Universidade Federal do ABC
Engenharia de Instrumentação, Automação e Robótica

Henrique Lima Werneck

Técnicas de Inteligência Artificial e Visão Computacional para Controle de Carro Robô

Santo André
2018

Universidade Federal do ABC
Engenharia de Instrumentação, Automação e Robótica

Henrique Lima Werneck

Técnicas de Inteligência Artificial e Visão Computacional para Controle de Carro Robô

Trabalho apresentado à Universidade Federal do ABC para obtenção do título de Engenheiro de Instrumentação, Automação e Robótica

Orientador:
Prof. Dr. Luiz Antônio Celiberto Júnior

Santo André
2018

Henrique Lima Werneck

Técnicas de Inteligência Artificial e Visão Computacional para Controle de Carro Robô

Trabalho julgado e aprovado para a obtenção do grau de Engenheiro de Instrumentação, Automação e Robótica pelo curso de Graduação em Engenharia de Instrumentação, Automação e Robótica da Universidade Federal do ABC.

Santo André - SP, 15 de julho de 2018.

Prof. Dr. Luis Alberto Martínez Riascos - Coordenador do curso

_____

Prof. Dr. Luiz Antonio Celiberto Junior
Orientador

_____

Prof. Dr. Roberto Luiz da Cunha Barroso Ramos
Avaliador

Agradecimentos

Agradeço primeiramente aos meus pais, Luciomar e Isabel, por sempre terem priorizado minha educação e sempre terem feito de tudo por isso, e por mim de maneira geral. Infelizmente, não tem como agradecer o suficiente só com esse parágrafo.

Agradeço ao meu irmão, Igor, por ser meu melhor amigo sempre em toda a minha vida.

Agradeço à minha namorada, Paloma, por sempre me apoiar, me empurrar e me ajudar, mesmo quando ela não sabia como, mas principalmente quando nem eu acreditava em mim mesmo.

Agradeço a todos os meus amigos, professores e colegas da Universidade Federal do ABC, por todo apoio e ajuda que me deram e que me permitiram de alguma forma completar este trabalho.

Agradeço ao professor Doutor Monael Pinheiro. Graças a uma ótima conversa com ele, meu tema foi escolhido e um esqueleto de trabalho criado e direcionado.

Agradeço ao professor Doutor Roberto Ramos, que se dispôs a avaliar meu trabalho, tanto administrativamente, mas também me direcionando para produzir um trabalho mais organizado, bem escrito e de melhor qualidade.

Agradeço por fim ao professor Doutor Luiz Celiberto, que me orientou e me ajudou sempre que eu precisei.

Resumo

O objetivo do projeto foi a criação de um carro robô capaz de detectar faixas que delimitam uma pista, identificar sua orientação como retas ou curvas, e utilizando uma inteligência artificial fuzzy, executar o controle da movimentação do robô de modo que ele se mantivesse dentro da pista.

O trabalho foi feito montando-se primeiro a estrutura física do robô, em seguida estudando-se seus sistemas isolados, destaque para o sistema de inteligência artificial fuzzy e para o sistema de visão computacional, testando-os e calibrando-os para cada situação possível com a qual o robô pudesse se deparar.

Os sistemas isolados apresentaram resultados satisfatórios, retornando respostas com valores exatos ou muito próximos do esperado. Quando dos testes do carro autônomo com todos os seus sistemas conectados, seu funcionamento em retas provou-se um sucesso. Contudo, foram observados resultados incorretos quando o robô se encontrava em curvas, não relacionados exatamente com a precisão dos movimentos calculados, mas com o momento de execução desses movimentos.

Palavras chave: Robótica, Carro Autônomo, Inteligência Artificial, Lógica Fuzzy, Visão Computacional

Abstract

The objective of the project was the creation of a robot car capable of detecting lanes that bound a track, identify their orientation as straight or curves, and using a fuzzy artificial intelligence, execute the control of the movement of the robot in a way that it keeps itself within the tracks.

The work was done assembling first the physical structure of the robot, next studying its systems isolated, highlighting the fuzzy artificial intelligence system and the computer vision system, testing them and calibrating them for every possible situation with which the robot could come across.

The isolated systems presented satisfactory results, returning responses with exact values or very close to the ones expected. When during the tests of the autonomous car with all its systems connected, its operation in straights has proven itself a success. However, it was observed incorrect results when the robot found itself in curves, not related exactly with the precision of the calculated moves, but with the moment of execution of these moves.

Keywords: Robotics, Autonomous Car, Artificial Intelligence, Fuzzy Logic, Computer Vision

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ISO	International Organization for Standardization
IFR	International Federation of Robotics
ABS	Anti-lock Braking System (Sistema de frenagem anti-travamento)
km	Quilômetro
CR	Centro de rotação
ml	Mililitro
OpenCV	OpenCV - Open Computer Vision
ROI	Region of interest (Região de interesse)
DC	Direct current (Corrente contínua)
DNIT	Departamento Nacional de Infraestrutura de Transportes
cm	Centímetro
LED	Light Emitting Diode (Diodo emissor de luz)
MotorD	Conjunto dos motores à direita do robô
MotorE	Conjunto dos motores à esquerda do robô
ms	Milisegundo

1. Introdução

1.1. Definindo um robô

Robótica é a área do conhecimento destinada ao estudo de robôs, suas estruturas e construção, programação e funcionamento.

Definir o robô de forma absoluta é uma tarefa um pouco mais complexa, uma vez que existem diversas definições de o que é um robô e elas evoluem conforme o modo como a tecnologia e nossa percepção sobre o assunto evoluem[1]. Algumas definições são mais abertas e generalistas, enquanto outras definições são bem restritivas.

A definição de robô se torna menos restritiva, e muitas vezes menos abrangente, sob a ótica do público leigo em geral. Uma visão mais técnica pode descrever apenas tipos específicos de robôs. Toma-se como exemplo as definições a seguir:

1. Enciclopédia Larousse Cultural: “Aparelho automático capaz de manipular objetos ou de executar operações segundo um programa fixo, modificável ou adaptável.”[2];

2. International Organization for Standardization (ISO):

- Robô industrial: “manipulador automaticamente controlado, reprogramável, multifuncional, programável em três ou mais eixos, que pode ser fixo no lugar ou móvel para uso em aplicações de automação industrial”[3] (Tradução livre)[1];
- Robô de serviço: “robô que executa tarefas úteis para humanos ou equipamento excetuando-se aplicações de automação industrial”[3] (Tradução livre)[2].

Observando o primeiro exemplo, da Enciclopédia Larousse, segundo estas definições, entende-se que o robô precisa ser automático e funciona segundo um programa, mas não é especificado como o controle do robô acontece de forma que

1 “automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications”[3]

2 “robot that performs useful tasks for humans or equipment excluding industrial automation applications”[3]

ele consiga realizar alguma tarefa de forma organizada ou precisa. As definições oferecidas pela ISO são utilizadas por diversas instituições, como a International Federation of Robotics (IFR)[4], por exemplo. Segundo a ISO, há uma diferenciação entre as definições de robôs industriais e robôs de serviços. A ambos é necessário um nível de automação mínimo para realização de tarefas, através do uso de sensores segundo um programa. No entanto, há especificidades para cada tipo de robô descrito e não um conceito unificado de robô.

Com as definições anteriores em mente, uma definição completa e unificada do que é um robô é dada por Maja J. Matarić[1]:

"Um robô é um sistema autônomo que existe no mundo físico, pode sentir o seu ambiente e pode agir sobre ele para alcançar alguns objetivos."

1.2. O carro autônomo

Partindo da definição de robô segundo Matarić[1], a expressão "robô autônomo" pode ser considerada redundante. Se um robô não for autônomo, ele será apenas uma máquina operada por alguém ou outro dispositivo mecatrônico similar a um robô, ou alguma máquina do gênero.

A palavra "autônomo" fica melhor empregada quando associada justamente a máquinas guiadas por seres humanos, quando estas estão sendo estudadas de forma a conferi-las a possibilidade de funcionamento independente de um controle humano direto. Podemos citar empilhadeiras autônomas, um sistema de controle autônomo, ou uma casa autônoma, como exemplos. Automaticamente, essas máquinas e sistemas passam a poder ser classificadas como robôs, uma vez que funcionam de forma independente a partir de sensoriamento e atuação no ambiente.

Os carros representam um dos grandes desafios de automação da atualidade. Muitos esforços têm sido realizados pela indústria automotiva e empresas do ramo da tecnologia para criação e aumento da confiabilidade de carros autônomos[5].

Atualmente, existem diversas dificuldades à implementação de um carro autônomo de uma forma completamente confiável e segura[6]. A parte física do controle de movimento, ambos velocidade e direção, é possível e já apresenta uma performance robusta e confiável. O maior problema com relação à criação de um carro autônomo está na parte de software: desde o sensoriamento do ambiente à inteligência do robô quando aplicada em situações reais.

Os sensores mais utilizados, como câmeras, radares e sensores ultrassônicos, estão sujeitos a severas variações em suas leituras de acordo com as condições do ambiente[6]. O tratamento de imagem e dos dados lidos pelos sensores são afetados quando chove, ou quando neva, por exemplo. Isso dificulta a detecção de pista e elementos relevantes, pois mesmo com os sensores sendo calibrados quando se sai com o carro, conforme o clima muda no mesmo dia, as leituras e dados também serão afetados[6].

Além dos problemas relacionados com a captação das imagens por parte de câmeras, as próprias condições da paisagem podem ser um empecilho. A detecção de faixas de rolamento, por exemplo, é muito prejudicada em vias com baixa infraestrutura, onde podem não haver faixas pintadas ou limites de pista claramente demarcados. Em estradas de terra, ou vias com calçamento de pedra, esse problema pode ser aumentado, uma vez que o carro terá maiores dificuldades em identificar um padrão para movimentação adequada na pista em questão.

A inteligência artificial presente no carro também é objeto de estudo e pesquisa direcionados à melhoria de seu desempenho e robustez. Situações adversas e inusitadas que podem acontecer em circunstâncias reais são testes alvo para essa constante melhoria. O encontro do carro robô com outros motoristas em condições como: embriagados, distraídos com celular, sonolentos, ou que simplesmente não respeitam regras e sinalização de trânsito de um modo geral, apresentam e representam comportamentos inesperados à autonomia segura de um carro robô. Veículos autorizados a burlar as leis em situações especiais (no geral, veículos de polícia, ambulância e bombeiros) também são uma dificuldade na implementação da programação dos carros robô[6].

A inteligência é muitas vezes aplicada em apenas algumas áreas do carro, de forma a se otimizar cada parte antes de unir todo esse trabalho em um componente único que controle o carro todo[5]. Por exemplo: na captura e tratamento da imagem oriunda das câmeras, para que esta se adapte da melhor forma às condições do ambiente; no controle de frenagem, como nos sistemas ABS (do inglês, Anti-lock Braking System)[7]. Eventualmente, essa tecnologia é reunida em um carro autônomo e temos exemplos de sucesso como o carro autônomo do Google[8], que já ultrapassou a marca de 3 milhões de km rodados.

Mais ainda, dentre as coisas que se busca melhoria com relação à inteligência, dependendo da técnica utilizada, a inteligência pode precisar de muito

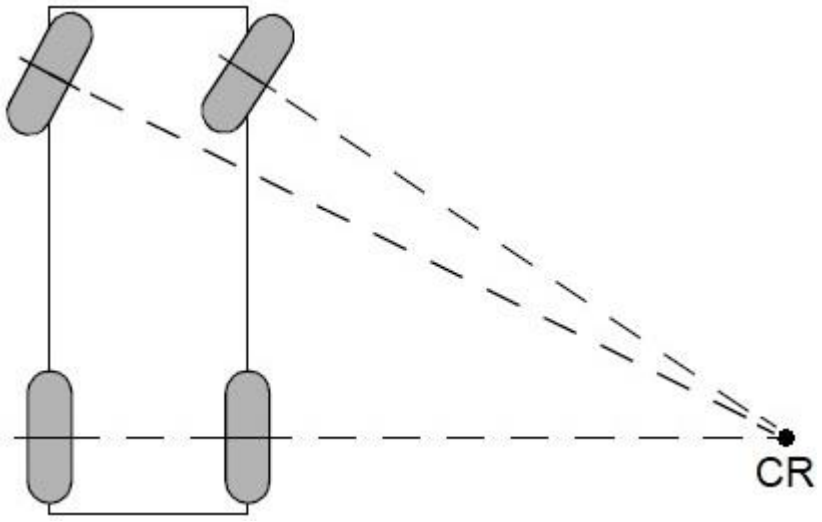
tempo de treinamento antes de ser capaz de guiar o carro completamente sozinha. Busca-se, portanto, um menor tempo de treinamento, além também de um melhor tempo de processamento e resposta nas tomadas de decisão do carro, além de precisão e exatidão no planejamento de rotas e controle dos sistemas do carro para correta navegação, especialmente em situações adversas de emergência ou que exijam uma tomada de decisão rápida.

1.3. Tipos de direção

Um carro autônomo, assim como qualquer veículo com rodas, precisa de um sistema de direção. Os tipos de direção variam com o número e tipo de rodas utilizadas, e também com a forma com que esse conjunto de rodas interage com a superfície por onde ela se locomove. O tipo de direção mais comum aplicado a veículos terrestres é a direção de Ackermann.

Na direção de Ackermann[9], não importa quantas rodas o veículo possua, ele sempre terá apenas um centro de rotação. O centro de rotação (CR) do veículo é determinado como o ponto de encontro de todas as linhas perpendiculares ao centro de cada roda do veículo, como mostrado na Figura 1.

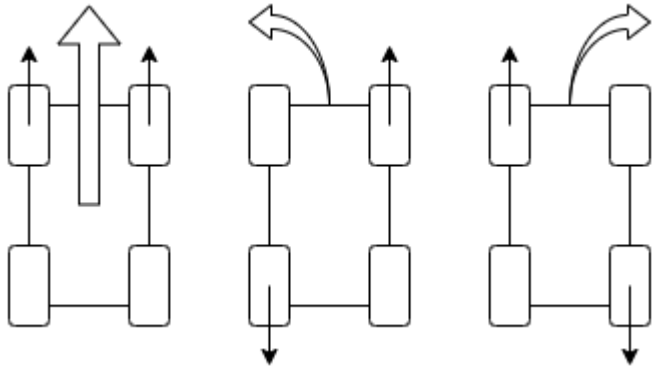
Figura 1. Esquema de direção de Ackermann[9]



Outro dos tipos de direção, muito utilizado para projetos de carros robô pela simplicidade de montagem e funcionamento do sistema quando comparado à direção de Ackermann, é a direção diferencial. Na direção diferencial o veículo possui motores em suas laterais esquerda e direita e o controle da direção é feito a

partir de uma diferença na velocidade dos motores presentes em cada lateral. Um esquemático da direção diferencial aplicada em um carro robô é apresentada na Figura 2.

Figura 2. Direção diferencial aplicada em robô



Portanto, para movimentação em linha reta, o veículo precisa ativar ambos os motores com velocidades iguais. De outra forma, ele viraria: caso o veículo tenha as rodas da direita em velocidade maior que as rodas da esquerda, ele virará para a esquerda; para o caso de as rodas da esquerda estarem mais rápidas que as da direita, o veículo fará uma curva à direita.

O veículo diferencial é analisado de forma segmentada. Inicialmente é estudado a velocidade de cada roda e o efeito causado por essa velocidade sobre o veículo como um todo. Após o estudo de cada roda, temos a soma dos efeitos de cada roda para determinação da movimentação do veículo diferencial. As fórmulas que regem a movimentação de um veículo com sistema de direção diferencial estão apresentadas abaixo:

$$v_e = r\omega_e \text{ (1)} \qquad \text{e} \qquad v_d = r\omega_d \text{ (2)}$$

$$V = \frac{v_e + v_d}{2} \text{ (3)} \qquad \text{e} \qquad \omega\ \frac{v_d - v_e}{L} \text{ (4)}$$

Em (1) e (2) calcula-se as velocidade das rodas, esquerda e direita. As velocidades das rodas esquerda e direita são representadas, respectivamente, por v_e e v_d. O raio das rodas é representado por r e a velocidade angular de cada roda por ω_e e ω_d. A velocidade do veículo é dada como a média da soma das velocidades

de cada roda, e na equação (3) acima é representada por V, enquanto que ω, na equação (4), representa a velocidade angular do veículo.

1.4. A inteligência artificial

Inteligência artificial é uma área de estudo e pesquisa da computação que tem por objetivo a produção de programas "inteligentes". A intenção da palavra "inteligente" aqui oferece objetivos diferentes a cada tipo de programa em questão, dependendo dos problemas e óticas utilizadas para analisá-los. Como citado por Russell e Norvig[10], programas inteligentes podem ser criados para pensar problemas ou agir sobre eles de forma racional, analisando fisicamente, matematicamente ou estatisticamente, ou ainda simulando processos ou comportamentos biológico, como o de seres humanos, animais ou outros tipos de seres vivos.

Sistemas inteligentes são aplicados entre outras coisas para otimização de processos, resolução de problemas, tomadas lógicas de decisão, criação de previsões e estimações. Alguns desses sistemas trabalham em problemas dinâmicos adaptando-se às alterações inerentes a esses problemas e agindo conforme seu "aprendizado". Essa adaptação pode ocorrer, por exemplo, através do histórico de interações entre a inteligência e o problema, e as tomadas de decisão e resultado dessas decisões no cenário geral do problema.

Os sistemas inteligentes podem ser completamente virtuais ou utilizados em implementações no mundo físico, podem ser centralizados ou distribuídos[11], indutivos, dedutivos ou abdutivos[12], supervisionados, semi-supervisionados ou não supervisionados[13], entre outras classificações possíveis.

Existem diferentes técnicas para se implementar uma inteligência artificial. Destacam-se, por exemplo, entre as mais estudadas e utilizadas o aprendizado por reforço, redes neurais, algoritmos genéticos e lógica fuzzy. Cada uma dessas técnicas busca tratar um problema de diferentes formas: três dessas técnicas são baseados à forma que sistemas e seres biológicos tratam esses problemas, enquanto que o outro pode ser visto como semelhante à forma que aprendemos quando recebemos retroalimentações de nossas ações.

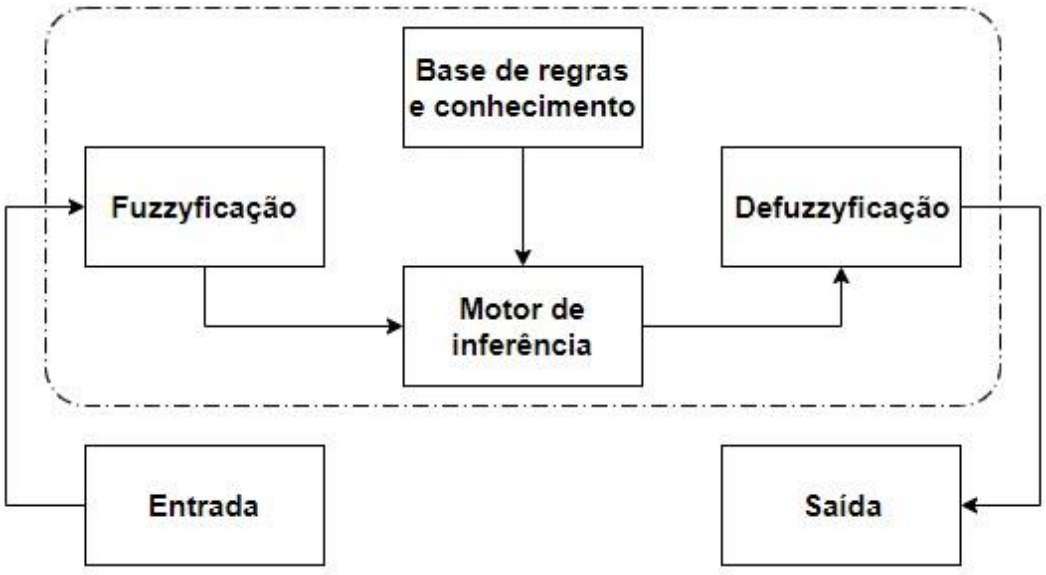
Redes neurais simulam um cérebro humano, desde a estrutura até a forma com que elas lidam com determinadas informações[10]. Algoritmos genéticos

analisam o problema de forma análoga ao modo como a evolução de espécies ocorre através de gerações, mantendo como modelo das próximas gerações sempre as soluções (ou indivíduos) mais adequadas e aplicando cruzamentos e mutações a cada nova geração, para que se possa ver algo novo que possa melhorar o modelo utilizado anteriormente[14]. Aprendizagem por reforço deve fornecer ao programa condições de analisar se o resultado de suas ações foi bom ou ruim, baseado no reforço recebido. Assim o programa aprende a agir da melhor maneira conforme ele passa por experiências de aprendizado que oferecem recompensa, seja ela boa ou ruim[10]. Por fim, a lógica fuzzy se baseia na valoração de conceitos de definição imprecisa, comumente interpretada de forma tão natural e diferenciada por seres humanos.

A lógica fuzzy é um tipo de inteligência artificial que incorpora a forma humana de pensar, procurando tratar conceitos imprecisos, vagos ou qualitativos, em tomadas lógicas de decisão, parecido com como os humanos tomam decisões quando deparados com problemas desse mesmo tipo[15]. Dessa incerteza também vem o significado de "fuzzy", do inglês "nebuloso" ou "difuso"[16].

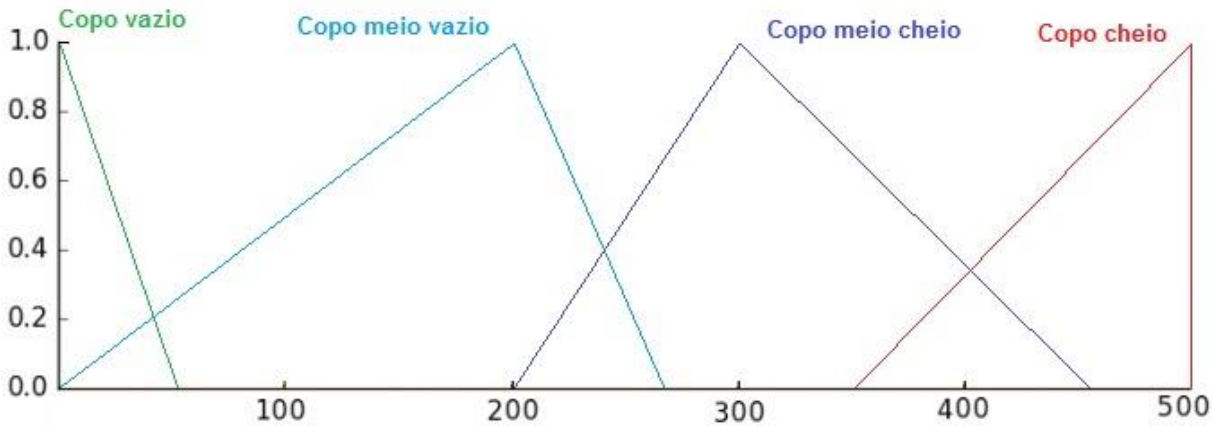
Um controlador lógico fuzzy é um sistema de controle inteligente que usa a lógica fuzzy. Funcionando como um sistema de controle, esse sistema recebe uma entrada, analisa e controla essa entrada, e retorna uma saída, onde não necessariamente entradas e saídas são únicas. Uma malha de controle exemplificando esse sistema é apresentada na Figura 3 abaixo.

Figura 3. Sistema de controle com lógica fuzzy



O conceito primordial da lógica fuzzy é aplicada no tratamento dos sinais de entrada e saída, uma vez que ambos não consistem obrigatoriamente de informações definidas e precisas. É realizada então uma operação de fuzzyficação, como mostrado na Figura 3, para transformar as variáveis de entrada de valores bem definidos em valores fuzzy. Essa transformação de variáveis é realizada através de uma função de pertinência. A função de pertinência recebe os valores nítidos e os relativiza em relação aos termos fuzzy escolhidos, mapeando-os em valores fuzzy entre 0 e 1[17]. Esses termos são comumente adjetivos, como "vazio" e "cheio", ou advérbios, como "talvez", "quase", "nunca" e "sempre". Cada termo possui uma função de pertinência. Um exemplo de função de pertinência para uma função simples é mostrada na Figura 4.

Figura 4. Função de fuzzyficação: copo cheio, copo vazio



Na imagem, a entrada da função de pertinência é nítida e bem definida: os mililitros contidos em um copo de 500 mL que será avaliado em vazio ou cheio por um controlador fuzzy.

Com os valores fuzzyficados, o motor de inferência realiza o controle a partir da entrada em conjunto com a base de regras e conhecimento pré-programados na inteligência. As regras fuzzy relacionam as variáveis fuzzy umas com as outras. Esse conjunto de regras fuzzy é dado em declarações "IF-THEN" (do inglês, "se-então"). Isto quer dizer que as regras relacionam diretamente variáveis de entrada e variáveis de saída: as variáveis relacionadas à parte "IF" da regra são as variáveis de entrada, ou antecedentes; as variáveis vinculadas à parte "THEN" são as variáveis de saída, ou consequentes.

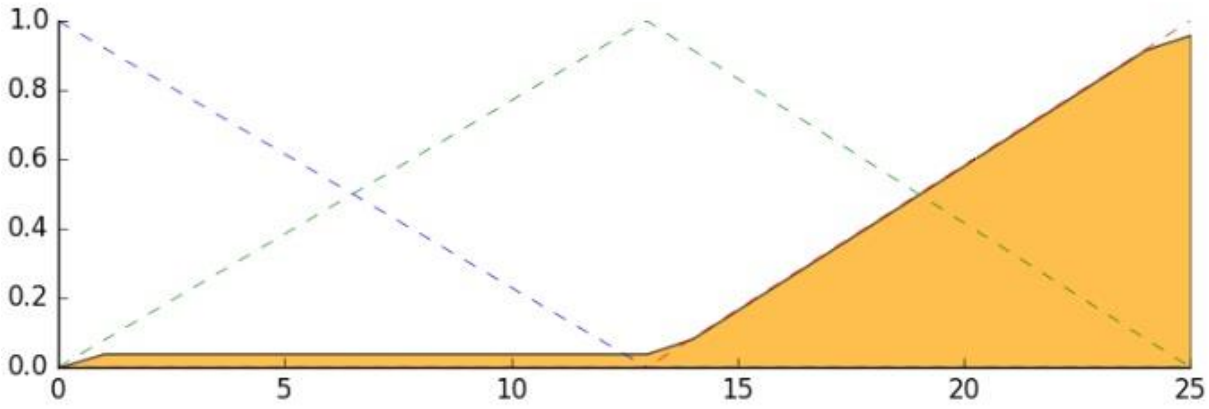
Por exemplo, para um controlador fuzzy relacionando o ritmo do carro com a posição de um carro mais à frente poderíamos ter as seguintes regras:

- Se "carro da frente" = "longe", então "ritmo" = "acelerado";
- Se "carro da frente" = "perto", então "ritmo" = "cauteloso".

Neste conjunto de regras, "carro da frente" é a variável de entrada e "longe" e "perto" são os termos fuzzy correspondentes. "acelerado" e "cauteloso" são os termos fuzzy atrelados à variável de saída, "ritmo".

Após a computação das regras, uma (ou mais) saída é calculada com base nos resultados das aplicações das regras. É realizada então a defuzzyficação da variável de saída fuzzy encontrada, sendo então novamente o valor traduzido para um valor nítido e bem definido, como mostrado na Figura 5 abaixo com o recorrente problema da gorjeta.

Figura 5. Problema da gorjeta[17]



A função de pertinência da saída relaciona as variáveis fuzzy que vão, como de costume, de 0 a 1. No exemplo da figura, a gorjeta baixa fica representada por valores entre 0% e 13%, a gorjeta média entre 0% e 25% e a gorjeta alta entre 13% e 25%. No exemplo da Figura 5, as regras já foram computadas e uma amostra do resultado do controle fuzzy por meio do motor de inferência já é apresentado. A partir desse resultado, faz-se a defuzzyficação do controle da inteligência fuzzy para encontrar a saída do sistema.

O processo do cálculo da variável defuzzyficada pode ser realizado com a relação matemática que o usuário desejar, sendo as formas mais comuns de cálculo da função de pertinência resultada na saída do controlador fuzzy: o cálculo do centroide, escolha do primeiro ou do último dos máximos, ou cálculo da média dos máximo ou ainda escolha do ponto máximo.

1.5. Visão computacional

Visão computacional é um campo da computação que tem por finalidade o tratamento de uma imagem, extração de informações e percepção da imagem captada através de técnicas de computação[18]. Essas informações são obtidas através de técnicas de tratamento de imagem e processamento de dados obtidos da imagem original ou tratada. Com visão computacional é possível o reconhecimento, detecção e identificação de objetos e elementos de imagem, como pessoas, animais ou veículos, por exemplo. Além disso é possível estimar e calcular grandezas físicas e outros dados relacionados aos elementos presentes na imagem, como a distância e velocidade de um carro a frente ou a trajetória de uma bola lançada ao ar.

Um sistema de visão computacional consiste de uma ou mais câmeras para captação da imagem e algoritmos de tratamento de imagem para detecção e verificação de elementos na imagem que são pertinentes à análise sendo feita. O sistema de visão também pode ser usado para execução de estimativas e medidas a partir da imagem tratada, além da possibilidade de ser usada como interface gráfica entre o usuário e o sistema.

Nos carros autônomos, o sistema de visão computacional é geralmente a maior fonte de informação de sensoriamento de entrada para controle do carro. Com o uso de um sistema de visão computacional podem ser extraídos da imagem os limites da pista, faixas de rolamento, obstáculos na pista (como pedestres, ciclistas ou outros veículos), sinalização, entre outras informações que podem ajudar a tornar os sistemas de direção autônomos cada vez mais confiáveis e robustos.

No trabalho foi utilizada a biblioteca OpenCV, sigla de Open Source Computer Vision[19], da qual foram utilizadas diferentes formas para tratar a imagem antes da detecção da pista. A detecção de pista foi feita utilizando o algoritmo de detecção de bordas de Canny, que tem como saída uma imagem binarizada, contendo as bordas na imagem em pixels não-nulos. O algoritmo de Canny foi aplicado à imagem

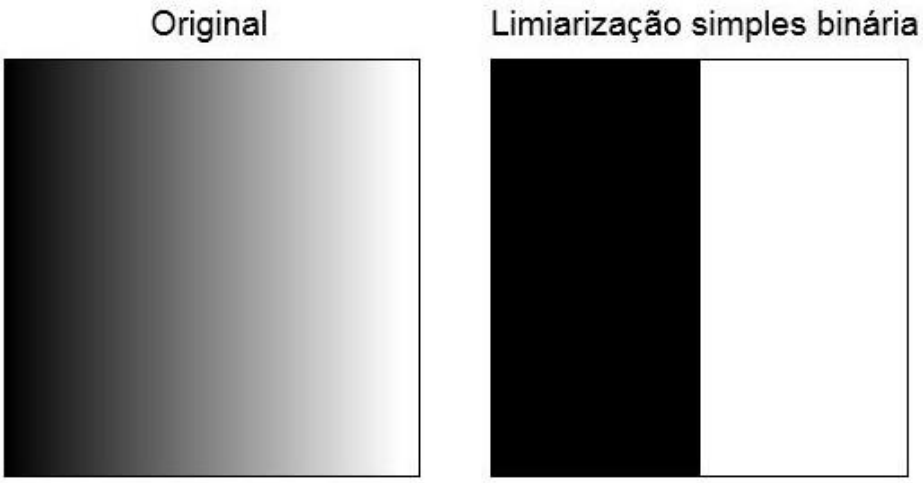
original captada pela câmera, sua versão em preto e branco e com filtros de limiarização, um simples e um filtro adaptativo de Gauss. O princípio de funcionamento destes algoritmos está explicado a seguir:

- Limiarização:

No processo de limiarização, uma imagem é comparada, pixel a pixel, a um limite definido de intensidade de luz (escala de cinza na imagem). Se o pixel tiver valor maior que o limite, receberá uma cor; caso seja menor que o limite estabelecido, o pixel receberá outra cor. Geralmente, as cores escolhidas para valores menores e maiores que o valor de limiar são respectivamente preto (bit 0) e branco (bit 1). A entrada de um processo de limiarização é uma imagem em preto e branco, de forma que a intensidade de luz analisada corresponde ao valor dos pixels dentro de uma escala de cinza (geralmente indo de 0 a 255 bits). Como saída, geralmente tem-se imagens binarizadas, com todos os pixels da imagens tomando valores de bits iguais a 0 ou 1. Entretanto, também é possível outros tipos de limiarizações, com saída também em escala de cinza, mas com alguma segmentação de elementos da imagem. Dentro dos algoritmos utilizados neste trabalho, foram avaliados os algoritmos de limiarização simples e de limiarização adaptativa de Gauss.

O processo de limiarização simples é o descrito acima. Imagem preto e branca em escala de cinza na entrada é comparada a um limiar que transformará a imagem em valores binários correspondentes às cores preta e branca. Esse tipo de limiarização é também chamada de limiarização global[20]. A função de chamada em Python para utilização da limiarização simples em OpenCV é cv2.threshold (in1, in2, in3, in4), onde o primeiro argumento é a imagem de entrada, o segundo argumento é o valor de limiar usado para comparação dos valores de intensidade de pixels, o terceiro argumento é o valor que o pixel receberá caso ultrapasse o valor de limiar, e o quarto argumento é o estilo de limiarização. A chamada da função deve ser feita com duas variáveis que receberão a saída da função, uma vez que a saída da função de limiarização simples é uma tupla, ou seja, um objeto que guarda um conjunto de objetos (no caso, dois objetos). Um exemplo de limiarização simples é mostrada na Figura 6 abaixo:

Figura 6. Exemplo de limiarização de imagem

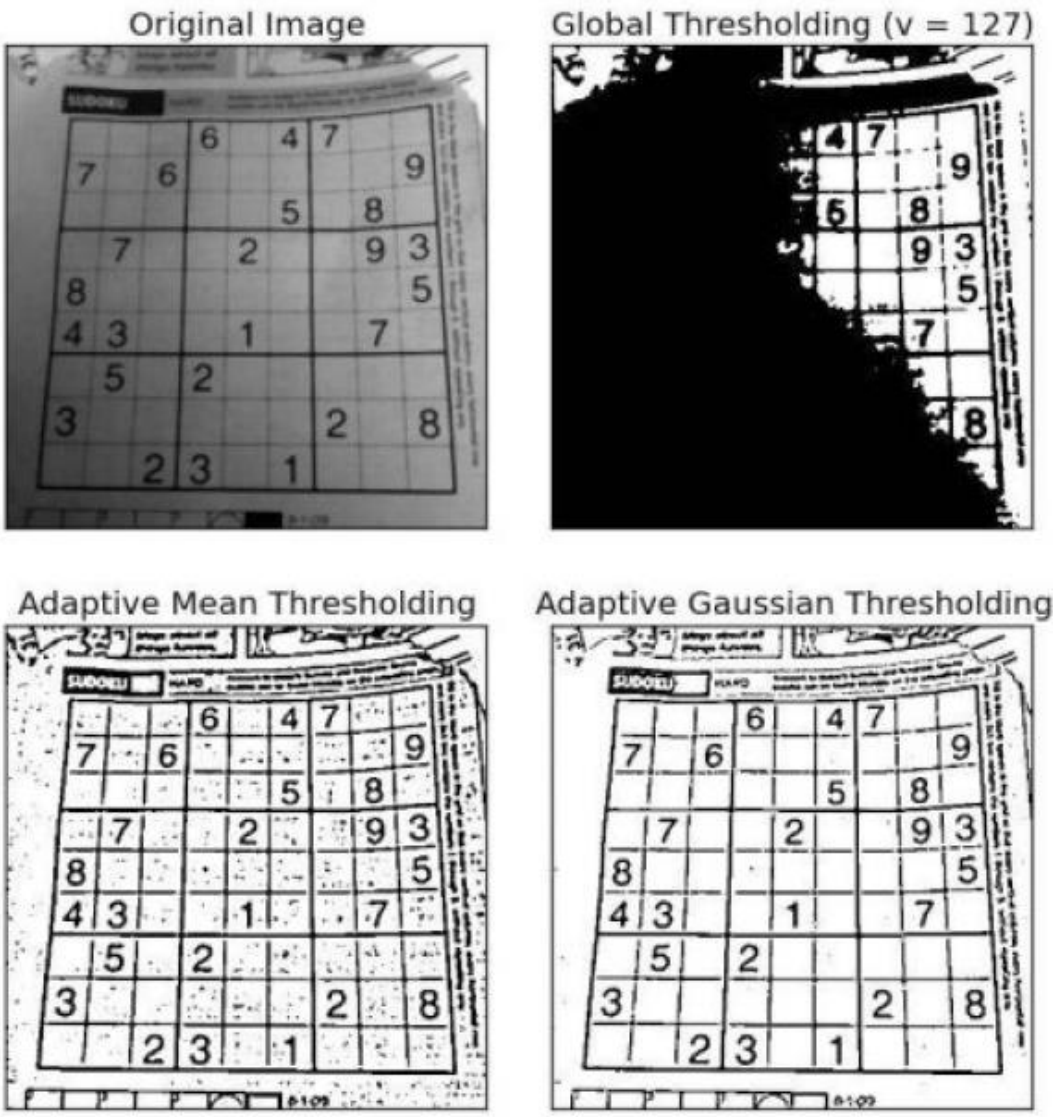


Algoritmos de limiarização adaptativa, por sua vez, levam em consideração a região do pixel analisado. Isso quer dizer que no lugar de um valor de limiar ser utilizado para a imagem como um todo, a cada pixel analisado, o valor de limiar é recalculado de acordo com a área em volta desse pixel. Essa técnica é utilizada para transmitir à imagem efeitos sofridos por ela na captura, principalmente a iluminação do ambiente. Geralmente o valor de limiar é calculado como a média da região do pixel analisado. O tamanho dessa região é definida pelo usuário e afeta diretamente o valor de limiar calculado pelo algoritmo.

A limiarização adaptativa de Gauss atua de forma um pouco diferente. Neste algoritmo, o valor de limiar é recalculado utilizando-se uma média ponderada da região do pixel, onde os pesos de ponderação são retirados de um filtro de Gauss.

A função de chamada da limiarização adaptativa é comum a ambos os modos de limiarizar uma imagem. A função e Python é cv2.adaptiveThreshold (in1, in2, in3, in4, in5). O primeiro argumento é a imagem de entrada; o segundo é o valor que o pixel que ultrapassa o valor de limiar receberá; o terceiro argumento é o método adaptativo; o quarto é o tamanho da área de pixels relativa ao pixel analisado; por fim, o quinto é o valor de uma constante que será subtraída do valor de limiar médio calculado. A diferença entre a limiarização adaptativa e a limiarização adaptativa de Gauss está no terceiro argumento, onde só se troca o método de limiarização adaptativa. Abaixo é mostrado um exemplo comparativo entre a limiarização simples global, a limiarização adaptativa e a adaptativa de Gauss.

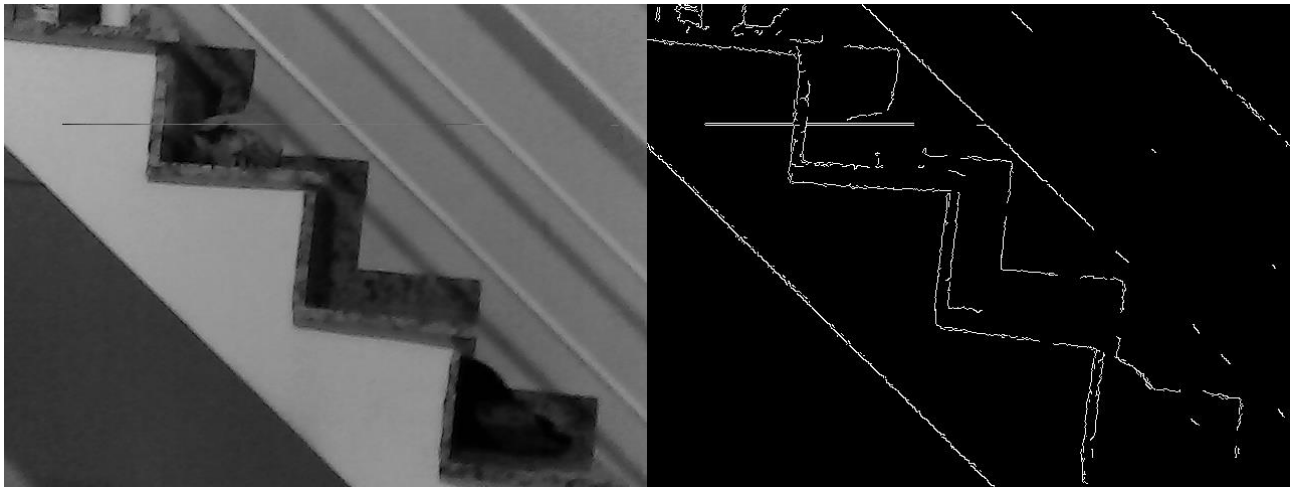
Figura 7. Comparação entre alguns tipos de limiarização[20]



- Algoritmo de detecção de bordas de Canny:

O algoritmo de detecção de bordas de Canny funciona seguindo alguns passos. O princípio do algoritmo é de procurar nas imagens uma diferença significativa na intensidade das cores de pixels subsequentes ou regiões de pixels subsequentes. O algoritmo destaca essas diferenças como uma borda, a divisão na imagem entre um objeto e outro. A entrada do algoritmo não exige um tipo de imagem específica. A saída, no entanto, será sempre uma imagem binarizada. Um exemplo da aplicação do algoritmo de detecção de bordas de Canny pode ser visto na Figura 8 abaixo.

Figura 8. Detecção de bordas com o algoritmo de Canny



Para isso, o algoritmo realiza primeiro uma suavização da imagem para redução de ruídos presentes na imagem. Na sequência, o algoritmo calcula as intensidades de gradientes da imagem, onde os gradientes são perpendiculares às bordas contidas na imagem. Depois, é feita uma supressão não-máxima da imagem. Neste passo, o pixel da borda é comparado aos pixels da vizinhança para verificar se o pixel selecionado representa um máximo local. Caso ele não seja, ele é descartado e transformado em um pixel preto (bit 0). Os pixels selecionados como máximos locais são filtrados por um limiar de histerese. Esse limiar toma dois valores como limites de comparação. O valor máximo determina valores que são definitivamente bordas na imagem. Caso o valor da intensidade do pixel seja maior que o valor máximo, aquele pixel será indicado como uma borda. O valor mínimo limita quais valores podem ser considerados bordas. Se um pixel tiver valor menor que o valor mínimo, ele será descartado como borda da imagem. Para o caso de o pixel se encontrar entre os valores máximo e mínimo, ele será considerado borda caso esteja ligado a pixels acima do valor máximo, direta ou indiretamente. Caso esteja conectado apenas a valores entre os valores máximo e mínimo, ou ligado apenas a valores menores que o valor mínimo, ele será descartado como borda.

Os pixels classificados como borda terão seus valores alterados para pixel branco (bit 1). Os valores descartados serão pixels pretos (bit 0) e a saída da imagem será assim uma imagem binarizada.

A função disponível para utilização do algoritmo de Canny em Python é cv2.Canny (in1, in2, in3), onde o primeiro argumento é a imagem de entrada, o segundo argumento é o valor mínimo e o terceiro é o valor máximo que a função utiliza para determinar o que é e o que não é borda.

Com a imagem binarizada conseguida através do algoritmo de Canny, recorta-se na imagem uma área de interesse, onde realiza-se uma análise detalhada dos elementos da imagem. Essa área de interesse é conhecida como ROI (Region of interest, do inglês). Como a ROI é geralmente uma parte da imagem, os algoritmos de tratamento de imagem aplicados em uma ROI tem um tempo de execução menor do que teriam se fossem aplicados na imagem completa.

- Transformação de Hough:

Utilizou-se a transformação probabilística de Hough após destacar-se a ROI da imagem capturada pela câmera. A transformação de Hough é uma função usada para detecção de linhas na imagem. Com essa função é possível encontrar linhas que atendem a requisitos de tamanho do usuário, que estão em algum ponto de interesse. Esta função também pode ser usada para eliminar segmentos de reta ou conjuntos de pixels isolados que sejam ruído não desejado na imagem.

A transformação de Hough busca representar as linhas capturas da imagem como pontos parametrizados. Em uma imagem de entrada qualquer, assim como enxergamos, pontos são pontos, e linhas são linhas. Já no espaço parametrizado de Hough, linhas são transformadas em pontos e uma linha representa um ponto na imagem capturada. A equação que rege esta transformação é apresentada abaixo:

$$\rho = x\,cos\theta + y\,sen\theta \quad (5)$$

Na equação acima, ρ é a distância perpendicular da reta à origem do plano parametrizado, e θ representa o ângulo formado entre a reta perpendicular e o eixo horizontal, no sentido anti-horário.

Dessa forma, para encontrar linhas grandes o suficiente na imagem capturada, linhas que representem as faixas de rolagem, linhas que atravessam o maior número possível de pixels, procuramos no espaço de Hough por um ponto por onde passam o máximo possível de linhas. Cada linha atravessando um ponto

corresponde a um voto. Quanto mais votos um ponto tiver, maior é o tamanho da linha que ele representa.

A função de chamada da transformação de Hough em Python é cv2.HoughLines (in1, in2, in3, in4), onde os parâmetros de entrada são, na ordem: a imagem binarizada a ser analisada; a precisão de ρ e θ como segundo e terceiro argumentos, respectivamente; e um limiar representando o número mínimo de votos. Escolhendo um número mínimo de votos, determina-se um tamanho mínimo de linha, excluindo linhas menores que possam representar ruído na imagem binarizada.

A função retorna um vetor com pontos (ρ, θ) no plano parametrizado que atendem aos parâmetros passados na chamada da função. Os pontos (ρ, θ) no vetor de saída correspondem cada um a uma reta na imagem capturada.

No entanto, a função utilizada foi a transformação probabilística de Hough, que é uma otimização da transformação original. A primeira diferença desta função é que ela não analisa todos os pontos parametrizados da imagem. Esta segunda função separa aleatoriamente um subgrupo de pontos suficiente para detecção de linhas e trabalha em cima deste subgrupo. A segunda diferença desta função é a saída: na transformação probabilística de Hough, a saída é dada como um vetor de quatro valores: as coordenadas x e y dos dois pontos extremos de cada linha detectada já nas coordenadas da imagem original fornecida como entrada.

A função utilizada para chamada da transformação probabilística de Hough em Python é cv2.HoughLinesP (in1, in2, in3, in4, in5, in6). Os quatro parâmetros de entrada iniciais são os mesmos da função de transformação de Hough. O quinto argumento é o comprimento mínimo da linha detectada e o sexto argumento é a separação máxima entre segmentos de linha para que eles sejam considerados como uma linha só. Recomenda-se a redução do valor de limiar, quarto argumento, em relação ao que seria utilizado na transformação original de Hough.

2. Objetivos

O objetivo do trabalho foi a criação de um carro robô capaz de detectar faixas de rodagem em uma pista, identificá-las como retas ou curvas, para a direita e para a esquerda, e executar um controle dos seus motores através de uma inteligência artificial fuzzy afim de manter-se dentro do limite das faixas da pista.

O trabalho consistiu desde a definição, o projeto e montagem da estrutura dos carros robô, até a programação de seus sistemas de inteligência artificial e visão computacional para controle de motores.

O trabalho também apresenta um estudo preliminar dos principais sistemas do carro robô: a inteligência artificial e o sistema de visão computacional. Nesses estudos, foram avaliados principalmente a precisão dos resultados de cada programa isolado e os tempos de execução dos programas.

3. Metodologia

A metodologia foi dividida apresentando em um primeiro momento a proposta e contextualização do trabalho para o projeto do carro robô, seguidos da apresentação dos principais materiais utilizados no trabalho e os cuidados com a montagem do carro e pista. Seguiu-se com a apresentação dos principais subsistemas do robô: os sistemas de visão computacional e inteligência artificial.

3.1. Proposta e contextualização do trabalho

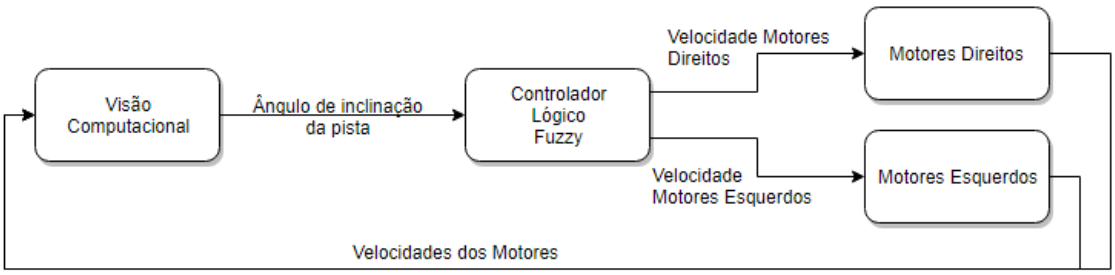
Como descrito brevemente na seção 2., o objetivo principal do trabalho foi a criação de um carro robô, cujas capacidades abrangeriam a detecção das faixas que delimitam uma pista e o controle de seus motores a partir de uma inteligência artificial para se manter entre as faixas detectadas. Essa é uma simplificação básica de um conceito que hoje tenta-se aplicar no mundo: o conceito do carro autônomo.

Impulsionados por incontáveis motivos, pesquisadores e empresas do mundo todo tentam criar um carro perfeitamente autônomo, que se guie em qualquer condição de tempo e pista, e que mesmo se encontrando nas situações mais adversas, que o carro sozinho seja autossuficiente ao lidar da melhor maneira possível nessas situações. As melhorias atreladas a um carro perfeitamente autônomo vão de um trânsito mais fluído até um menor número de acidentes, menos mortes, maior segurança, menos (ou nenhuma) infrações de trânsito, melhoria no atendimento de saúde pública[21] e menos gastos causados por acidentes[22], entre muitas outras.

O desafio de criação de um carro autônomo é acima de tudo um desafio de automação e robótica. A solução para este problema engloba o sensoriamento do ambiente para aquisição de dados, sobre os quais haverá um controle calculado através, no geral, de um sistema de controle ou uma inteligência artificial. Esse controle é transmitido aos motores e então o carro se movimenta.

O projeto apresentado neste trabalho se propôs a seguir uma versão simples desse modelo, contando com um sistema de visão computacional para o sensoriamento, uma inteligência artificial fuzzy como sistema de controle e um conjunto de motores como atuadores. O diagrama de blocos da Figura 9 ilustra a proposta de como o projeto será executado seguindo esse modelo.

Figura 9. Diagrama



Acompanhando o fluxo do diagrama, o começo do projeto acontece com o sistema de visão. O sistema de visão detecta a pista e identifica seu sentido como reta ou curva. O sentido para o qual a pista ruma é passado na saída do sistema de visão como a soma entre os ângulos de inclinação das faixas, considerando a inclinação para a esquerda como negativa e a direita como positiva.

Essa diferença entre as inclinações das faixas será o sinal de entrada da inteligência. Caso negativo, o controlador fuzzy gerará um controle dos motores para virar o carro robô para a esquerda. Caso positivo, o controle dos motores levará o carro à direita. Por fim, caso a entrada da inteligência artificial seja 0 (zero), a inteligência fuzzy comandará o carro a acelerar com velocidade máxima com ambos os motores.

Por fim, o controle encontrado pelo código fuzzy será repassado aos atuadores, para que enfim o carro se mova de acordo com a pista analisada.

Nas seções seguintes, será explicado em maiores detalhes o funcionamento de cada parte do diagrama da Figura 9.

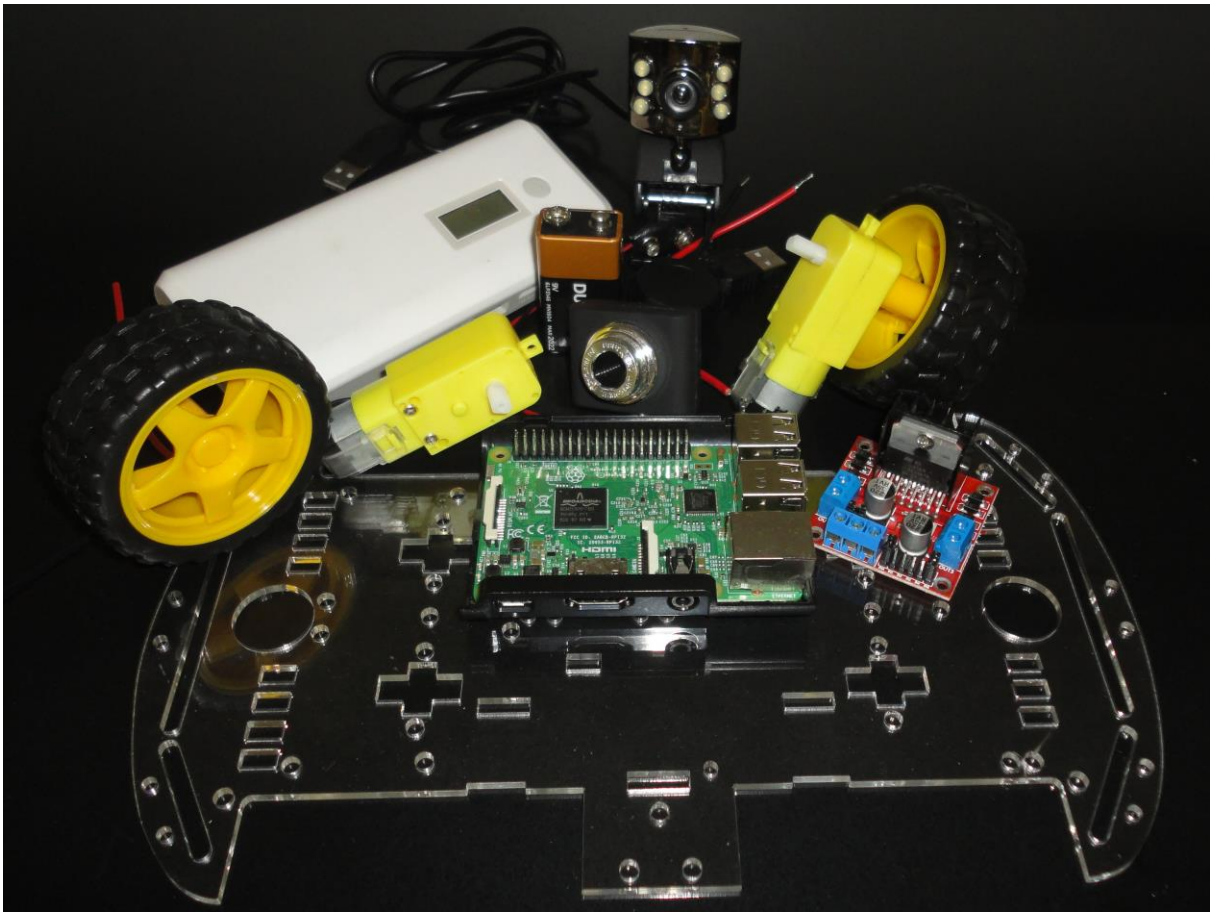
3.2. Lista de materiais, equipamentos e softwares utilizados

Os equipamentos utilizados estão dispostos a seguir, classificados de acordo com o sistema onde foram empregados. Abaixo, a Tabela 1 mostra os principais materiais utilizados no carro robô, com especificações e quantidades. Os elementos da Tabela 1 também são exibidos na Figura 10. A Tabela A1, no Anexo, apresenta a lista completa de materiais utilizados na montagem do carro robô.

Tabela 1. Relação dos principais materiais

Componente	Aplicação	Quantidade
Raspberry Pi 3 Modelo B	Sistema integrador/Inteligência Artificial	1
Motor DC 5V	Sistema de direção	4
Roda	Sistema de direção	4
Chassi acrílico	Estrutura do robô	2
Ponte H L298N	Sistema de direção	1
Câmera Multilaser WC040	Sistema de visão computacional	1

Figura 10. Materiais utilizados

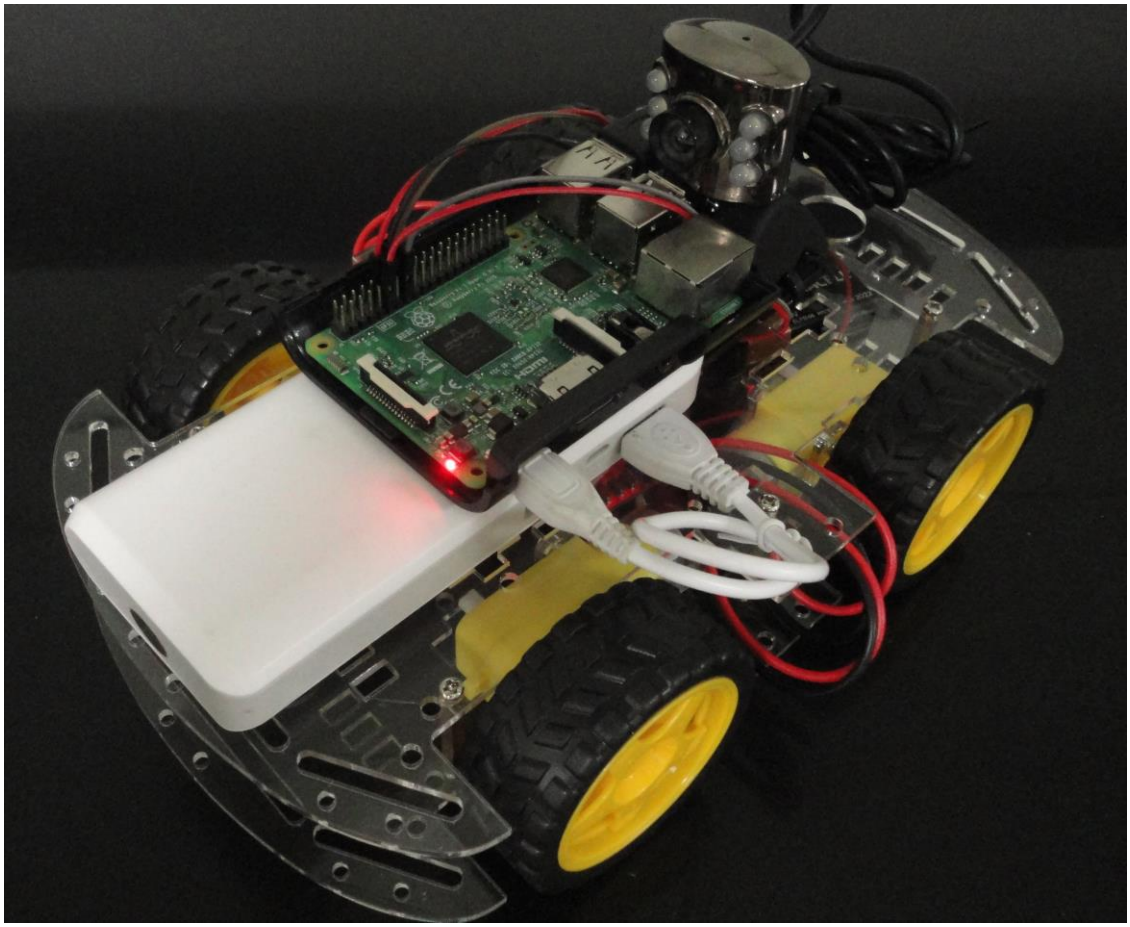


Na parte de software, foram estudados e utilizados a linguagem de programação Python, a biblioteca gpiozero, OpenCV, scikit-fuzzy e numpy. Na Tabela A2, no Anexo, estão especificados cada software e biblioteca - e suas respectivas versões - utilizados no trabalho.

3.3. Montagem do chassi e pista

O chassi utilizado foi escolhido entre duas opções: um chassi para robô diferencial com quatro motores e outro chassi com direção de Ackermann. O chassi diferencial foi comprado pronto, desmontado, enquanto que o chassi com direção de Ackermann foi projetado e fabricado. Optou-se pelo chassi diferencial para simplificar o comando da inteligência artificial, de forma que ela trabalhasse com apenas um tipo de motor. A Figura 11 mostra o carro robô montado em sua forma final.

Figura 11. Carro robô

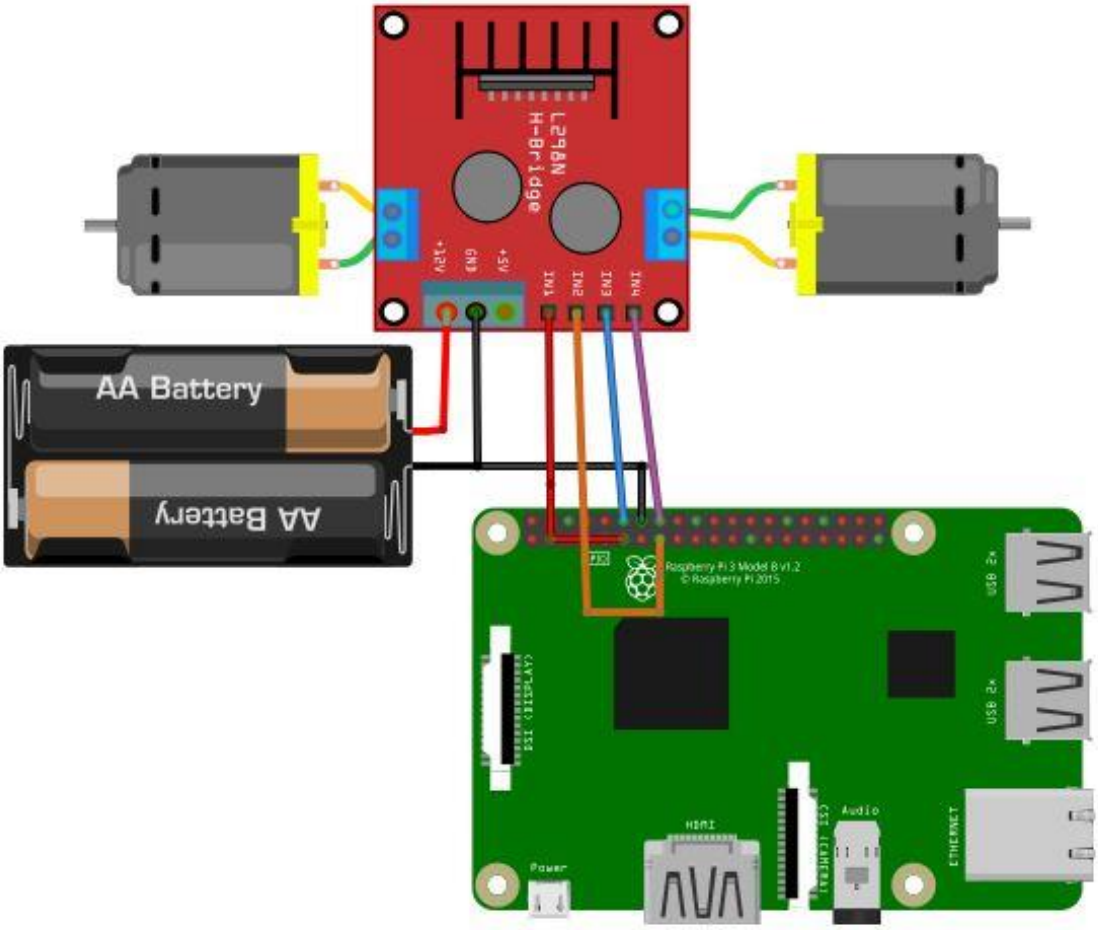


A montagem do chassi começou pelos motores. Após a soldagem dos fios dos motores, cada motor foi testado sendo ligado diretamente a uma bateria para verificação do funcionamento dos mesmos (determinar se havia algum mal contato, além da verificação do funcionamento do motor em si e sentido de sua rotação).

A seguir, o chassi foi montado. Foi necessário organizar os motores de cada lado de forma que quando os dois motores fossem ligados na mesma bateria, os dois rodassem na mesma direção.

O resto da montagem dos circuitos do carro robô seguiu o esquema apresentado na Figura 12. O Raspberry foi alimentado por uma bateria externa, não presente na Figura 12. A fonte de energia alimentando a ponte H L298N durante os testes independentes, fonte que alimenta os motores, foi uma fonte de 15 V ligada diretamente à rede elétrica normal (110 V). Quando dos testes do carro robô com todos os sistemas integrados, foi utilizada um conjunto de pilhas de 6 V totais como fonte para os motores.

Figura 12. Circuito do carro robô



A pista foi projetada tendo-se em mente pistas reais (mais especificamente Indianápolis e Daytona), escolhendo seu formato de forma que o carro pudesse

rodar de forma mais fluida. As faixas que delimitam a pista foram feitas com fita isolante preta, diretamente no chão. A Figura 13 mostra como ficou a pista vista de cima; a Figura 14 mostra a vista da câmera do robô.

Figura 13. Pista (Vista superior)

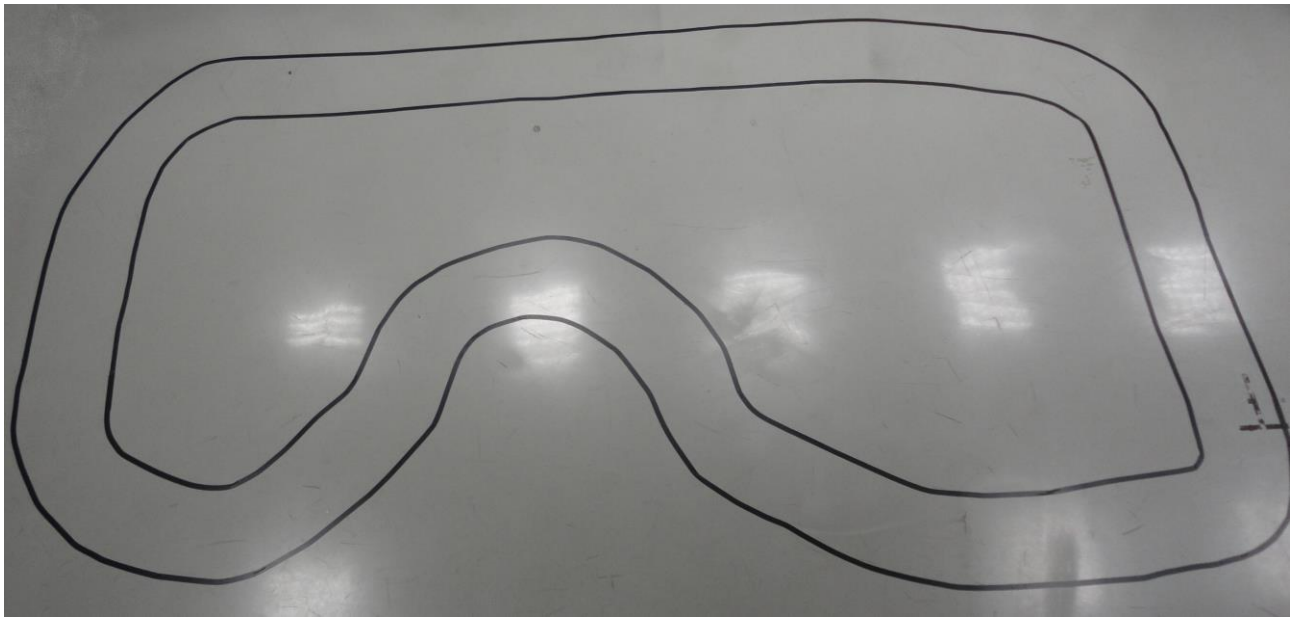


Figura 14. Pista (Vista do carro)

O projeto seguiu, em proporção, as normas de projeto de estradas do DNIT[23] para determinação da largura da pista de rolamento. Considerando uma largura média de 1,8 metros para carros de passeio (L_{carro}), uma pista (L_{pista}) com largura entre 3 e 3,5 metros tem a largura do carro como aproximadamente 60% de sua própria medida. O carro robô tem 16 cm de largura ($L_{robô}$). A largura da pista de rolamento (L) foi escolhida a partir do cálculo das larguras máxima e mínima. A largura de pista escolhida foi de 30 cm e abaixo estão apresentados os cálculos:

$$L = L_{robô} \times \frac{L_{pista}}{L_{carro}} \tag{6}$$

$$L_{mín} = 16\ cm \times \frac{3\ m}{1{,}8\ m} = 26{,}67\ cm$$

$$L_{máx} = 16\ cm \times \frac{3{,}5\ m}{1{,}8\ m} = 31{,}11\ cm$$

3.4. Configuração do Raspberry

Para a configuração do Raspberry Pi 3, iniciou-se seu sistema com a instalação do sistema operacional Linux Raspbian. Após o Raspberry se tornar utilizável, foram criados pequenos programas teste para confirmação da existência de algumas bibliotecas. As bibliotecas que apontaram erros foram instaladas ou atualizadas, de acordo com a necessidade de cada uma, para funcionamento correto dos sistemas do robô. As bibliotecas utilizadas e suas versões estão dispostas no Anexo, na Tabela A2, contendo uma lista dos softwares utilizados no trabalho.

Os programas teste utilizados fizeram uso de códigos simples com ativação de componentes básicos, como um LED associado a um resistor, através do uso de funções das bibliotecas. No caso da biblioteca OpenCV, a câmera utilizada nos testes foi a mesma câmera aplicada no carro robô.

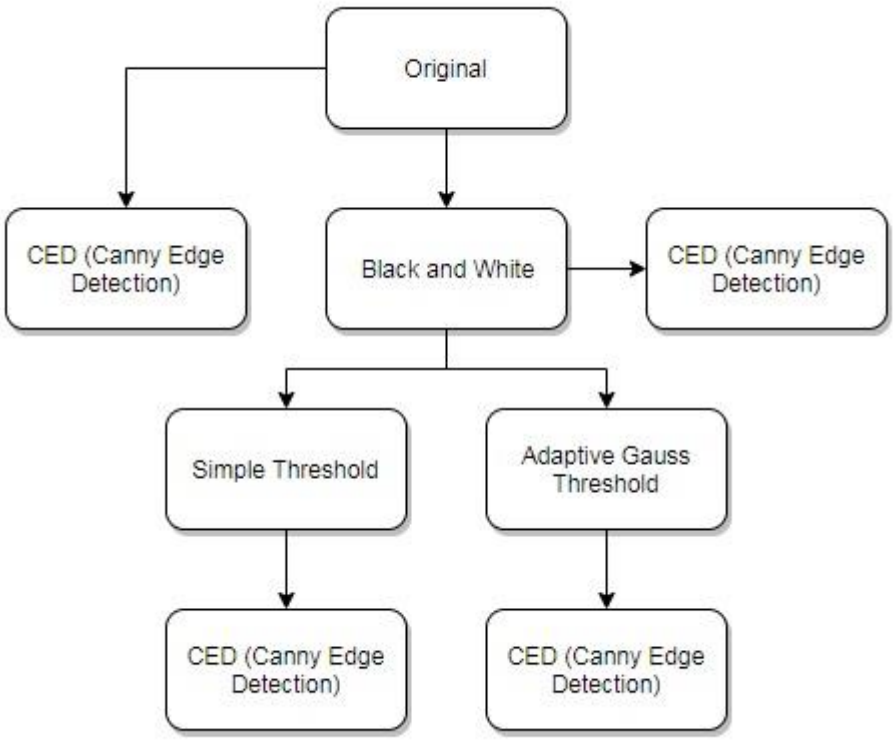
Por fim, configurou-se o Raspberry para rodar o programa do carro robô assim que iniciado. Para isso, foi utilizado um script bash simples, fixado na pasta etc/init.d/ na raiz do Raspberry[24][25].

3.5. Câmera e sistema de visão computacional

O objetivo do sistema de visão do carro foi detectar as faixas direita e esquerda que delimitam a faixa de rolagem disponível para o carro e retornar um valor de controle à inteligência que indicasse a posição do carro robô em relação à orientação da pista. Buscou-se a inclinação das faixas detectadas pelo algoritmo de Canny, aplicado a uma ROI especificada a partir de uma aproximação de deslocamento calculada com a velocidade atual dos motores do carro robô.

Para o tratamento da imagem, antes da detecção de bordas das faixas, foram utilizados algumas rotinas diferentes. O intuito foi o de comparar a velocidade e qualidade de processamento da imagem para detecção de bordas com o algoritmo de Canny para cada abordagem no tratamento da imagem, e escolher assim o tipo de tratamento que retornaria a melhor qualidade de resposta por parte do sistema de visão. Os programas testes seguiram o fluxograma com as rotinas de tratamento de imagem apresentado abaixo na Figura 15.

Figura 15. Organograma de tratamento da imagem

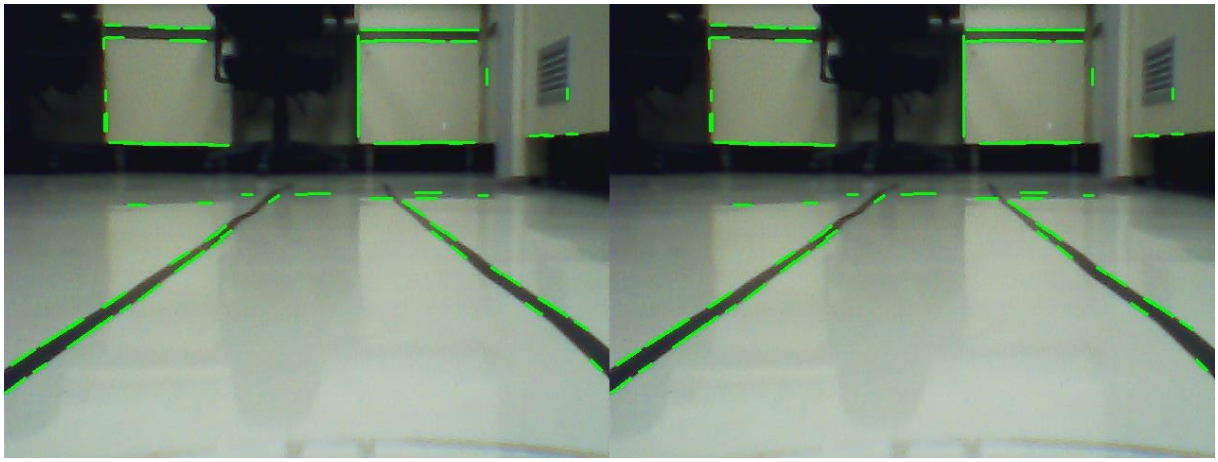


As rotinas foram executadas em programas diferentes, considerando-se apenas os tempos para tratamento da imagem, uma vez que o processo de captação da imagem era comum a todos os programas com os diferentes tipos de tratamentos de imagem. Cada rotina realizou a análise do tempo gasto para detecção de bordas em três testes diferentes: tempo gasto para 20, 50 e 100 ciclos de processamento da imagem. As rotinas adicionam cada tempo de processamento a um vetor entre cada iteração. Ao final, cada elemento do vetor é somado e dividido pelo respectivo número de ciclos, retornando o tempo médio gasto por cada rotina para detectar bordas nas imagens.

Todos os testes foram realizados com a mesma câmera, mirando sempre no mesmo lugar e dentro de um laboratório fechado e iluminado de forma uniforme durante todo o processo de teste de tratamento de imagem. Os resultados desse estudo estão apresentados no capítulo 4 do trabalho.

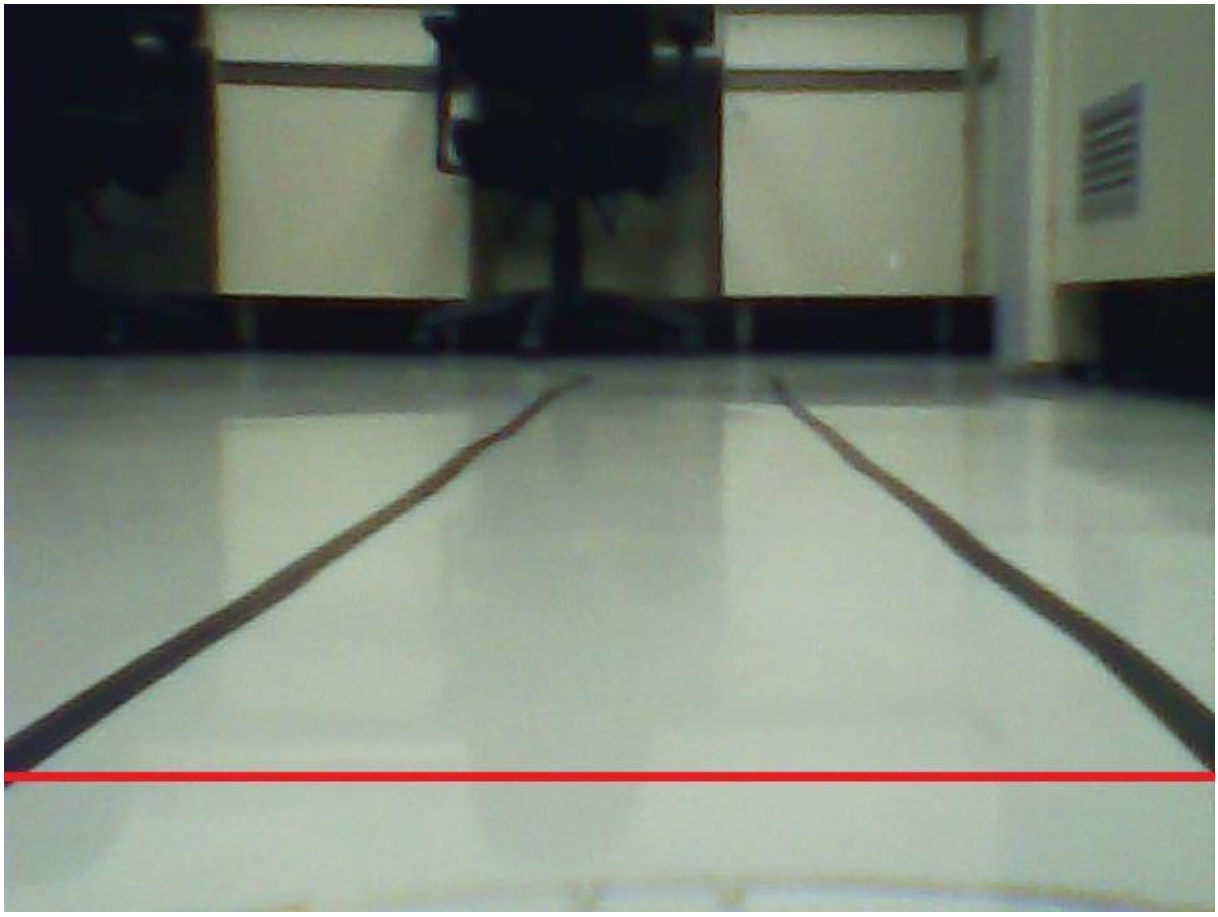
Ao final dos testes, as rotinas mais rápidas para o processamento foram selecionadas. No teste seguinte, foi aplicada a elas o algoritmo de Hough. O objetivo foi o de verificar a qualidade das linhas obtidas: tamanho das linhas, ruído na imagem identificado como linhas, segmentação e orientação de linhas foram os principais critérios observados. Nesta segunda parte do teste, foi selecionada apenas a rotina que mais se adequou aos requisitos de tratamento da imagem.

Figura 16. Comparação entre as linhas de Hough detectadas para o algoritmo que utiliza o algoritmo de Canny sobre a imagem preto e branca (à esquerda) e sobre a imagem limiarizada (à direita)



Em seguida, já no carro robô, a imagem foi destacada com aplicação da ROI antes da detecção de bordas com o algoritmo de Canny. A ROI aplicada à imagem capturada pela câmera foi posicionada sempre com referência no canto superior esquerdo e com altura máxima de 405 pixels (essa altura delimita o limite inferior da ROI, lembrando que a altura aqui é medida a partir do canto superior esquerdo). Esse limite inferior foi definido como o ponto mínimo onde ambas as faixas da pista podem aparecer juntas na tela capturada pela câmera, como delimitado pela faixa vermelha na Figura 17.

Figura 17. Limite de altura da ROI

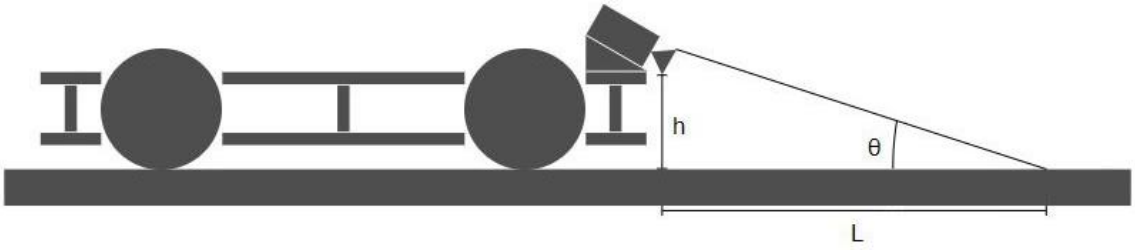


A ROI aplicada tem a forma de um retângulo com a largura original da imagem captada e tem altura igual a 100 pixels. A coordenada y do pixel central da ROI é definido de acordo com uma previsão aproximada da velocidade dos motores, calculada com a equação (3) e levando-se em consideração a porcentagem de velocidade dos motores em relação à velocidade máxima, como explicado mais

adiante. Em outras palavras, a análise da imagem é feita tendo-se em mente onde o robô estará na próxima análise de imagem.

Considerando o carro rodando em uma pista horizontal plana e a câmera ligada sempre com o mesmo ângulo sobre o robô e sempre com a mesma distância vertical h entre a câmera e o chão, temos o seguinte esquema:

Figura 18. Esquemático apresentando relação entre o posicionamento da câmera e características da imagem capturada



A imagem da pista captada pela câmera terá sempre o mesmo ângulo de incidência θ, e uma vez que a pista é plana, a distância L capturada pela câmera é fixa, e portanto, a distância de um pixel a outro em y (na imagem capturada pela câmera) é aproximadamente a mesma. O valor máximo do pixel y corresponde à velocidade máxima do carro robô em linha reta.

Para encontrar esse valor máximo, criou-se uma rotina que mantinha os motores ligados em aceleração máxima por 1 segundo. Em seguida, mediu-se a distância percorrida pelo carro robô nesse segundo e dividiu-se pelo tempo médio de ciclo de processamento do programa do robô. O robô foi colocado novamente na posição inicial do teste e foi feita uma marcação na pista correspondente à distância percorrida pelo robô com aceleração máxima durante um ciclo de processamento médio. A câmera foi ligada e mediu-se em pixels a distância da parte de cima da imagem (dentro de computação gráfica, a contagem é sempre feita a partir do canto superior esquerdo) até a marcação feita na pista. Essa distância é o pixel y que delimita a ROI na sua parte superior da imagem analisada.

A cada análise de imagem, a rotina de visão recebe as velocidades dos conjuntos de motores. Com essas velocidades, calcula-se a velocidade do robô no eixo y, de acordo com a equação (3) apresentada na seção 1.3. Aplicando os valores de entrada dos conjuntos dos motores entre -1 e 1, temos como saída uma porcentagem da velocidade dos motores, referente à máxima aceleração. A partir

dessa porcentagem em relação à velocidade máxima do robô, temos que o pixel y de cada análise será proporcional ao pixel y observado durante a máxima aceleração do robô.

Com o y encontrado a cada iteração, realiza-se o corte da imagem, destacando-se nela uma ROI. A ROI terá 50 linhas de pixels acima e 49 linhas de pixels abaixo do pixel y central encontrado, totalizando 100 pixels de altura da ROI. A largura da ROI será a mesma da imagem original. Em seguida, é aplicado o algoritmo de detecção de bordas de Canny na ROI criada na imagem.

Com as bordas encontradas apenas dentro da ROI pelo algoritmo de Canny, aplica-se então a transformação probabilística de Hough sobre a imagem binarizada. O ideal aqui é que a transformação de Hough retorne apenas 4 linhas: duas para cada faixa da pista, sendo uma para cada borda da faixa na pista.

Cada linha retornada pela transformação de Hough será informada através de quatro pontos, como especificado na seção 1.5., ou seja, coordenadas x e y de cada extremo de cada linha encontrada. Com essas coordenadas, é calculado o coeficiente de inclinação da reta, ou o coeficiente angular da reta, com a equação (7):

$$m = tg\ \alpha = \frac{y2-y1}{x2-x1} \tag{7}$$

Como mencionado por Hardwick[26], por causa do modo como é orientado o sistema de coordenadas em imagens no computador, o coeficiente virá com sinal invertido. Assim que calculado, o coeficiente é invertido, para normalização de seu valor.

Ângulos positivos indicam retas que se movem, olhando a partir da parte inferior da imagem, para a borda lateral direita da imagem. Ao contrário, retas com ângulos de inclinação negativos se movem, novamente a partir da parte inferior da imagem, em direção à borda lateral esquerda da imagem. Sendo assim, as retas encontradas foram agrupadas em retas que levam à esquerda e à direita.

Se ambas as faixas apresentarem inclinações inversas na imagem capturada pela câmera do robô, isto é, com mesmo valor em módulo, porém uma faixa com inclinação positiva e a outra com inclinação negativa, a soma das inclinações das faixas será zero, configurando essa parte analisada da pista como uma reta (a imagem capturada pela câmera possui perspectiva paralela, onde suas linhas de

fuga representam linhas paralelas no mundo físico, mas convergem para um ponto de fuga central ao fundo da imagem, fazendo com que essas linhas possuam inclinação).

Se analisarmos uma curva na pista, ambas as faixas terão inclinação ou positiva, ou negativa, indo ou para a direita, ou para a esquerda. Como as bordas da fita isolante utilizada para fazer a faixa da pista são paralelas, idealmente, os coeficientes e ângulos de inclinação das retas das bordas de cada fita serão iguais. Tendo a pista sendo feita buscando-se a paralelidade de faixas ao longo de seu trajeto, as faixas são próximas de paralelas entre si o suficiente para uma aproximação. Assim, será feita uma média dos ângulos de inclinação de todas as retas capturadas dentro de cada grupo de retas. A média calculada das inclinações será aplicada aos grupos de retas que vão para a esquerda e o grupo de retas que vão para a direita.

Após o cálculo do coeficiente de inclinação de cada reta, foi aplicado a esses coeficientes uma função arco tangente. A função arco tangente utilizada (math.atan()) retorna o valor do ângulo em radianos. Foi realizada então uma conversão de radianos para graus (função math.degrees()). O coeficiente encontrado se torna um ângulo. Em seguida, foi realizada a média dos ângulos positivos e negativos, que levam para a esquerda e para a direita, encontrando-se o ângulo de inclinação médio dos conjuntos de retas que vão para a esquerda e para a direita. Por fim, esses ângulos são somados e seu resultado é repassado como variável de entrada para o controlador lógico fuzzy utilizado no trabalho.

3.6. Inteligência artificial e a lógica fuzzy

O sistema de inteligência do carro foi feito utilizando-se um controlador lógico fuzzy. A função de pertinência da entrada do robô foi definida a partir do tipo desejado de entrada. A inteligência recebe o ângulo-soma dos ângulos de inclinação das faixas detectadas pelo algoritmo de visão. Caso esse ângulo-soma de inclinações das faixas seja 0º, significa que as faixas possuem a mesma inclinação em módulo, mas são opostas entre si, fazendo com que a orientação da pista seja interpretada como uma reta (como exemplo, a Figura 13 se aproxima desse caso). Caso a soma das inclinações das faixas seja negativa, o robô precisa acertar o

curso para a direita; caso a soma seja positiva, o robô deve acertar seu trajeto para a esquerda.

O espaço de entrada para a função de pertinência da entrada da inteligência varia de -90º até 90º, sendo -90º a representação de uma curva máxima à esquerda e 90º uma curva máxima à direita. A variável de entrada foi chamada de "Pista" e os termos fuzzy escolhidos para a função de pertinência de entrada foram "Curva à direita", "Meio da pista" e "Curva à esquerda" e todas são funções triangulares. A função de pertinência está apresentada na Figura 19.

Figura 19. Função de pertinência de fuzzyficação

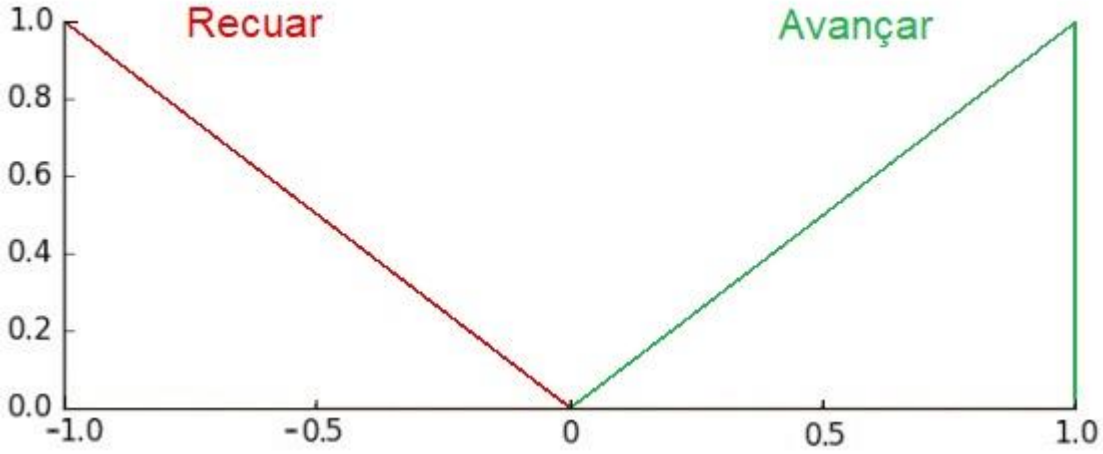
O conjunto de regras escolhido para inferência do sistema e cálculo da resposta na saída é:

- Se "Pista" = "Meio da Pista", então "MotorD" = "Avançar" e "MotorE" = "Avançar";
- Se "Pista" = "Curva à direita", então "MotorD" = "Recuar" e "MotorE" = "Avançar";
- Se "Pista" = " Curva à esquerda", então "MotorD" = "Avançar" e "MotorE" = "Recuar".

Nas regras acima, "MotorD" é o conjunto dos motores presentes à direita do robô enquanto que "MotorE" é a representação dos motores da esquerda. "Avançar" e "Recuar" são comandos para, respectivamente, acelerar para frente e para trás

Com as regras estabelecidas, foram definidas na sequência as funções de pertinência das saídas. As variáveis de saída foram definidas como os dois motores, "MotorD" e "MotorE". Para cada uma das variáveis de saída foram definidos os dois termos fuzzy: "Avançar" e "Recuar". As funções de pertinência de cada termo são iguais para as duas variáveis e estão apresentadas na Figura 20 abaixo.

Figura 20. Função de pertinência de defuzzyficação



Por fim, a defuzzyficação das variáveis fuzzy de saída ocorreu através de um somatório dos pontos máximos das funções de pertinência de cada termo para cada variável. Para os dois motores, a soma envolveu três valores: o valor fuzzy para as funções de pertinência dos termos "Curva à esquerda", "Meio da pista" e "Curva à direita".

Com a inteligência artificial programada, dois testes preliminares foram realizados. No primeiro, foram fornecidas ao controlador fuzzy entradas aleatórias, de respostas conhecidas ou não, para verificação do funcionamento do controlador. Para esse teste, foi analisada principalmente a precisão da resposta na saída do controlador fuzzy. Além da precisão, em um segundo teste foi realizado também um estudo do tempo de execução do código do controlador fuzzy durante o funcionamento da inteligência artificial. Os resultados dos testes são apresentados na discussão dos resultados do trabalho.

Confirmado o funcionamento do código da inteligência, foi realizada, por fim, a integração dos sistemas.

3.7. Integração de sistemas e aplicação do robô na pista

A integração dos sistemas começou unindo a inteligência artificial e os motores. Testou-se o funcionamento dos motores com entradas específicas na inteligência artificial. Escolheu-se entradas que retornariam saídas com visível diferença na atuação dos conjuntos de motores da direita e esquerda, ou seja, entradas em que um dos conjuntos de motores estivessem desligados, ou ambos os motores ligados em velocidade máxima.

Após verificação do funcionamento dos motores em concordância com os cálculos da inteligência, adicionou-se o sistema de visão ao programa principal. Os primeiro testes foram realizados com o carro suspenso. O robô foi posicionado em diversos pontos da pista com uma pequena plataforma apoiando o chassi, de forma a deixar as rodas sem qualquer contato com o piso. O robô foi posicionado de forma a enxergar a pista e atuar os motores como se estivesse solto na pista. No entanto, suspenso o robô não sairia do lugar, o que teoricamente faria com que ele enxergasse a pista com a mesma inclinação. Isso permitiu observar mais isoladamente o controle exercido para cada situação testada do carro na pista.

Por fim, o carro robô foi testado livremente na pista. Em todas as partidas iniciadas, o robô teve sua rotina de direção ativada com o carro parado, em repouso. Ele não foi testado sempre do mesmo ponto de partida: o programa do robô foi iniciado com ele partindo de retas na pista, na entrada ou no meio de curvas para um lado e para outro. Além disso, ele também foi iniciado mirando pontos na pista onde a frente estava livre, ou que tinham paredes ou armários à frente (obstáculos que o carro também pudesse detectar como linhas) ou ainda não estando em paralelo com as faixas da pista, ou seja, não estando virado exatamente para a frente, mesmo em retas. Todas essas posições iniciais foram escolhidas aleatoriamente na pista, mas sempre tentando estudar o comportamento tomado pelo carro robô em cada situação.

Além da variação nas posições de partida, também foi variada a potência máxima de ativação dos motores. Uma variável foi criada e atribuída como 1 no começo dos testes. Após um ciclo de acionamento do carro robô em cada situação de partida possível, a potência foi gradativamente reduzida para observação mais lenta da resposta encontrada pelo robô para cada situação.

Os testes foram todos abortados assim que o robô parava de se locomover, seja por não enxergar as faixas da pista, falta de potência para saída do repouso, ou impedimento de deslocamento por alguma peça travando as rodas ou problema do gênero. Os testes também foram parados quando o robô saia completamente para fora da pista.

Todos os testes com o carro robô completo foram realizados na mesma pista, no mesmo laboratório, iluminado de forma uniforme durante todos os processos, dependentes ou não do funcionamento adequado da câmera. Os resultados dos testes são apresentados no capítulo 5, na parte de conclusões e perspectivas.

4. Resultados e discussão

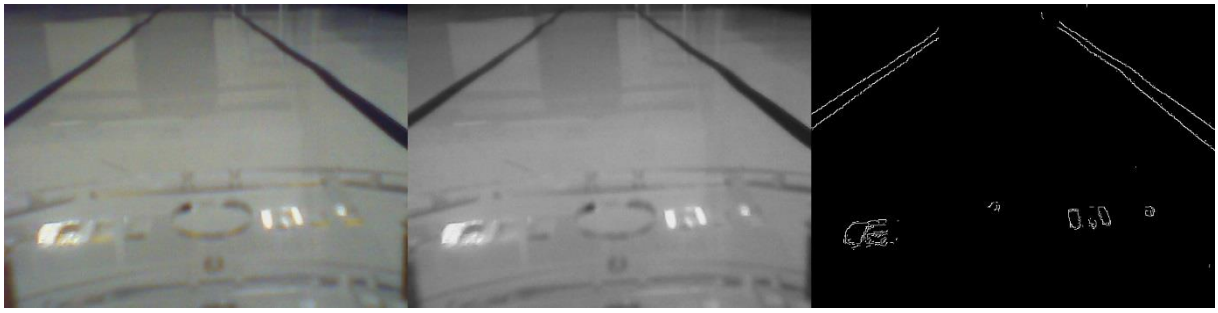
4.1. Análise de tempos e qualidade de processamento de imagem

O primeiro experimento realizado foi a comparação entre os tempos de execução das rotinas de tratamento da imagem até a detecção de bordas através do algoritmo de Canny, para uma quantidade pré-fixada de ciclos. Em seguida, foi realizada uma análise qualitativa das linhas captadas com o algoritmo de Hough para cada rotina de tratamento que foi aprovada no primeiro teste. Os critérios de avaliação consistiram dos seguintes elementos presentes na pega das linhas da imagem: tamanho das linhas capturadas, ruídos na imagem identificados como linhas, segmentação e orientação de linhas.

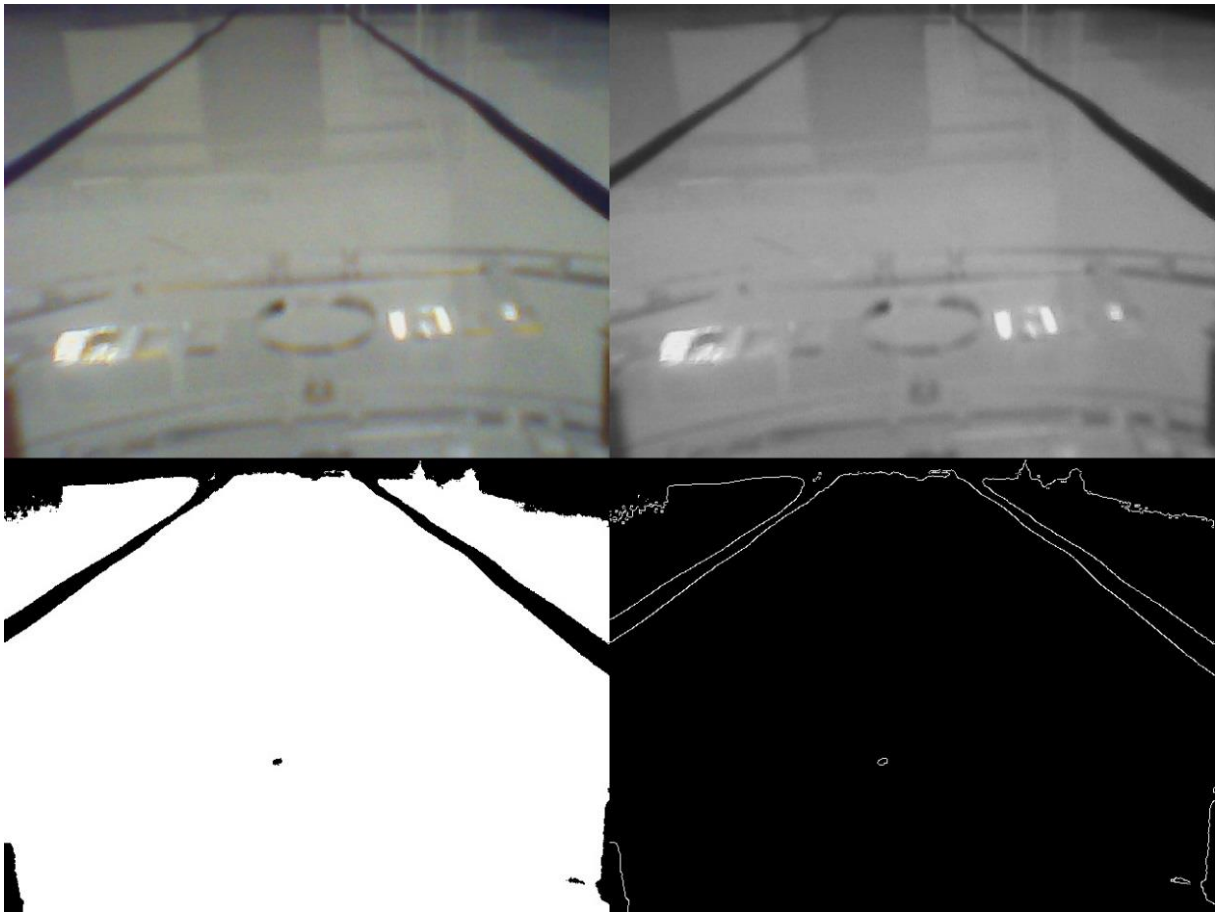
Como primeiro passo desse experimento, as câmeras foram calibradas para que cada tratamento de imagem pudesse gerar a melhor imagem possível, com bordas claras e o mínimo de ruído, principalmente na região onde a pista se encontra na imagem. A Figura 21 mostra, em sequências de imagens, o resultado de cada conjunto de processos utilizados para tratamento das imagens após a calibração da câmera para cada forma proposta de detectar as faixas de rolamento.

Figura 21. Resultado da detecção de bordas para as imagens com os seguintes efeitos após a calibração da câmera para cada tratamento: (a) Original; (b) Preto e branco; (c) Limiarização simples; (d) Limiarização adaptativa de Gauss

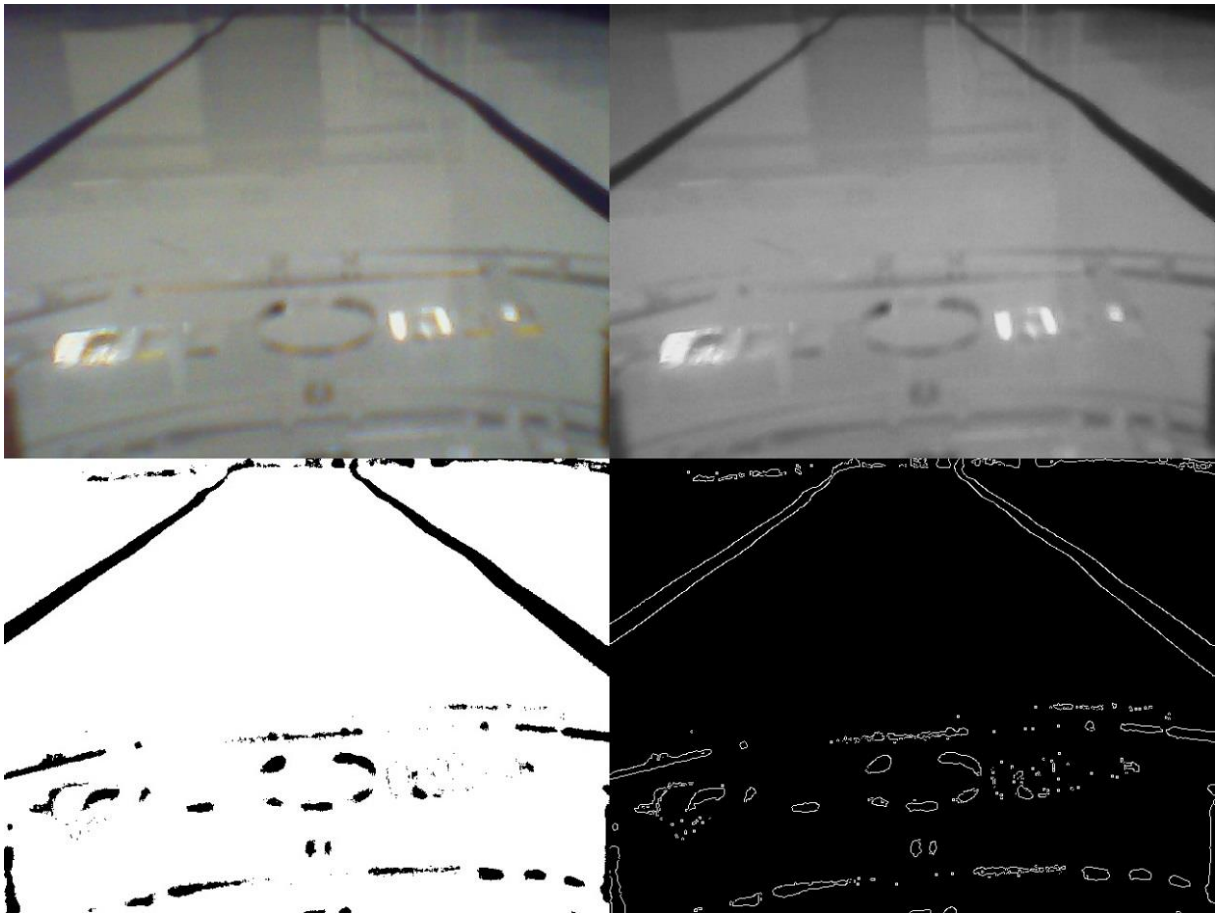
(a) Original



(b) Preto e branco



(c) Limiarização simples



(d) Limiarização adaptativa de Gauss

Os parâmetros que atingiram um nível bom de calibração para cada método de detecção estão listados a seguir (apresentados como parâmetros como são utilizados nas respectivas funções de chamada):

- Original:
  - Detecção de bordas de Canny:
    cv2.Canny(image, 100, 150)
- Preto e branco:
  - Detecção de bordas de Canny:
    cv2.Canny(image, 80, 100)
- Limiarização simples:
  - Limiarização:
    cv2.threshold(blackwhite, 95, 255, cv2.THRESH_BINARY)
  - Detecção de bordas de Canny:
    cv2.Canny(image, 100, 120)
- Limiarização adaptativa de Gauss:

- Limiarização adaptativa de Gauss: cv2.adaptiveThreshold(blackwhite,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY, 101, 10)
- Detecção de bordas de Canny: cv2.Canny(image, 110, 120)

Os dados coletados a partir dos testes de tempo dos tratamentos das imagens estão apresentados abaixo na Tabela 2.

Tabela 2. Tempo médio de processamento dos tratamentos de imagem

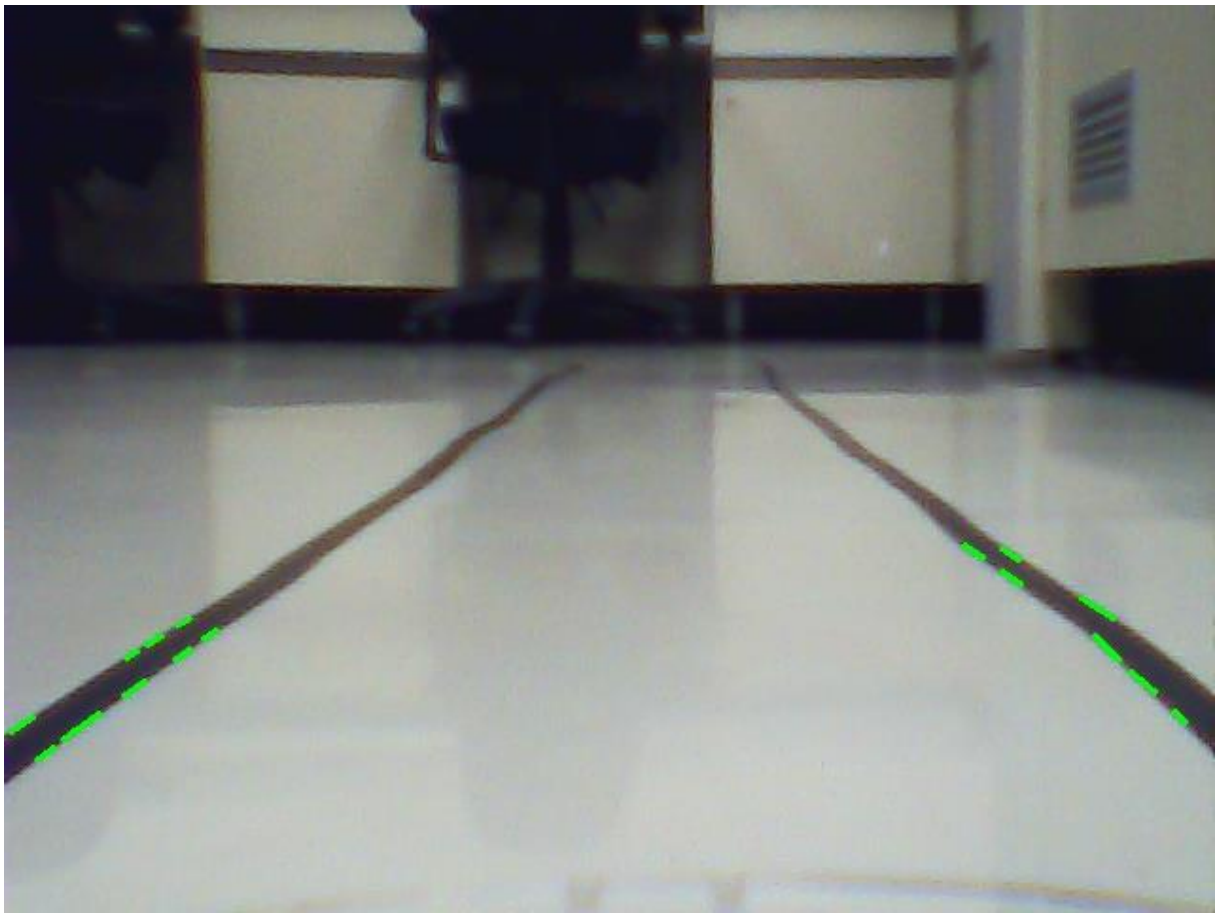
Canny aplicado a	20 Ciclos (ms)	50 Ciclos (ms)	100 Ciclos (ms)
Original	113,310	118,438	112,165
Preto e branco	47,793	46,956	47,105
Limiarizada	50,887	51,495	51,971
Limiarizada de Gauss	491,094	498,540	502,913

A partir do resultado observado, escolheu-se a partir da média de tempo duas rotinas para avaliação da qualidade de pega das linhas: a rotina que faz a detecção de bordas de Canny diretamente da imagem em preto e branco e a rotina que aplica o algoritmo de Canny à imagem limiarizada de forma simples.

Um resultado preliminar foi apresentado na Figura 16. Na Figura 22 abaixo, é apresentado uma ROI da imagem com as linhas já tratadas, obtidas através da câmera calibrada com a rotina de limiarização simples, exemplificando a imagem que será avaliada pela visão computacional do robô. Essa rotina foi escolhida por sua menor quantidade de ruído do ambiente na imagem final. A Figura 23 apresenta as linhas capturadas dentro da ROI, aplicadas à imagem original capturada pela câmera.

Figura 22. Linhas identificadas dentro da ROI

Figura 23. Limiarização simples: linhas identificadas na ROI exibidas junto com a imagem original da pista



A calibração do algoritmo probabilístico de Hough para o resultado apresentado nas Figuras acima foi feito de acordo com a função de chamada e parâmetros a seguir: cv2.HoughLinesP (treatedImage, 2, np.pi/180, 20, 20, 2).

4.2. Teste de precisão e tempo do controlador lógico fuzzy

Para uma avaliação da precisão do controlador lógico fuzzy, foram realizados testes separados da câmera, fornecendo entradas dentro da faixa de possibilidades definida. Alguma entradas eram conhecidas, por terem sido base para definição das funções de pertinência da inteligência artificial. A Tabela 3 mostra as entradas fornecidas (soma dos ângulos de inclinação das faixas da direita e da esquerda) e as saídas apresentadas pela inteligência artificial para valores com saída definida para os motores direito e esquerdo.

Tabela 3. Teste de precisão da inteligência artificial para valores com saída conhecida

Teste	1	2	3	4	5
Entrada	0	90	- 90	45	- 67,5
Motor Direito	1	- 1	1	0	1
Motor Esquerdo	1	1	- 1	1	- 0,5

As entradas com saídas conhecidas são aquelas que representam ou pontos extremos ou pontos médios da função de pertinência do controlador fuzzy. Se a entrada é 0 (ou seja, 0º de diferença entre a faixa esquerda e a direita), o robô está numa reta e deve acelerar o máximo possível. Caso a entrada seja 90 ou - 90 (90º e -90º respectivamente), o robô tem pela frente uma curva muito fechada e ele deve girar no mesmo lugar para não sair da pista e ajustar sua posição. Os pontos médios são 45; - 45; 67,5; - 67,5; 22,5 e - 22,5; onde um motor está ligado em velocidade máxima e o outro ou está desligado (caso de 45 e - 45), ou está ligado em 50% de sua velocidade, no sentido para frente ou para trás, dependendo de cada caso. Essas eram as entradas conhecidas. Durante o teste da precisão da inteligência artificial para calcular os testes, todos os resultados retornaram uma resposta lógica, desde que a entrada também o fosse, estando a inteligência correta todo o tempo para valores dentro da faixa esperada de valores.

Em um último teste, entretanto, foi tentado um valor fora da faixa determinada para o universo de possibilidades da função de pertinência de entrada. Quando a entrada de 101 foi fornecida, o programa acusou o erro, ao ser utilizado um valor fora da faixa determinada para esse estudo e deixou de ser executado de imediato.

Na sequência, foram fornecidas entradas aleatórias à inteligência, e a partir delas foram observadas as saídas retornadas. A partir dos valores conhecidos e saídas das funções de pertinência, é possível determinar se as respostas dadas pelo código fuzzy estão ou não de acordo com o controle esperado. Os resultados deste segundo teste estão na Tabela 4.

Tabela 4. Teste de precisão da inteligência artificial para entradas desconhecidas

Teste	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Entrada	- 26	30	- 60	8	- 49	71	85	- 12
Motor Direito	1	0,333	1	0,822	1	-0,578	-0,889	1
Motor Esquerdo	0,422	1	0,333	1	0,089	1	1	0,733

Mais uma vez, o comportamento das variáveis de saída do controlador fuzzy se mostrou previsível, o que contribui para a fácil verificação da assertividade do programa. A aceleração dos motores é sempre máxima quando se vira para o lado oposto ao lado do conjunto de motores: motores do lado direito tem velocidade máxima em curvas à esquerda; motores esquerdos possuem velocidade máxima em curvas à direita. O conjunto de motores que não está acelerado ao máximo tem seu ritmo controlado de forma linear, a partir do ponto central da função de pertinência até o ponto de curvatura mais extremo: quando não há inclinação, o motor se encontra em 100%; conforme ocorrer curva para algum lado, o respectivo motor começa a desacelerar, chegando a 0% em 45º de diferença, positivos ou negativos e entrando em marcha ré caso a curva se torne mais fechada, culminando em 90º de diferença, onde os conjuntos de motores do mesmo lado da curva serão ligados em 100% de velocidade, mas em marcha ré. Nesse ponto, o robô gira no mesmo lugar alterando apenas sua orientação sobre a pista, mas não sua posição.

Com a precisão da inteligência artificial confirmada, realizou-se um teste para verificação do tempo de processamento da inteligência artificial para cada ciclo de resposta. Novamente, variou-se a entrada da inteligência, utilizando-se valores aplicados aos testes anteriores, com saídas conhecidas para confirmar a contagem do tempo para cálculo de rotina com resposta correta. Os tempos de processamento para cada entrada, juntamente com sua média, estão na Tabela 5.

Tabela 5. Teste de tempo de processamento do controlador lógico fuzzy

Teste	1	2	3	4	5	6	7	8	Média
Entrada	- 26	30	90	- 12	- 45	71	85	8	-
Tempo (ms)	3,814	3,963	4,141	3,871	3,733	3,785	3,920	4,115	3,918

Analisando os resultados do teste de tempo de processamento da inteligência, aparentemente, não há variação no tempo de processamento em função das entradas utilizadas. Sejam entradas correspondentes a pontos limite da faixa de valores possíveis ou valores centralizados nessa faixa de valores, entradas positivas e negativas, o tempo médio para todo o cálculo da inteligência aconteceu sempre por volta de 4 ms.

5. Conclusões e Perspectivas

5.1. Sistemas isolados

Os sistemas isolados foram mais extensivamente e intensamente estudados do que os sistemas já unidos no robô. Isso ocorreu em uma tentativa de antecipar todas as situações possíveis com as quais o robô se depararia. Além disso, quanto mais simples o sistema, mais facilmente ele é estudado, avaliado e eventuais problemas diagnosticados e tratados. Com todos os sistemas unidos, um problema na visão computacional poderia ser interpretado como um problema na velocidade ou potência dos motores, ou um problema com a inteligência artificial poderia ser uma falha na variável de entrada.

Os sistemas estudados no trabalho foram os sistemas de visão computacional e de inteligência artificial. Com diversos testes realizados, o sistema de visão computacional forneceu respostas satisfatórias na saída, com representação gráfica que confirma a detecção de faixas. Os resultados numéricos na saída representando os graus da diferença entre as inclinações da faixa direita e esquerda também corresponderam aos resultados esperados, indicando retas, curvas à direita e esquerda, analisando mesmo que a inclinação de apenas uma faixa. O sistema de visão apresentou isoladamente uma análise satisfatória da pista, com tempos de processamento baixos para duas principais formas de tratamento de imagem estudados.

O sistema de inteligência artificial funcionou exatamente como esperado, calculando os valores de saída a partir da entrada com precisão de 100%. As situações de posição e orientação do carro robô na pista tiveram respostas coerentes com o que se esperava do comportamento final do robô calculado pela inteligência artificial fuzzy. Isoladamente, o controlador fuzzy foi o sistema mais bem organizado e executado dentro do projeto.

5.2. Sistemas integrados e funcionamento do carro robô

Como explicado na seção 3.7., a integração de sistemas começou pela união dos motores e inteligência artificial. Até esse momento, nenhum problema foi observado e os sistemas continuaram funcionando de acordo com o esperado. Ao adicionar o sistema de visão, mantendo-o suspenso, observou-se um

comportamento incerto em alguns pontos. Em retas, o carro demonstrou controle aparentemente coerente com a pista detectada: aceleração máxima em retas perfeitas e pequena variação na velocidade de um motor ou outro de acordo com uma leve inclinação na orientação do sentido do robô em retas. Porém, quando posicionado com a câmera mirando curvas, os motores do carro foram acionados com velocidade equivalente ao necessário para fazer as curvas, sem, no entanto, estar sobre as curvas para fazer a curva.

Por fim, foram realizados os testes com o carro livre na pista. Em concordância com o que foi observado no teste anterior com o carro suspenso, o carrinho apresentou resultados ruins nesse teste final.

Nas retas, o comportamento do robô foi extremamente satisfatório, se mantendo sempre entre as faixas e sempre acertando o curso para se manter no meio da pista. Quando colocado em uma reta, porém orientado para a lateral da pista e não para a frente, o carro também conseguiu acertar o curso e seguir dentro das faixas satisfatoriamente. Nas curvas, por outro lado, o robô confirmou o erro observado no teste anterior. Ele realizou todas as curvas logo que entrava nelas ou um pouco antes de entrar na curva. Esperava-se que quando fosse testado livre, a velocidade do carro permitisse que ele chegasse à curva e realizasse a curva sem sair de por entre as faixas. Isso não se concretizou e o robô em teste livre continuou fazendo o comando de viragem da curva muito cedo.

Faz-se uma análise detalhada dos problemas observados com relação ao comportamento do carro robô na pista:

1. Sistema de visão:

1.1. Câmeras - Enquanto hardware, as imagens capturadas pelas câmeras foram nítidas o suficiente para identificação de diversos elementos de imagem. O código, como apresentado no trabalho por meio de imagens, foi capaz de extrair das imagens capturadas os limites de pista, entre outros objetos presentes no ambiente, como cadeiras e armários. Definitivamente não é um problema para solução do problema da realização antecipada da curva.

1.2. Visão computacional - Analisando a parte de software do sistema de visão, verifica-se que as bordas dos limites de pista foram corretamente identificados e filtrados nas imagens. A detecção de linhas, por outro lado, não conseguiu, mesmo nas retas, identificar linhas com comprimento significativo. Como a entrada do sistema de controle da inteligência foi a média da inclinação dessas

linhas, erros de inclinação poderiam mais facilmente ser corrigidos nas retas, uma vez que com linhas paralelas, ambas teriam a mesma inclinação. Já no caso da parte curva da pista, isso poderia ser traduzido em um erro de curvatura na movimentação do robô. Além disso, a maior dificuldade para selecionar linhas com um comprimento contíguo em curvas pode também ter contribuído para um maior erro no direcionamento do robô. No entanto, o maior problema observado foi a antecipação com que o robô realizava as curvas, e não a precisão de seu movimento enquanto na curva.

2. Sistema de inteligência artificial:

2.1. O sinal de entrada - A entrada do código fuzzy utilizado é a soma das inclinações das médias das linhas que tendem à direita e à esquerda. A entrada é uma média da direção das linhas capturadas pelo robô. Muito embora, após a aplicação da ROI e todo o tratamento, a maior parte observada pela câmera seja a pista, ruídos significativos podem alterar drasticamente o direcionamento do robô. Isso acontece especialmente quando o robô se aproxima de curvas, uma vez que há menos pista diretamente à frente para ser vista e um cálculo mais normalizado da média das inclinações da pista se torna cada vez mais difícil. Essa desorientação em curvas pode ser resultado do uso desse tipo de sinal de entrada.

2.2. Controlador lógico fuzzy - O controlador lógico fuzzy teve resposta satisfatória durante os trechos de reta e o código levava em consideração comportamentos importantes, como o de aceleração máxima no caso de uma reta perfeita. No entanto, o carro robô não foi capaz de executar corretamente as curvas. Isso não necessariamente pode ser atribuído ao código da inteligência artificial. Mesmo assim, uma mudança na função de fuzzyficação ou de defuzzyficação poderia solucionar o problema das curvas, ou pelo menos melhorar o comportamento do carro com relação à orientação da pista caso a inteligência não seja de fato o problema.

3. O sistema do robô:

3.1. Integração dos sistemas - A integração dos sistemas é onde mais provavelmente se resolveria os problemas observados no resultado final do trabalho. Uma vez que os sistemas foram pensados, implementados e testados separados, o passo anterior ao teste com falhas foi justamente a integração dos sistemas do robô. Entre as possibilidades, os sistemas isolados em correto funcionamento poderiam ser integrados de maneira menos direta. Isto é, no lugar de o robô atuar diretamente

os motores na sequência do comando da inteligência artificial, que por sua vez foi baseado em uma imagem que analisa a futura posição do robô, a integração dos sistemas poderia apresentar uma lista de possíveis comandos de atuação baseados na atual situação do robô. Somente a pretexto de ilustração, caso o robô se encontrasse em uma reta, mas estivesse enxergando uma curva fechada, ele não deveria necessariamente começar a realizar a curva de imediato. O comportamento antecipativo é importante, mas nem sempre é correto, uma vez que o robô, nesse exemplo, ainda não se encontrava na curva.

Outra informação que faltou ao sistema como um todo e que poderia ser decisiva é a de noção de posição do próprio robô com relação à pista. Sem saber em que parte da pista estava, sua movimentação e orientação poderia muito bem ser considerada um movimento de ponto a ponto na pista, onde o robô tentava se manter sempre entre as duas faixas no próximo ponto visualizado, mas não fazendo diferença se ele atravessava alguma faixa para chegar ao meio das duas faixas no próximo trecho de pista, por exemplo.

5.3. Perspectivas para trabalhos futuros

O primeiro ponto de melhoria seria o reparo do código do robô de forma que ele realizasse as curvas corretamente. Dentre as possibilidades, uma afinação nas funções de pertinência do robô de forma que a relação entre curva e controle dos motores não fosse linear. Outra possibilidade seria a criação de uma lista de comandos, preenchida de acordo com a equação de movimentação de robôs diferenciais, movimentação do robô e visão da câmera, de forma que os comandos não fossem diretamente executados e talvez normalizados e checados entre si para confirmação do melhor comando possível.

Além do reparo do projeto, seria interessante também a criação de um carro robô mais robusto, com componentes mais potentes e resistentes.

O robô também poderia ser aplicado em outras situações, de forma a testar variações aplicadas a cada sistema ou situações mais complexas. Outras pistas, outros locais, com possível variação no ambiente, seja de iluminação ou clima testariam o sistema de visão. A adição de mais carros na pista talvez seria o projeto mais interessante em um próximo estudo, considerando o conserto do funcionamento do carro sozinho na pista.

Por fim, outro possível trabalho futuro é a implementação de comportamento ao ritmo do carro, onde o carro poderia ter ritmo acelerado ou ritmo cauteloso dependendo de indicações na pista, sonoras ou visuais, como os sinais verde, amarelo e vermelho em um semáforo, por exemplo.

Ainda dentro da mesma ideia, seria interessante analisar como o robô se comportaria em um ambiente de movimentação livre, sem sinais explícitos de trânsito como semáforo ou placas, mas com obstáculos com uma movimentação e comportamento próprio, como uma simulação à atividade de um veículo autônomo em um ambiente rural ou em situação especial como a de movimentação dentro de um evento ou dentro de um shopping, no caso de robôs moveis não aplicados no trânsito.

6. Referências Bibliográficas

[1]. MATARIĆ, Maja J. **Introdução à Robótica.** 1. ed. São Paulo: Editora UNESP/Blucher, 2014.

[2]. **Enciclopédia Larousse Cultural.** São Paulo: Nova Cultural Ltda, 1995. v. 21.

[3]. International Organization for Standardization (ISO). **ISO 8373:2012(en) Robots and robotics devices.** Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en>. Acesso em: 11/06/2017.

[4]. International Federation of Robotics (IFR). **International robot standardization within ISO.** Disponível em: <https://ifr.org/standardisation>. Acesso em: 11/06/2017.

[5]. PRESSE, France. G1 (Globo). **Veículos autônomos ainda têm muitos desafios pela frente.** 2018. Disponível em: <https://g1.globo.com/carros/noticia/veiculos-autonomos-ainda-tem-muitos-desafios-pela-frente.ghtml>. Acesso em: 19/05/2018.

[6]. PLUMER, Brad. Vox. **5 big challenges that self-driving cars still have to overcome.** 2016. Disponível em: <https://www.vox.com/2016/4/21/11447838/self-driving-cars-challenges-obstacles>. Acesso em: 19/05/2018.

[7]. COSTA, Kawann. Ciências e Tecnologia. **IA: Lógica fuzzy aplicada à Engenharia de sistemas de frenagem.** 2017. Disponível em: <https://cienciaetecnologias.com/ia-logica-fuzzy-aplicada-engenharia-de-sistemas-de-frenagem/>. Acesso em: 19/05/2018.

[8]. BHUIYAN, Johana. recode. **After two million miles, Google's robot car now drives better than a 16-year-old.** 2016. Disponível em: <https://www.recode.net/2016/10/5/13167364/google-self-driving-cars-2-million-miles>. Acesso em: 19/05/2018.

[9]. SIEGWART, Roland; NOURBAKHSH, Ilah R. **Introduction to Autonomous Mobile Robots.** 1. ed. Cambridge: The MIT Press, 2004.

[10]. RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial.** 2. ed. Rio de Janeiro: Elsevier, 2004.

[11]. GRUVER, William A. IEEE Xplore. **Distributed intelligent systems: A paradigm shift.** Smolenice: IEEE, 2012. <https://ieeexplore.ieee.org/document/6319500/>. Acesso em: 20/05/2018.

[12]. CROWDER, James A.; CARBONE, John N. **Abductive Artificial Intelligence Learning Models.** Aurora: ICAI, 2017. <https://csce.ucmss.com/cr/books/2017/LFS/CSREA2017/ICA3123.pdf>. Acesso em: 20/05/2018.

[13]. RODRIGUEZ, Jesus. Medium. **Types of Artificial Intelligence Learning Models.** 2017. Disponível em: <https://medium.com/@jrodthoughts/types-of-artificial-intelligence-learning-models-814e46eca30e>. Acesso em: 20/05/2018.

[14]. MATLAB & Simulink. **What is the Genetic Algorithm?** 2018. <https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html>. Acesso em: 20/05/2018.

[15]. SIMÕES, Marcelo G.; SHAW, Ian S. **Controle e Modelagem Fuzzy.** 2. ed. São Paulo: Blucher, 2007.

[16]. CAVALCANTI, José H. F., et al. **Lógica Fuzzy Aplicada Às Engenharias.** João Pessoa: Câmara, 2012. Disponível em: <http://www.logicafuzzy.com.br/wp-content/ uploads/2013/04/logica_fuzzy_aplicada_as_engenharias.pdf>. Acesso em: 20/05/2018.

[17]. scikit-fuzzy development team. scikit-fuzzy. **Fuzzy Control Primer.** Disponível em: <https://pythonhosted.org/scikit-fuzzy/userguide/fuzzy_control_primer.html>. Acesso em: 22/06/2018.

[18]. RIOS, Luiz R. S. **Visão Computacional.** Salvador: UFBA, 2010. Disponível em: <http://homes.dcc.ufba.br/~luizromario/Apresenta%C3%A7%C3%A3o%20de%20IA/Artigo%20(final).pdf>. Acesso em: 22/05/2018.

[19]. LAGANIÈRE, Robert. **OpenCV 2 Computer Vision Aplication Programming Cookbook:** Over 50 recipes to master this library of programming functions for real-time computer vision. 1. ed. Birmingham: Packt, 2011.

[20]. MORDVINTSEV, Alexander; K, Abid. **OpenCV-Python Tutorials Documentation.** 2017. Disponível em: <https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/ opencv-python-tutroals.pdf>. Acesso em: 22/05/2018.

[21]. G1 (Globo). **Vítimas de acidente de trânsito ocupam 60% das UTIs no Brasil.** 2014. Disponível em: <http://g1.globo.com/bom-dia-brasil/noticia/2014/11/vitimas-de-acidente-de-transito-ocupam-60-das-utis-no-brasil.html>. Acesso em: 11/07/2018.

[22]. em discussão! (Senado). **Entre mortes e feridos, vítimas de acidentes no trânsito são custo para a sociedade e problema para o sistema de saúde pública do Brasil.** 2012. Disponível em: <http://www.senado.gov.br/noticias/

Jornal/emdiscussao/motos/saude-publica/entre-mortes-e-feridos-vitimas-de-acidentes-no-transito-sao-custo-para-a-sociedade-e-problema-para-o-sistema-de-saude-publica-do-brasil.aspx>. Acesso em: 11/07/2018.

[23]. Ministério dos Transportes - DNER/DNIT. **Normas para o Projeto das Estradas de Rodagem.** Rio de Janeiro, 1973. Disponível em: <http://www.dnit.gov.br/download/rodovias/operacoes-rodoviarias/faixa-de-dominio/normas-projeto-estr-rod-reeditado-1973.pdf>. Acesso em: 05/05/2018.

[24]. Comunidade Hardware. **Iniciar script python ao ligar (Debian).** 2016. Disponível em: <https://www.hardware.com.br/comunidade/v-t/1417634/>. Acesso em: 02/06/2018.

[25]. TAVARES. Caderno de Laboratório. **Inicializando um programa automaticamente no RaspberryPi.** 2015. Disponível em: <https://cadernodelaboratorio.com.br/2015/06/10/inicializando-um-programa-automaticamente-no-raspberrypi/>. Acesso em: 02/06/2018.

[26]. HARDWICK, Matt. Medium. **Simple Lane Detection with OpenCV.** 2017. Acesso em: <https://medium.com/@mrhwick/simple-lane-detection-with-opencv-bfeb6ae54ec0>. Acesso em: 22/05/2018.

Anexo

Tabelas complementares - Listas de materiais e softwares utilizados no trabalho

Tabela 1A. Lista completa de materiais do carro robô

Componente	Aplicação	Quantidade
Raspberry Pi 3 Modelo B	Sistema integrador/Inteligência Artificial	1
Motor DC 5V	Sistema de direção	4
Roda	Sistema de direção	4
Chassi acrílico	Estrutura do robô	2
Ponte H L298N	Sistema de direção	1
Câmera Multilaser WC040	Sistema de visão computacional	1
Bateria externa 10.000 mAh	Sistema integrador	1
Pilha 9 V	Fonte para alimentação da Ponte H	1
Espaçador M3 x 30 mm	Estrutura do robô	6
Parafuso M3 x 8 mm	Estrutura do robô	12
Presilha de acrílico	Estrutura do robô (fixação dos motores)	8
Parafuso M3 x 24 mm	Estrutura do robô (fixação dos motores)	8
Porca M3	Estrutura do robô (fixação dos motores)	8
Jumper fêmea-fêmea 20 cm	Ligação Raspberry - Ponte H	4
Jumper macho-fêmea 20 cm	Ligação Raspberry - Ponte H	1
Fonte 12V	Fonte para alimentação da Ponte H	1

Tabela 2A. Lista de softwares utilizados

Software	Versão	Aplicação
Raspbian	Debian	Sistema operacional do Raspberry Pi 3
Python	3.6.5	Linguagem de programação
gpiozero	1.4.1	Biblioteca para controle dos motores
OpenCV	3.4.1	Sistema de visão computacional
scikit-fuzzy	0.2	Biblioteca para inteligência artificial
numpy	1.14.5	Biblioteca para otimização de operações matemáticas