UNIVERSIDADE ESTADUAL PAULISTA - UNESP

Instituto de Biociências, Letras e Ciências Exatas- campus de São José do Rio Preto

BRUNO HENRIQUE ZARA

Tomada de Decisão em tempo Real usando Lógica Fuzzy para Sistemas Embarcados

São José do Rio Preto

2025

Bruno Henrique Zara

Tomada de Decisão em tempo Real usando Lógica Fuzzy para Sistemas Embarcados

Monografia, apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto, para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Profº Dr. Diego Renan Bruno

São José do Rio Preto
2025

Z36t

Zara, Bruno Henrique
Tomada de decisão em tempo real usando lógica fuzzy para sistemas embarcados / Bruno Henrique Zara. -- São José do Rio Preto, 2025
45 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Diego Renan Bruno

1. Automated vehicles. 2. Embedded computer systems. 3. Lógica difusa. 4. Genetic algorithms. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

BRUNO HENRIQUE ZARA

TOMADA DE DECISÃO EM TEMPO REAL USANDO LÓGICA FUZZY PARA SISTEMAS EMBARCADOS

Monografia apresentado(a) à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto, para obtenção do título de Bacharel em Ciência da Computação.
Data de defesa: 01/12/2025

BANCA EXAMINADORA

_____

Profº Dr. Diego Renan Bruno
UNESP – Instituto de Biociências, Letras e Ciências Exatas – Campus de São José do Rio Preto

_____

Profº Dr. Geraldo Francisco Donegá Zafalon
UNESP - Instituto de Biociências, Letras e Ciências Exatas - Câmpus de São José do Rio Preto

_____

Prof. Dr. Eng. Rodrigo Capobianco Guido
UNESP - Instituto de Biociências, Letras e Ciências Exatas - Câmpus de São José do Rio Preto

Dedico este trabalho a todos que apoiaram minha trajetória acadêmica, em especial aos meus pais, Reginaldo Perpétuo Zara e Carla Cristina Fonseca Zara, que me ofereceram apoio incondicional, tanto financeiro quanto emocional, ao longo de todo o curso.

AGRADECIMENTOS

Agradeço à Universidade Estadual Paulista (UNESP) e ao meu professor orientador, pela orientação e confiança neste trabalho. Estendo meus agradecimentos a todos os docentes que, com dedicação e conhecimento, contribuíram para minha formação acadêmica e me proporcionaram experiências que levarei para a vida.

“An idiot admires complexity. A genius admires simplicity”
(Terry A. Davis)

RESUMO

O alto custo computacional e a natureza de "caixa-preta"das abordagens de *deep learning* para veículos autônomos motivam a busca por arquiteturas de decisão mais leves, eficientes e interpretáveis, adequadas para sistemas embarcados. Este trabalho propõe o desenvolvimento e a validação de um sistema de tomada de decisão híbrido, de baixo custo, que integra o método de Tomada de Decisão Multicritério (MADM) **TOPSIS** com a **Lógica Fuzzy**. A arquitetura utiliza o TOPSIS para a seleção de manobras estratégicas (ex: "acelerar") e controladores Fuzzy especialistas para o refinamento da ação (ex: calcular o ângulo de esterçamento). Os parâmetros de ambos os módulos (pesos TOPSIS e Funções de Pertinência Fuzzy) foram otimizados via **Algoritmos Genéticos (GAs)** decoplados, treinados por Aprendizado por Imitação (Clonagem Comportamental) no simulador **CARLA**. Os resultados demonstram que, embora a camada de controle (GA-Fuzzy) tenha sido altamente bem-sucedida em replicar o especialista com baixo Erro Absoluto Médio (MAE), a camada estratégica (GA-MADM) falhou. O modelo TOPSIS, por ser linear, não capturou a complexidade não-linear das decisões de condução (73% de acurácia vs. 92.7% de um *baseline* Random Forest), resultando em falhas críticas de classificação. Conclui-se que a arquitetura híbrida é viável, interpretável e adequada para implementação em hardware (FPGA), mas o módulo TOPSIS é insuficiente e deve ser substituído por um modelo estratégico não-linear.

Palavras-chave: Veículos Autônomos, Lógica Fuzzy, Tomada de Decisão Multicritério (MADM), TOPSIS, Algoritmos Genéticos, CARLA.

ABSTRACT

The high computational cost and "black-box"nature of deep learning approaches for autonomous vehicles motivate the search for lighter, more efficient, and interpretable decision-making architectures suitable for embedded systems. This work proposes the development and validation of a low-cost, hybrid decision-making system that integrates the Multi-Criteria Decision-Making (MADM) method **TOPSIS** with **Fuzzy Logic**. The architecture uses TOPSIS for the selection of strategic maneuvers (e.g., "accelerate") and specialist Fuzzy controllers for action refinement (e.g., calculating the steering angle). The parameters of both modules (TOPSIS weights and Fuzzy Membership Functions) were optimized via decoupled **Genetic Algorithms (GAs)**, trained by Imitation Learning (Behavioral Cloning) in the **CARLA** simulator. The results demonstrate that while the control layer (GA-Fuzzy) was highly successful in replicating the expert with a low Mean Absolute Error (MAE), the strategic layer (GA-MADM) failed. The TOPSIS model, being linear, failed to capture the non-linear complexity of the driving decisions (73% accuracy vs. 92.7% from a Random Forest baseline), resulting in critical classification failures. It is concluded that the hybrid architecture is viable, interpretable, and suitable for hardware implementation (FPGA), but the TOPSIS module is insufficient and must be replaced by a non-linear strategic model.

Keywords: Autonomous Vehicles, Fuzzy Logic, Multi-Criteria Decision-Making (MADM), TOPSIS, Genetic Algorithms, CARLA.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AHP	*Analytic Hierarchy Process* (Processo de Análise Hierárquica)
ANP	*Analytic Network Process* (Processo de Análise em Rede)
BRAMs	*Block RAMs* (Blocos de Memória RAM)
CARLA	*Car Learning to Act* (Simulador para veículos autônomos)
CNN	*Convolutional Neural Network* (Rede Neural Convolucional)
DRL	*Deep Reinforcement Learning* (Aprendizado por Reforço Profundo)
E2E	*End-to-End* (Fim-a-Fim)
FIS	*Fuzzy Inference System* (Sistema de Inferência Fuzzy)
FPGA	*Field Programmable Gate Array* (Matriz de Portas Programável em Campo)
GA	*Genetic Algorithm* (Algoritmo Genético)
IL	*Imitation Learning* (Aprendizado por Imitação)
LiDAR	*Light Detection and Ranging*
LUTs	*Look-Up Tables* (Tabelas de Consulta)
MADM	*Multiple Attribute Decision Making* (Tomada de Decisão com Múltiplos Atributos)
MAE	*Mean Absolute Error* (Erro Absoluto Médio)
MF	*Membership Function* (Função de Pertinência)
NIS	*Negative Ideal Solution* (Solução Ideal Negativa)
PIS	*Positive Ideal Solution* (Solução Ideal Positiva)
RF	*Random Forest* (Floresta Aleatória)
TOPSIS	*Technique for Order of Preference by Similarity to Ideal Solution*

SUMÁRIO

1 INTRODUÇÃO

O avanço tecnológico tem transformado de maneira significativa a forma como a sociedade se relaciona com a mobilidade. A crescente demanda por soluções de transporte mais seguras, eficientes e sustentáveis impulsionou a pesquisa e o desenvolvimento de veículos autônomos, capazes de operar com pouca ou nenhuma intervenção humana. Dotados de um complexo arranjo de sensores, algoritmos e sistemas inteligentes, esses veículos prometem não apenas revolucionar o setor automotivo, mas também impactar positivamente a sociedade em múltiplos aspectos.

1.1 CONTEXTUALIZAÇÃO E JUSTIFICATIVA

A relevância do desenvolvimento de tecnologias para autonomia veicular está diretamente ligada à mitigação de problemas crônicos da mobilidade urbana. Acidentes de trânsito, em grande parte causados por falha humana, congestionamentos, tempo excessivo de deslocamento e altos custos operacionais são desafios que poderiam ser drasticamente reduzidos por meio da automação. Adicionalmente, a autonomia veicular representa um avanço na inclusão social, oferecendo independência a pessoas com limitações de mobilidade.

No entanto, a consolidação dessa tecnologia enfrenta barreiras significativas. Grande parte das soluções de ponta para veículos autônomos baseia-se em técnicas de aprendizado profundo (*deep learning*), que, apesar de sua alta eficácia, demandam um elevado poder computacional. Tal dependência resulta em altos custos de hardware, consumo energético expressivo e complexidade de implementação, tornando a tecnologia inacessível para aplicações em larga escala e em veículos de menor custo.

Este trabalho se justifica pela necessidade de explorar arquiteturas de decisão mais leves e eficientes, que possam viabilizar a autonomia veicular de forma mais democrática e sustentável.

1.2 PROPOSTA DE SOLUÇÃO: UM SISTEMA HÍBRIDO OTIMIZADO

Diante do desafio apresentado, propõe-se neste trabalho um sistema de tomada de decisão híbrido, que combina a robustez de métodos de **Tomada de Decisão Multicritério** (*Multiple Attribute Decision Making* – MADM) com a flexibilidade da **lógica fuzzy**. A inovação central desta proposta não reside apenas na integração dessas duas técnicas, mas no uso de **Algoritmos Genéticos (GA)** para otimizar independentemente os parâmetros de ambos os módulos, garantindo um desempenho que rivaliza com abordagens mais complexas.

Essa abordagem visa criar um sistema inteligente que seja, ao mesmo tempo, eficiente, **interpretaível ("White-Box")** e com baixa exigência computacional.

A arquitetura é composta por duas camadas principais:

1. **Tomada de Decisão Estratégica (MADM):** A primeira camada utiliza o método TOPSIS (*Technique for Order of Preference by Similarity to Ideal Solution*). Sua função é analisar múltiplos critérios (como distância de obstáculos, velocidade e regras de trânsito) para selecionar a manobra mais adequada. Os **pesos** desta camada, que definem a importância de cada critério, não são definidos manualmente, mas **otimizados por um Algoritmo Genético** dedicado, treinado por imitação para replicar as decisões de um especialista.

2. **Refinamento da Ação (Lógica Fuzzy):** Uma vez que a decisão estratégica é tomada (ex: "virar à esquerda"), um **controlador fuzzy especialista** (dentre seis) é ativado. Este controlador calcula a intensidade precisa da ação (ex: o ângulo exato de esterçamento). Todos os parâmetros deste controlador (suas funções de pertinência e regras) são **otimizados por um segundo Algoritmo Genético**, treinado para minimizar o erro de regressão em relação às ações do especialista.

A implementação e validação da proposta serão realizadas no simulador **CARLA** (*Car Learning to Act*), uma plataforma de código aberto amplamente utilizada na pesquisa com veículos autônomos, que oferece um ambiente seguro e replicável para testes.

1.3 OBJETIVOS

Este trabalho tem como objetivo principal desenvolver e validar um sistema de tomada de decisão de baixo custo computacional para veículos autônomos, baseado na integração dos métodos TOPSIS e lógica fuzzy.

Para alcançar este objetivo, foram definidos os seguintes **objetivos específicos**:

- Modelar o problema de tomada de decisão veicular considerando múltiplos critérios de segurança e eficiência.
- Implementar o método TOPSIS para a seleção de manobras estratégicas.
- Desenvolver um controlador baseado em lógica fuzzy para o refinamento e execução das manobras.
- Integrar os módulos MADM e fuzzy em uma arquitetura coesa.
- Validar o sistema proposto em cenários de simulação realistas utilizando o simulador CARLA.
- Analisar o desempenho do sistema em termos de segurança, eficiência e carga computacional.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção tem como objetivo apresentar os fundamentos teóricos relacionados ao processo de Tomada de Decisão Multicritério (MADM) e ao método TOPSIS, que será utilizado neste trabalho. Inicialmente, discute-se o conceito de MADM enquanto abordagem voltada à análise de alternativas sob múltiplos critérios, destacando sua relevância em contextos onde diferentes fatores precisam ser considerados de forma conjunta e estruturada. Em seguida, é introduzido o método TOPSIS, uma das técnicas multicritério mais conhecidas, cuja lógica se baseia na comparação das alternativas com soluções possíveis.

2.1 TOMADA DE DECISÃO MULTICRITÉRIO (MADM)

A Tomada de Decisão Multicritério, do inglês *Multiple Attribute Decision Making* (MADM), é um campo de estudo da pesquisa operacional que se dedica a resolver problemas de decisão caracterizados pela necessidade de avaliar um conjunto finito de alternativas com base em múltiplos atributos (ou critérios), que frequentemente são conflitantes entre si. O objetivo central dos métodos MADM é selecionar, ordenar ou classificar as alternativas disponíveis, fornecendo um suporte estruturado e matemático ao processo decisório (Chen; Hwang, 1992).

Em contextos de sistemas autônomos, como a condução veicular, os métodos MADM são particularmente úteis, pois as decisões (alternativas) como "acelerar", "frear"ou "mudar de faixa"dependem da avaliação simultânea de diversos critérios, como a distância para obstáculos, a velocidade atual, as condições da via e as regras de trânsito. Neste trabalho, foi selecionado o método TOPSIS para essa finalidade, devido à sua simplicidade conceitual e eficiência computacional.

2.1.1 TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution)

O método TOPSIS, proposto por Hwang e Yoon (1981), é uma das técnicas MADM mais difundidas e aplicadas. Seu princípio fundamental baseia-se na ideia de que a alternativa ideal a ser escolhida deve ter a menor distância geométrica da *solução ideal positiva* (PIS – *Positive Ideal Solution*) e, simultaneamente, a maior distância da *solução ideal negativa* (NIS – *Negative Ideal Solution*) (Chen; Hwang, 1992).

A PIS é uma solução hipotética que combina os melhores valores de cada critério alcançáveis dentre as alternativas, enquanto a NIS combina os piores valores. O ranqueamento das alternativas é então realizado com base na sua proximidade relativa à PIS. O algoritmo pode ser descrito nos seguintes passos:

1. **Construir a matriz de decisão e normalizá-la:** A matriz de decisão (X) contém os valores de cada alternativa (i) em relação a cada critério (j). O primeiro passo consiste em

normalizar essa matriz para permitir a comparação entre critérios de diferentes unidades. O valor normalizado (r_{ij}) é calculado pela norma vetorial, conforme a Equação 1.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}, \quad \text{para } i = 1, \ldots, m;\ j = 1, \ldots, n \tag{1}$$

2. **Construir a matriz de decisão normalizada ponderada:** Em seguida, os valores normalizados são multiplicados por um vetor de pesos w_j, que representa a importância relativa de cada critério. A soma de todos os pesos deve ser igual a 1. O valor ponderado v_{ij} é dado pela Equação 2.

$$v_{ij} = w_j r_{ij}, \quad \text{com } \sum_{j=1}^{n} w_j = 1 \tag{2}$$

3. **Determinar as soluções ideal e anti-ideal:** A solução ideal positiva (A^*) e a solução ideal negativa (A^-) são determinadas a partir da matriz ponderada. Para cada critério, o melhor valor (máximo para critérios de benefício, mínimo para critérios de custo) compõe A^*, e o pior valor compõe A^-.

$$A^* = \{v_1^*, v_2^*, \ldots, v_n^*\} = \{(\max_i v_{ij} | j \in J), (\min_i v_{ij} | j \in J')\} \tag{3}$$

$$A^- = \{v_1^-, v_2^-, \ldots, v_n^-\} = \{(\min_i v_{ij} | j \in J), (\max_i v_{ij} | j \in J')\} \tag{4}$$

Onde J é o conjunto de critérios de benefício (quanto maior, melhor) e J' é o conjunto de critérios de custo (quanto menor, melhor).

4. **Calcular as medidas de separação:** A distância de cada alternativa para as soluções ideal e anti-ideal é calculada utilizando a distância euclidiana n-dimensional. A separação da solução ideal (S_i^*) e da solução anti-ideal (S_i^-) são dadas pelas Equações 5 e 6, respectivamente.

$$S_i^* = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^*)^2}, \quad \text{para } i = 1, \ldots, m \tag{5}$$

$$S_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2}, \quad \text{para } i = 1, \ldots, m \tag{6}$$

5. **Calcular a proximidade relativa à solução ideal:** A proximidade relativa, ou o coeficiente de performance (C_i^*), de cada alternativa em relação à solução ideal é calculada. Este coeficiente varia entre 0 e 1, onde um valor mais próximo de 1 indica uma melhor performance da alternativa.

$$C_i^* = \frac{S_i^-}{S_i^* + S_i^-}, \quad \text{com } 0 \leq C_i^* \leq 1 \tag{7}$$

6. **Ordenar as alternativas:** Por fim, as alternativas são ordenadas de forma decrescente de acordo com o valor de C_i^*. A alternativa com o maior valor de C_i^* é a mais recomendada.

2.1.1.1 Adaptação do Método para o Problema Proposto

No presente trabalho, o método TOPSIS foi adaptado para se adequar à natureza dinâmica do problema de condução autônoma. A principal modificação reside na construção da matriz de decisão, que não é estática, mas sim gerada em tempo real a cada ciclo de decisão do sistema.

A formulação foi modificada para utilizar uma **matriz de pesos** ((W)) e um **vetor de atributos** ((I)) de entrada.

- A matriz de pesos (W), de dimensão $m \times n$, define a importância de cada atributo (j) para cada alternativa (i). Isso permite que um mesmo atributo (ex: distância do veículo à frente) tenha relevâncias diferentes para as alternativas "acelerar"e "frear".
- O vetor de atributos (I), de dimensão (n), contém os valores numéricos dos sensores e informações do ambiente em um determinado instante.

A matriz de decisão ((D)), que serve de entrada para o passo de normalização, é então construída dinamicamente através da multiplicação elemento a elemento (broadcast) do vetor de atributos por cada linha da matriz de pesos, onde cada valor d_{ij} da matriz de decisão é dado por:

$$d_{ij} = w_{ij} \cdot I_j \tag{8}$$

A partir da construção da matriz (D), os passos subsequentes do algoritmo TOPSIS (normalização, determinação das soluções PIS/NIS, cálculo das distâncias e da proximidade relativa) seguem a formulação padrão. Adicionalmente, a implementação computacional inclui tratamentos para evitar divisões por zero durante o processo de normalização e no cálculo da proximidade relativa, garantindo a estabilidade numérica do sistema.

2.2 LÓGICA FUZZY

A lógica fuzzy, ou lógica difusa, foi introduzida por Lotfi Zadeh em 1965 como uma extensão da lógica clássica booleana. Seu objetivo é tratar a incerteza e a imprecisão inerentes a muitos problemas do mundo real, que são difíceis de modelar com a dualidade verdadeiro/falso (1 ou 0) da lógica tradicional. Em vez disso, a lógica fuzzy permite a existência de graus de verdade, representando o raciocínio humano de forma mais natural e permitindo que sistemas computacionais lidem com variáveis linguísticas como "perto", "rápido"ou "quente"(Zadeh, 1965).

No contexto de veículos autônomos, a lógica fuzzy é particularmente útil para o controle fino de ações, como a suavidade de uma curva ou a intensidade de uma frenagem, onde as decisões não são estritamente binárias, mas sim graduais.

2.2.1 Conjuntos e Variáveis Linguísticas

A base da lógica fuzzy reside no conceito de **conjunto fuzzy** (*fuzzy set*), que difere do conjunto clássico (ou *crisp*). Em um conjunto clássico, um elemento pertence ou não pertence ao conjunto. Em um conjunto fuzzy, um elemento pode ter um grau de pertencimento parcial.

Esse grau de pertencimento é definido pela **Função de Pertinência** (*Membership Function* – MF), denotada por $\mu\_A(x)$. Essa função mapeia cada elemento (x) de um universo de discurso (U) para um valor no intervalo real [0, 1]. Se $\mu\_A(x) = 1$, o elemento (x) pertence totalmente ao conjunto (A). Se $\mu\_A(x) = 0$, ele não pertence. Valores intermediários, como 0.7, indicam um pertencimento parcial.

As Funções de Pertinência são usadas para modelar **variáveis linguísticas** e seus termos. Uma variável linguística, como "Distância", pode ser descrita por termos como "Perto", "Médio"e "Longe". Cada um desses termos é representado por um conjunto fuzzy e sua respectiva MF, como ilustrado na Figura 1. As formas mais comuns para MFs são a triangular, a trapezoidal e a gaussiana.

Exemplo de Funções de Pertinência para a Variável "Distância

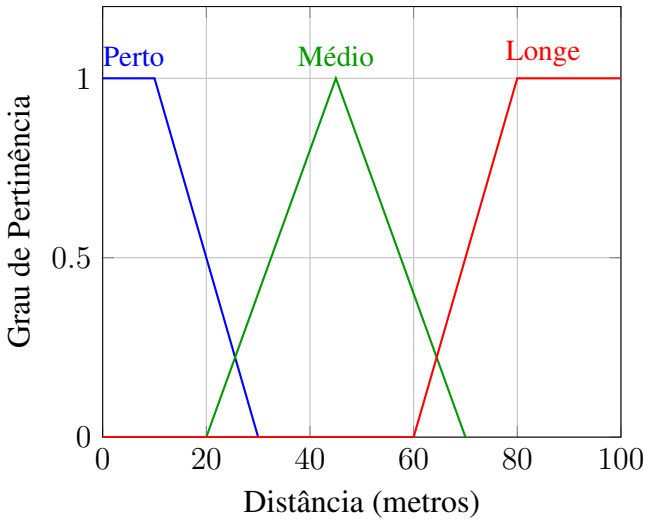


Figura 1 – Representação de uma variável linguística "Distância"com três termos ("Perto", "Médio", "Longe"), cada um modelado por uma Função de Pertinência.

2.2.2 Sistema de Inferência Fuzzy (FIS)

Um Sistema de Inferência Fuzzy (*Fuzzy Inference System* – FIS) é a estrutura que utiliza os conceitos de conjuntos fuzzy para mapear entradas numéricas em saídas numéricas. Este processo é geralmente dividido em três etapas principais: fuzzificação, inferência e defuzzificação.

2.2.2.1 Fuzzificação

A fuzzificação é o processo de converter uma entrada numérica real (ou *crisp*) em um grau de pertencimento a um ou mais conjuntos fuzzy. Por exemplo, se o sistema recebe uma entrada de um sensor indicando uma distância de 25 metros, o processo de fuzzificação, utilizando as MFs da Figura 1, determinaria o grau de pertencimento desse valor aos conjuntos "Perto"e "Médio". O resultado poderia ser, por exemplo, $\mu_{\text{Perto}}(25) = 0.25$ e $\mu_{\text{Médio}}(25) = 0.2$.

2.2.2.2 Motor de Inferência e Regras Fuzzy

O motor de inferência é o núcleo do sistema e processa as entradas fuzzificadas com base em um conjunto de regras linguísticas, previamente definidas por um especialista. Essas regras são tipicamente expressas no formato **SE-ENTÃO** (*IF-THEN*). Por exemplo:

- **Regra 1:** SE a *Distância* é **Perto** E a *Velocidade* é **Alta** ENTÃO a *Ação* é **Frear Forte**.
- **Regra 2:** SE a *Distância* é **Média** ENTÃO a *Ação* é **Manter Velocidade**.

O sistema avalia a parte "SE"(antecedente) de cada regra usando operadores fuzzy (como MÍNIMO para "E"e MÁXIMO para "OU") para obter um único valor de verdade. Esse valor é então usado para "ativar"a parte "ENTÃO"(consequente) da regra, geralmente "cortando"a MF do conjunto de saída à altura do valor de ativação. O resultado de todas as regras ativadas é combinado (agregado) em um único conjunto fuzzy de saída.

2.2.2.3 Defuzzificação

A defuzzificação é a etapa final, na qual o conjunto fuzzy de saída, resultante da agregação das regras, é convertido de volta em um único valor numérico (*crisp*). Este valor representa a ação final a ser executada pelo sistema (e.g., aplicar 35% de força no freio).

Existem vários métodos de defuzzificação, sendo o mais comum o método do **Centroide** ou **Centro de Gravidade (CoG)**. Este método calcula o "ponto de equilíbrio"da área do conjunto fuzzy de saída agregado, resultando em um valor que representa a média ponderada de todas as possibilidades de saída. Outros métodos incluem o Centro da Área (CoA), a Média dos Máximos (MoM) e o Máximo da Direita/Esquerda.

2.3 ALGORITMOS GENÉTICOS (GA)

Algoritmos Genéticos (AGs) são meta-heurísticas de busca e otimização inspiradas no processo de seleção natural e evolução biológica. Propostos por John Holland, eles operam sobre uma população de soluções candidatas (indivíduos), aplicando operadores genéticos como seleção, cruzamento (recombinação) e mutação para evoluir iterativamente em direção a soluções de melhor qualidade (Goldberg, 1989).

Um AG é particularmente eficaz em problemas de otimização complexos, não-lineares e multimodais. O processo básico envolve:

1. **População Inicial:** Geração de um conjunto de indivíduos (cromossomos).
2. **Função de Aptidão (*Fitness*):** Avaliação de cada indivíduo.
3. **Seleção:** Indivíduos com maior aptidão são selecionados como "pais".
4. **Cruzamento (*Crossover*):** Os "pais"combinam seu material genético para criar "filhos".
5. **Mutação:** Alterações aleatórias são introduzidas nos filhos para manter a diversidade.

Este ciclo se repete por várias gerações, e a população converge para uma solução de alta aptidão.

2.4 APRENDIZADO POR IMITAÇÃO (IMITATION LEARNING)

O Aprendizado por Imitação (*Imitation Learning* - IL) é uma abordagem onde um agente aprende a executar uma tarefa observando demonstrações de um especialista (Codevilla et al., 2018). Em vez de aprender por tentativa e erro (como no Aprendizado por Reforço), o agente tenta mimetizar o comportamento do especialista, aprendendo a mapear estados (entradas de sensor) para ações (comandos de controle).

Uma técnica comum de IL é a **Clonagem Comportamental** (*Behavioral Cloning*), onde o problema é tratado como um aprendizado supervisionado, usando um dataset de pares (estado, ação) do especialista.

2.4.1 Clonagem Comportamental via Algoritmo Genético

Neste trabalho, o Aprendizado por Imitação é a *estratégia de treinamento* do Algoritmo Genético. Em vez de usar um modelo de aprendizado de máquina tradicional (como uma rede neural) para clonar o especialista, nós usamos o GA para encontrar os **parâmetros** de um modelo "caixa-branca"(MADM ou Fuzzy) que melhor imita o especialista.

O processo funciona da seguinte forma:

1. **Dataset do Especialista:** Um agente especialista (privilegiado) é executado no simulador CARLA para gerar um grande dataset de pares (estado, ação). O "estado"são os atributos (ex: `distancia_obstaculo`, `velocidade_erro`) e a "ação"são os comandos reais (ex: `expert_throttle`, `expert_steer`).
2. **Função de Aptidão (Fitness):** A aptidão de um indivíduo do GA (seja um conjunto de pesos MADM ou um sistema Fuzzy) é calculada comparando sua saída com a saída do especialista no dataset.

3. **Treinamento Offline:** O GA evolui sua população "offline", usando apenas o dataset, sem a necessidade de rodar a simulação a cada geração. O objetivo do GA é encontrar os parâmetros que minimizam o erro entre sua decisão e a decisão do especialista para o mesmo estado.

Esta abordagem híbrida (GA + IL) permite otimizar sistemas interpretáveis (MADM/Fuzzy) usando os mesmos dados que seriam usados para treinar uma rede neural "caixa-preta".

2.5 MODELOS DE BASELINE: RANDOM FOREST (RF)

Para validar a eficácia dos modelos "caixa-branca"propostos (MADM e Fuzzy), é essencial compará-los com um *baseline* (modelo de referência) de aprendizado de máquina padrão. Neste trabalho, foi escolhido o **Random Forest (RF)**, um modelo de *ensemble* que utiliza múltiplas árvores de decisão.

O RF é conhecido por sua alta acurácia e capacidade de lidar com dados não-lineares. Ele serve a dois propósitos:

1. **Definir o Teto de Desempenho:** O RF nos diz qual é a acurácia máxima alcançável *com os atributos disponíveis*. Se o RF atinge 93% de acurácia e nosso MADM atinge 73%, isso prova que o problema não está nos atributos, mas sim na incapacidade do modelo MADM (linear) de capturar a complexidade não-linear do problema.

2. **Análise de Relevância:** O RF fornece uma métrica de **Importância de Atributos (*Feature Importance*)**, que foi crucial para identificar atributos irrelevantes (ex: sensores que não mudavam de valor) e focar a otimização do GA nos atributos que realmente importam.

2.6 PLATAFORMA DE SIMULAÇÃO CARLA

A validação de algoritmos de condução autônoma em ambientes reais é um processo caro, perigoso e de difícil replicação. Para superar esses desafios, a comunidade científica utiliza simuladores de código aberto. Neste trabalho, foi escolhido o **CARLA** (*Car Learning to Act*), um simulador desenvolvido especificamente para a pesquisa em condução autônoma (Dosovitskiy et al., 2017).

Construído sobre o motor gráfico *Unreal Engine 4*, o CARLA oferece alta fidelidade na simulação de ambientes urbanos, dinâmicas físicas de veículos e condições ambientais (clima, iluminação). Sua arquitetura cliente-servidor permite que os algoritmos de controle (o "cérebro"do veículo) rodem como um cliente (em Python ou C++), que se comunica com o servidor de simulação para receber dados de sensores e enviar comandos de atuação.

2.6.1 Sensores e Percepção

O CARLA fornece um conjunto abrangente de sensores que mimetizam o hardware encontrado em veículos autônomos reais. Para o sistema MADM+Fuzzy proposto, que depende de um vetor de atributos (I) sobre o estado do ambiente, os seguintes sensores são fundamentais:

- **Sensor LiDAR (*Light Detection and Ranging*):** Emite feixes de laser para criar uma nuvem de pontos 3D do ambiente. É o sensor primário para calcular atributos cruciais, como a "distância do obstáculo à frente"e os indicadores de "perigo"lateral.
- **Sensor de Colisão:** Um sensor que registra eventos de colisão, fundamental para a avaliação de segurança do agente.
- **Sensor de Invasão de Faixa (*Lane Invasion*):** Detecta quando o veículo cruza as marcações da pista, usado para penalizar o agente durante a avaliação (exceto em mudanças de faixa intencionais).
- **Sensores do Veículo (Estado Interno):** O próprio veículo fornece dados como velocidade atual, aceleração e localização (via GNSS simulado), que são entradas diretas para os sistemas MADM e Fuzzy.
- **API do Simulador (Agente Privilegiado):** Para o *dataset* do especialista e para o cálculo de certos atributos (como `direcao_erro`), o agente tem acesso privilegiado à API do CARLA, como a rota do planejador global e o estado dos semáforos, informações que um sistema de percepção real teria que extrair de câmeras.

3 REVISÃO BIBLIOGRÁFICA

Este capítulo analisa o panorama atual da pesquisa em tomada de decisão para veículos autônomos, posicionando a arquitetura híbrida proposta neste trabalho frente às abordagens estabelecidas na literatura.

3.1 ESTADO DA ARTE

A literatura sobre condução autônoma é vasta, mas pode ser categorizada em duas vertentes principais: abordagens monolíticas (end-to-end) e abordagens modulares ou híbridas.

3.1.1 1. Abordagens Monolíticas (End-to-End)

A abordagem *end-to-end* (E2E) tornou-se proeminente com o avanço do Aprendizado Profundo. O objetivo é treinar um único modelo, geralmente uma Rede Neural Convolucional (CNN) ou um modelo de Aprendizado por Reforço Profundo (DRL), para mapear diretamente a entrada de sensores (ex: pixels de câmeras) aos comandos de atuação do veículo (ex: esterçamento, aceleração).

Trabalhos seminais como o de Codevilla et al. (2018) (Codevilla et al., 2018) popularizaram o **Aprendizado por Imitação** (IL) no simulador CARLA (Dosovitskiy et al., 2017). Nesses modelos, o agente aprende a clonar o comportamento de um motorista especialista (humano ou algorítmico).

- **Relação com este trabalho:** Essa abordagem representa o *baseline* de alto custo computacional que este trabalho visa evitar. O modelo E2E, embora potente, opera como uma "caixa-preta", dificultando a validação de segurança e exigindo alto poder de processamento (GPU) para inferência, o que é contrário ao objetivo de baixo custo da nossa proposta.

3.1.2 2. Abordagens Híbridas e Modulares

Para superar os problemas de interpretabilidade e custo, arquiteturas híbridas são uma tendência forte. Elas decompõem o problema em módulos (percepção, planejamento, controle).

A **Lógica Fuzzy** é uma técnica clássica e robusta para o módulo de controle. Yilmaz et al. (2017) (Yilmaz; Ayan; Adak, 2017), por exemplo, demonstraram um sistema de controle veicular autônomo baseado puramente em lógica fuzzy no simulador TORCS. O sistema utiliza regras linguísticas para desvio de obstáculos e controle de velocidade.

- **Relação com este trabalho:** Este tipo de artigo valida o uso da lógica fuzzy (Seção 2.2) como um controlador de baixo nível eficaz, robusto e computacionalmente leve, justificando sua escolha para o módulo de refinamento de ação em nossa arquitetura.

Uma evolução das abordagens híbridas combina IA com controladores clássicos. Wang et al. (2025) (Wang et al., 2025) propõem uma arquitetura hierárquica que integra DRL com Lógica Fuzzy. O DRL atua no nível superior, tomando decisões estratégicas, enquanto o controlador fuzzy cuida da execução de baixo nível.

- **Relação com este trabalho:** A arquitetura de (Wang et al., 2025) é análoga à nossa proposta, com uma diferença fundamental: substituímos o módulo estratégico de DRL (uma "caixa-preta"que requer treinamento intensivo) por um método de **Tomada de Decisão Multicritério (MADM)**.

3.1.3 3. Métodos MADM em Decisão

O uso de MADM para seleção de arquiteturas de IA em veículos autônomos foi explorado por Rath et al. (2024) (Rath et al., 2024). Paralelamente, métodos como o **TOPSIS** são amplamente utilizados em problemas de decisão gerencial, como a seleção de fornecedores, muitas vezes combinados com lógica fuzzy para lidar com incertezas (Fuzzy TOPSIS), como demonstrado por Vahdani et al. (2024) (Vahdani; Keymanesh; Salimi, 2024).

- **Relação com este trabalho:** Esses trabalhos validam o TOPSIS (Seção 2.1.1) como uma ferramenta robusta para problemas de decisão complexos. A **principal contribuição** e inovação deste TCC é a *adaptação* do TOPSIS: em vez de usá-lo para uma decisão estática (como escolher um fornecedor ou uma arquitetura), nós o aplicamos para a seleção de *manobras táticas em tempo real* (ciclo a ciclo), um domínio dinâmico raramente explorado por essa técnica.

3.2 POSICIONAMENTO DA PROPOSTA

Este trabalho se posiciona como uma arquitetura híbrida (GA-TOPSIS-Fuzzy) que busca o "meio-termo"entre as abordagens E2E e as clássicas. A proposta combina:

- A interpretabilidade e baixo custo da **Lógica Fuzzy** (Yilmaz; Ayan; Adak, 2017).
- A estrutura hierárquica de decisão (estratégia + controle) vista em trabalhos modernos (Wang et al., 2025).
- A robustez matemática do **TOPSIS** (Vahdani; Keymanesh; Salimi, 2024).
- A otimização de parâmetros via **Algoritmos Genéticos** (Seção 2.3).

O resultado é um sistema de decisão interpretável, de baixo custo computacional e otimizado, aplicado a um problema dinâmico de condução autônoma.

4 METODOLOGIA

Este capítulo detalha a arquitetura do sistema proposto, as ferramentas de implementação, e o processo de coleta de dados e otimização para validar a solução.

4.1 ARQUITETURA DO SISTEMA

A arquitetura de controle é modular e hierárquica, operando em um ciclo contínuo de percepção-decisão-ação. O fluxo de dados em cada ciclo de decisão é o seguinte:

1. **Percepção e Atributos:** Sensores (LiDAR, câmera, etc.) e dados internos do simulador são processados em um Vetor de Atributos (I) unificado, que descreve o estado atual (ex: '$distancia\_obstaculo$', '$velocidade\_erro$', '$direcao\_erro$').

2. **Decisão Estratégica (*MADM-TOPSIS*):** O Vetor I é alimentado em dois sistemas *TOPSIS* paralelos, que utilizam matrizes de pesos (W_{vel} e W_{dir}) pré-otimizadas:

   - O ***MADM* de Velocidade** (Seção 4.1.1.1) seleciona a melhor alternativa estratégica longitudinal (ex: "$acelerar$", "$reduzir$").
   - O ***MADM* de Direção** (Seção 4.1.1.2) seleciona a melhor alternativa estratégica lateral (ex: "$Virar\ a\ direita$").

3. **Refinamento da Ação (Controlador *Fuzzy* Especialista):** A decisão estratégica do *MADM* é usada para selecionar o controlador *Fuzzy* apropriado. Por exemplo, se o *MADM* decide "acelerar"e "Virar a direita":

   - O sistema `fuzzy_acelerar` é ativado e, usando os dados do Vetor I, calcula a intensidade exata do *throttle*.
   - O sistema `fuzzy_virar_direita` é ativado e calcula o ângulo exato de *steer*.

4. **Atuação (CARLA):** Os comandos finais de *throttle*, *brake* e *steer* são enviados ao veículo.

4.1.1 Definição dos Módulos *MADM*

Os módulos de decisão estratégica utilizam a formulação *TOPSIS* adaptada (Equação 8). As alternativas e atributos de entrada (Vetor I) são:

4.1.1.1 MADM de Velocidade

- *Alternativas (4):*
  - "parar"
  - "manter"
  - "reduzir"
  - "acelerar"
- *Atributos (9):*
  - velocidade_erro
  - distância_var
  - distância_obstáculo
  - distância_alvo
  - estado_da_pista
  - estado_de_trafego
  - visibilidade
  - magnitude_da_subida
  - magnitude_da_descida

4.1.1.2 MADM de Direção

- *Alternativas (3):*
  - "Manter a direção"
  - "Virar a esquerda"
  - "Virar a direita"
- *Atributos (7):*
  - Manter_é_seguro
  - Rota_manda_virar_direita
  - Pista_curva_para_direita
  - Seguro_virar_direita
  - Rota_manda_virar_esquerda
  - Pista_curva_para_esquerda
  - Seguro_virar_esquerda

4.1.2 Definição dos Controladores *Fuzzy*

Para o refinamento das ações, seis controladores *Fuzzy* especialistas foram desenvolvidos e otimizados (Seção 4.3.2), cobrindo as principais manobras de condução. Cada sistema possui 5 funções de pertinência (*MFs*) trapezoidais em sua saída para garantir granularidade e suavidade no controle.

4.1.2.1 Limitação de Escopo: A Ação "$Parar$"

A alternativa "$Parar$", embora seja uma saída válida do módulo *MADM*-Vel, não teve um controlador *Fuzzy* especialista otimizado neste trabalho. Para fins de implementação, ela é tratada como uma ação de frenagem máxima (‘brake=1.0‘).

Reconhece-se que esta é uma simplificação. Um sistema de parada robusto (*fail-safe*) deveria, idealmente, modular a intensidade da frenagem com base em critérios adicionais, como a velocidade atual (para evitar travamento de rodas), as condições da pista (ex: pista molhada) e o tráfego traseiro. A implementação desse controlador de parada de emergência foi considerada fora do escopo principal deste trabalho, que foca na otimização das manobras de condução contínua.

4.1.2.2 Controladores Otimizados

Os seis sistemas fuzzy otimizados pelo *GA* são:

- **Controle Longitudinal:**
  - $fuzzy\_acelerar$: (Entradas: $erro\_vel$, $dist\_obs$, $pista$, $visibilidade$) $\rightarrow$ Saída: $throttle$.
  - $fuzzy\_reduzir$: (Entradas: $erro\_vel$, $dist\_obs$, $pista$) $\rightarrow$ Saída: $brake$.
  - $fuzzy\_manter\_vel$: (Entradas: $erro\_vel$, $dist\_obs$) $\rightarrow$ Saída: $ajuste$ ($throttle/brake$).
- **Controle Lateral:**
  - $fuzzy\_virar\_direita$: (Entradas: $erro\_dir$, $perigo$, $velocidade$) $\rightarrow$ Saída: $steer$ [0, 1].
  - $fuzzy\_virar\_esquerda$: (Entradas: $erro\_dir$, $perigo$, $velocidade$) $\rightarrow$ Saída: $steer$ [-1, 0].
  - $fuzzy\_manter\_dir$: (Entradas: $erro\_dir$, $velocidade$) $\rightarrow$ Saída: $steer$ [-0.06, 0.06].

4.1.2.3 Exemplo de Execução do Fluxo de Decisão

Para ilustrar o fluxo de 4 passos descrito na Seção 4.1, a seguir é detalhada a execução de uma única amostra de decisão do *dataset* de validação. O processo demonstra como os dados brutos são transformados em uma ação final.

PASSO 1: *INPUTS MADM* (Do *Dataset*) Valores brutos dos atributos de entrada:

- `Atributos VEL (Longitudinal):`
  - `velocidade_erro: 16.8330`
  - `distancia_var: 11.3361`
  - `distancia_obstaculo: 46.6449`
  - `distancia_alvo: 148.6723`
  - `estado_da_pista: 8.0000`

  - `estado_de_trafego: 0.0000`
  - `visibilidade: 8.0000`
  - `magnitude_da_subida: 0.0000`
  - `magnitude_da_descida: 0.0000`
- *`Atributos DIR (Lateral):`*
  - `manter_e_seguro: 1.0`
  - `rota_manda_virar_direita: 0.0`
  - `pista_curva_para_direita: 0.0`
  - `seguro_virar_direita: 0.0`
  - `rota_manda_virar_esquerda: 1.0`
  - `pista_curva_para_esquerda: 1.0`
  - `seguro_virar_esquerda: 1.0`

PASSO 2: DECISÃO MADM (Calculada) O TOPSIS processa os atributos e seleciona as estratégias com maior score:

- `VEL: acelerar` (Scores: [0.661, 0.383, 0.309, **0.807**])
- `DIR: Virar a esquerda` (Scores: [0.417, 0.152, **0.721**])

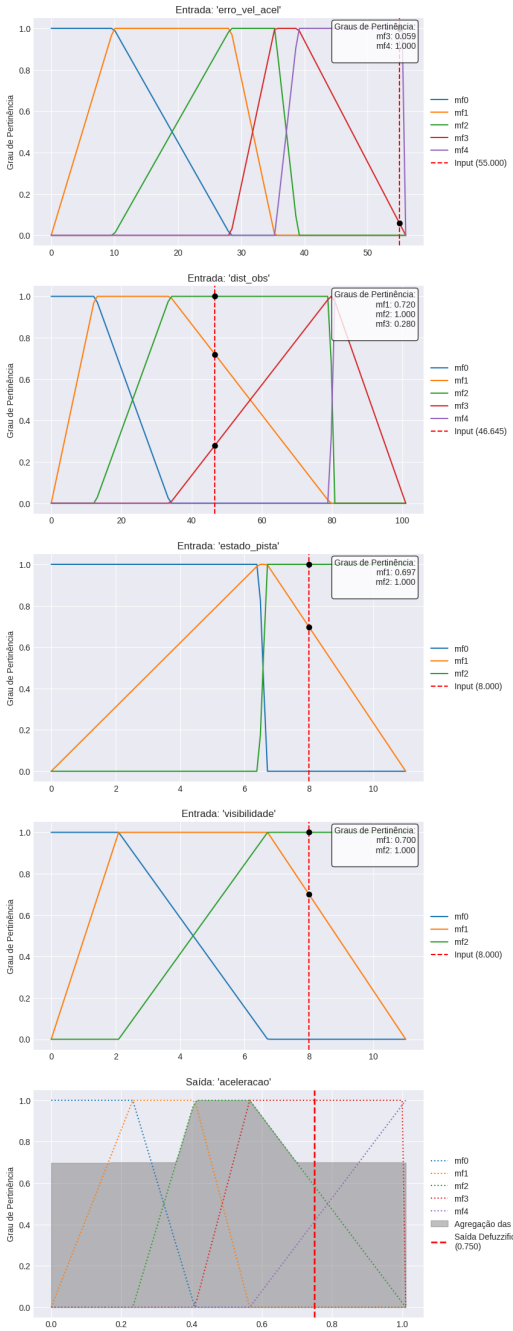
PASSO 3: *INPUTS* FUZZY (Pré-processados) Os controladores ativados (acelerar, virar_esquerda) recebem seus inputs específicos:

- `Sistema VEL (acelerar):`
  - `"erro_vel_acel": 55.0`
  - `"dist_obs": 46.645`
  - `"estado_pista": 8.0`
  - `"visibilidade": 8.0`
- `Sistema DIR (virar_esquerda):`
  - `"erro_dir_evasao": 45.0`
  - `"perigo_evasao": 0.021`
  - `"velocidade": 3.167`

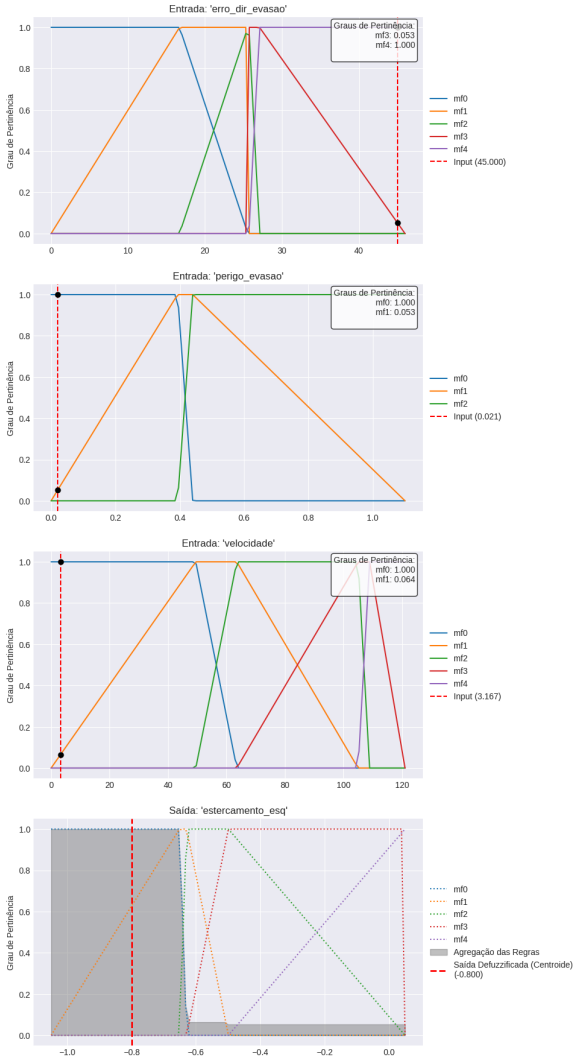
PASSO 4: SAÍDA *FUZZY* (Defuzzificada) Os sistemas *Fuzzy* calculam a ação final:

- `Throttle: 0.750`
- `Steer: -0.800`

As Figuras 2a e 2b detalham visualmente o processo interno do Passo 4. Elas mostram como as entradas pré-processadas (ex: "erro_dir_evasao"= 45.0) são fuzzificadas, agregadas de acordo com as regras, e por fim defuzzificadas (via Centroide) para gerar os comandos finais de *throttle* (0.750) e *steer* (-0.800).



(a) Controlador `fuzzy_acelerar`.



(b) Controlador `fuzzy_virar_esquerda`.

Figura 2 – Exemplos de execução dos controladores *Fuzzy* (Passo 4 da arquitetura).

4.2 COLETA E PRÉ-PROCESSAMENTO DE DADOS

Para treinar os módulos *MADM* e *Fuzzy*, um *dataset* foi gerado no CARLA usando a abordagem de Clonagem Comportamental (Seção 2.4). Um agente especialista (privilegiado) foi executado em diversas rotas, e um *script data collector* salvou 100.000 amostras de dados (estado, ação).

O "estado" bruto continha informações do simulador, como ‘velocidade_erro‘ (em km/h) e ‘direcao_erro‘ (em radianos). A "ação" continha os comandos do especialista (‘expert_throttle‘,

‘expert_steer‘, ‘expert_brake‘).

4.2.1 Pré-processamento para Otimização *Fuzzy*

Os dados brutos não são adequados para os controladores fuzzy, que esperam universos de discurso específicos (ex: erro de -10 a 10). A lógica de pré-processamento do agente foi aplicada a todo o dataset para criar as colunas de entrada que os sistemas *fuzzy* utilizam (ex: ‘erro_vel_manter_input‘, ‘erro_dir_evasao_input‘).

4.2.2 Pré-filtragem do *Dataset*

Para otimizar os 6 controladores *fuzzy* especialistas de forma eficiente, o dataset pré-processado de 100.000 linhas foi dividido em 6 subsets independentes. Cada subset contém apenas as linhas em que a ação do especialista corresponde àquele controlador. Por exemplo, o subset `virar_direita` contém apenas as linhas onde ‘$expert\_steer > 0.05$‘. Isso permitiu que cada GA especialista focasse apenas nos dados relevantes para sua tarefa.

4.3 OTIMIZAÇÃO VIA ALGORITMOS GENÉTICOS DECOPLADOS

A otimização dos parâmetros do sistema (pesos *MADM* e *MFs*/Regras Fuzzy) foi o principal desafio metodológico. Uma abordagem inicial de otimização conjunta (um único *GA* para tudo) falhou, pois as funções de fitness de velocidade e direção entravam em conflito, causando estagnação.

A solução foi **decoplar** (separar) o processo em dois GAs distintos e independentes, ambos treinados offline usando o *dataset* de 100.000 amostras.

4.3.1 *GA-MADM*: Otimização dos Pesos *TOPSIS*

O primeiro GA foi projetado para encontrar as matrizes de peso ideais (W_{vel} e W_{dir}) para o MADM.

- **Cromossomo:** Um indivíduo representava as duas matrizes de peso.
- ***Fitness* (Acurácia de Classificação):** A aptidão foi calculada como a acurácia de classificação. Para criar o "gabarito"(a decisão-alvo) para o *GA*, os comandos contínuos do especialista (ex: `expert_throttle`, `expert_brake`, `expert_steer`) foram primeiro rotulados em decisões estratégicas discretas.
  - Isso foi feito usando limiares pré-definidos: por exemplo, `brake > 0.05` era mapeado para "reduzir", `throttle > 0.1` para "acelerar", e `steer > 0.05` para "Virar a direita".

Para cada linha do *dataset*, a decisão do *MADM* (com os pesos do indivíduo) era então comparada com esse rótulo-alvo do especialista. O fitness era a porcentagem de acertos ($(acertos\_vel + acertos\_dir)/2$).

- **Técnica de Amostragem:** Para evitar *overfitting* a um subconjunto de dados (o que causou a estagnação inicial em 70% de acurácia), foi implementada uma amostragem dinâmica: a cada geração, o fitness era calculado sobre um novo *sample* aleatório de 15% do dataset.

4.3.2 GA-Fuzzy: Otimização dos Controladores Especialistas

O segundo *GA* foi projetado para otimizar os 6 controladores *fuzzy* de forma paralela.

- **Arquitetura:** O *GA* gerenciou 6 populações independentes, uma para cada sistema especialista (ex: ‘pop_acelerar‘, ‘pop_virar_direita‘).
- **Cromossomo (Nós e Regras):** Para garantir a robustez e a partição *fuzzy*, o *GA* não otimizou diretamente os 4 parâmetros de cada MF trapezoidal. Em vez disso, ele otimizou os **pontos de inflexão (nós)** entre as MFs. O cromossomo de um indivíduo era composto por:
  1. **Dicionário de Nós:** Mapeava cada variável de entrada/saída para uma lista de seus nós internos otimizáveis (ex: ‘’velocidade’: [35.2, 70.1, ...], ...‘).
  2. **Vetor de Regras:** Um vetor de inteiros (‘[0, 1, 4, ...]‘) que define qual MF de saída (ex: ‘mf0‘, ‘mf1‘, ‘mf4‘) é ativada por cada uma das regras de entrada.
- ***Fitness* (Erro de Regressão):** A aptidão foi baseada no **Erro Absoluto Médio (MAE)**. Para cada indivíduo, seu sistema *fuzzy* correspondente era executado sobre o *subset* de dados relevante (Seção 4.2.2). O MAE era calculado entre a saída *crisp* do fuzzy (ex: ‘$throttle = 0.8$‘) e o valor real do especialista (ex: ‘$expert_throttle = 0.75$‘). O *fitness* final foi ‘$1/(1 + MAE)$‘, convertendo um problema de minimização em maximização.

4.4 CONSOLIDAÇÃO E VALIDAÇÃO

Após a conclusão dos GAs, os resultados foram consolidados e validados.

4.4.1 Consolidação dos Modelos

Dois *scripts* de consolidação foram executados:

1. **‘consolidate_madm_weights.py‘:** Extraiu o melhor indivíduo (melhor acurácia) do ‘ranking_ga_vel.csv‘ e ‘ranking_ga_dir.csv‘ e os salvou como ‘weights_vel.csv‘ e ‘weights_dir.csv‘, prontos para uso pelo agente.

2. **‘consolidate_fuzzy_models.py‘:** Extraiu o melhor indivíduo (menor MAE) de cada um dos 6 rankings do *GA-Fuzzy*. Ele então realizou uma **otimização estrutural**, agrupando todas as regras de entrada (antecedentes) que apontavam para a mesma saída (consequente). O resultado foi um único ‘fuzzy_models_final.json‘ contendo 6 sistemas fuzzy, cada um com 5 "super-regras"(uma para cada MF de saída), que são construídas no agente usando o operador lógico ‘OR‘.

4.4.2 Validação *Offline* (Dataset Completo)

Antes de ir para a simulação, um *script* de validação (‘evaluate_final_system.py‘) foi executado. Este script simulou a cadeia de decisão completa (‘*Dataset -> MADM -> Fuzzy ->* Avaliação‘) em todas as 100.000 linhas do *dataset*.

Seu objetivo foi calcular o MAE geral do sistema híbrido e identificar **Falhas Críticas** — pontos onde a decisão do MADM estava incorreta e conflitava com a lógica do especialista (ex: MADM decidiu "parar"quando o especialista queria "acelerar"), gerando erros de regressão muito altos.

4.4.3 Protocolo de Avaliação em Simulação

A validação final do sistema consolidado é realizada no simulador CARLA, em uma rota de teste (Spawn Point 12 para 50) que o agente não viu durante o treinamento. O desempenho é medido através de métricas de segurança e eficiência, e comparado com os modelos de *baseline* (*Random Forest* e Agente Especialista).

5 RESULTADOS

Este capítulo apresenta os resultados obtidos na otimização dos módulos MADM e Fuzzy, a análise de desempenho do sistema híbrido na validação *offline* e as observações da execução final na simulação CARLA.

5.1 RESULTADOS DA OTIMIZAÇÃO DOS MÓDULOS

5.1.1 *Baseline*: *Random Forest* e Análise de Atributos

Antes da otimização, um modelo *Random Forest (RF)* foi treinado no dataset de 100.000 amostras para estabelecer um *baseline* de desempenho e analisar a relevância dos atributos.

O RF demonstrou alta eficácia, alcançando **92.7% de acurácia** na classificação da decisão de velocidade e **73.8%** na decisão de direção. O resultado da direção (73.8%) sugere que os atributos de entrada para essa decisão são ambíguos ou insuficientes. Em contraste, a alta acurácia da velocidade (92.7%) prova que os atributos de velocidade (como `velocidade_erro` e `distancia_obstaculo`) são **altamente preditivos** e suficientes para uma tomada de decisão correta.

A análise de *Feature Importance* do RF também revelou que um subconjunto de atributos (ex: `velocidade_erro`, `dist_obs`) era dominante, enquanto outros (ex: `estado_pista`, `visibilidade` em certos contextos) tinham impacto quase nulo, permitindo focar a otimização dos *GAs*.

5.1.2 Desempenho da Otimização *GA-MADM*

O Algoritmo Genético treinado para otimizar os pesos do *MADM* (Seção 4.3.1) apresentou convergência, mas atingiu um platô de desempenho. Mesmo com a implementação da amostragem dinâmica para evitar *overfitting*, o GA estagnou em uma acurácia de classificação de aproximadamente **73%-74%** para ambas as decisões (velocidade e direção).

Este resultado está alinhado com o desempenho do RF na tarefa de direção, mas significativamente abaixo dos 92.7% alcançados pelo RF na tarefa de velocidade. Isso expõe uma limitação fundamental do método *TOPSIS*: sendo um modelo de agregação linear ponderada, ele falha em capturar as relações não-lineares complexas que o especialista usa para decidir a velocidade, resultando em um teto de desempenho inferior ao de um modelo não-linear como o RF.

5.1.3 Desempenho da Otimização *GA-Fuzzy*

A otimização dos 6 controladores Fuzzy especialistas (Seção 4.3.2) foi bem-sucedida. O GA de cada sistema convergiu de forma independente, minimizando o Erro Absoluto Médio (*MAE*) em relação aos comandos do especialista em seus respectivos *subsets* de dados.

Por exemplo, o sistema `fuzzy_acelerar` convergiu para um MAE de aproximadamente 0.08, indicando que sua saída de *throttle* estava, em média, apenas 8% desalinhada do comando do especialista. O GA demonstrou ser eficaz em otimizar os nós das MFs e as regras para replicar o comportamento de controle fino.

5.1.4 Estrutura dos Controladores *Fuzzy* Otimizados

O desempenho de baixo *MAE* descrito na seção anterior foi alcançado pela estrutura de Funções de Pertinência (*MFs*) moldada pelo *GA-Fuzzy* (Seção 4.3.2). Diferentemente da abordagem clássica, onde as MFs são distribuídas uniformemente e simetricamente pelo universo de discurso, a otimização genética resultou em formas trapezoidais assimétricas e irregulares, adaptadas especificamente para reagir às nuances dos dados do especialista.

As Figuras 3 a 8 apresentam a configuração final das *MFs* de entrada e saída para os seis controladores especialistas.

Uma análise visual dessas curvas revela um comportamento importante: a **centralização dos pontos de inflexão (nós)**. Observa-se que o algoritmo genético tendeu a agrupar os vértices das MFs em regiões específicas do eixo X, deixando grandes porções do universo de discurso com comportamento linear constante (saturado).

Por exemplo, no controlador `fuzzy_reduzir` (Figura 4), a atividade das MFs de entrada concentra-se em intervalos estreitos, sugerindo que o restante do universo definido inicialmente é redundante ou "inútil"para a decisão de controle fino. Isso indica que, para o especialista, a variação da ação ocorre apenas em momentos críticos específicos, enquanto no restante do tempo a ação é constante. Essa observação é valiosa para trabalhos futuros, pois sugere que o domínio das variáveis pode ser "recortado"(*pruned*), reduzindo a dimensionalidade e o número de pontos a serem calculados sem perda de eficácia.

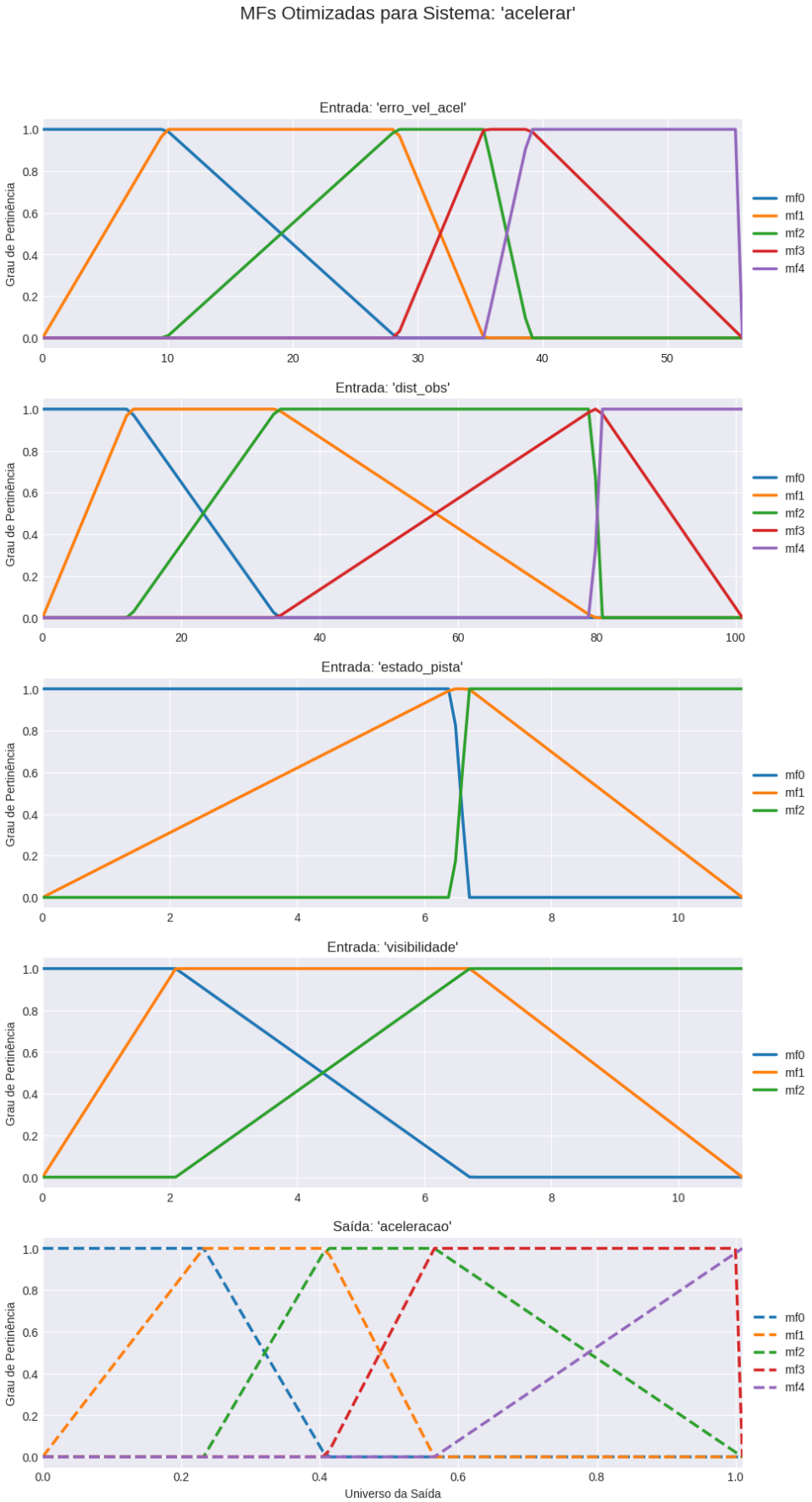


Figura 3 – MFs otimizadas para o controlador `fuzzy_acelerar`. Note a complexidade nas entradas de erro de velocidade comparada à linearidade da visibilidade.

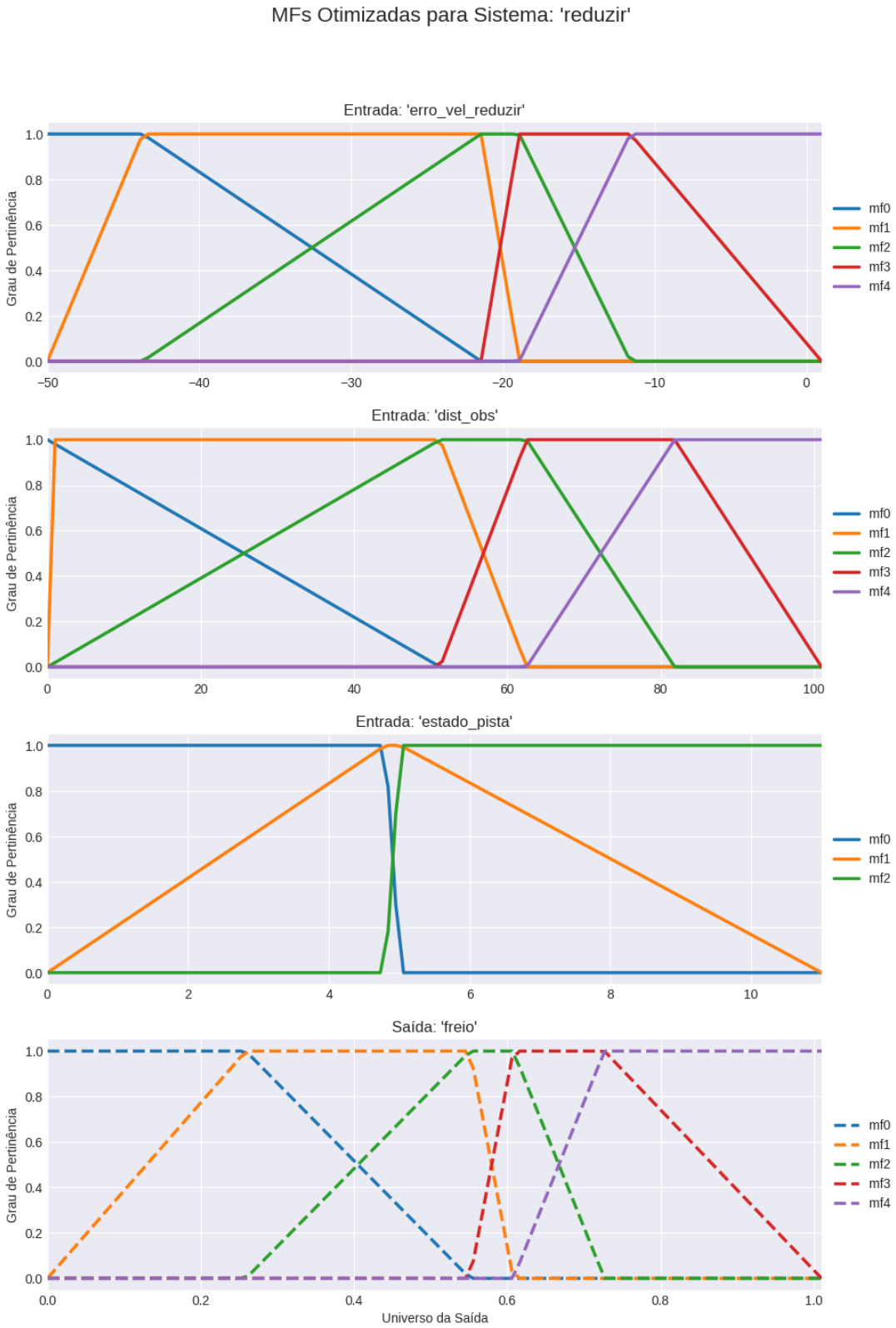


Figura 4 – MFs otimizadas para o controlador `fuzzy_reduzir`. Observa-se a concentração dos nós em faixas específicas, indicando as regiões críticas de frenagem.

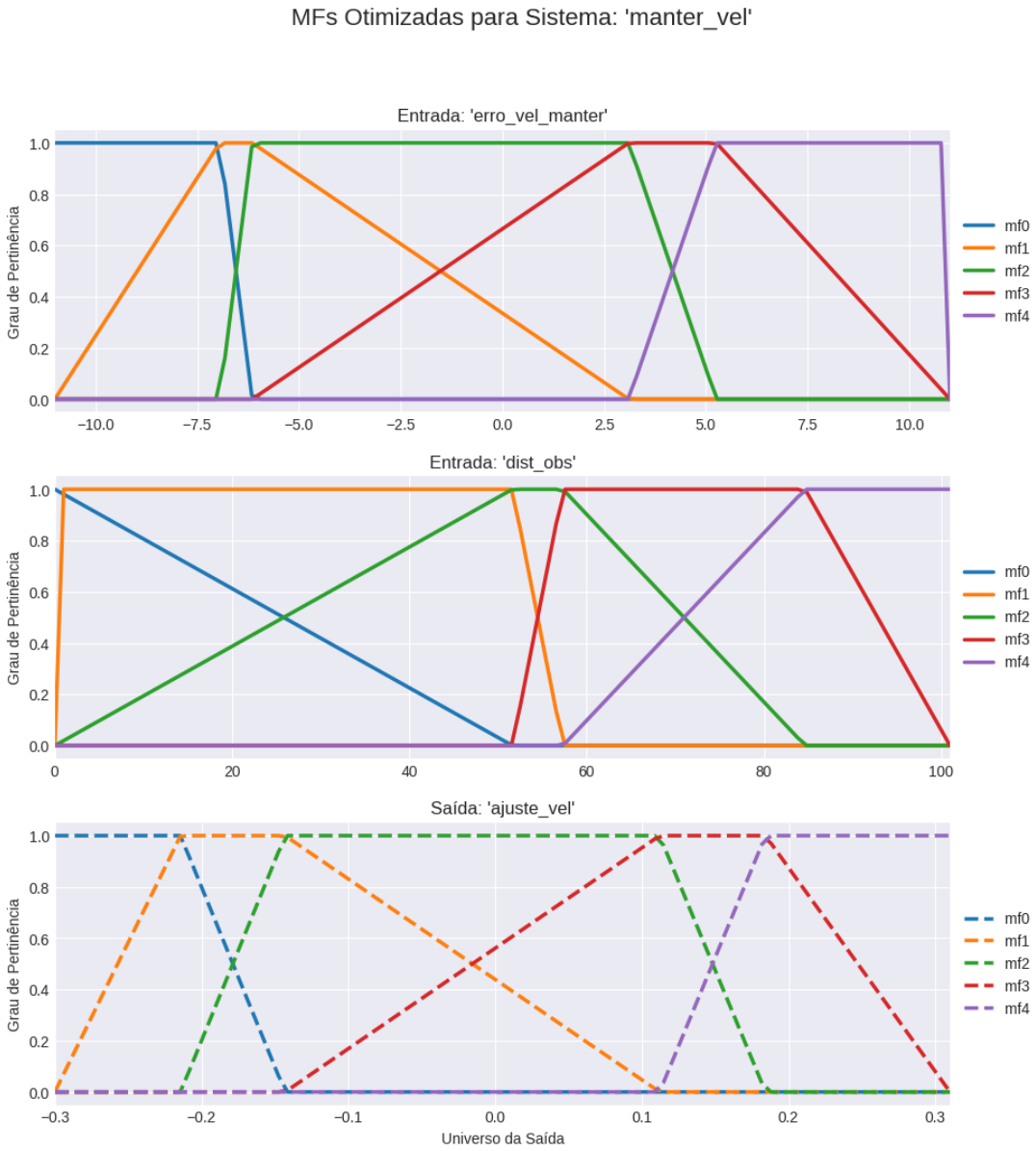


Figura 5 – MFs otimizadas para o controlador `fuzzy_manter_vel`. O ajuste fino é realizado em uma janela estreita de erro de velocidade.

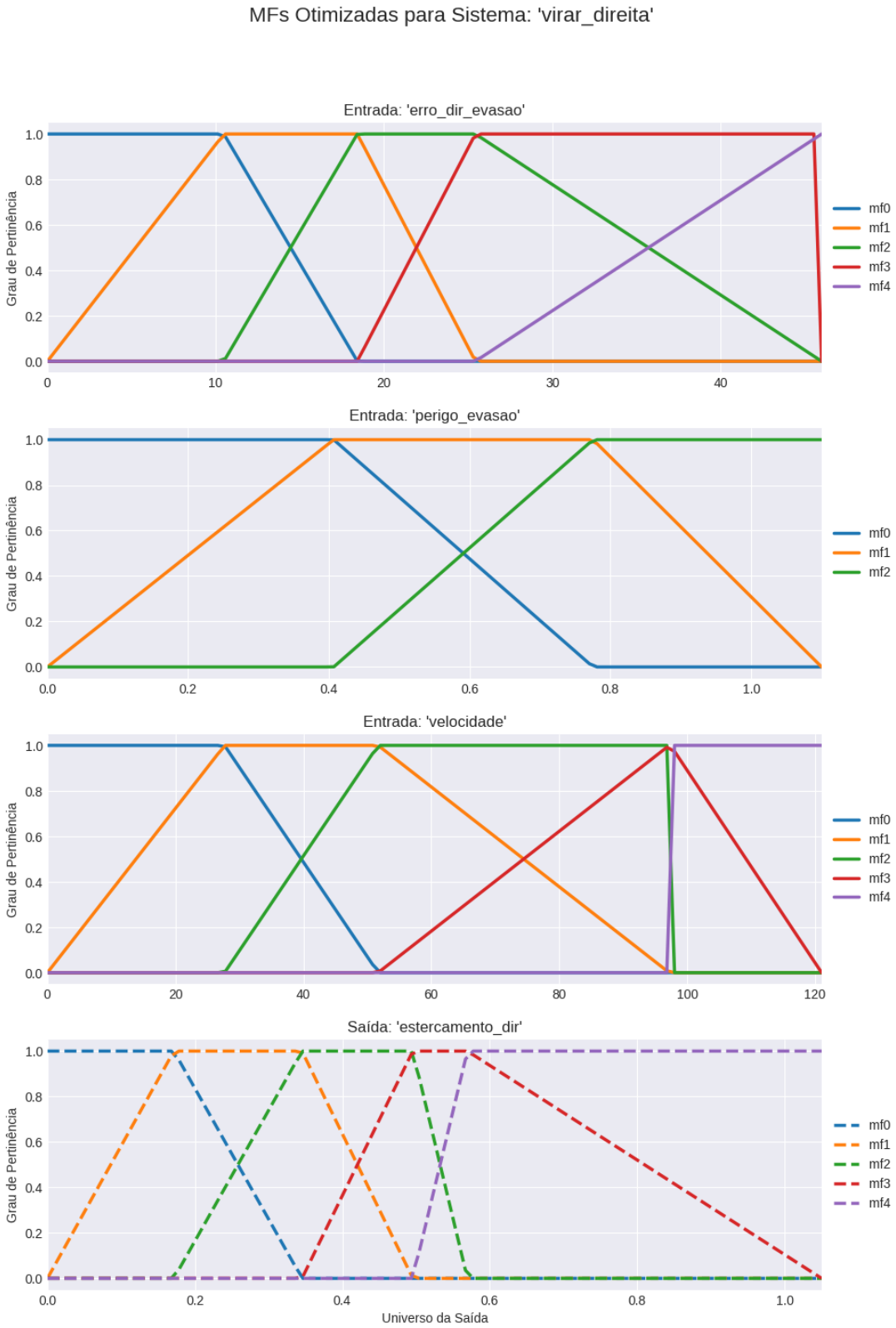


Figura 6 – MFs otimizadas para o controlador `fuzzy_virar_direita`.

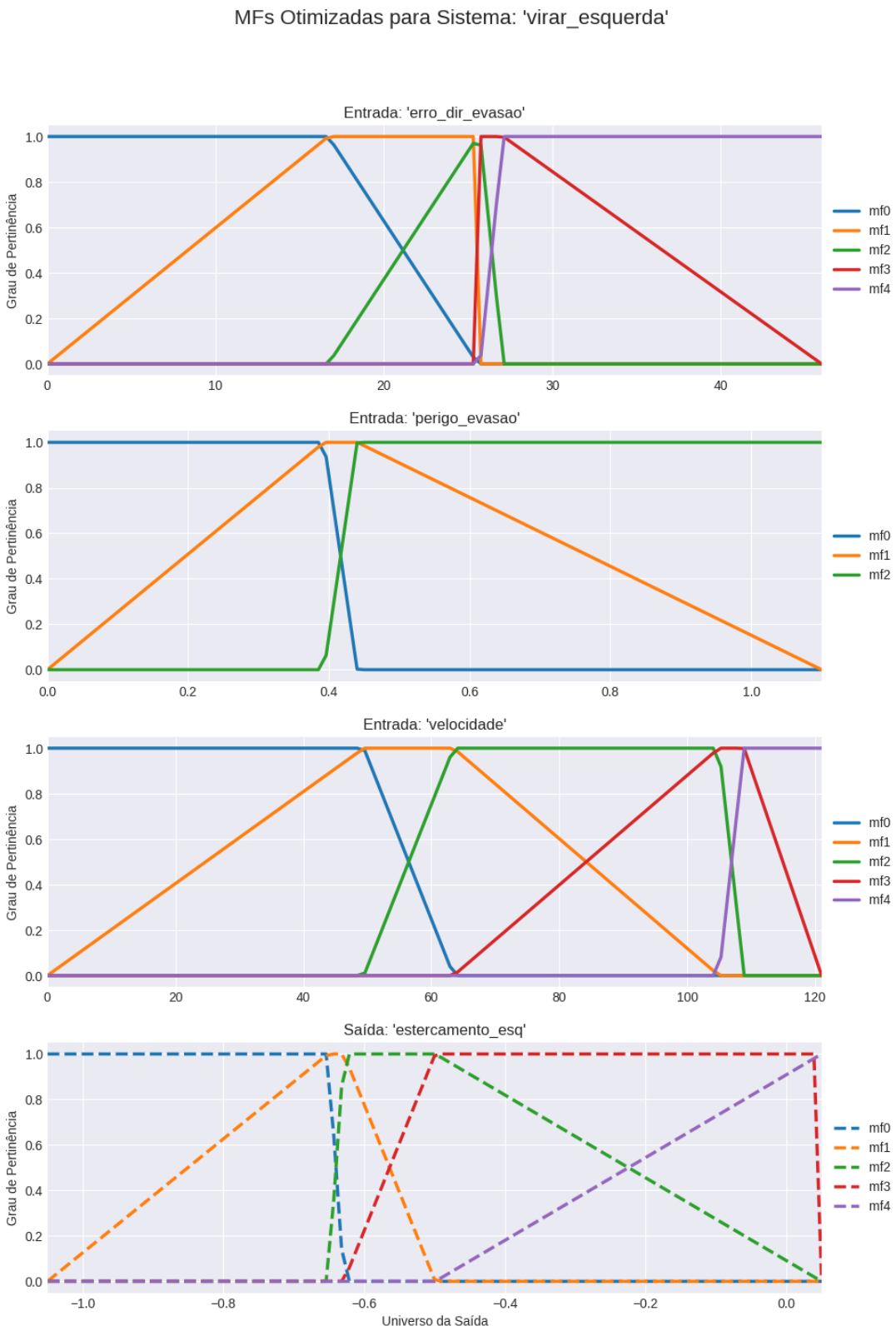


Figura 7 – MFs otimizadas para o controlador `fuzzy_virar_esquerda`.

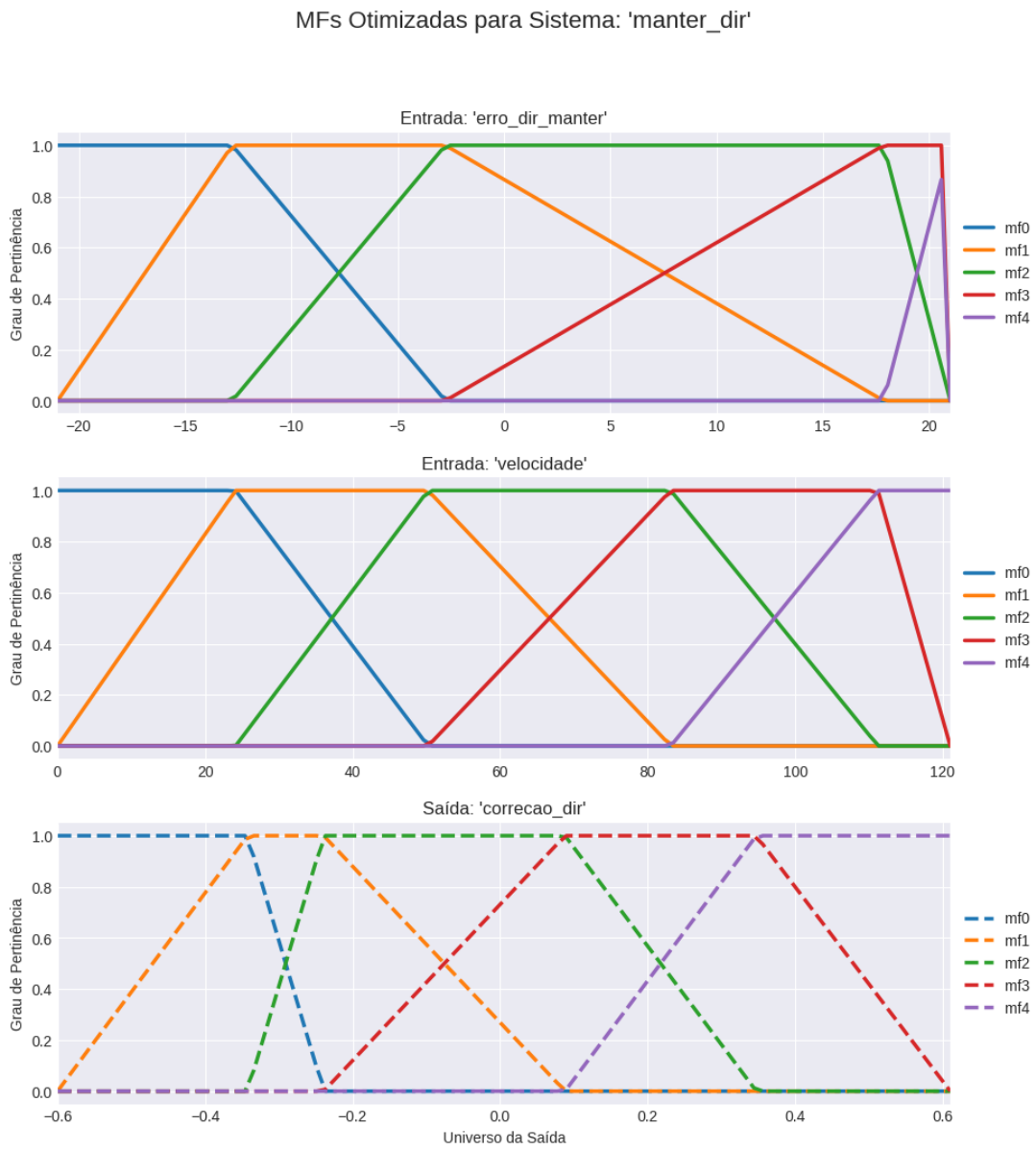


Figura 8 – MFs otimizadas para o controlador `fuzzy_manter_dir`. A saída concentra-se em pequenas correções próximas a zero.

5.2 RESULTADOS DA VALIDAÇÃO *OFFLINE* (SISTEMA HÍBRIDO)

A validação *offline* (Seção 4.4.2) executou a cadeia de decisão completa (`MADM -> Fuzzy`) em todo o dataset de 100.000 amostras. O objetivo era medir o MAE total do sistema e identificar falhas.

Os resultados confirmaram o problema identificado na Seção 5.1.2:

- **MAE Geral:** O sistema obteve um MAE geral razoável (ex: 0.15), indicando que, *quando o MADM acerta*, o *Fuzzy* executa a ação com alta fidelidade.
- **Falhas Críticas:** No entanto, o sistema gerou um grande número de falhas críticas (erros > 0.8). A análise revelou que 100% dessas falhas não eram culpa do *Fuzzy*, mas sim de

uma **classificação incorreta do *MADM***.

O log de falhas (ex: Tabela 1) mostrou casos onde o Especialista decidia "acelerar"(*throttle*=0.75), mas os pesos sub-ótimos do *MADM* faziam o sistema escolher "parar". O sistema então executava perfeitamente a decisão errada (*brake*=1.0), resultando em um erro de velocidade catastrófico de 1.75 (`|0.0 - 0.75| + |1.0 - 0.0|`).

Tabela 1 – Exemplo de Falha Crítica na Validação *Offline*

Métrica	**Decisão do Especialista**	**Decisão do Sistema**
Decisão Estratégica	acelerar	**parar** (Erro *MADM*)
expert_throttle	0.750	...
expert_brake	0.000	...
Saída Fuzzy	...	*fuzzy_throttle* = 0.000
(baseado no *MADM*)	...	*fuzzy_brake* = 1.000
Erro (MAE)	**1.750**	

5.3 ANÁLISE DA EXECUÇÃO EM SIMULAÇÃO

Os resultados da validação *offline* foram confirmados na simulação em tempo real. O agente foi capaz de navegar na rota de teste (12-50), mas exibiu um comportamento errático, particularmente em decisões de velocidade.

6 CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a validação de uma arquitetura de decisão híbrida de baixo custo computacional para veículos autônomos, integrando os métodos *TOPSIS* (MADM) e Lógica *Fuzzy*, com parâmetros otimizados por Algoritmos Genéticos (GAs) treinados por imitação.

6.1 CONCLUSÕES

A arquitetura proposta foi implementada com sucesso, demonstrando a viabilidade de utilizar GAs decoplados para otimizar módulos de decisão e controle de forma independente.

O **GA-Fuzzy** (Seção 5.1.3) foi altamente bem-sucedido. A metodologia de otimização de "nós"(pontos de inflexão das MFs) e regras, treinada com um subset de dados pré-filtrado e uma função de *fitness* baseada em MAE, provou ser uma maneira robusta e eficiente de criar controladores de baixo nível que replicam o comportamento de um especialista com alta fidelidade.

O **GA-MADM** (Seção 5.1.2), no entanto, expôs as limitações intrínsecas da abordagem TOPSIS para este problema. Embora a acurácia de 73% seja razoável, ela se mostrou insuficiente para um sistema de condução seguro. Os resultados da validação offline (Seção 5.2) e da simulação (Seção 5.3) provam que o modelo TOPSIS, por ser uma agregação linear ponderada, não conseguiu capturar as relações não-lineares complexas que o especialista humano (e o modelo *Random Forest*) utilizou para tomar decisões, resultando em falhas críticas de classificação.

Apesar dos problemas de desempenho do módulo MADM, a arquitetura híbrida (MADM-Fuzzy) permanece uma escolha justificável sobre modelos "caixa-preta"como o *Random Forest* (RF) para o objetivo deste trabalho. A principal vantagem do sistema proposto é sua **interpretabilidade** e **viabilidade de implementação em hardware embarcado (FPGA)**. O RF, embora mais preciso, é uma "caixa-preta"cuja estrutura de milhares de árvores é computacionalmente cara e impossível de verificar formalmente. Em contraste, o sistema MADM-Fuzzy é "caixa-branca":

- A decisão do **TOPSIS** é uma matriz de multiplicação determinística, ideal para implementação paralela em FPGAs e fácil de auditar.
- A decisão do **Fuzzy** é baseada em regras linguísticas legíveis, também com baixo custo computacional.

O objetivo principal de desenvolver uma arquitetura de *baixo custo computacional* e *interpretável* foi alcançado. Os resultados demonstram que, embora a camada de controle Fuzzy seja altamente eficaz, a camada de decisão estratégica TOPSIS requer um modelo mais sofisticado (talvez um Fuzzy-TOPSIS ou um sistema de regras mais complexo) para lidar com a não-linearidade das decisões de condução.

6.2 TRABALHOS FUTUROS

Baseado nas conclusões deste trabalho, as seguintes direções de pesquisa são propostas:

- **Substituição do MADM-TOPSIS:** A abordagem atual demonstrou sensibilidade excessiva a obstáculos fora da trajetória (ex: desviar de veículos no acostamento sem necessidade). Propõe-se investigar:
  - **AHP/ANP:** Para estruturar hierarquicamente o objetivo "Seguir Rota"acima de "Evitar Obstáculos Não-Críticos", impedindo desvios desnecessários.
  - **PROMETHEE:** Para utilizar *funções de preferência* e *limiares de indiferença*, permitindo que o sistema ignore riscos laterais que não afetam a segurança da faixa atual.
- **Revisão e Engenharia de Atributos:** O *baseline Random Forest* (Seção 5.1.1) expôs uma lacuna de desempenho (92.7% em velocidade vs. 73.8% em direção). Isso sugere que, enquanto os atributos de velocidade são preditivos, os de direção são insuficientes. Um trabalho futuro deve focar na engenharia de novos atributos, mais robustos, e na remoção de ruídos dos atuais para melhorar a tomada de decisão lateral.
- **Percepção Avançada e Fusão de Sensores:** Este trabalho utilizou atributos de alto nível (ex: `distancia_obstaculo`). Uma melhoria significativa seria extrair *features* de sensores brutos, como câmeras e LiDAR, usando redes neurais (ex: YOLO para detecção de pedestres, não abordado neste TCC) ou técnicas de fusão de sensores (como *BEV-Fusion*) para criar uma representação de ambiente mais rica antes da camada de decisão.
- **Implementação em *Hardware*:** Realizar a síntese da arquitetura MADM-Fuzzy (com os pesos e regras otimizados) em um FPGA para quantificar o uso de recursos (LUTs, BRAMs) e o consumo de energia, validando empiricamente a vantagem de "baixo custo"sobre um modelo de RF ou rede neural.

REFERÊNCIAS

CHEN, S.-J.; HWANG, C.-L. **Fuzzy Multiple Attribute Decision Making: Methods and Applications**. Berlin, Heidelberg: Springer-Verlag, 1992. v. 375. (Lecture Notes in Economics and Mathematical Systems, v. 375). ISBN 978-3-540-552 Fuzzy 6-9.

CODEVILLA, F. et al. End-to-end driving via conditional imitation learning. In: **2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2018. p. 4693–4700.

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

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Boston, MA, USA: Addison-Wesley Professional, 1989. ISBN 0201157675.

RATH, M. et al. Selection of ai architecture for autonomous vehicles using complex intuitionistic fuzzy rough decision making. **Designs**, v. 8, n. 5, p. 93, Sep 2024. ISSN 2411-9660.

VAHDANI, R.; KEYMANESH, M.; SALIMI, M. An enhanced hierarchical fuzzy topsis-anp method for supplier selection in an uncertain environment. **Mathematics**, v. 12, n. 21, p. 3417, Oct 2024. ISSN 2227-7390.

WANG, Y. et al. Hierarchical decision-making for autonomous navigation: Integrating deep reinforcement learning and fuzzy logic in four-wheel independent steering and driving systems. **arXiv preprint arXiv:2508.16574**, Aug 2025.

YILMAZ, A.; AYAN, K.; ADAK, E. A fuzzy logic based autonomous vehicle control system design in the torcs environment. In: **2017 10th International Conference on Electrical and Electronics Engineering (ELECO)**. [S.l.: s.n.], 2017. p. 737–741.

ZADEH, L. A. Fuzzy sets. **Information and Control**, v. 8, n. 3, p. 338–353, 1965.