura 30 é possível observar a trajetória gerada, como o veículo se encontrava deslocado à esquerda, a trajetória sugeriu um alinhamento ao centro para facilitar na futura conversão.

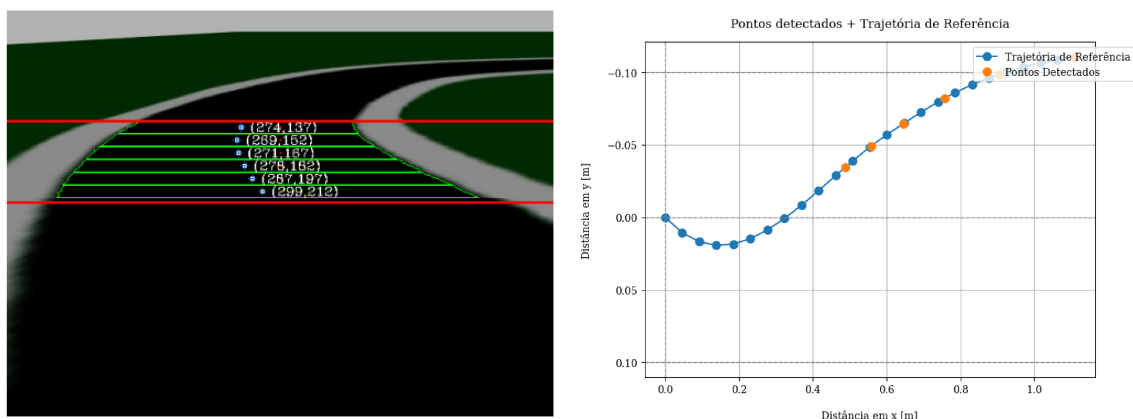


Figura 30 – Veículo em um momento anterior a curva

5.3 Sistema de Direção Autônoma

O sistema de direção autônoma é o resultado da integração de todos os componentes desenvolvidos no projeto. Nesse caso, após a verificação do funcionamento individual de cada uma das partes anteriormente, o presente tópico avalia os resultados gerais obtidos do veículo sobre uma pista. Como referência durante a simulação foi utilizada a pista com o traçado inspirado no Autódromo José Carlos Pace (vide 4.3.2). A Figura 31 mostra, à esquerda, a visão superior do robô sobre a pista e, à direita, uma visão ampliada do veículo sendo simulado no mapa.

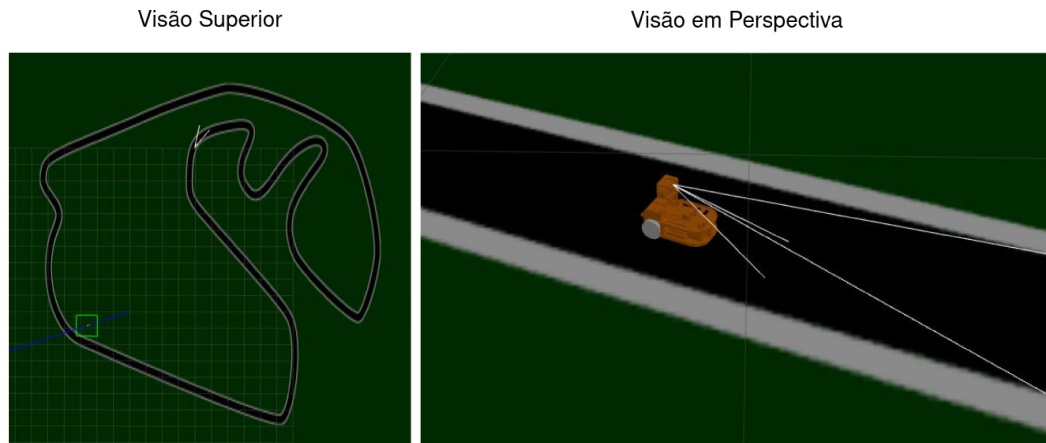


Figura 31 – Veículo em ambiente de simulação.

As imagens da pista são capturadas pela câmera virtual e processadas com os algoritmos de identificação de pista e de coordenadas, como resultado, o algoritmo de controle recebe uma trajetória para ser utilizada como referência.

O algoritmo de controle gera uma sequência de saídas a fim de minimizar os erros de rastreamento em relação a trajetória de referência, essas saídas são calculadas levando em consideração as restrições físicas do veículo impostas na confecção do modelo. A primeira das saídas resultantes é aplicada ao sistema e, a medida que o veículo se desloca, o ciclo é repetido para as novas imagens da câmera. A quantidade de saídas calculadas é conhecida como o Horizonte de Predição, o seu tamanho está relacionado ao seu custo computacional, podendo tornar mais lento a resolução do problema.

A trajetória prevista pelo MPC indica uma previsão da aplicação de todas as saídas planejadas. Essa trajetória calculada se ajusta em relação a posição e orientação em que o robô se encontra. A figura 32 traz a captura dos pontos identificados, bem como, a curva indicando a trajetória prevista calculada pelo algoritmo do MPC.

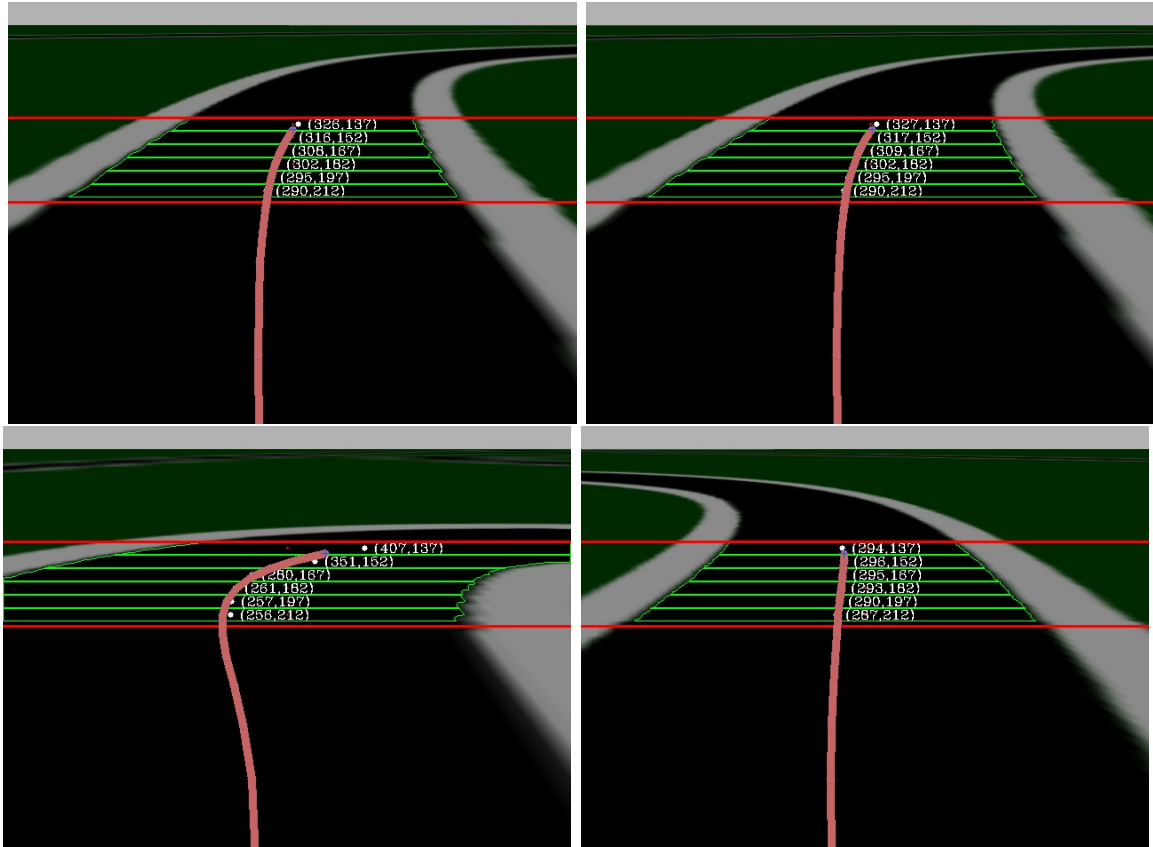


Figura 32 – Previsão de trajetória sobre o percurso.

Dessa forma, foi possível colocar o robô para completar o percurso virtual e coletar as informações referentes a odometria do processo. Foram coletados 149002 pontos referentes a posição do veículo durante a simulação, a figura 33 traz o trajeto percorrido pelo robô e a pista de referência.

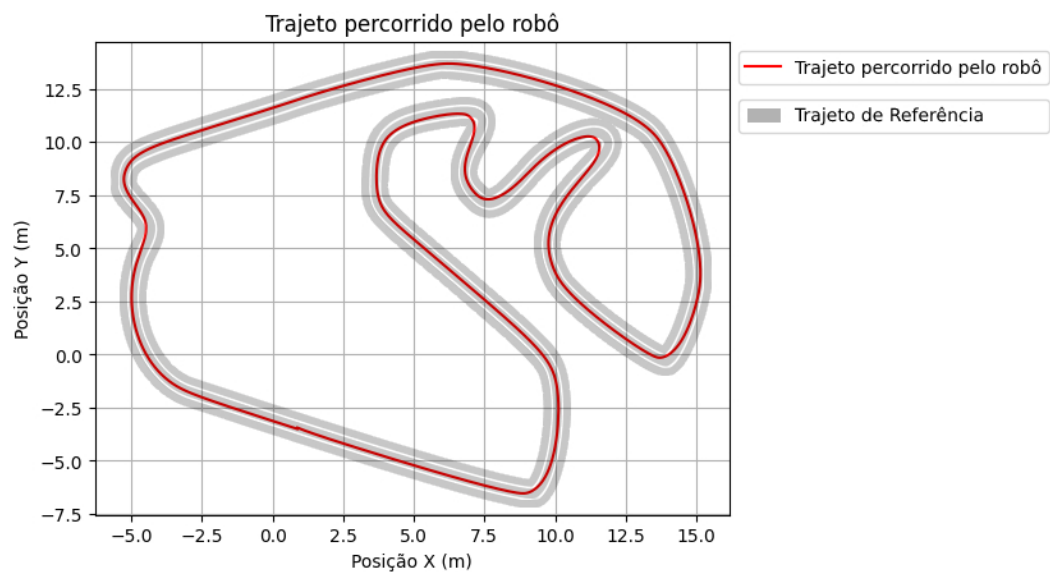


Figura 33 – Odometria coletada do veículo sobre o percurso.

É possível observar que o veículo desempenha uma boa trajetória, conseguindo seguir as curvas complexas do traçado. Ao se analisar a odometria, é possível identificar antecipações no desenvolvimento das curvas, devido a trajetória de referência coletar informações referentes a pontos futuros da pista.

As estatísticas coletadas da odometria deste processo apresentaram os resultados descritos na tabela 4.

Tabela 4 – Estatísticas de Odometria:

Item	Valor
Velocidade média (m/s):	0,06
Velocidade máxima (m/s):	0,06
Velocidade mínima (m/s):	0,00
Distância total percorrida (m):	100,61
Tempo total gasto (s):	1717,85
Tempo total gasto (min):	28,63

Ao se analisar os dados de odometria é possível perceber que a velocidade média do veículo se tornou próxima a velocidade máxima, de forma que o veículo utilizou em quase todo momento a sua capacidade máxima determinada pelo modelo. A restrição de velocidade imposta no teste foi de no máximo 0,06 m/s, dessa forma, foi possível observar que o veículo não ultrapassou em nenhum momento. No entanto, nessa velocidade, o veículo demorou cerca de 28,63 minutos para completar o percurso.

6 Conclusões e sugestões de trabalhos futuros

O trabalho realizado destaca as etapas realizadas no desenvolvimento de um sistema de direção autônoma em tempo real para um veículo com o auxílio de uma câmera. O projeto começou com a concepção do veículo e a seleção dos seus componentes eletrônicos, garantindo a eficiência, autonomia e operabilidade do sistema. A manufatura, realizada por meio da impressão 3D, permitiu a criação de uma estrutura adequada às necessidades do projeto, enquanto a descrição do veículo para simulações virtuais permitiu a utilização de um ambiente de teste seguro e eficiente.

A calibração da câmera, baseada no método de (ZHANG, 2000) e utilizando a biblioteca OpenCV, provou ser eficaz para o levantamento dos parâmetros intrínsecos e extrínsecos da câmera. Essa etapa possibilitou a conversão dos pontos capturados pela câmera em distância do mundo real. O algoritmo de identificação da pista mostrou-se capaz de reconhecer os limites da pista e extrair pontos de referência para a navegação do veículo em diversos cenários.

O algoritmo de controle, baseado na abordagem de Controle Preditivo Baseado em Modelo, demonstrou ser capaz de lidar com restrições dinâmicas e prever trajetórias com eficiência. Durante a simulação na pista virtual de Interlagos, o veículo manteve uma velocidade média de 0,06 m/s e completou o trajeto de maneira estável. Futuras modificações podem incluir ajustes nos limites de velocidade para alcançar maior desempenho sobre a pista.

A área de robótica móvel e navegação autônoma oferece um amplo campo para continuidade das pesquisas e do projeto desenvolvido. Estudos futuros podem focar na operação do algoritmo em velocidades maiores, no reconhecimento e desvio de obstáculos, bem como na interação com outros veículos. Essas melhorias podem contribuir para sistemas de navegação autônoma mais robustos e aplicáveis a uma quantidade maior de cenários e desafios.

Referências

- BROWN, D. Close-range camera calibration. In: . [s.n.], 1971. Disponível em: <<https://api.semanticscholar.org/CorpusID:56490170>>.
- CUTLER, C. R.; RAMAKER, B. L. Dynamic matrix control??a computer control algorithm. *Joint Automatic Control Conference*, v. 17, p. 72, 1980.
- GOEBEL, P. *Ros Arduino Bridge*. 2012. Pacote para integração do arduino com o ros. Disponível em: <https://github.com/hbrobotics/ros_arduino_bridge>.
- HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. 2. ed. New York, NY, USA: Cambridge University Press, 2003. ISBN 0521540518.
- LYNCH, K. M.; PARK, F. C. *Modern Robotics: Mechanics, Planning, and Control*. 1st. ed. USA: Cambridge University Press, 2017. ISBN 1107156300.
- MAZON, H.; ZACCHI, G. P.; MARTINS, R. Calibração de câmeras e fontes de erros para triangulação fotogramétrica. *Florianópolis: UFSC*, 2011. Disponível em: <<http://marte.sid.inpe.br/col/dpi.inpe.br/marte/2011/07.29.19.16/doc/p1602.pdf>>.
- OPENCV. *Camera Calibration*. 2024. OpenCV. Disponível em: <https://docs.opencv.org/4.5.4/d9/d0c/group___calib3d.html>.
- OPENCV. *Camera Calibration Tutorial*. 2024. OpenCV. Disponível em: <https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html>.
- ROBOTICS, O. *Gazebo Simulador*. 2002. Simulador de robótica. Disponível em: <<https://gazebosim.org/about>>.
- ROS2_CONTROL. *ros2 control*. 2024. The ros2 control is a framework for (real-time) control of robots using (ROS 2). Disponível em: <<https://control.ros.org/rolling/index.html>>.
- Rossiter JA. *Model-based predictive control: a practical approach*. CRC Press, 2003. Accessed on 2024/08/09. ISBN 978-0849312915. Disponível em: <>.
- SICILIANO, B. et al. *Robotics: Modelling, Planning and Control*. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1849966346.
- WÄCHTER, A.; BIEGLER, L. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, v. 106, n. 66, p. 25–57, 2006. Disponível em: <<http://dx.doi.org/10.1007/s10107-004-0559-y>>.
- ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, p. 1330–1334, December 2000. MSR-TR-98-71, Updated March 25, 1999. Disponível em: <<https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/>>.

Apêndices

APÊNDICE A – Código Desenvolvido

O Código do projeto pode ser encontrado em: <https://github.com/ooiuri/lia-mpc/>