Assistente de estacionamento para veículos de passeio

Lucas Pereira Lopes - 42887

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho orientado por:

Prof. Doutor Rui Pedro Lopes

Prof. Doutora Tatiane Tambarussi Thomaz

Bragança

Novembro de 2020

Assistente de estacionamento para veículos de passeio

Lucas Pereira Lopes - 42887

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho orientado por:

Prof. Doutor Rui Pedro Lopes

Prof. Doutora Tatiane Tambarussi Thomaz

Bragança

Novembro de 2020

Agradecimentos

Gostaria de agradecer a todas as pessoas que de alguma forma me ajudaram, influenciaram e incentivaram nesta trajetória, mas em específico a minha família, a qual fiquei distante e ausente por muito tempo para que tudo fosse possível, e por fim, a minha namorada, que sempre me apoiou e esteve comigo mesmo nos momentos mais estressantes.

Resumo

O assistente para estacionamento de veículos tem, como objetivo, auxiliar o condutor no processo de estacionamento de veículos de passeio, atendendo às dificuldades que envolvem o ato de estacionar, principalmente em horários de grandes movimentos em grandes centros urbanos, o que acaba gerando desconforto para o condutor e um efeito cascata para os engarrafamentos. Dessa forma, em um primeiro momento, utilizou-se pesquisa bibliográfica para dar suporte teórico, sobre quais seriam as melhores tecnologias a serem utilizadas durante o trajeto do protótipo, para atingir o objetivo específico de elaborar um modelo computacional que auxilie no processo de estacionamento paralelo. Para tanto, em um segundo momento, utilizou-se para validar o modelo o simulador CoppeliaSim, as linguagens de programação Python, JavaScript e TypeScript, e FrameWorks como p5.js, Node.js e a geração de uma trajetória de estacionamento por meio da aproximação de uma função sigmoide. Previamente tem-se como hipótese que o assistente é uma solução mais simples, proposta da aproximação de uma função e consegue cumprir de forma aceitável, a tarefa de propor uma trajetória de estacionamento viável para vagas paralelas.

Palavras-chave: estacionamento, simulador, trajetória, trânsito.

Abstract

D

The vehicle parking assistant aims to assist the driver in a more simplified way of parking passenger vehicles, given the problems that involve the act of parking, especially during busy hours in large urban centers, which ends up generating discomfort for the driver and a ripple effect for traffic jams. Thus, first, bibliographic research was used to provide theoretical support, about what would be the best technologies to be used during the prototype's journey, to achieve the specific objective of developing a computational model that helps in the parallel parking process. For that, in a second moment, the CoppeliaSim simulator, the programming languages Python, JavaScript and TypeScript, and FrameWorks such as p5.js, Node.js and the generation of a parking path by approximation of a sigmoid function. Previously, it is hypothesized that the assistant is a simple solution, proposed to approximate a function and manages to accept, in an acceptable way, the task of proposing a viable parking path for parallel spaces.

Keywords: parking, simulator, path, traffic.

Conteúdo

Lista de Figuras

Siglas

ABS Anti-lock Brake System. 7, 8

API Application Programming Interface. 32, 40

CLMR car-like mobile robot. 15

CNH Carteira Nacional de Habilitação. 2

CSS Cascading Style Sheets. 35

DARPA Defense Advanced Research Projects Agency. 11

EBA Emergency Brake Assitance. 9

ESP Eletronic Stability Program. 7

FIS Fuzzy Inference Systems. 17

HTML Hypertext Markup Language. 33, 35

HTTP Hypertext Transfer Protocol. 39, 40, 42

SEVA3D Simulador de Estacionamentos de Veículos Autônomos em um ambiente tridimensional. 18

TCS Traction Control System. 8

URL Uniform Resource Locator. 45

Capítulo 1

Introdução

Segundo Franz e Seberino [1], para muitos cientistas a roda é considerada a maior invenção de todos os tempos, originada a partir do transporte de materiais com troncos de àrvore. Com o surgimento da roda, foi possível a criação de novos meios de locomoção, como por exemplo o automóvel, o qual em sua época de invenção possuía somente essa função, de transportar seres e objetos de maneira mais ágil. No entanto, de acordo com Giucci [2], o automóvel foi definido por Karl Benz como representante da era dos motores.

O automóvel surgiu como objeto de luxo para classes sociais elevadas, contrapondo-se à utilização do cavalo e do trem como meio transporte, indicando seu impacto e alta influência por meio das publicidades [2]. No entanto, com o desenvolvimento da tecnologia, o automóvel expandiu-se por todo o globo terrestre, tornando-se acessível aos mais diversos públicos e assumindo os mais diversos usos como, por exemplo, o transporte de cargas e pessoas, hobby, colecionáveis e outros. Contudo, este crescimento ocasionou na eclosão de novos problemas, tecnológicos e sociais.

Este trabalho surge diante da problemática de estacionar um veículo de maneira precisa e ágil, auxiliando os condutores a realizar uma baliza com tranquilidade e segurança. Propõe-se então um modelo computacional capaz de identificar uma vaga e fornecer comandos aos motoristas de maneira que este os consiga reproduzir e efetuar uma baliza de sucesso.

Como resultado, espera-se obter a diminuição de acidentes que possam ocorrer como

consequência da tomada de decisão incorreta por parte de condutores. Além disso, proporcionar segurança aos motoristas ao saírem de casa com seus automóveis sem se preocuparem com o momento da parada, propiciando maior conforto ao indivíduo, visto que o assistente detecta a vaga de maneira mais rápida e o estacionamento de maneira mais eficaz.

1.1 Objetivos

Este trabalho tem por objetivo geral elaborar um modelo computacional que auxilie no processo de estacionamento de veículos de passeios. Mais especificamente, pretende-se:

1. Estudar as técnicas necessárias para realização da baliza;
2. Identificar os dispositivos necessários para obtenção de dados;
3. Criar um ambiente de simulação para validação do sistema.

1.2 Contribuições do trabalho

Pressupõe-se que todo motorista tenha tido uma formação sobre o processo de estacionar um veículo, visto que é um quesito obrigatório para obtenção da Carteira Nacional de Habilitação (CNH) no Brasil. Porém no trânsito não é tão simples e usual para todos os motoristas. Este cenário é caracterizado por um alto nível de incerteza pois, dentre uma das razões, a posição com que os carros estão posicionados nas vagas é variado, necessitando de diferentes conhecimentos para tomada de decisão em cada situação.

Os impactos relacionados com a falta de conhecimento sobre o processo de baliza, na prática, podem ocasionar na falta de confiança por parte do motorista, impedindo-o de posicionar seu veículo na vaga pretendida. Em vias públicas de muito movimento, essa dificuldade interfere diretamente no trânsito, visto que quando um motorista está efetuando uma manobra, o próximo deve aguardar, pois não há espaço para uma ultrapassagem, criando assim um efeito de cascata.

Neste contexto, existem sistemas capazes de realizar o procedimento de baliza autonomamente sem a interferência do condutor, porém estes ainda se encontram restritos a veículos cujo valor é elevado e o proprietário deve optar pelo acessório no momento da compra na concessionária. Assim, o segmento de sistemas assistentes de estacionamento carecem de um sistema capaz de auxiliar o próprio condutor a estacionar seu veículo de maneira fácil e segura. Com o desenvolvimento deste trabalho espera-se contribuir nos seguintes tópicos:

- Auxiliar o condutor durante o processo de baliza;
- Contribuir no aprendizado sobre o processo de baliza por meio do uso contínuo do sistema;
- Diminuir desconfortos e inseguranças por parte dos motoristas durante o estacionamento.

1.3 Justificativa

Modelos analíticos e de simulação possibilitam entender um processo, propor uma resposta baseado em modelos matemáticos existentes, analisar seus principais pontos de falha, bem como simular o procedimento de funcionamento do modelo. Desta forma, este trabalho propõe-se a criar um modelo capaz de entender e simular o processo de estacionamento.

Para aplicação será necessário a utilização de uma interface capaz de interagir com o usuário de forma a passar as instruções para o condutor, tendo a necessidade então da construção de um sistema para realizar tal tarefa. Além da interação com o usuário, a interação com o sistema será feita a partir da leitura dos dados de distância do veículo até cada possível obstáculo.

Espera-se, com a elaboração do modelo, contribuir no auxílio e construção de futuros projetos envolvendo os temas empregados neste trabalho que abordam ambientes simulados, matemática, entre outros.

1.4 Delimitações do trabalho

O modelo proposto neste trabalho destina-se somente a veículos de passeio como carros ou caminhonetes, não se aplicando a caminhões, por exemplo. A ferramenta destina-se a auxiliar o processo de baliza em que vagas e veículos estacionados estejam paralelos ao sentido da via em que o condutor percorre, de forma que não será disponibilizada uma resposta para vagas perpendiculares ou que apresentem algum tipo de ângulo para com a via, como por exemplo vagas a 45º. O modelo será simulado computacionalmente devido ao elevado custo e a complexidade técnica para se utilizar um veículo real.

O sistema só irá entrar em ação quando o usuário apertar um botão. Sensores irão verificar o tamanho da vaga e não os espaços ao seu redor.

1.5 Estrutura do documento

Este documento está dividido em 6 capítulos, sendo este primeiro a introdução, ilustrando os objetivos e intenções do trabalho de modo geral. No Capítulo 2 é apresentado o contexto histórico acerca do automóvel e suas tecnologias, conceitos sobre estacionamento autônomo, estudo sobre o processo de baliza e por fim uma revisão dos trabalhos relacionados. Em sequência, o Capítulo 3 define a metodologia utilizada para desenvolvimento deste trabalho e o Capítulo 4 exibe todos os pormenores sobre o desenvolvimento do trabalho. O Capítulo 5 traz os resultados encontrados após os testes realizados no modelo desenvolvido e por fim, o Capítulo 6 traz as considerações finais e sugestões para trabalhos futuros.

Capítulo 2

Referencial teórico

O automóvel está presente na sociedade moderna atual marcando a ânsia pela modernidade. Melo [3] afirma que as invenções, como o automóvel, caracterizam o século XIX além de reestruturar a forma de viver, abrangendo as mais diversas utilizações, como meio de locomoção, transporte de pessoas ou de carga, ainda ampliava a possibilidade de passeio ou a sua utilização na prática de esportes, como por exemplo, o automobilismo.

A percepção histórica descrita por Giucci [2] compara diferentes situações que ganharam novos entendimentos com o automóvel:

> O automóvel é o símbolo por excelência do moderno no início do século XX. Sua chegada a diferentes partes do mundo ilustra a trajetória irresistível da mobilidade. Chega a máquina bufante, o novo sáurio mecânico, o carro de fogo, envolvido numa nuvem de pó. E montado no cavalo mecânico chega o mensageiro da motorização. Enquanto o arauto medieval levava mensagens, determinava as festas de cavalaria e organizava os registros da nobreza, o piloto introduz o não visto e o estranho, na forma de antecipação do futuro. Vem de longe anunciando grande transformação.

Segundo Corassa [4], a interação que os condutores possuem com seus veículos é responsável por atribuir novos sentidos, como, por exemplo, as mudanças no cenário descrito anteriormente para o contexto atual onde o automóvel é popular dentro da sociedade,

estabelecendo uma afinidade que simetriza a relação e sentimentos que possuem com suas próprias casas. Serafim et al [5] relatam que o automóvel foi o responsável por substituir carruagens movidas por tração animal, visto que se fazia necessário um meio de locomoção mais rápido.

Nubel [6], Broy e Schmidt [7] indicam que o primeiro automóvel com motor de combustão interna foi desenvolvido e patenteado por Karl Benz, denominado Benz Patent Motorwagen em 1886, conforme ilustrado na Figura 2.1.

Figura 2.1: Automóvel Benz Patent Motorwagen [8]

De acordo com Melo [3], o primeiro automóvel a desembarcar em território brasileiro fora fabricado pela marca Peugeot na Europa, e trazido ao Brasil por Alberto Santos Dumont em 1891. Pode-se dizer que o crescimento no número de importações nos períodos seguintes deve-se ao impacto causado ao membros da elite com a chegada do veículo em 1891 no Porto de Santos.

Do ponto de vista da produção no Brasil, existe uma incerteza a respeito de qual teria sido o primeiro automóvel a ser produzido em território nacional. Segundo Marson [9] teria sido o Romi-Isetta quando em 1954 a empresa Romi teria obtido o direito para fabricar o veículo (Figura 2.2). Contudo, de acordo com Brandão [10], o primeiro automóvel nacional

Figura 2.2: Romi-Isetta [11]

Figura 2.3: perua DKW-Vemag [12]

seria uma perua DKW produzida pela empresa DKW-Vemag em 1956 (Figura 2.3).

Casier, Moens e Appeltans [13] expõem que os primeiros automóveis eram compostos por sistemas mecânicos e não incluíam muitos sistemas elétricos, visto que a bateria não teria capacidade de fornecer energia suficiente e por não ser confiável. Porém, os avanços na tecnologia dos componentes resultaram no aumento da velocidade com que novos sistemas elétricos pudessem fazer parte dos veículos.

De acordo com Casier, Moens e Appeltans [13], após a década de 70 com a diminuição dos custos e o aumento da confiabilidade, os sistemas elétricos passaram a ser substituídos por sistemas eletrônicos, sendo utilizados em um primeiro momento em sistemas não críticos de forma não interferir no processo de direção, atuando em setores como conforto e conveniência. Com o aumento da complexidade e segurança dos sistemas eletrônicos, aliado à utilização de metodologias, os circuitos passam a atuar com sistemas críticos relacionados à segurança, como por exemplo Anti-lock Brake System (ABS), Eletronic

Stability Program (ESP), airbags, controle de emissão de poluentes, entre outros.

Com o avanço da implantação de sistemas eletrônicos nos veículos é possível perceber novos horizontes e maneiras para se lidar com os automóveis, assim como com as funções e recursos que um veículo pode disponibilizar para o condutor e os passageiros. Neste contexto, Casier, Moens, Appeltans [13] relataram que os sistemas eletrônicos utilizados inicialmente não exerciam influência sobre a condução do veículo, ficando este por total responsabilidade do condutor, mas auxiliavam na eficiência do automóvel e podiam reagir a possíveis erros do motorista.

Conforme Wei, Pissardini e Sousa Júnior [14], atualmente existem diversas implementações de recursos e tecnologias nos veículos de maneira a fornecer uma experiência de condução melhorada, como por exemplo:

- Sistemas capazes de fornecer uma melhoria no ambiente de direção, como reduzir distrações de maneira a aumentar o foco do condutor no processo de direção, focando na via em que se encontra. Exemplos desse tipo de tecnologia podem ser o acionamento automático dos faróis, sistemas e sensores capazes de detectar desvio do foco para fora da via ou até mesmo a sonolência por meio do fechar das pálpebras;

- Sistemas embarcados com o objetivo de auxiliar no ato de dirigir, atuando diretamente sobre o comportamento do veículo de maneira a mantê-lo dentro das condições do ambiente em que o veículo se encontra e realizar a correção de pequenos erros ou aumentar a eficiência de um recurso do automóvel. Dentre este tipo de auxílio, destaca-se o ABS que evita o travamento das rodas durante a aplicação de uma força sobre o pedal de freio do veículo, consequentemente melhorando a eficácia de uma frenagem. Outro tipo de assistência é o Traction Control System (TCS) que controla individualmente a força que é distribuída às rodas, permitindo uma melhora no controle do veículo em situações de condução sobre diferentes tipos de superfícies de atrito;

- Auxílios que estendem o alcance dos sentidos do condutor, como, por exemplo, sensores capazes de detectar pedestres, obstáculos, sinais de trânsitos e enviar avisos

ao condutor de veículo de maneira a prever situações e tomar uma decisão antecipada com mais informações sobre o ambiente;

- Em casos que o acidente se torna uma situação inevitável, alguns sistemas dos veículos podem entrar em ação de maneira prévia reduzindo os danos para os ocupantes e pessoas externas ao veículo, como, por exemplo, o Emergency Brake Assitance (EBA) que aciona os freios automaticamente ao detectar uma situação de impacto iminente, conforme ilustra a Figura 2.4. Outra importante ação que pode ser realizada é desligar automaticamente o fluxo de combustível de modo a evitar possíveis explosões após a colisão.

De acordo com a Figura 2.4 é possível observar a aplicação da tecnologia em situações de risco, onde no caso de um veículo sem o sistema EBA resultaria em uma colisão iminente, visto que o motorista não conseguiria manter uma correta pressão nos freios com o intuito de parar o automóvel. Porém no caso de um cenário com EBA, a colisão poderia ser evitada uma vez que o poder de frenagem aplicado resultaria em uma distância menor durante este processo.

2.1 Veículos autônomos

Segundo Kröger [16], a ocorrência de acidentes envolvendo automóveis possuía, como causa primária, decisões incorretas por parte dos condutores, ocasionando um crescente número de acidentes fatais nos Estados Unidos em 1920, tornando-se um grande problema social.

Para Pissardini, Wei e Sousa Júnior [17], a construção de veículos autônomos de transporte terrestre (Figura 2.5), é motivada pela expectativa de diminuição de acidentes de trânsito obtidos a partir da eliminação da condução humana, seja em tempo integral ou parcial, alcançada por meio de um sistema computacional capaz de combinar algoritmos e ferramentas para sensoriamento e tomada de decisão. De acordo com Stiller, Farber e Kammel [18] sistemas para auxílio de direção devem ter como objetivo a capacidade de

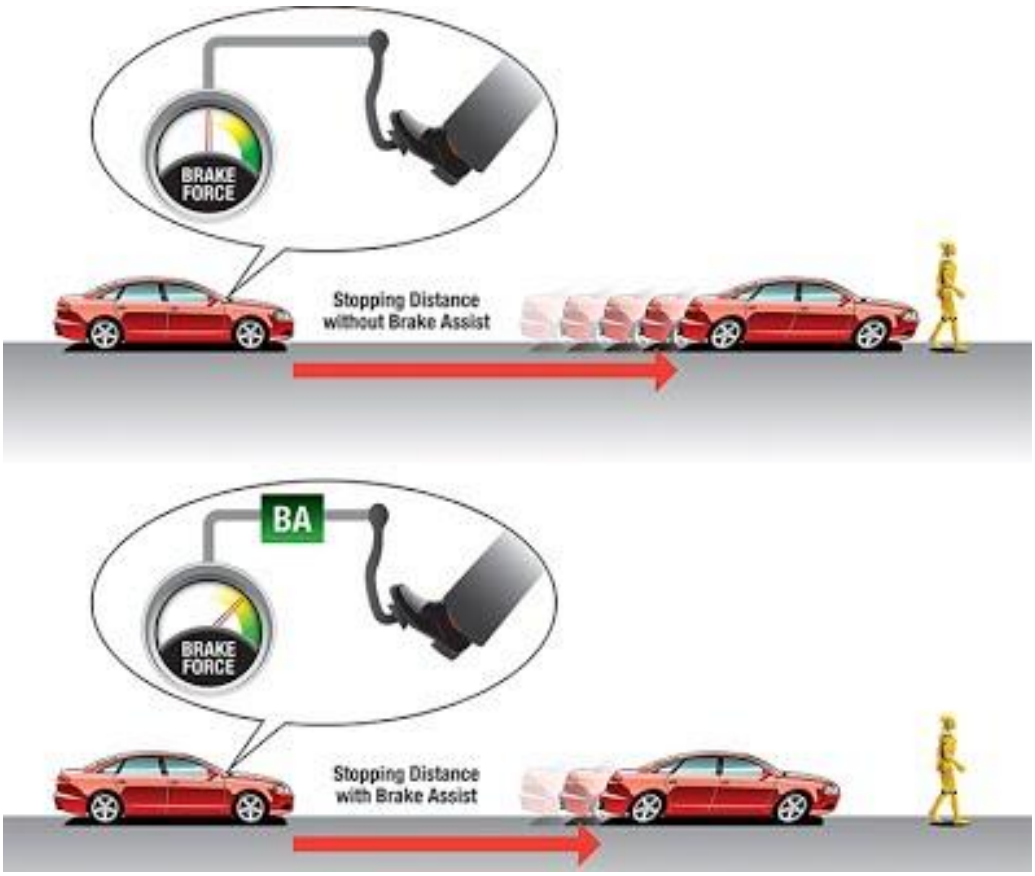

Figura 2.4: Emergency Brake Assistance [15]

entender a situação e tomar uma ação adequada ao ambiente.

Para Waiselfisz [20], a justificativa da utilização de veículos autônomos é perceptível quando se trata de números, atendendo a que a frota de veículos automotores no Brasil em 1996 era próxima de 17 milhões e, em 2010, esse número alcançou 31 milhões. Nesse mesmo período, foram registrados meio milhão de óbitos relacionados diretamente com acidentes de trânsito.

Segundo Shiller e Gwo [21], a criação de veículos autônomos consiste em selecionar um caminho a seguir, determinar a velocidade do automóvel, evitando obstáculos e minimizando alguma função de custo como, por exemplo, tempo ou energia. Desta forma, foi proposto um método para planejar a movimentação de um veículo autônomo sobre superfícies em geral. O método apresentado consiste em obter um melhor caminho sem obstáculos entre dois pontos, aliado à otimização de tempo, considerando um ambiente

Figura 2.5: Carro autônomo do Google [19]

dinâmico com outros veículos, a topografia do terreno e a mobilidade do caminho.

Tendo como motivação e validação do experimento no desafio urbano proposto pela Defense Advanced Research Projects Agency (DARPA) em 2007, no qual veículos devem navegar autonomamente sobre a área de estacionamentos, Dolgov et al. [22] descreveram um algoritmo para planejamento de caminhos para um veículo autônomo, no qual, os obstáculos serão detectados online pelos sensores do robô.

Dessa forma, a aplicação da solução é composta por duas fases, a primeira consiste da utilização de uma variante do algoritmo A* para obter uma trajetória viável. A segunda fase incrementa a qualidade da solução por meio de uma otimização numérica não linear. Posteriormente, estende-se ao algoritmo utilizar um conhecimento topológico do ambiente, guiando a procura por um caminho, obtendo uma busca mais rápida e uma trajetória final mais adequada ao ambiente.

Conforme Osório, Heinen e Fortes [23], atribuir a um sistema a capacidade de raciocínio inteligente e a interação com o meio tem motivado um grande número de pesquisadores. Neste contexto, existem atuações diversas onde este tipo de tecnologia está sendo aplicado, como por exemplo, robôs desarmadores de bombas ou carros autônomos, em ambos é compartilhada a capacidade de receber leituras de sensores que lhes dão informações sobre o ambiente e são capazes de gerar comandos permitindo seu deslocamento de forma segura.

2.2 Técnicas para baliza

Uma vez que o condutor identifica uma vaga e inicia o processo de estacionamento, faz-se elementar entender o cenário no qual o veículo está inserido, bem como, propor um trajeto a ser seguido, desta maneira, uma forma de compreender computacionalmente como o automóvel irá se descolar é fundamental. Heinen et al. [24] expressam que uma maneira muito utilizada de configurar tal forma de operar robôs móveis autônomos baseia-se no modelo cinemático de Ackerman, onde um sistema de direção é composto geometricamente por rodas frontais com direito de giro e um eixo traseiro fixo, resultando em um centro de rotação para movimento do veículo, de maneira que é possível entender o deslocamento por meio de equações matemáticas.

Existem diversas implementações e técnicas que possibilitam criar um percurso para o veículo com o objetivo de o estacionar em uma vaga. Prado [25] aborda a criação de um sistema autônomo capaz de criar e executar um planejamento de caminho para estacionamento de um veículo, no qual mapeia computacionalmente o ambiente a partir da leitura de sensores, define os movimentos que o veículo poderá executar baseando-se nas equações do modelo de Ackerman e, por fim, gera uma conjunto de possíveis estados formados por coordenadas (x, y, θ), em que é o ângulo formado entre um determinado ponto estabelecido previamente no carro (centro de massa) e a vaga pretendida, partindo da posição atual até totalmente parado na vaga, sendo a trajetória obtida por meio da aplicação de um algoritmo de busca.

Por outro lado, Osório, Heinen e Fortes [23] utilizaram o conhecimento humano como forma de entender quais ações e passos devem ser efetuados para concluir com sucesso o processo de baliza, na qual, com o auxílio de um especialista humano, criaram um autômato finito, que continha os estados e ações que o veículo enfrenta e quais ações realiza. Contudo, o processo de definição de regras para o autômato é trabalhoso, por esta razão, utilizaram-se de uma rede neural artificial para obtenção do conhecimento necessário e criar um sistema que fosse eficiente em diversas situações.

2.3 Trajetórias de estacionamento

A partir dos estudos de Gupte et al. [26], é possível perceber que existem inúmeros tipos de veículos, os quais se diferenciam por seu comprimento, largura, tipo de trajetória que descrevem e entre outras características. Nesta abordagem, os autores têm de lidar com tais atributos de maneira a mapear os objetos computacionalmente, que representa um problema complexo mesmo partindo de padrões predefinidos.

Isto posto, é possível supor que o movimento descrito por cada veículo é diferente, como por exemplo, o ângulo máximo que o mesmo irá executar durante uma curva, impactando diretamente no processo de estacionamento, fazendo com o que o movimento descrito pelo veículo se altere. No projeto proposto por Demirli e Khoshnejad [27], ao buscar por uma trajetória viável de estacionamento existem inúmeras trajetórias e a alteração da posição inicial, por exemplo, altera a forma como o percurso será efetuado, como é ilustrado na Figura 2.6, onde W indica a posição inicial mínima e máxima onde o veículo pode começar o processo de estacionamento e concluí-lo com sucesso.

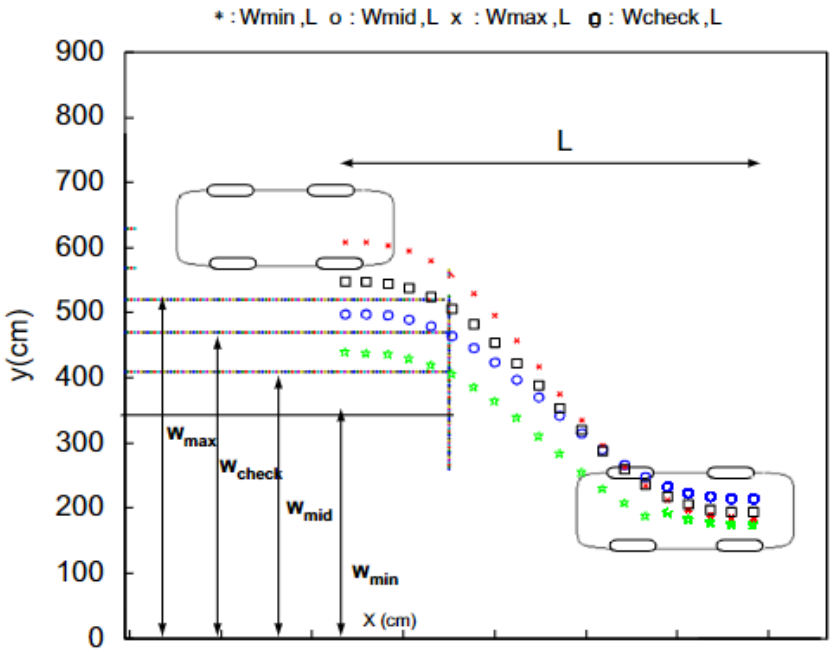

Figura 2.6: Impacto das posições para início do estacionamento [27]

No modelo descrito acima, os autores utilizam inteligência artificial com o intuito de obter o melhor percurso para o veículo estacionar com segurança, porém utilizam como base um polinômio de quinto grau para geração dos possíveis percursos, que depois serão selecionados de acordo com o ambiente (Figura 2.7).

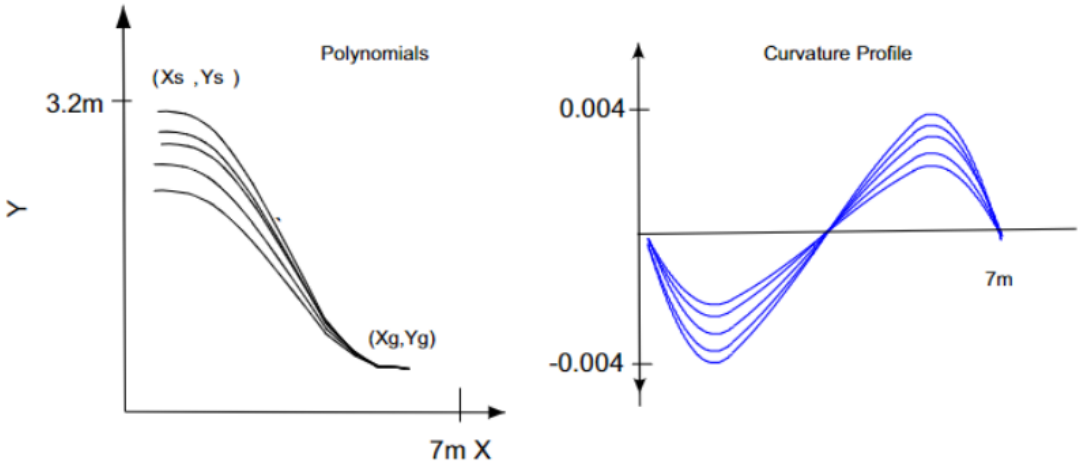

Figura 2.7: Perfil das trajetórias de acordo com o polinômio [27]

De maneira análoga, Pinheiros [28], para definir a trajetória de estacionamento em seu trabalho, coloca em prática uma abordagem que consiste na combinação dos arcos gerados por duas circunferências, posicionadas de acordo com o arranjo do veículo em movimento e aos demais automóveis estacionados que delimitam a vaga desejada. Neste caso, como pré-requisito, o autor parte da ideia de que a vaga identificada possuí um comprimento mínimo, que é dado de acordo com o comprimento, ângulo máximo de esterçamento das rodas e a posição do centro de massa do veículo (Figura 2.8).

Outra abordagem para obter uma trajetória de estacionamento a partir dos movimentos do veículo mapeados por circunferências é vista no trabalho de Vorobieva et al. [29], onde expõe que o estacionamento pode ser realizado em diversos cenários, sendo este em uma única manobra com entrada a frente ou com o veículo em marcha ré, e também com diversas manobras sendo realizadas durante o trajeto, situação está encontrada quando o comprimento for reduzido porém ainda consiga conter o automóvel.

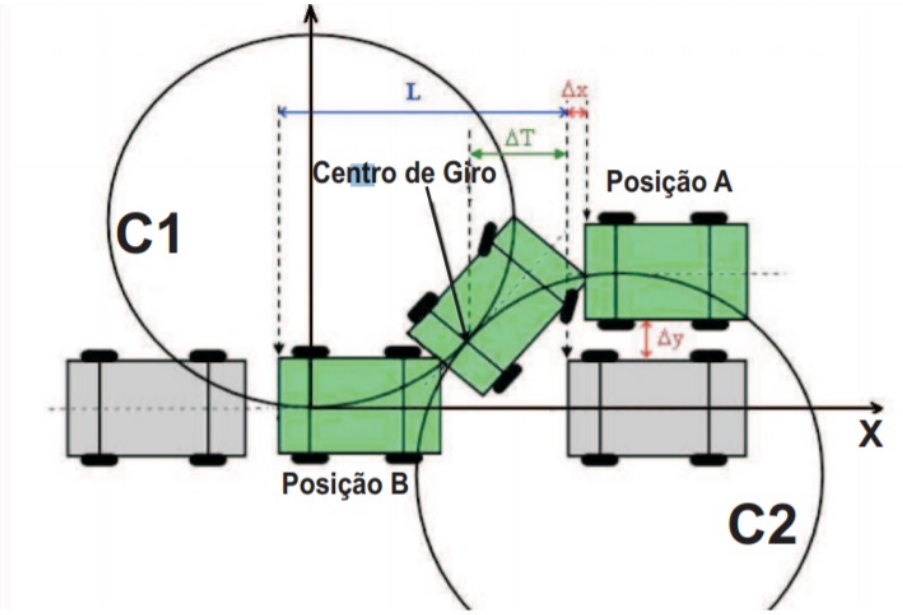

Figura 2.8: Trajetória obtida a partir das circunferências descritas [28]

2.4 Estacionamento com auxílio de inteligência artificial

Abordagens voltadas para o estacionamento autônomo são propostas por Osório, Heinen e Fortes [30], em que um sistema capaz de controlar um veículo autônomo, com forma retangular similar a um carro, equipado com seis sensores infravermelhos de baixo custo, tem como objetivo fazer com que o veículo seja capaz de localizar uma vaga e realizar o estacionamento sem a intervenção de um condutor humano, sendo que o controle pode ser realizado por meio de uma Rede Neural Artificial com aprendizado supervisionado ou com um sistema especialista composto por um conjunto de regras heurísticas.

Li e Chang [31] com o intuito de solucionar o problema de estacionamento, propõem uma abordagem capaz de auxiliar em diversas situações de baliza como o estacionamento em vagas paralelas e a entrada na garagem (vagas a 90º em relação ao sentido de movimentação do veículo). Os autores atingem seu objetivo por meio da modelagem de um carro como um robô móvel, ou car-like mobile robot (CLMR), com processamento de

imagem em tempo real e criando dois controladores fuzzy, um para cada tipo de estacionamento desejado, tendo como entrada uma rota factível que permite levar o veículo de uma posição inicial até a posição final, correspondente ao veículo posicionado dentro da vaga que foi identificada.

Veículos capazes de estacionar de maneira autônoma podem ter um impacto significativo na redução de acidentes e colisões visto que, segundo Matsubara [32]:

> Pequenas batidas podem acontecer na hora de parar o carro. No entanto, um levantamento realizado pela Allianz aponta que esse tipo de ocorrência é mais comum do que imaginamos: cerca de 40% dos acidentes veiculares que resultam em perdas ou danos materiais ocorrem durante as manobras de estacionamento.

Ainda neste âmbito, a utilização de algoritmos e abordagens fuzzy tem procurado otimizar o desempenho de veículos em diferentes aspectos, na melhoria de performance em termos de economia de combustível, na performance do motor e autonomia [33].

No segmento relacionado a desenvolver automóveis autônomos, Sordi Filho et al [34] propuseram um controle de navegação de um carro autônomo utilizando um sistema nebuloso (Fuzzy), de maneira que o veículo seria capaz de desviar automaticamente de obstáculos em ambientes desconhecidos, este ambiente seria o local por onde o veículo está se deslocando, e conforme sua navegação o sistema mapeia o cenário de maneira a permitir o planejamento das trajetórias através do algoritmo A*, tendo como parâmetro a posição na qual o veículo se encontra, a posição alvo, sendo o algoritmo responsável por calcular os possíveis movimentos necessários a atingir a posição final.

Bonissone e Aggour [35] afirmam que muitos acidentes em que um veículo colide na traseira de outro tem relação com o tempo em que o motorista demora a identificar uma potencial situação de risco aliado ao tempo necessário para aplicar uma pressão sobre o pedal de freio, de maneira que é gerado um atraso entre o tempo de identificação e a execução de uma medida corretiva. Deste modo, Rehman, Mushtaq e Qamar [36] propõem um sistema utilizando-se de lógica fuzzy capaz de prevenir a colisão com outro veículo,

baseado na aplicação autônoma dos freios de acordo com a distância entre os veículos envolvidos, a velocidade atual e o coeficiente de atrito estático, de modo que a aplicação do sistema fuzzy em questão, demonstra a eficiência da utilização de fuzzy no quêsito de atuação autônoma para prevenção de acidentes.

O trabalho de Selekwa, Dunlap e Collins [37] refere-se a veículos autônomos terrestres em ambientes com muitos obstáculos, onde a configuração dos obstáculos não é conhecida. Nesse cenário o trabalho descreve a dificuldade em tomar decisões, logo a importância do uso de lógica fuzzy no controle de navegação. No sistema são consideradas duas ações de controle o controle de direção e de velocidade. No primeiro busca-se o objetivo, evita obstáculo e rastreia-se as bordas laterais. O controle de velocidade evita obstáculos e capotagem. Todas as decisões foram baseadas nas informações coletadas pelos sensores.

Zhao e Collins [38] apresentam um projeto baseado em controle fuzzy considerando que o algoritmo de controle com essa estrutura possa ser transferido para protótipos diferentes. Os autores destacam que controladores fuzzy possuem flexibilidade para aplicar uma regra, se aproximando ao raciocínio humano por meio de regras de decisão, além de ser capaz de combinar comandos.

A aprendizagem de veículos autônomos pode ser feita com um algoritmo capaz de extrair o conhecimento de manobras realizadas por motoristas humanos por meio da utilização de Fuzzy Inference Systems (FIS), ao invés de técnicas de inteligência artificial em algoritmos programados por especialistas. No contexto desta proposta, os dados estão armazenados em um banco de dados contento exemplos de treinamento do sistema de controle que se deseja modelar. A utilização de conceitos fuzzy foi escolhido devido a facilidade em representar o conhecimento de maneira simples, por meio da linguagem natural humana [39].

2.5 Abordagens alternativas

Pesquisas na área de veículos autônomos demandam um momento para validação e testes, porém, a utilização de veículos reais pode ocasionar em alguns problemas, como por

exemplo o risco de danificar o veículo e demais elementos do ambiente enquanto o sistema não estiver totalmente configurado. Antes de iniciar os testes, o veículo deverá estar equipado com todos os equipamentos e dispositivos necessários, sendo que, pode acontecer de nem todos serem necessários a uma determinada tarefa, e a utilização de veículos reais necessita de cuidados como manutenção e reabastecimento de combustível ou recarga de baterias em caso de veículos elétricos.

Desse modo, foi proposto a utilização do sistema Simulador de Estacionamentos de Veículos Autônomos em um ambiente tridimensional (SEVA3D), o qual permite criar um ambiente virtual com um carro e ter o controle do mesmo por meio da leitura de dados de sensores, gerar comandos de aceleração e movimento na direção, permitindo simular a ação de estacionar em uma vaga paralela. As ações executadas pelo veículo são provenientes de um controlador implementado utilizando um autômato finito baseado em regras [24].

Outra abordagem utilizada, parte da ideia de que o veículo se move de acordo com modelo cinemático de Ackerman, o trabalho proposto por Heinz [40], propõe um protótipo de um carro em escala reduzida com tração traseira e posteriormente, toma como base a posição inicial do veículo, para então, determinar os parâmetros geométricos necessários para realizar cada manobra. Para obtenção de tais parâmetros, toma como base informações como o raio de curvatura gerado pelo protótipo, distância entre eixos, ângulo de esterçamento máximo, entre outros.

2.6 Considerações finais

Neste capítulo foi apresentado um contexto histórico da evolução dos automóveis e suas respectivas tecnologias, com objetivo de entender a motivação deste trabalho no cenário global em relação a quantidade de veículos em circulação e seu uso. Foi apresentado também uma visão geral do funcionamento da tecnologia de veículos autônomos, as técnicas utilizadas para o estacionamento de automóveis e algumas de suas possíveis trajetórias, fazendo-se necessário para compreender a solução que será proposta neste trabalho.

A partir da análise dos trabalhos relacionados, foi possível observar que existem diversas formas diferentes de se obter uma trajetória viável para o estacionamento de um automóvel, principalmente de acordo com os parâmetros tidos como base no ambiente em que o veículo está inserido. Como base para o desenvolvimento a partir do que já foi feito no cenário científico atual, foi justificável a escolha do modelo de Ackerman para traçar o movimento do veículo no ambiente que será simulado neste trabalho.

Capítulo 3

Metodologia

Para o desenvolvimento do modelo proposto neste projeto é necessário entender como se estabelece o cenário no qual será aplicado, ou seja, como se pretende abordar o problema e sua base de funcionamento. A assistência ao estacionamento terá início a procura por uma vaga disponível e verificar se existe um espaço suficiente para posicionar o veículo. Uma vez identificada a vaga, o sistema irá traçar uma rota partindo da posição inicial do veículo até estar completamente estacionado, ficando a cargo da aplicação exibir na interface gráfica a posição do veículo e a trajetória a ser seguida, tendo como base leituras provenientes de sensores. Contudo, nenhum movimento será efetuado pelo sistema, ficando por total responsabilidade do condutor realizar as manobras. O fluxograma na Figura 3.1 representa o princípio de funcionamento do sistema.

3.1 Arquitetura

O funcionamento básico do sistema consiste no simulador com um automóvel e seus respectivos sensores, um servidor Python responsável por ler e repassar em tempo real as informações de posição do veículo, bem como informar se encontrou uma vaga válida para estacionamento, fazer a leitura dos sensores e, por fim, repassar todos estes dados para a interface gráfica, a qual, por sua vez é responsável por receber os parâmetros, exibir os dados do veículo que foram recebidos e, em caso de encontrar uma vaga, também é

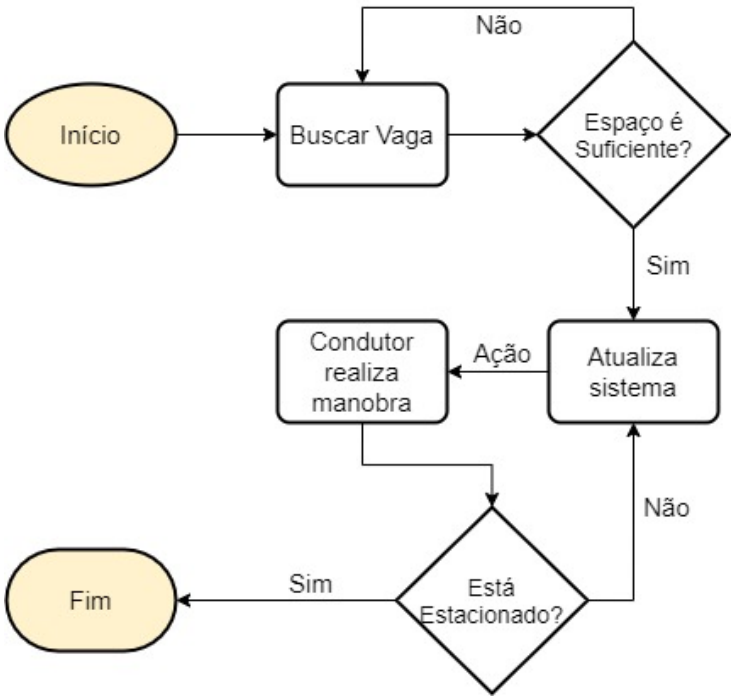

Figura 3.1: Fluxograma de funcionamento do sistema

responsável pela geração da trajetória (Figura 3.2).

3.2 Ambiente simulado

A principal característica deste ambiente será trazer para o projeto uma execução que permita recriar de maneira confiável um ambiente que será encontrado no dia a dia dos motoristas ao se depararem com o movimento de estacionamento. Para que tal representação semelhante ao normal, deverá ser utilizada a representação de um veículo, que descreve os movimentos de um automóvel de passeio real, baseando-se no modelo cinemático de Ackerman, que modela matematicamente a trajetória de movimento de um veículo com eixo dianteiro e rodas que são capazes de se movimentar angularmente, e um eixo traseiro rígido sem movimento angular por parte das rodas (Figura 3.3).

Como também é de responsabilidade do simulador permitir a inserção de sensores no veículo, sendo estes:

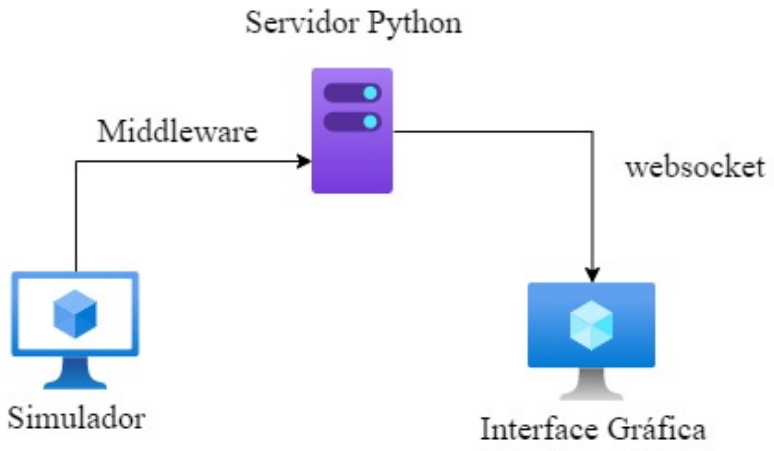

Figura 3.2: Arquitetura do sistema

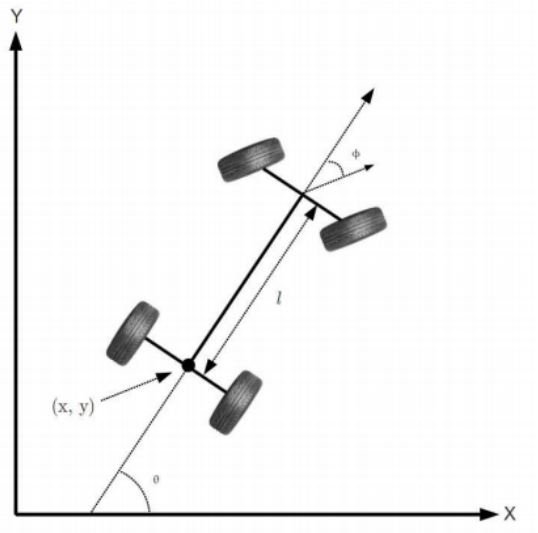

Figura 3.3: Modelo cinemático de Ackerman

- 3 sensores ultrassônicos na parte frontal;
- 3 sensores ultrassônicos na parte traseira;
- 1 sensor ultrassônico no meio da roda dianteira direita;
- 1 sensor ultrassônico no meio da roda traseira direita;
- 1 sensor de GPS para permitir a identificação da posição atual do veículo;
- 1 acelerômetro de modo a obter a orientação em relação ao movimento angular do veículo, posicionado no centro do eixo traseiro;

Após a configuração do simulador, o servidor Pyhton irá conectar-se a ele e começar a fazer de modo constante uma leitura de todos os sensores até que o veículo se encontre em um cenário de estacionamento concluído.

3.3 Servidor Python

Por sua vez, o servidor Pyhton será a parte central de comunicação do sistema, pois será o responsável por estabelecer uma comunicação com o simulador, iniciar uma conexão com a interface gráfica, e em tempo real efetuar todas as leitura necessárias ao processo de estacionamento.

Em um primeiro momento, após as conexões terem sido iniciadas, o servidor Python irá se fundamentar no sensor dianteiro direito, de modo a verificar seu status (Identificando um obstáculo ou não), com o intuito de validar se o automóvel está passando próximo a um veículo e possivelmente encontrará uma vaga. Ao identificar que o veículo começou a percorrer um espaço vazio, o mesmo deverá salvar sua posição, como ilustra a Figura 3.4

Quando sair de um estado, em que sua leitura anterior é um espaço vazio e a atual é um obstáculo, demonstrado na Figura 3.5, o servidor deverá fazer um cálculo entre a posição que está salva e sua posição atual, obtendo desta forma, o tamanho da vaga encontrada.

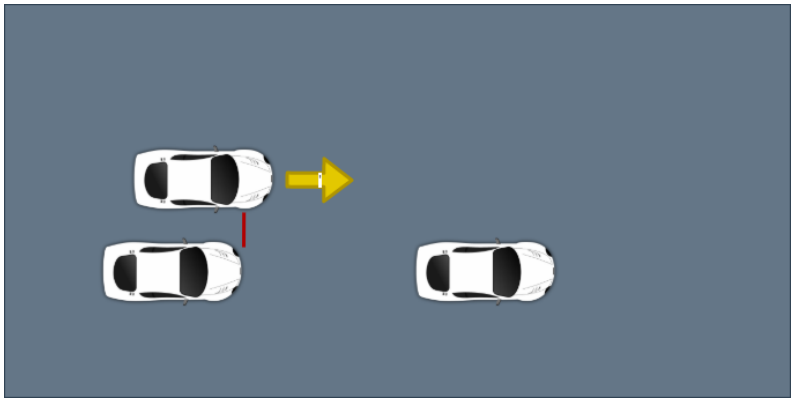

Figura 3.4: Momento de identificação do espaço vazio após obstáculo

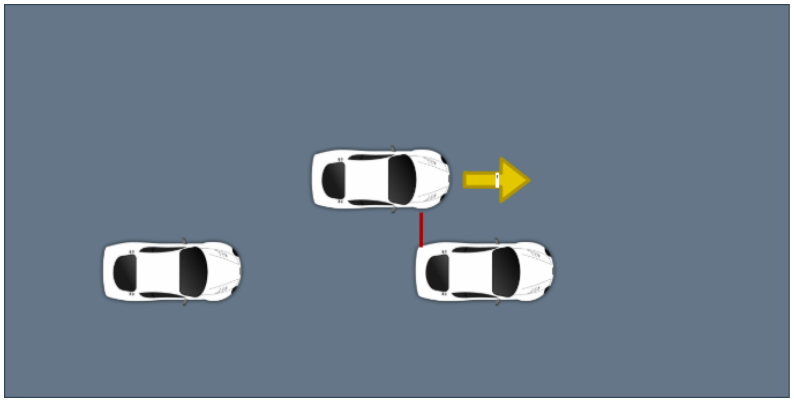

Figura 3.5: Momento de identificação do obstáculo após espaço vazio

Caso a vaga seja do tamanho mínimo necessário para definição de um trajetória, emite um evento a interface gráfica para instruir o condutor a parar e apresentar a trajetória disponível.

Por outro lado, caso a vaga não tenha o comprimento mínimo necessário, o servidor irá reiniciar os seus parâmetros ao estado inicial, continuando a procura por uma possível vaga para estacionar o veículo. É importante ressaltar que a todo momento o servidor deverá estar se comunicando com a interface gráfica, mesmo que nada esteja sendo exibido, garantindo sua conexão.

De maneira geral, o servidor será responsável pela leitura e transmissão dos dados do veículo de maneira correta, e em escalas corretas, bem como, identificação de mudança de estado, partindo de procurando uma vaga para pronto para estacionar, emitir para a interface gráfica caso seja necessário algum movimento diferente da trajetória por parte do condutor.

3.4 Trajetória de estacionamento

A partir da observação e identificação de similaridades com os estudos definidos na pesquisa teórica deste trabalho, foi possível identificar que a trajetória pode ser calculada por meio da representação de funções e polinômios que, quando combinados, geram uma trajetória aceitável para estacionamento do veículo, assumindo como parâmetro a posição inicial, o comprimento do veículo, ângulo máximo de esterçamento das rodas e entre outros demais pontos.

Para cálculo de trajetória deste trabalho foi escolhido a definição de um percurso a partir da aproximação de função já existente. Tal escolha foi feita observando que funções geométricas podem ser possíveis para uma trajetória de estacionamento, e trabalhos que utilizam de inteligência artificial para escolha da melhor trajetória a cada momento do cenário. Em alguns casos, também se baseiam na geração a partir de uma função, contudo, seu diferencial é a capacidade de recalcular em tempo real todo o processo de estacionamento.

Posto que a abordagem deste trabalho tem o intuito de auxiliar de maneira fácil e prática o condutor a manobrar seu veículo, a perspectiva de aproximação para uma função que não se altera em tempo real atende a estes requisitos, de modo que o condutor é levado a prestar atenção nos movimentos que está fazendo, com a finalidade de manter seu veículo na trajetória correta que está sendo exibida na interface gráfica, sem alterações súbitas.

Levando em consideração a criação de uma trajetória de estacionamento a partir do do gráfico descrito por uma função, foi escolhida a função sigmoide, cuja saída é descrita pela Figura 3.6.

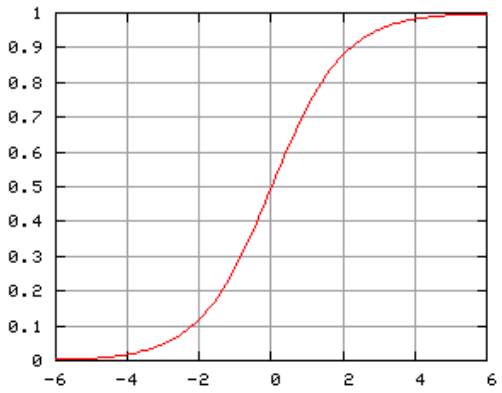

Figura 3.6: Representação gráfica da função sigmoide

Seguindo a mesma linha de raciocínio, outra função que poderia ser utilizada para descrever tal movimento são as funções hiperbólicas (Figura 3.7).

3.5 Interface gráfica

A interface gráfica deverá ser minimalista, exibindo em tela somente o necessário para que o motorista seja capaz de interpretar sua posição, o trajeto a ser seguido e a posição dos outros carros ou objetos que delimitam sua vaga.

Sua configuração inicial deve mostrar ao usuário uma indicação de que está procurando por uma vaga disponível, sendo atualizada constantemente, pois no momento em que o servidor enviar um mensagem para iniciar o processo de estacionamento, este deve exibir

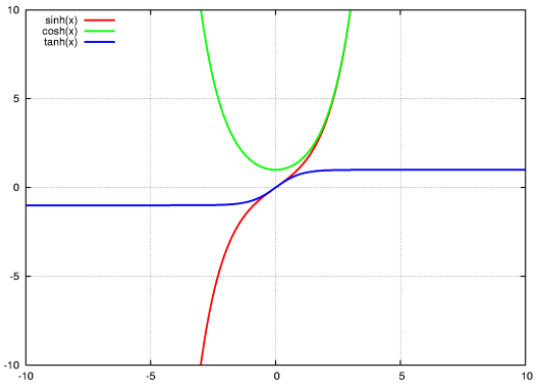

Figura 3.7: Representação gráfica das funções hiperbólicas

em tela o automóvel do condutor, a vaga identificada e os veículos ou objetos que a delimitam.

Outra característica necessária na interface com o utilizador é de que precisa ser atualizada em tempo real com as coordenadas corretas do veículo, pois esta será o único meio de identificação visual do espaço ao redor e sua respectiva posição para o condutor, além de ser necessário para assegurar ao condutor que as manobras executadas por ele estão corretas e conduzindo o veículo para um cenário de estacionamento correto.

Por fim, é de total obrigação da interface gráfica, no momento em que receber do servidor uma mensagem para inciar o processo de estacionamento, utilizar-se das técnicas definidas para geração de trajetória e apresentar em tela o percurso ao condutor, que deve segui-lo para estacionar seu veículo.

3.6 Testes e validações

Visto que o modelo proposto neste trabalho não realiza movimentos autônomos, não é possível avaliar sua eficiência pelo resultado de sucesso ou fracasso após o processo de baliza, pois mesmo que o sistema envie comandos corretos, o condutor pode não os executar ou efetuar uma manobra diferente da sugerida pelo sistema.

Desta forma, para avaliação do modelo serão realizadas simulações com diferentes características de ambientes, ou seja, variando a posição dos veículos estacionados, alinhamento na vaga (simulando um carro parado desalinhado) e tamanho das vagas disponíveis. Em virtude de o condutor realizar as manobras conforme seu julgamento e habilidades, durante as simulações também serão reproduzidas situações nas quais o condutor realiza uma parte das manobras correta, porém durante o processo não segue o direcionamento do modelo ocasionando em uma ação errada, avaliando a capacidade de retomada do trajeto de forma a obter o sucesso, equivalente ao veículo estacionado

3.7 Considerações finais

Neste capítulo foi descrito de maneira objetiva as principais funções de cada parte do sistema, com intenção de exemplificar da melhor maneira possível qual deve ser o comportamento do sistema em condições boas, com todos os parâmetros sendo satisfatórios, pois é o caminho mais fácil de entendimento. A partir da definição dos fluxos mais básicos do sistema, foi possível que seu nível de complexidade é alto, visto que demanda do correto funcionamento das integrações entre diversos sistemas.

Capítulo 4

Desenvolvimento

Para desenvolvimento do modelo proposto neste trabalho, foram utilizadas diversas tecnologias e linguagens de programação, de maneira a possibilitar a simulação de um ambiente próximo ao real, no qual está inserido o veículo e os obstáculos, um servidor capaz de processar todos os dados e uma interface gráfica responsável por exibir ao condutor o trajeto a ser seguido para realizar o processo de estacionamento com sucesso. De maneira facilitar o entendimento do trabalho, será listado a seguir as tecnologias utilizadas e algumas de suas características que foram imprescindíveis para sua respectiva escolha.

4.1 Tecnologias utilizadas

O desenvolvimento do protótipo implica a utilização de um simulador, onde é possível testar os algoritmos e abordagens descritos neste documento, linguagens de programação, para implementar e executar a instância do sistema, e algumas frameworks, de forma a otimizar o esforço e tempo de desenvolvimento.

4.1.1 Simulador

Com objetivo de propor um modelo que seja viável para utilização no mundo real, foi importante a escolha de uma tecnologia capaz de simular com precisão os movimentos de

um veículo, bem como sua atualização em tempo real, de modo a que não houvesse atrasos entre o comando enviado e sua efetiva realização. Outro fator crucial para a escolha, é a possibilidade de comunicação com servidores e *scripts* externos ao ambiente simulado, de modo que fosse possível a interação do simulador com um servidor externo.

Dado estes requisitos, foi escolhido o CoppeliaSim, anteriormente conhecido como V-rep. De acordo com Rohmer, Singh e Freese [36], o simulador traz diversas opções para integração com programas externos a partir de diversas opções de Application Programming Interface (API) que podem ser escritas de acordo com a linguagem de programação desejável dentre as elegíveis. Adicionalmente, uma cena, representação do cenário criado no simulador, pode ser composta por diversos objetos de criação própria ou modelos já prontos, dentre eles, um modelo que representa um veículo baseado no modelo cinemático de Ackerman, essencial para o desenvolvimento deste trabalho.

4.1.2 Linguagens de programação

Durante a execução de uma simulação, a quantidade de dados lidos no simulador e a necessidade de transmissão de maneira prática e eficiente, conduziu as escolhas das linguagens de programação utilizadas no trabalho. Para a interação com o simulador foi escolhida a linguagem Python. Para comunicação e tratamento dos dados e responsável por transmitir para a interface gráfica foi escolhida a linguagem TypeScript. Por fim, para criação da interface gráfica de modo mais ágil e de fácil entendimento, o JavaScript.

Python

Visto que, o simulador gera uma grande quantidade de dados, enviando em tempo real a posição dos veículos, leitura de sensores e entre outras diversas informações, se fez necessário a escolha de uma linguagem de programação simples porém robusta que fosse capaz de lidar com tal cenário, e por sua vez, que fosse compatível com as opções disponíveis para comunicação com a API do simulador escolhido.

Python vem ganhando grande visibilidade no cenário tecnológico, Oliphant [41] diz

que foi surpreendido em sua procura por uma linguagem que permitisse a criação de algoritmos capazes de atuar com grupos de dados grandes e que sua experiência aumentou ainda mais com a facilidade de expressar ideias complexas na sintaxe do Python. Trazendo funcionalidades como:

- Possibilidade de comunicação com uma grande variedade de outros softwares;
- Grande números de bibliotecas e módulos disponíveis, auxiliando na construção de programas complexos;
- Extensa comunidade utilizando, o que significa respostas a diversos problemas que podem ser encontrados de maneira fácil e rápida.

Assim sendo, essa linguagem mostrou-se uma alternativa promissora de acordo com a necessidade para o desenvolvimento do modelo deste trabalho.

JavaScript

Com o intuito de criar uma aplicação gráfica eficiente e também de maneira simples, o principal ponto para escolha da linguagem é a sua versatilidade, a qual seria utilizada para comunicação com um servidor e também para representação dos elementos gráficos.

Neste cenário, o JavaScript é comumente aliado a criação de arquivos Hypertext Markup Language (HTML), ou seja a interface do cliente ou a parte gráfica com o qual o usuário tem a interação. Contudo, dado suas diversas funcionalidades, foram criados interpretadores de código que possibilitam o uso da linguagem também na camada do servidor, comprovando sua versatilidade e eficiência [42].

Como o objetivo deste trabalho é apresentar ao condutor de maneira simples e eficiente uma interface gráfica com o que é necessário para estacionar seu automóvel, o JavaScript foi a melhor escolha justamente por ser capaz de interagir com ferramentas que geram a parte gráfica, mas também por receber os dados que estão acontecendo em tempo real.

TypeScript

Com a necessidade de receber uma quantidade de dados gerados pelo simulador e transmiti-los para um outro ambiente, fez-se a necessidade de uma linguagem potente mas que ao mesmo tempo seja rápida para lidar com uma grande quantidade de requisições e identificar o que está sendo recebido e transmitido.

Desta maneira, o TypeScript oferece diferenciais como uma notação conveniente para a escrita do código, visto que existe a possibilidade de definir um tipo para o dado que será tratado, orientação ao objeto e todos as funcionalidades oferecidas pelo JavaScript, visto que qualquer código TypeScript é compilado para JavaScript [43].

Dada a necessidade de receber os dados emitidos pelo simulador e repassar para uma interface gráfica, a escolha do TypeScript se justifica pela sua capacidade de definir um tipo para os dados, garantindo que os dados transmitidos sempre irão seguir um contrato fixo, e em caso contrário irá parar de funcionar, o que garante o funcionamento do fluxo de execução do modelo com consistência e robustez.

4.1.3 Frameworks

A escolha por linguagens de programação simples, com alta capacidade na construção de aplicativos é o requisito básico para o desenvolvimento de software, complementarmente, existem diversos frameworks, abstrações de códigos que juntos provem uma funcionalidade, que se fazem necessário para permitir e facilitar a construção do modelo de estacionamento deste trabalho, abaixo foram listadas as opções escolhidas.

Node.js

A necessidade de servidores cada vez mais rápidos, robustos e de fácil configurabilidade são alguns dos fatores mais importantes em sua escolha. Neste cenário uma vez definida que a linguagem utilizada será o JavaScript e o TypeScript. surge o problema de escolher um servidor com tais características e que suporta essas linguagens.

O Node.js é um ambiente de desenvolvimento JavaScript no lado do servidor, focado

em performance e no baixo uso de memória [44]. Uma de suas principais diferenças em relação às demais opções disponível é a utilização de eventos assíncronos para execução de tarefas, o que resulta em uma maior facilidade para construção e entendimento de programas, fácil manutenção e maior eficiência em momento de execução [44].

Dada sua aplicabilidade e também a possibilidade de utilização do TypeScript, visto que o código é compilado para JavaScript, a escolha da construção de um servidor em Node.js facilita o desenvolvimento e garante eficiência para gerenciar uma grande quantidade de requisições, tanto recebidas quanto transmitidas.

p5.js

A construção de interfaces gráficas dá-se de acordo com o contexto que será utilizada e onde será implementada. Por exemplo, a construção de um website pressupõe, em sua grande maioria, a utilização da linguagem HTML, Cascading Style Sheets (CSS) e JavaScript. Por outro lado, para criação de jogos temos como exemplo a utilização da Unity ou Unreal Engine. Considerando a necessidade de criar formas, representar funções e, como objetivo secundário, minimizar a quantidade de tecnologias diferentes dentro de um projeto, o framework p5.js surge como uma oportunidade a ser considerada.

O p5.js busca facilitar a construção de softwares que criam imagens, animações e suas respectivas interações, de modo que, ao escrever uma linha de código, seja possível criar um círculo e exibi-lo em tela, com objetivo de explorar diversas ideias em um curto período de tempo, possibilitando a criação de cenários cada vez mais complexos [45]. O framework funciona a partir da criação de um arquivo HTML base, com a definição das bibliotecas utilizadas, e a escrita e criação de código utilizando JavaScript.

A escolha do framework para a construção da interface gráfica surge, essencialmente, por utilizar JavaScript, o que contribui para uma maior flexibilidade, uma vez que é a mesma utilizada no lado do servidor, diminuindo a curva de aprendizagem para entendimento de uma nova tecnologia, mas também com sua característica de desenvolvimento fácil e ágil.

Socket.IO

Com a grande quantidade de tecnologias disponíveis para construção de aplicações tanto para servidores quanto para clientes, a demanda por formas de comunicação entre tais aplicações também é grande. Neste cenário, novamente, precisa ser analisado de acordo com as necessidades e demandas da aplicação, para este trabalho, é necessário uma forma de comunicação em tempo real, onde ao veículo mover no simulador, a interface seja atualizada ao mesmo tempo, ou o mais próximo possível.

O Socket.IO traz o conceito de eventos, onde aplicativos clientes e servidores podem escutar eventos e emitir mensagens, as quais são transmitidas por um canal. Desta forma, sempre que uma mensagem é emitida no canal, todos os dispositivos que estão conectados e escutando o evento, recebem a mensagem, a qual pode conter qualquer tipo de dado desejado. O framework, é responsável por criar estes canais e transmitir as mensagens, podendo ser executado de modo autônomo ou dentro de um servidor existente, como no caso do Node.js [46].

Após algumas experiências práticas com o framework, com o intuito de validar se o mesmo consegue lidar com uma grande quantidade de mensagens sendo transmitidas, seja no lado do cliente em receber e exibir as mensagens em tempo real, como também no lado do servidor em transmitir, o Socket.IO comprovou ser eficiente para o objetivo deste trabalho, que irá transmitir mensagens constantes contendo a posição do veículo, por exemplo.

4.2 Criação e configuração do ambiente simulado

A criação do ambiente simulado é feito de maneira simples, visto que utiliza de diversos objetos e modelos já prontos, sendo de fácil utilização a partir do menu drag-and-drop lateral. Por exemplo, o primeiro item adicionado a cena, criada pelo CoppeliaSim, foi uma superfície (ResizableFloor), para permitir a inserção dos demais objetos.

O próximo item, e mais importante nesta simulação foi a inclusão do veículo, o simulador já tem disponível entre seu conjunto de dados, um modelo que representa o veículo

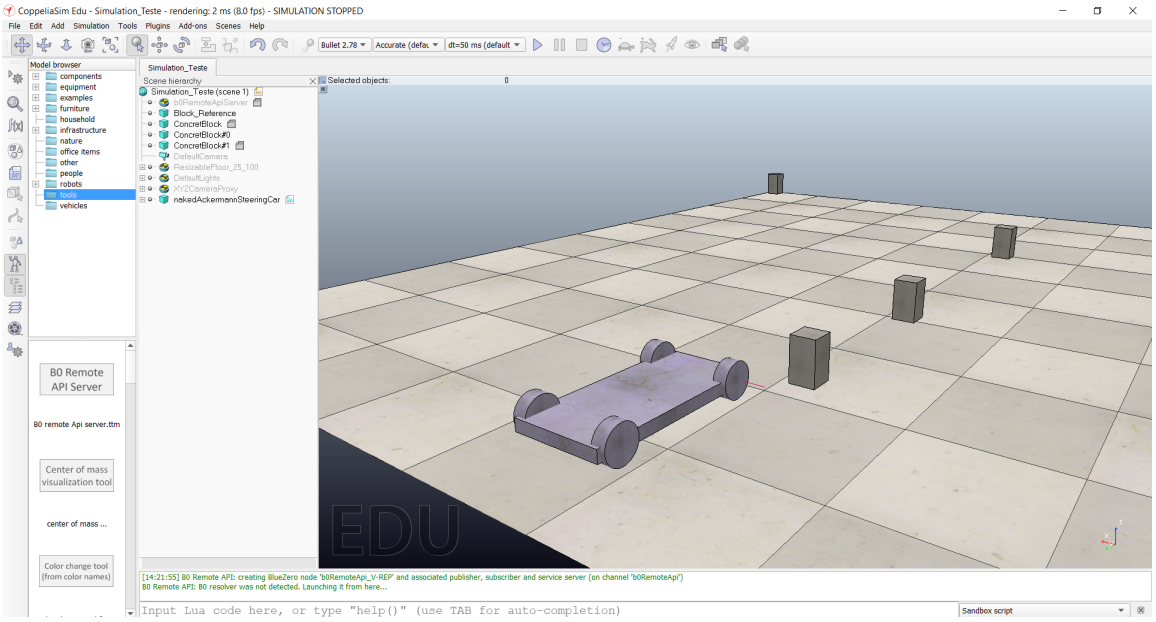

Figura 4.1: Representação do ambiente do simulador

baseado no modelo cinemático de Ackerman (nakedAckermannSteeringCar). Desta maneira, seu movimento é descrito corretamente conforme o necessário no ambiente, conforme ilustrado na Figura 4.1. Restando somente, adicionar os sensores necessário para realização do trabalho.

Por sua vez, ao tentar utilizar um sensor GPS disponível na plataforma, houve uma grande variação nos dados que estavam sendo retornados, como se não tivesse um referencial bem definido, ocasionando uma leitura incorreta da posição do veículo. Visto que em um cenário real a utilização de um GPS se dá de maneira correta, para o prosseguimento do trabalho foi utilizado uma função disponibilizada pelo simulador, que retorna a posição atual do veículo, de modo que foi possível encapsular sua leitura como se fosse um sensor que estava a ser lido, não ocasionando em nenhum impacto para o trabalho proposto.

O mesmo problema foi encontrado na utilização do acelerômetro. Inicialmente foi suposto que o mesmo não fornecia leituras corretas devido a não ter uma correta referência. Por este motivo, e também para facilitar o entendimento das coordenadas entre o simulador e a interface gráfica, foi inserido um bloco, na posição (0,0), de modo a ser a referência. Contudo, a solução encontrada foi utilizar uma função do simulador que retorna o ângulo

formado pelas rodas frontais do veículo em radianos, de modo a possibilitar a reprodução do seu movimente na interface gráfica.

De maneira geral, sua criação foi simples, embora tenham existidos inconsistências com dados retornados pelos sensores. Outro ponto importante, a simulação da cena é feita em três dimensões, porém, como a interface gráfica terá somente duas, nos cálculos e posições utilizadas, foi considerado apenas posições no formato (x, y).

4.3 Implementação da arquitetura

Antes de começar o desenvolvimento definitivo, foi criada uma versão básica de maneira a validar sua viabilidade, composta pelo simulador, um servidor Python que comunica com o simulador e a interface com o utilizador. A comunicação é feita por meio da funcionalidade B0-based remote API, que se trata de um conjunto de funções disponibilizado pelo simulador CoppeliaSim, responsável por estabelecer a relação client/server e publisher/subscriber por meio de um middleware que transporta as mensagens.

Outra função do servidor Python foi a criação de um servidor WEB, que permite disponibilizar a interface gráfica, e realizar a comunicação por meio de um WebSocket. Contudo, quando o teste foi executado, ao movimentar o veículo no simulador e tentar exibir sua posição em tempo real na interface gráfica, foi observado um atraso muito grande entre os movimentos, ocorrendo a movimentação do veículo na interface gráfica após diversos segundos, não atendendo as necessidades deste projeto.

Ao analisar de maneira crítica e também auxiliado da documentação do framework p5.js, este expõem que quando a aplicação no servidor WEB é feita com uma determinada biblioteca Python, sua otimização não é feita de forma eficiente. Não foi utilizada a mesma biblioteca descrita na documentação, contudo, foi possível supor que este atraso na representação poderia estar sendo ocasionado pela incompatibilidade com o servidor Python.

Tendo em vista essa situação, foi necessário encontrar uma nova abordagem de arquitetura, e foi visto na documentação do p5.js que a maior eficiência era obtida através

da disponibilização do servidor WEB com o framework Node.js, o que resultou um novo design na arquitetura conforme ilustra a Figura 4.2 a seguir.

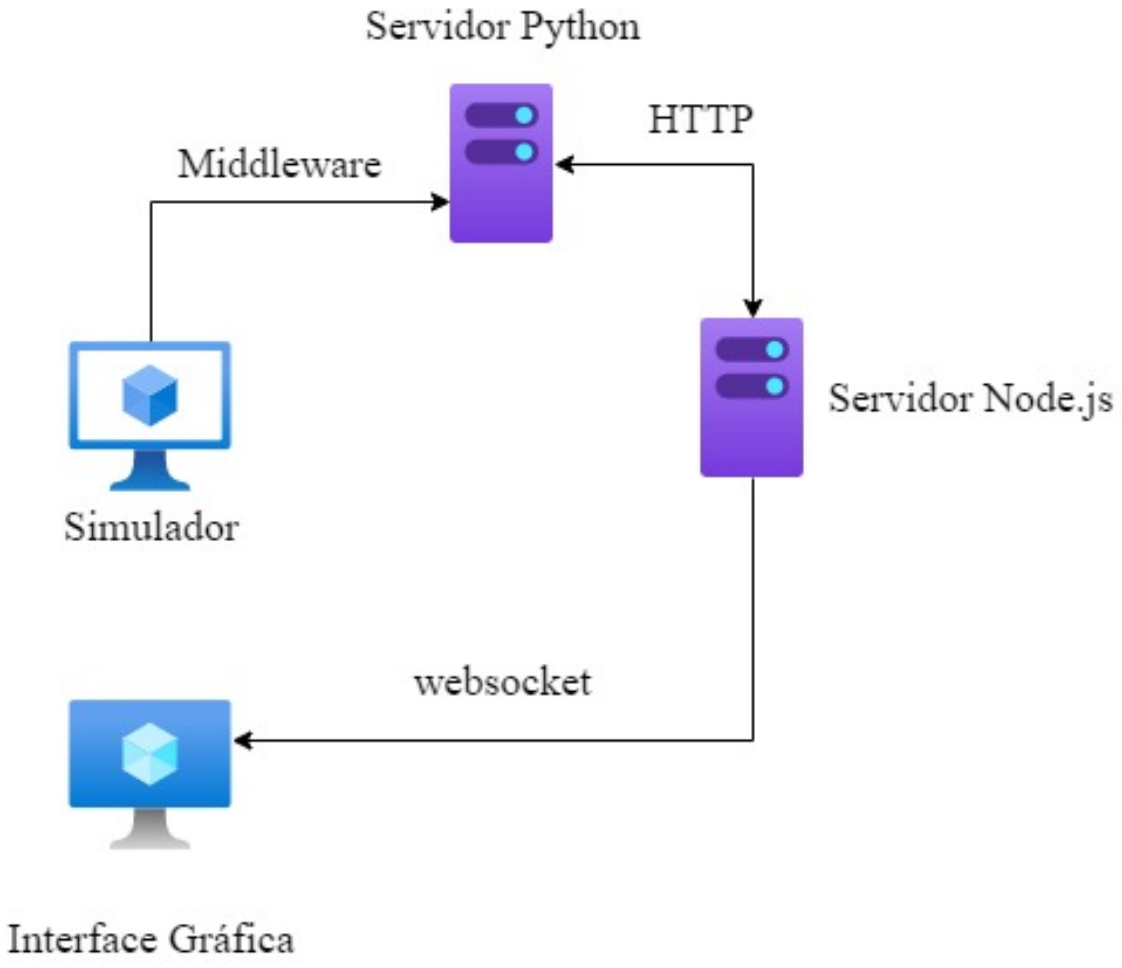

Figura 4.2: Arquitetura atualizada

Nesta nova arquitetura, o servidor Python continua responsável por comunicar com o simulador, todavia, este irá efetuar uma requisição Hypertext Transfer Protocol (HTTP) para o servidor Node.js, transmitindo os dados. Por sua vez, o servidor Node.js irá estabelecer um servidor WEB contendo a interface gráfica criada no p5.js, e retransmitir os dados recebidos em HTTP através de um WebSocket.

Após alguns testes preliminares, a nova arquitetura foi validada, mostrando-se robusta e capaz de atualizar em tempo real as posições do veículo no simulador e na interface gráfica, garantindo os requisitos necessários para a realização deste projeto.

4.3.1 Configuração do servidor Python

O servidor Python tem como principal função o controle sobre a cena criada, tendo como função os seguintes itens:

- Estabelecer conexão com o simulador por meio da API do CoppeliaSim;
- Efetuar o controle da cena criada, de modo a identificar quando uma vaga tem o tamanho mínimo necessário para estacionamento, sendo definido pelo tamanho do veículo acrescido de 30%;
- Calcular o tamanho da vaga, com uma margem de 5% sobre o valor real obtido, de modo a garantir uma variação de segurança;
- Transmitir os dados por meio de uma requisição HTTP para o servidor Node.js

Uma vez estabelecida a conexão, por meio da API fornecida pelo CoppeliaSim, o próximo passo é o envio das posições e dos dados necessário para o servidor Node.js, para isso, foram usadas algumas funções da API como por exemplo: simxGetObjectHandle, simxGetFloatSignal, simxGetObjectPosition, sendo estas responsáveis por obter a referência do objeto desejado, efetuar a leitura de um sinal do tipo float e obter a posição de um determinado objeto, respectivamente. Estes são somente alguns exemplos das funções usadas, tendo como base a documentção disponibilizada no site do CoppeliaSim através da página B0-based remote API, Python.

Um fator determinante para o correto funcionamento da simulação diz respeito às unidades utilizadas, tendo em vista que existe diferença entre o ambiente simulado e a interface gráfica. Para garantir a integridade das dimensões, foi inserido um bloco de referência na cena do simulador, marcando o ponto (0,0) da simulação. Contudo, as unidades utilizadas pelo simulador estavam em escala menor e com orientações diferentes, para garantir as mesmas unidades, antes de fazer a transmissão dos dados, as coordenadas são multiplicadas por 20 e calculado seu módulo, desta maneira, os eixos x e y terão como menor e maior valor, respectivamente, 0 e 500. Com a conversão realizada, qualquer par ordenado (x,y) estará contido na representação gráfica.

De modo a garantir que o movimento feito pelo veículo no simulador e a representação do mesmo, feita na interface gráfica, fossem coerentes, também houve necessidade de alguns ajustes nos valores da velocidade e do ângulo das rodas lidos pelo servidor. A partir de inúmeros testes práticos realizadas com a arquitetura proposta, foi observado que quando o veículo estava realizando o movimento de curva para entrar na vaga, sua posição exibida na interface gráfica ficava distante da real, desta forma foi necessário adicionar um valor multiplicador nos dados da velocidade e no ângulo, sendo estes, 0,9 e 1,5 respectivamente. Com esta última conversão de unidades, o movimento do veículo se mostrou aceitável para representação durante a trajetória.

Por fim, para definir se um espaço encontrado tem o tamanho mínimo necessário para ser considerado como vaga, é realizado um conjunto de verificações a partir do sensor posicionado no centro da roda frontal direita, de modo que, quando o sensor identifica algum obstáculo, sua leitura tem o valor 1 e em caso negativo retorna o valor 0. Foi criado um cenário mapeando as leituras do sensor em cada situação de reconhecimento de obstáculos, onde quando o servidor Pyhton reconhece um obstáculo ele salva a posição atual do veículo, e ao encontrar um obstáculo seguinte, efetua o cálculo da distância entre a posição salva e a posição atual.

A distância é calculada por meio da fórmula de distância euclidiana entre dois pontos, é definido 95% de seu total, devido a margem de segurança, e então verificado se esse valor é maior ou igual ao mínimo necessário, sendo este, para os dados dessa simulação, de 4,57m.

O valor mínimo definido, está relacionado com as dimensões do veículo utilizado nessa simulação e os parâmetros definidos como margem de segurança para a leitura dos dados. O veículo utilizado neste cenário, possuí 3,35m de comprimento e 1,8m de largura. O cálculo é dado pela Equação 4.1:

$$D_{min} = 3,35 * 1,3 * 1,05 \tag{4.1}$$

A partir da análise da Equação 4.1, é utilizado o o comprimento do veículo, um

multiplicador de 1,3 para garantir que a vaga tenha um tamanho mínimo de 30% maior que o comprimento do automóvel, e um último multiplicador de 1,05, referente a margem de segurança de 5% em relação ao valor real calculado pela leitura dos sensores.

4.3.2 Configuração do servidor Node.js

Uma vez estabelecida a leitura de dados do simulador, é necessária a construção de um ambiente para transmitir estes dados para a parte que será vista pelo condutor, e consequentemente, um ambiente para expor a interface gráfica. Neste contexto, foi construído um servidor Node.js com utilização da linguagem TypeScript. Os principais pontos deste servidor são:

- Criar um ambiente para o servidor WEB;
- Receber os dados do servidor Python;
- Comunicar com a interface gráfica por meio de um WebSocket.

o Node.js dispõe de uma sintaxe para criação fácil de um servidor e ainda mais simples em caso de expor arquivos estáticos, como é o caso, visto que a interface gráfica não possui diversas telas, mas atualiza-se a cada iteração. Em paralelo, foram criadas algumas rotas para receber as chamadas HTTP do servidor Python, sendo que, para cada rota foi criado um evento que será emitido no WebSocket (Figura 4.3).

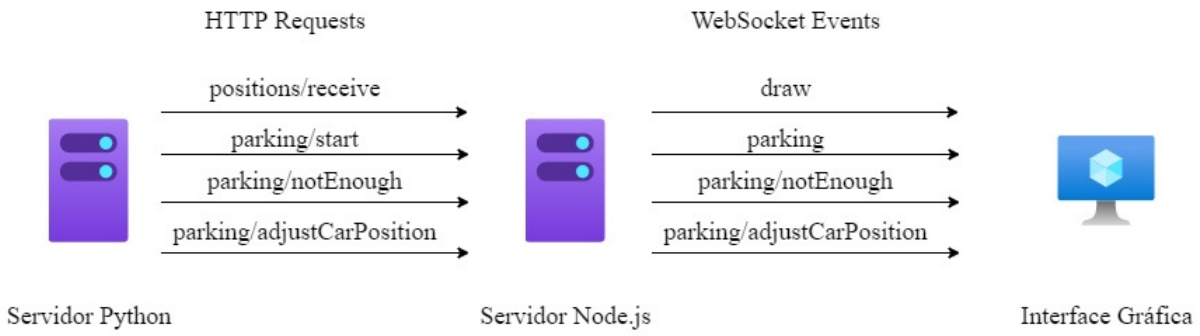

Figura 4.3: Requisições e eventos no Node.js

Para recebimento dos dados foram criados alguns endpoints, cada um com sua função específica de iniciar alguma ação na interface gráfica ou simplesmente transmitir dados recebidos. Para o endpoint positions/receive, ele é responsável por receber os dados com a posição (x,y) do veículo, ângulo das rodas e velocidade. Por sua vez, parking/start dá início ao processo de estacionamento, após ter identificado com sucesso uma vaga apta para o veículo atual, transmitindo o dado slotSize que representa o tamanho, em metros, da vaga encontrada, valor este já convertido pelo servidor Python.

Os demais endpoints são responsáveis por atualizar os textos que estão sendo exibidos na interface gráfica, não transmitindo nenhuma dado. O endpoint parking/notEnough inicia a exibição na interface gráfica de uma mensagem indicando que o veículo identificou um espaço vazio, porém, este não possuía espaço suficiente para acomodar o veículo.

Por fim, parking/adjustCarPosition, acontece quando o veículo encontrou uma vaga apta para o processo de estacionamento, e o condutor precisar avançar mais uma determinado espaço, até que o veículo fique na posição correta para o início do estacionamento. Neste último, é exibida em tela uma mensagem orientando o condutor para diminuir a velocidade e continuar em frente, em seguida, exibe seu veículo.

4.3.3 Criação do ambiente de visualização/Web

Inicialmente, foi projetado para o funcionamento da interface gráfica movimentar o veículo utilizando somente as suas coordenadas (x,y) e sua orientação em relação ao plano. Contudo, esta abordagem não provou ser eficiente, pois no momento de rotação das rodas, as trajetórias do veículos não se mostraram equivalentes. Por este motivo, foi escolhida uma abordagem de representação do veículo na interface gráfica, a partir da implementação de movimento baseado no modelo de Ackermann com os mesmos dados que estão sendo lidos pelos sensores no simulador.

No momento em que a simulação é iniciada pelo servidor Python, a interface gráfica utiliza os dados recebidos para inicializar seus parâmetros, de modo que, mesmo que o veículo não esteja sendo exibido ao condutor, quando este aparecer estará em posição

equivalente e com o movimento correto.

O funcionamento da interface gráfica está totalmente baseada nos eventos recebidos por meio do WebSocket, por meio do qual sai do estado inicial predefinido até o momento de estacionamento completo. O estado inicial é composto pela exibição de um canvas, ou quadro, com a dimensão de 500x500 pixeis, exibindo a mensagem de que está buscando uma vaga conforme a Figura 4.4.

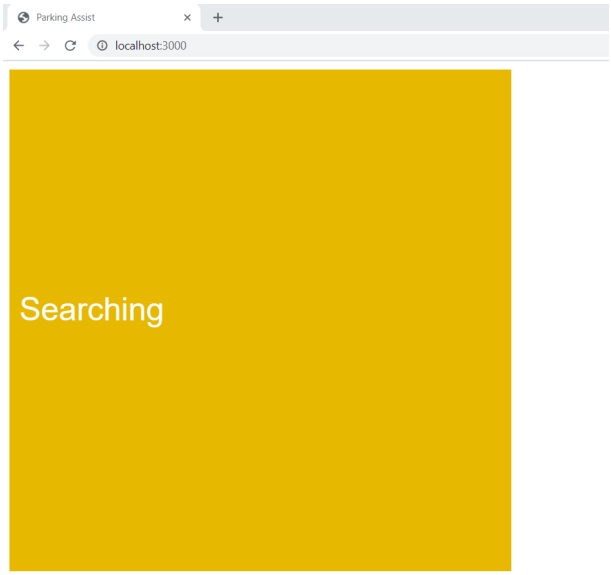

Figura 4.4: Estado inicial da interface gráfica

A partir deste ponto, o veículo pode se movimentar pelo plano, e o próximo evento possível de ser recebido pelo framework p5.js é o de que encontrou uma vaga e iniciar o processo de ajuste da posição de estacionamento, ou encontrar um espaço que não comporta o veículo, tendo de exibir em tela a respectiva mensagem conforme ilustra a Figura 4.5.

Por sua vez, se a vaga encontrada cumpre os requisitos mínimos é exibido em tela uma mensagem orientando o condutor a diminuir a velocidade e seguir em frente, em seguida exibe seu veículo em tela, pois, quando o sensor posicionado na roda traseira do veículo ler o começo do obstáculo, o veículo estará na posição correta para estacionamento, sendo

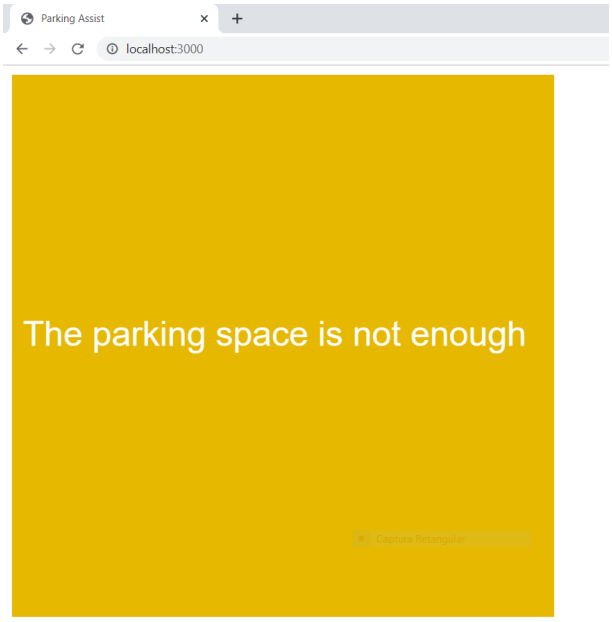

Figura 4.5: Espaço encontrado não é o suficiente

então exibido em tela a trajetória calculada e os obstáculos encontrados.

Todas as mensagens exibidas em telas são emitidas a partir dos eventos recebidos no WebSocket, o qual é configurado indicando a Uniform Resource Locator (URL) no qual o servidor está sendo executado, neste caso, é o endereço do servidor Node.js.

4.3.4 Cálculo de trajetória

Quando o sistema identifica que uma vaga tem o comprimento mínimo necessário, o p5.js recebe este evento e começa os cálculos para definição de trajetória. Para a sua criação, foi utilizada uma alteração da função sigmoide, adaptando seus coeficientes de acordo o cenário e em relação a largura do veículo, visto que, um veículo mais largo irá necessitar de um espaço lateral maior para conseguir efetuar a manobra.

A função utilizada foi inicialmente multiplicada por -1, para ajustar sua posição após rotacionar o gráfico gerado, pois quando posicionado ao lado do veículo sua representação foi observada de maneira invertida. A largura do veículo tem relação com o ângulo que

o veículo terá de fazer para entrar na vaga desejada, por este motivo, para escolha do coeficiente responsável por essa alteração, foi feita a escolha com base na largura do veículo e o ângulo gerado (Equação 4.2).

$$f(x) = -1 * (7.2 * 0.75) * \frac{1}{1 + e^{-x}} \tag{4.2}$$

Na Fórmula 4.2 é possível identificar o multiplicador para corrigir a inversão (-1) e o valor 7.2, que é obtido através da conversão dos dados do veículo utilizado no simulador, cujo comprimento é de 1,8m. Para obter este valor, foi necessário converter conforme o padrão utilizado no servidor Python, cujo multiplicador é 20 e, após isso, também é necessário realizar uma divisão por 5, visto que ao representar o gráfico da função sigmoide na interface gráfica foi necessário utilizar uma escala para aumentar sua representação e torná-la visível ao usuário, sendo a escala aplicada pelo framework p5.js.

O valor 0.75 foi obtido a partir de tetes realizados durante o processo de estacionamento, pois somente a largura total efetiva do veículo, gera um ângulo de movimento do veículo, no momento de entrada na vaga, elevado, o que torna inviável a reprodução pelo condutor. Desta maneira, foi feito uma proporção de 75% do valor de sua largura, gerando o gráfico conforme representado na Figura 4.6.

Uma vez com a função definida, foi escolhido o intervalo para exibição na interface gráfica, representando a trajetória de estacionamento, dado pelos valores máximos e mínimos no eixo dos XX de 7 para máximo e, para o valor mínimo foi utilizada uma proporção de acordo com o tamanho da vaga encontrada, ou seja, aproximadamente -10. Desta forma, foram obtidos os valores para a função sigmoide criada, substituindo para X os números no intervalo [-10,7].

Por fim, em um cenário onde o sistema é capaz de identificar uma vaga apta para estacionamento do veículo, é feito o cálculo da trajetória e representado na interface gráfica juntamente com os obstáculos encontrados (Figura 4.7).

Para concluir o desenvolvimento deste modelo, foram realizados testes com diferentes

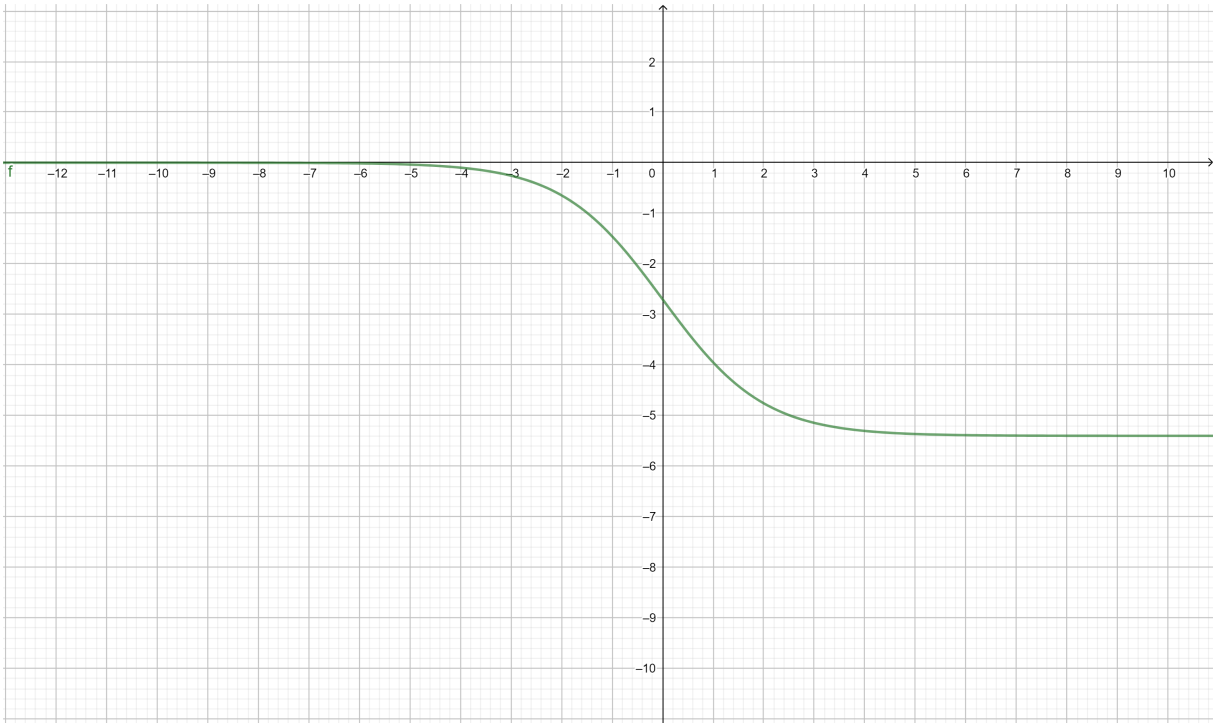

Figura 4.6: Gráfico da função sigmoide gerada

tamanhos de vagas disponíveis, sendo algumas menores que o mínimo necessário a algumas maiores. Nos cenários onde é possível estacionar o veículo, foram feitas simulações, controlando os movimentos do veículo tentando reproduzir a trajetória exibida em tela, e o resultado foi um sucesso na conclusão no processo de baliza.

4.4 Considerações finais

Com a busca por tecnologias capazes de atender aos requisitos necessários deste projeto, ficou evidente a grande quantidade de linguagens de programação, frameworks e ambientes disponíveis, tendo cada um destes suas respectivas vantagens e desvantagens de acordo com a aplicação. As tecnologias escolhidas em conjunto para resolução do problema colocado em questão neste projeto, não são exclusivamente únicas, podendo ser alteradas e chegar ao resultado da mesma forma, ficando a critério do desenvolvedor.

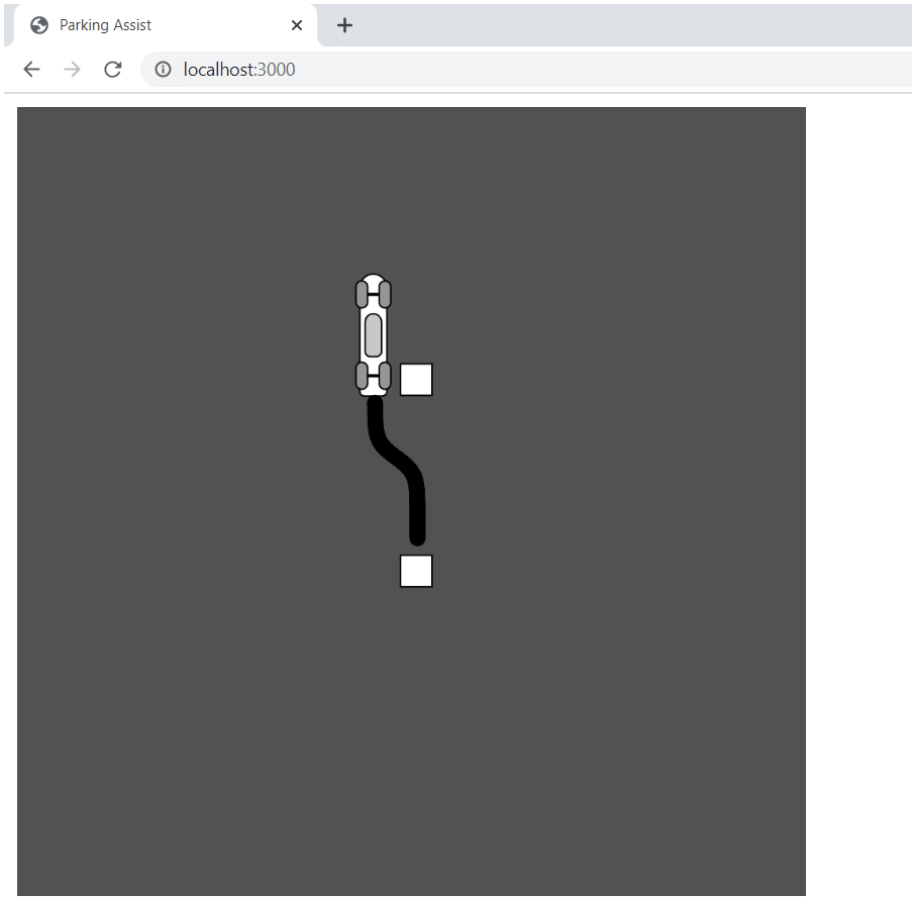

Figura 4.7: Trajetória final gerada

Capítulo 5

Resultados e discussão

O desenvolvimento do sistema teve, por base, estudar um sistema de apoio ao estacionamento paralelo de veículos, em específico tomando como principal referência a obtenção de uma trajetória por meio da aproximação de uma função sigmoide. Com o intuito de comprovar sua eficácia, foi criado um ambiente virtual por meio do uso do simulador CoppeliaSim, responsável por criar um cenário mais próximo do real, visto que o custo de teste em veículos reais é muito elevado.

O sistema teve como objetivo mostrar ao condutor uma trajetória para estacionar seu veículo, usando para isso um servidor em Python para entrada dos dados e, por ser mais rápido, um servidor em Node.js para hospedar a plataforma Web e transmitir os dados recebidos.

5.1 Caracterização dos resultados

De modo a validar o modelo proposto, foram adicionados três obstáculos em formatos de cubos, com espaçamento variado entre si, de modo a gerar um espaço vazio com comprimento de 3,75m e um outro espaço com 4,59m. Estes espaçamentos foram selecionados de modo a gerar dois cenário para o veículo:

1. Uma vaga com tamanho menor que o mínimo definido;

2. Uma outra vaga com tamanho muito próximo ao mínimo necessário.

Desta maneira, foi obtido o cenário, conforme ilustrado na Figura 5.1.

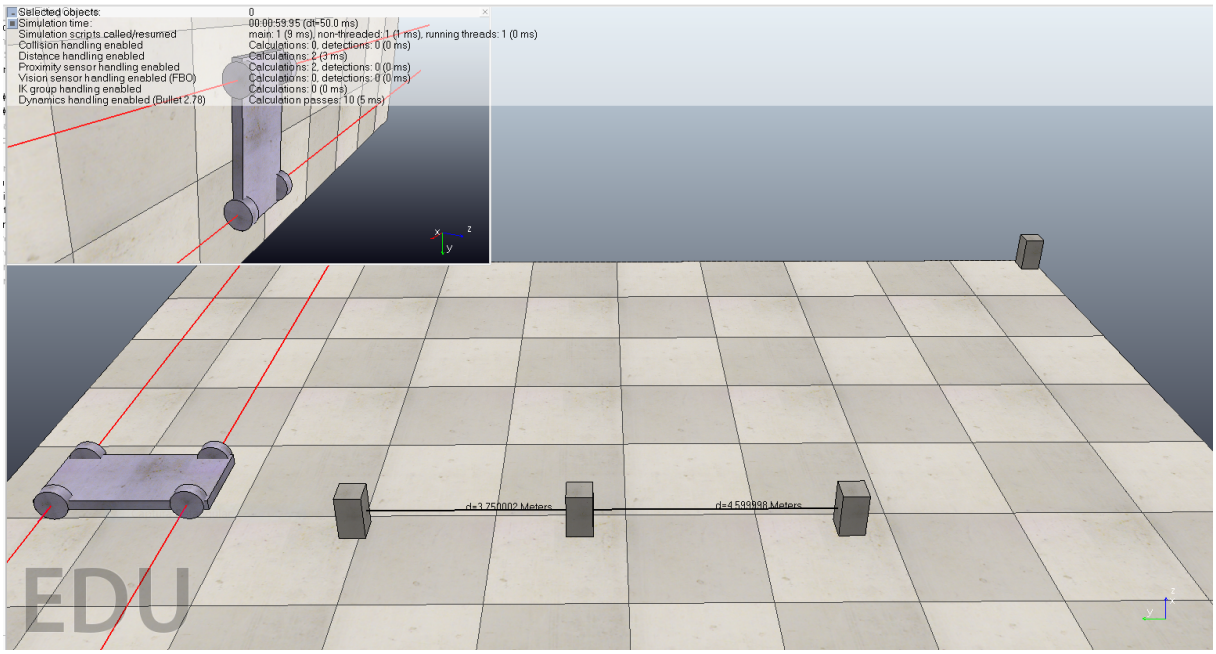

Figura 5.1: Estado inicial do ambiente de teste

Em paralelo foi executado o sistema desenvolvido para iniciar o procedimento de teste, resultando no seguinte cenário conforme a Figura 5.2.

Foi então, a partir da interação com o simulador, feito o movimentação do veículo a frente, de modo a alcançar os obstáculos. Ao passar pelo primeiro obstáculo nenhuma alteração foi exibida na interface gráfica, contudo, ao passar pelo segundo obstáculo, o sistema realizou os cálculos de distância e apresentou a mensagem de que a vaga encontrada não possuía o tamanho mínimo necessário, conforme a Figura 5.3 a seguir, e após alguns segundos, voltou a exibir a mensagem de que estava buscando por uma vaga.

Seguindo em frente, quando as rodas da frente do veículo passaram ao lado do terceiro obstáculo, foi exibida a mensagem para diminuir a velocidade e continuar em frente, indicando ter encontrado uma vaga disponível para o estacionamento (Figura 5.4).

Logo após passar pelo obstáculo foi e veículo exibido na interface gráfica, porém sem indicação dos demais obstáculos e sem a trajetória, sendo estes exibidos somente quando

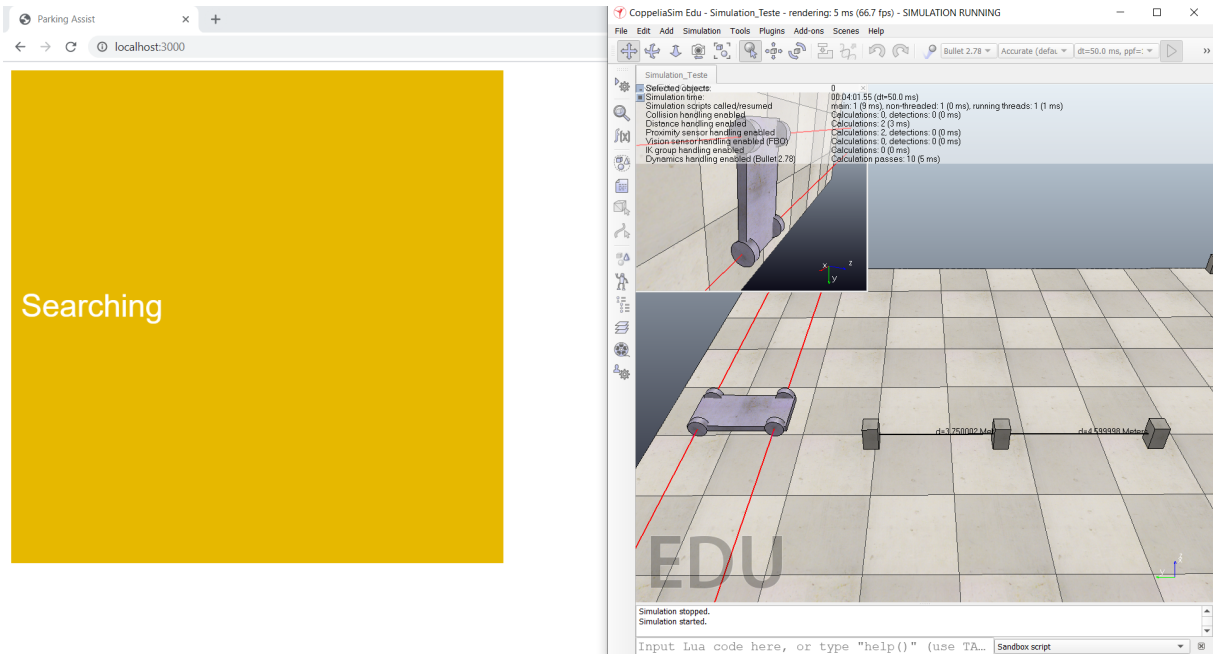

Figura 5.2: Estado inicial com exibição da aplicação desenvolvida

a roda traseira identifica o obstáculo 3 (Figura 5.5).

Neste momento, através dos controle do simulador foram feitos os comandos para o veículo parar completamente e, em seguida, começar o movimento em marcha ré. Quando o veículo, representado na interface gráfica, atingiu o início da curvatura exibida na trajetória, foi feito o movimento no volante de modo a tentar manter o centro do veículo na trajetória, como se estivesse seguindo a linha desenhada (Figura 5.6).

Em seguida, quando a parte traseira do veículo, atingiu o começo da linha reta da trajetória, foi feito o movimento para o lado contrário da vaga, simulando o movimento no volante do veículo, sem parar seguir em marcha ré, até que o mesmo se alinhasse na vaga, como mostra a Figura 5.7.

Quando o veículo atingiu o ponto máximo onde é possível ir em marcha ré, sem colidir com o carro que está atrás, foi feito o movimento de parada, alinhar as rodas e seguir em linha reta a frente para posicionar o veículo no centro da vaga atual, sendo possível observar na Figura 5.8.

Foi realizado um outro teste, partindo do momento que é exibida a trajetória e os

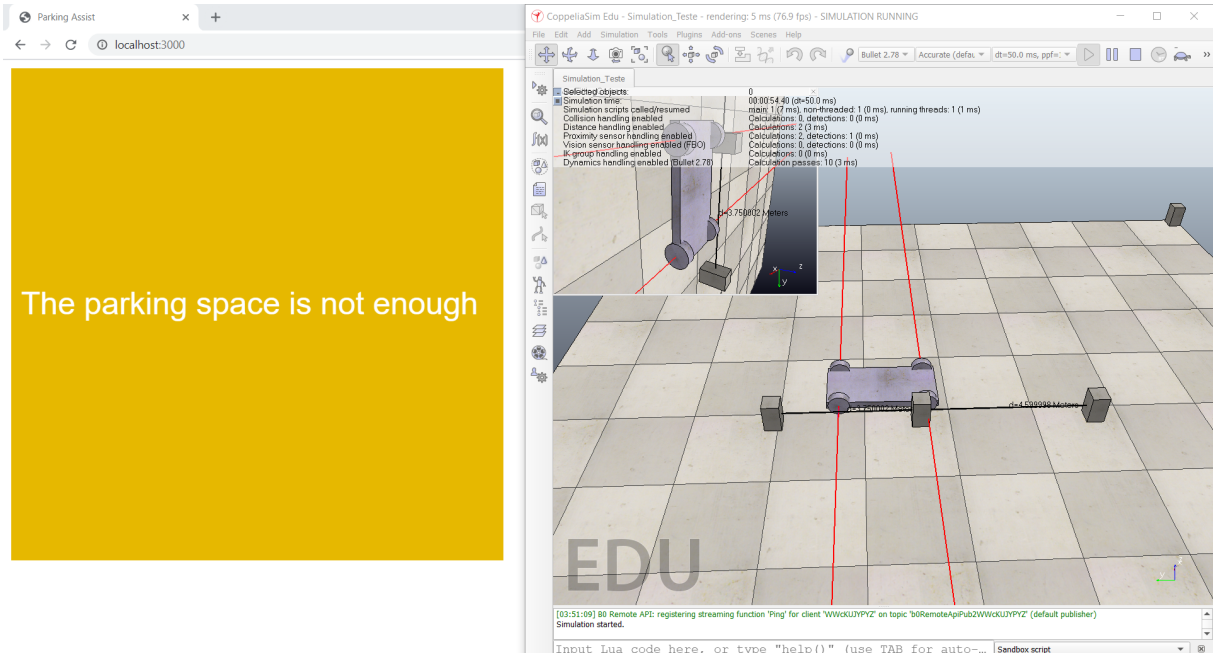

Figura 5.3: Cenário de vaga com tamanho menor que o mínimo

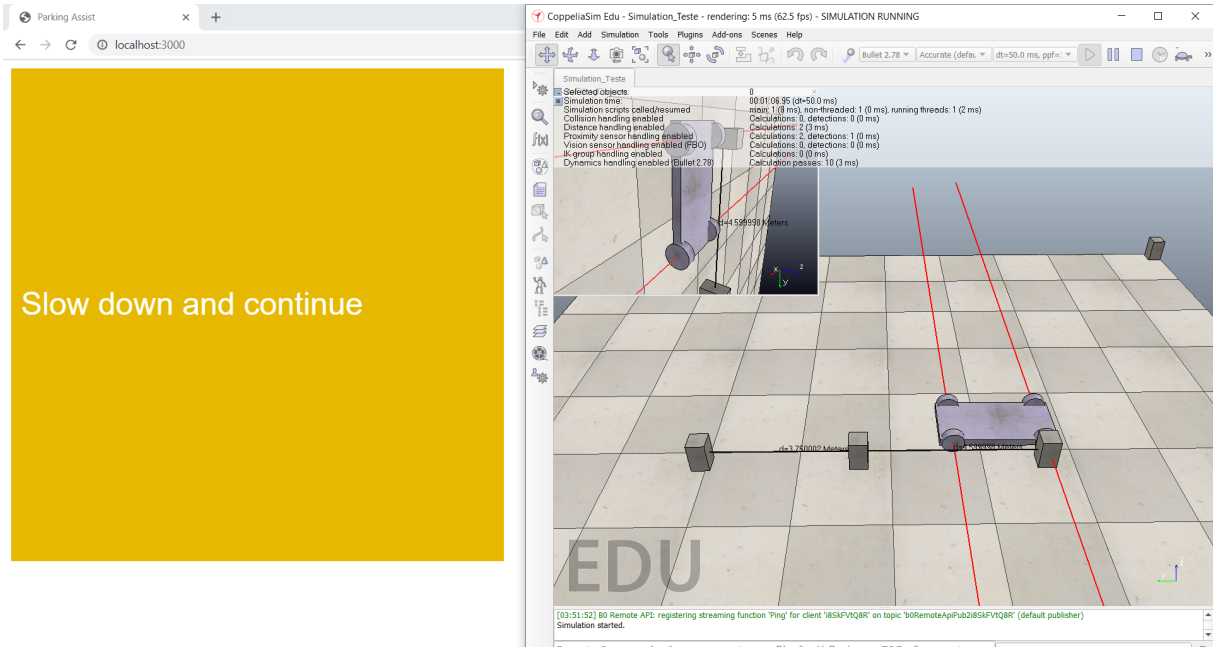

Figura 5.4: Vaga com tamanho mínimo encontrada

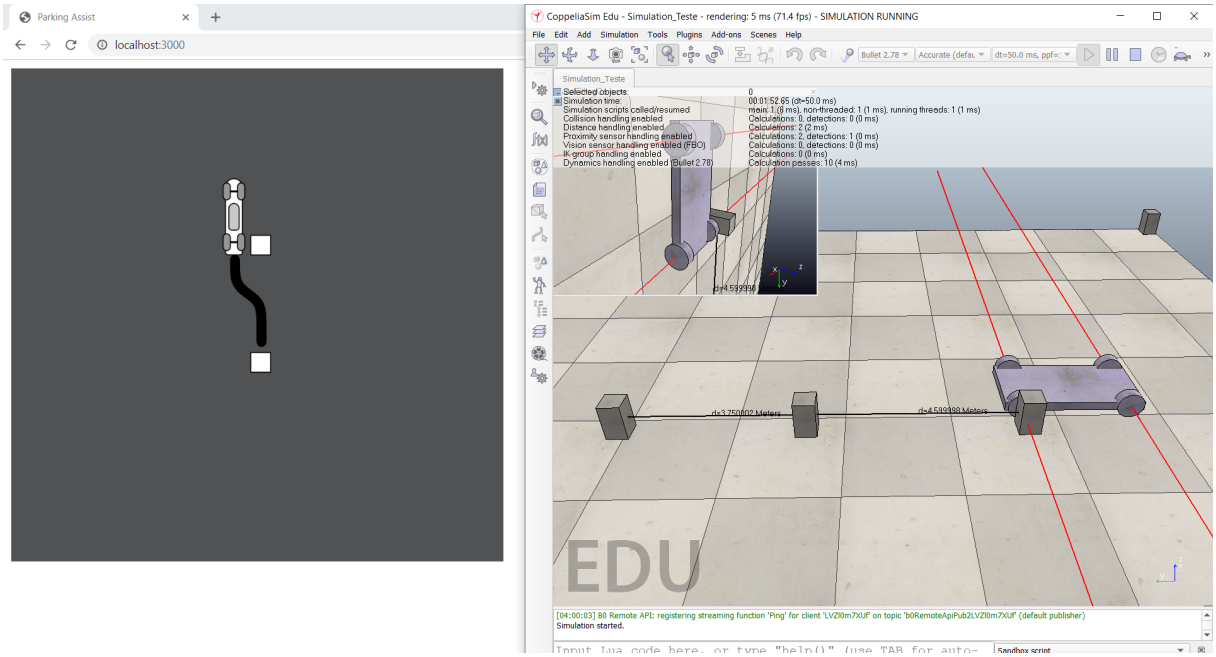

Figura 5.5: Exibição da trajetória e demais obstáculos

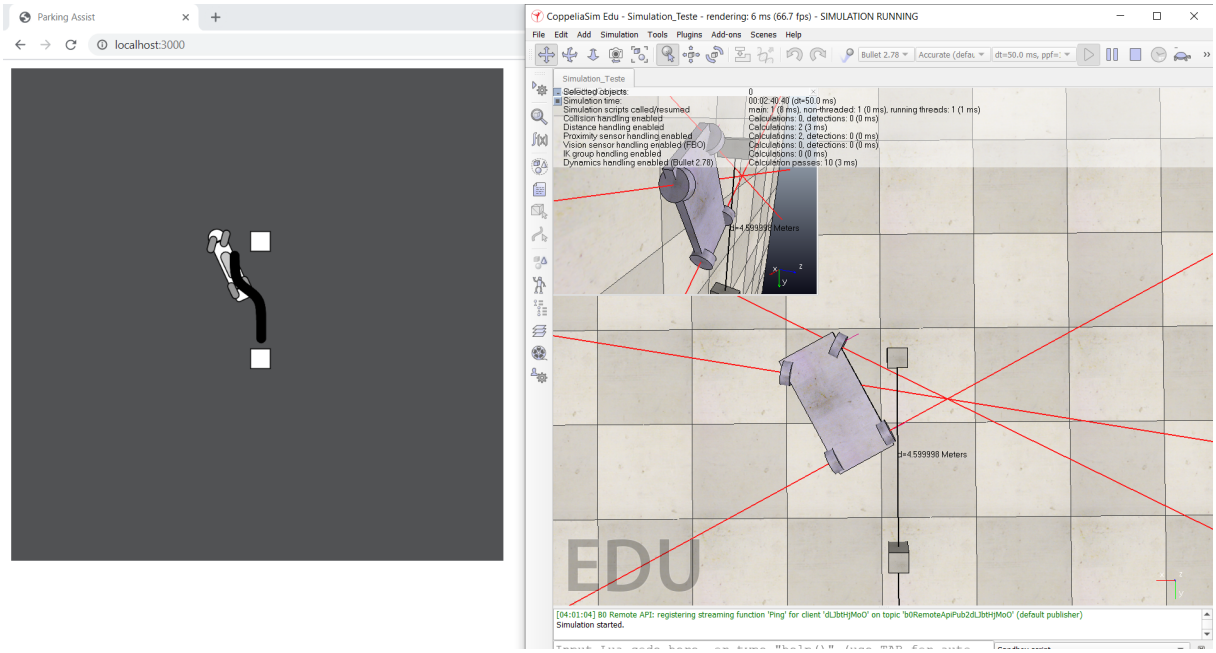

Figura 5.6: Movimento de curva durante a execução da trajetória

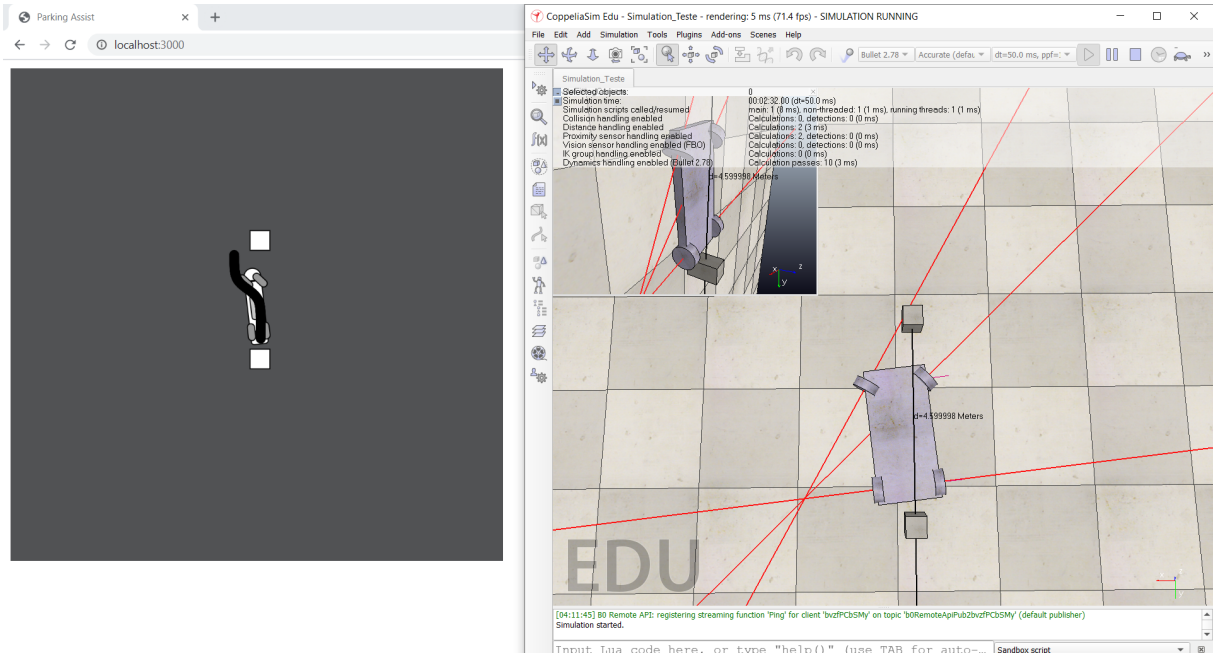

Figura 5.7: Movimento de troca de direção durante a trajetória

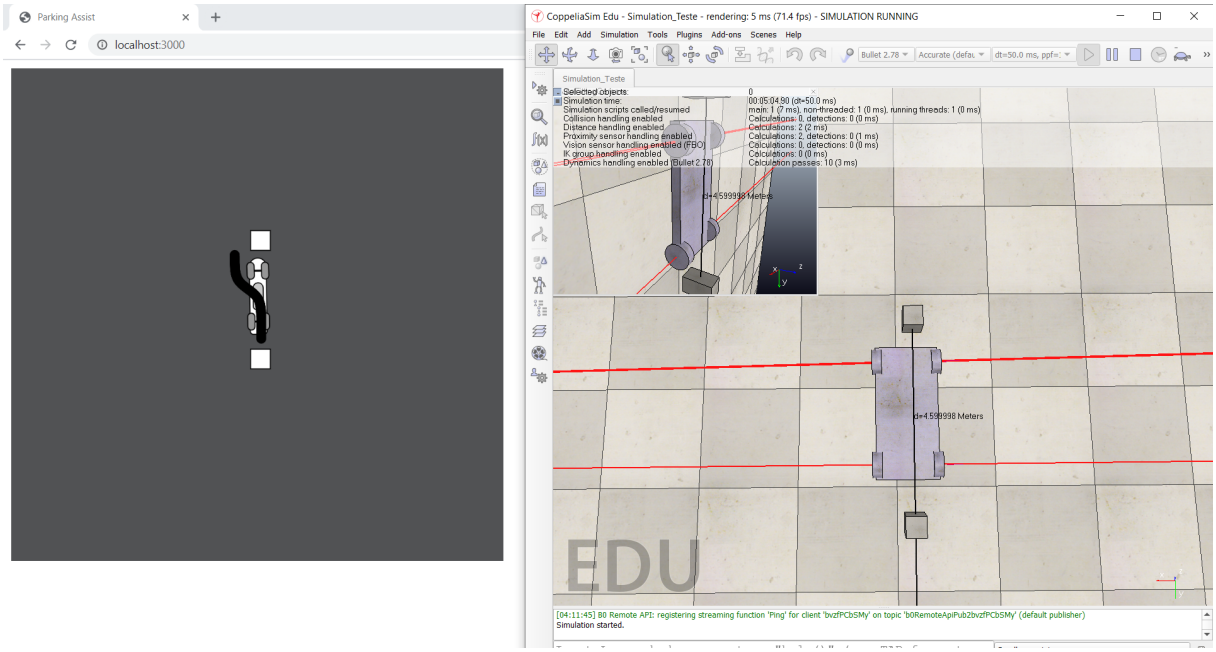

Figura 5.8: Ajuste após chegar ao fim da trajetória

obstáculos encontrados. Porém, ao invés de fazer o movimento em marcha, como recomendado pela interface gráfica, foi feito movimento de virada das rodas no sentido de entrada na vaga e então feito o movimento de marcha ré, até ao momento em que o veículo chega ao final do ângulo da trajetória (Figura 5.9).

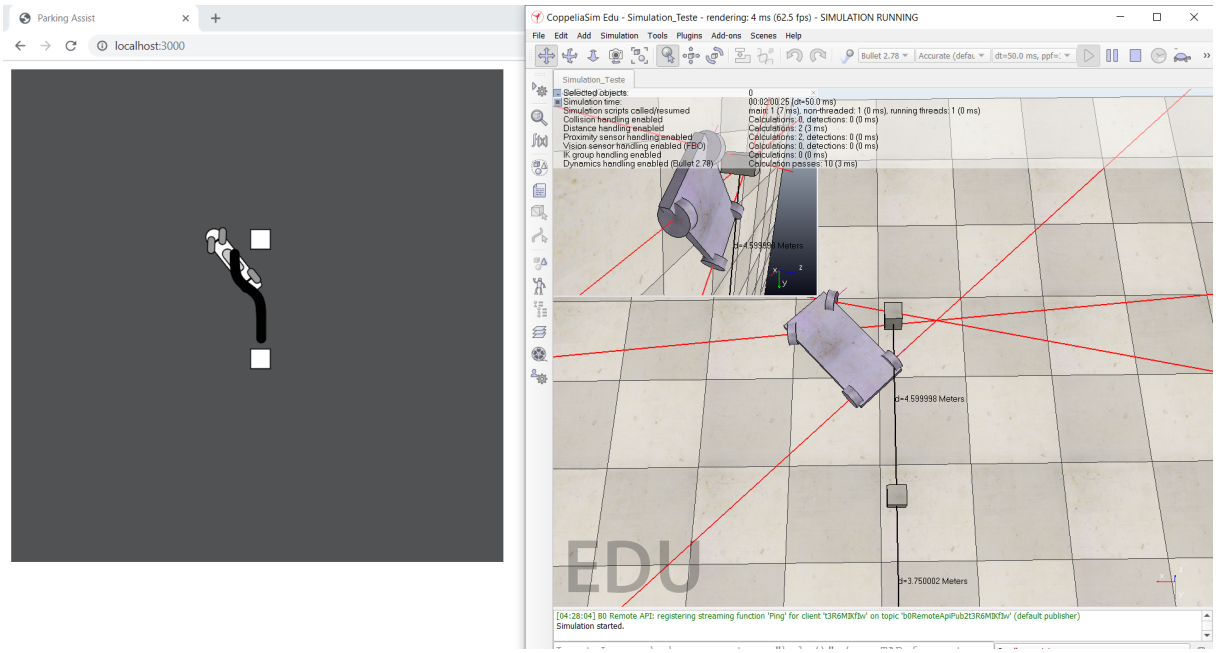

Figura 5.9: Execução de movimento incorreto

Neste momento foi possível perceber que se fossem feitos os mesmos comandos para virar as rodas no ângulo contrário para entrar na vaga, o veículo iria colidir com o obstáculo a sua frente. Visto que a trajetória não é recalculada, foram feitos movimentos de modo a centrar o veículo na trajetória, saindo da vaga, sem retornar a posição inicial e retornando. Nesta última situação, mesmo com a entrada inicial sido uma falha, com alguns movimentos e correções, o veículo foi capaz de ser posicionado na vaga encontrada.

5.2 Discussão

Foi possível analisar que o modelo desenvolvido atende à proposta inicial desse trabalho, em prestar auxílio ao motorista a estacionar seu veículo, sendo assistido por uma trajetória gráfica. Embora o sistema não tenha instruções do momento em que deve ser realizada

cada ação, o sistema mostrou-se capaz de atender as necessidade básicas de auxílio, como por exemplo, identificar se o espaço em questão tem o comprimento mínimo necessário e propor uma sugestão de trajetória.

Nos cenários em que o condutor não segue a trajetória, seja por falta de orientação do sistema ou por desconhecimento do processo de baliza, é possível retornar ao estado inicial para começar novamente o processo de estacionamento, e mesmo sem retornar ao estado inicial, com algumas manobras também mostrou-se possível continuar o estacionamento, contudo, necessitando de mais habilidade por parte do motorista, principalmente em relação ao controle de seu veículo.

5.3 Considerações finais

Após os testes no modelo desenvolvido, o mesmo mostrou-se capaz de atender as necessidades de auxiliar no processo de estacionamento, sendo coerente quando encontrou um espaço que não cumpre as distâncias mínimas definidas, e também para o cenário limite, no qual o espaço para estacionamento é igual ao mínimo. Outro fator perceptível nos testes, foi de que, quando o condutor não segue os movimentos para manter o veículo na trajetória, ainda é possível realizar o estacionamento, porém, neste cenário, os movimentos são dependentes do condutor para voltar a trajetória ou finalizar por si próprio o estacionamento.

Capítulo 6

Conclusões e trabalhos futuros

Nesta tese foi desenvolvido um assistente de estacionamento para vagas paralelas, aplicável a veículos de passeios, utilizando de uma aproximação da função sigmoide com coeficientes baseadas nas especificações do veículo criado no simulador. Para validação do modelo, foi criada uma arquitetura composta por um simulador, um servidor Python, um servidor Node.js e uma interface gráfica. Como resultado, foi gerado um modelo assistente de estacionamento e uma arquitetura pra validação.

A partir da conclusão deste trabalho e da análise dos resultados obtidos, foi possível observar que uma solução mais simples, proposta da aproximação de uma função, consegue cumprir de forma aceitável, a tarefa de propor uma trajetória de estacionamento viável. Provando que sua simplicidade pode ser estendida aos demais cenários de estacionamento, então como sugestão para trabalhos futuros seria interessante abordagens para vagas a 45º e vagas perpendiculares.

Uma vez criadas implementações para os demais tipos de vagas, seria interessante a extensão dos testes a um veículo real, com objetivo de validar as alterações necessárias em um veículo real, e posteriormente oferecer para um grupo de pessoas testarem, com o intuito de uma pesquisa estatística para comprovar se o sistema realmente é capaz de auxiliar no aprendizado dos condutores com seu uso contínuo.

Bibliografia

[1] C. M. FRANZ e J. R. V. SEBERINO, *A história do trânsito e sua evolução*, Monografia apresentada ao Curso de Pós-Graduação Lato Sensu, como requisito parcial para obtenção do certificado de Especialista em Gestão, Educação e Direito de Trânsito, Profa. Orientadora Ma. Denise Raquel Rosar, 2012.

[2] G. GIUCCI, *A vida cultural do automóvel: percursos da modernidade cinética.* Editora Record, 2004.

[3] V. A. D. MELO, "O automóvel, o automobilismo e a modernidade no Brasil (1891-1908)", *Revista Brasileira de Ciências do Esporte*, vol. 30, n.º 1, pp. 201–203, 2008.

[4] N. CORASSA, "Uso do carro como uma extensão da casa e os conflitos no trânsito.", em. Casa do Psicólogo, 2003, pp. 61–74.

[5] C. B. SERAFIM, M. d. S. AMARANTE, S. H. d. A. M. JUNIOR, T. M. Franco, A. C. CALIXTO, K. B. VICENTE e E. E. SILVA, "O automóvel, o automobilismo e a modernidade no Brasil (1891-1908)", *Revista Brasileira de Ciências do Esporte*, vol. 4, n.º 1, pp. 266–280, 2018.

[6] O. NÜBEL, "The Beginnings of the Automobile in Germany", em. The Economic e Social Effects of the Spread of Motor Vehicles, 1987, pp. 55–66.

[7] M. BROY e A. SCHMIDT, "Challenges in engineering cyber-physical systems", *Computer*, vol. 47, n.º 2, pp. 70–72, 2014.

[8] Mercedes-Benz, 2019. URL: `https://www.mercedes-benz.com/en/classic/`.

[9] M. D. MARSON, "A evolução da indústria de máquinas e equipamentos no Brasil: Dedini e Romi, entre 1920 e 1960", *Nova Economia*, vol. 24, n.º 3, pp. 685–710, 2014.

[10] R. d. L. BRANDÃO, *O automóvel no Brasil entre 1955 e 1961: a invenção de novos imaginários na era JK*, 2011.

[11] *Conheça a história do automóvel no Brasil: Rica e apaixonante. 2018*, 2018. URL: https://revistacarro.com.br/historia-do-automovel-no-brasil-rica-e-apaixonante/.

[12] G. GARCIA, *A história da fábrica da DKW Vemag: Brasileiros produzindo veículos para o Brasil.* 2009. URL: http://www.saopauloantiga.com.br/vemag-uma-%20fabrica-que-agoniza-no-tempo/..

[13] H. CASIER, P. MOENS e K. APPELTANS, "Technology considerations for automotive", em *Proceedings of the 30th European Solid-State Circuits Conference*, IEEE, 2004, pp. 37–41.

[14] D. C. M. WEI, R. d. S. PISSARDINI e E. S. d. F. JUNIOR, "CONVERGÊNCIA DE VEÍCULOS INTELIGENTES E VEÍCULOS AUTÔNOMOS", em *Anais do XXVII Congresso de Pesquisa e Ensino em Transportes*, ANPET, 2004.

[15] M. PRETORIUS, *How does brake assist work?*, 2019. URL: https://www.autotrader.co.za/cars/news-and-advice/automotive-news/how-do%20es-brake-assist-work/4582.

[16] F. Kröger, "Automated driving in its social, historical and cultural contexts", em *Autonomous Driving*, Springer, 2016, pp. 41–68.

[17] R. d. S. PISSARDINI, D. C. M. WEI e E. S. d. F. JUNIOR, "Veículos Autônomos: conceitos, histórico e estado-da-arte", em *Anais do XXVII Congresso de Pesquisa e Ensino em Transportes–ANPET*, 2013.

[18] C. Stiller, G. Farber e S. Kammel, "Cooperative cognitive automobiles", em *2007 IEEE Intelligent Vehicles Symposium*, IEEE, 2007, pp. 215–220.

[19] D. Melvin, "Cop pulls over Google self-driving car finds no driver to ticket", *CNN*, 2015. URL: `https://edition.cnn.com/2015/11/13/us/google-self-driving-car-pulled-over/index.html`.

[20] J. J. Waiselfisz, "Mapa da violência 2012. Caderno complementar 2: acidentes de trânsito", *Instituto Sangari, São Paulo*, 2012.

[21] Z. Shiller, Y.-R. Gwo et al., "Dynamic motion planning of autonomous vehicles", *IEEE Transactions on Robotics and Automation*, vol. 7, n.º 2, pp. 241–249, 1991.

[22] D. Dolgov, S. Thrun, M. Montemerlo e J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments", *The International Journal of Robotics Research*, vol. 29, n.º 5, pp. 485–501, 2010.

[23] F. Osório, F. Heinen e L. Fortes, "Controle da Tarefa de Estacionamento de um Veículo Autônomo através do Aprendizado de um Autômato Finito usando uma Rede Neural J-CC", *VII Simpósio Brasileiro de Redes Neurais*, 2002.

[24] M. R. Heinen, F. S. Osório, F. J. Heinen e C. Kelber, "Uso de Realidade Virtual no Desenvolvimento de um Sistema de Controle do Estacionamento de Veículos Autônomos", em *Proc. of VIII Symposium on Virtual Reality*, 2006, pp. 245–256.

[25] M. G. Prado, "Planejamento de trajetória para estacionamento de veículos autônomos", tese de doutoramento, Universidade de São Paulo, 2013.

[26] S. Gupte, O. Masoud, R. F. Martin e N. P. Papanikolopoulos, "Detection and classification of vehicles", *IEEE Transactions on intelligent transportation systems*, vol. 3, n.º 1, pp. 37–47, 2002.

[27] K. Demirli e M. Khoshnejad, "Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller", *Fuzzy sets and systems*, vol. 160, n.º 19, pp. 2876–2891, 2009.

[28] B. C. Pinheiro, "Sistema de controle tempo real embarcado para automação de manobra de estacionamento", tese de mestrado, Universidade Federal de Santa Catarina, 2009.

[29] H. Vorobieva, S. Glaser, N. Minoiu-Enache e S. Mammar, "Automatic parallel parking in tiny spots: Path planning and control", *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, n.º 1, pp. 396–410, 2014.

[30] F. Osório, F. Heinen e L. Fortes, "Controle inteligente de veículos autônomos: Automatização do processo de estacionamento de carros", *Anais do SEMINCO*, 2001.

[31] T.-H. Li e S.-J. Chang, "Autonomous fuzzy parking control of a car-like mobile robot", *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 33, n.º 4, pp. 451–465, 2003.

[32] V. MATSUBARA, "Estudo revela que 40% dos acidentes ocorrem ao estacionar", *Quatro Rodas*, 2016. URL: `https://quatrorodas.abril.com.br/noticias/estudo-revela-que-40-dos-acidentes-ocorrem-durante-manobras-de-estacionamento`.

[33] N. J. Schouten, M. A. Salman e N. A. Kheir, "Fuzzy logic control for parallel hybrid vehicles", *IEEE transactions on control systems technology*, vol. 10, n.º 3, pp. 460–468, 2002.

[34] Á. L. Sordi Filho, L. P. de Oliveira, A. S. de Oliveira, J. A. Fabro e M. A. Wehrmeister, "Navegação e mapeamento simultâneo em um carro autônomo baseado na lógica nebulosa", em *Anais do XII Congresso Brasileiro de Inteligência Computacional*, 2015.

[35] P. P. Bonissone e K. S. Aggour, "Fuzzy automated braking system for collision prevention", em *10th IEEE International Conference on Fuzzy Systems.(Cat. No. 01CH37297)*, IEEE, vol. 2, 2001, pp. 757–760.

[36] E. Rohmer, S. Singh e M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework", 2020. URL: `https://coppeliarobotics.com/coppeliaSim_v-rep_iros2013.pdf`.

[37] M. F. Selekwa, D. D. Dunlap e E. G. Collins, "Implementation of multi-valued fuzzy behavior control for robot navigation in cluttered environments", em *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 3688–3695.

[38] Y. Zhao e E. G. Collins Jr, "Robust automatic parallel parking in tight spaces via fuzzy logic", *Robotics and Autonomous Systems*, vol. 51, n.º 2-3, pp. 111–127, 2005.

[39] L. L. G. VERMAAS, "Aprendizado supervisionado de sistemas de inferência fuzzy aplicados em veículos inteligentes", tese de doutoramento, Universidade Federal de Itajubá, 2010.

[40] A. A. Heinz, "Sistema de Detecção de Vagas Paralelas e Estacionamento Automático Utilizando Sensores Ultrassônicos", 2014.

[41] T. E. Oliphant, "Python for scientific computing", *Computing in Science & Engineering*, vol. 9, n.º 3, pp. 10–20, 2007.

[42] M. S. Silva, *JavaScript-Guia do Programador: Guia completo das funcionalidades de linguagem JavaScript*. Novatec Editora, 2010.

[43] A. Rastogi, N. Swamy, C. Fournet, G. Bierman e P. Vekris, "Safe & efficient gradual typing for TypeScript", em *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2015, pp. 167–180.

[44] S. Tilkov e S. Vinoski, "Node. js: Using JavaScript to build high-performance network programs", *IEEE Internet Computing*, vol. 14, n.º 6, pp. 80–83, 2010.

[45] L. McCarthy, C. Reas e B. Fry, *Getting Started with P5. js: Making Interactive Graphics in JavaScript and Processing*. Maker Media, Inc., 2015.

[46] R. Rai, *Socket. IO Real-time Web Application Development*. Packt Publishing Ltd, 2013.