**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS**

Gustavo Alves Prudencio de Moraes

**Aprendizagem profunda por reforço combinada com
controlador recursivo robusto para controle lateral
baseado em visão computacional**

São Carlos

2020

Gustavo Alves Prudencio de Moraes

**Aprendizagem profunda por reforço combinada com
controlador recursivo robusto para controle lateral
baseado em visão computacional**

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências - Programa de Pós-Graduação em Engenharia Elétrica.

Área de concentração: Sistemas Dinâmicos

Orientador: Prof. Dr. Valdir Grassi Jr.

**São Carlos
2020**

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da
EESC/USP com os dados inseridos pelo(a) autor(a).

M827a   Morais, Gustavo Alves Prudencio de
              Aprendizagem profunda por reforço combinada com
              controlador recursivo robusto para controle lateral
              baseado em visão computacional / Gustavo Alves
              Prudencio de Moraes; orientador Valdir Grassi Jr.. São
              Carlos, 2020.

              Dissertação (Mestrado) - Programa de
              Pós-Graduação em Engenharia Elétrica e Área de
              Concentração em Sistemas Dinâmicos -- Escola de
              Engenharia de São Carlos da Universidade de São Paulo,
              2020.

              1. Controle baseado em visão . 2. Veículos
              autônomos. 3. Regulador Linear Quadrático. 4. Regulador
              Linear Quadrático Robusto. 5. Aprendizagem Profunda por
              Reforço. 6. Algoritmos Evolucionários. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **GUSTAVO ALVES PRUDENCIO DE MORAIS**.

Título da dissertação: "Aprendizagem profunda por reforço combinada com controlador recursivo robusto para controle lateral baseado em visão computacional".

Data da defesa: 24/06/2020

Comissão Julgadora:

Resultado:

Prof. Dr. **Valdir Grassi Junior**
(Orientador)

(Escola de Engenharia de São Carlos – EESC/USP)

Aprovado

Prof. Associado **Denis Fernando Wolf**

(Instituto de Ciências Matemáticas e de Computação/ICMC-USP)

Aprovado

Prof. Dr. **Danilo Alves de Lima**

(Universidade Federal de Lavras/UFLA)

Aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:

Prof. Titular **Ivan Nunes da Silva**

Presidente da Comissão de Pós-Graduação:

Prof. Titular **Murilo Araujo Romero**

Eu dedico este trabalho aos meus pais, Geysa e Domingos.

AGRADECIMENTOS

Antes de tudo, eu agradeço à Deus por ter me proporcionado uma vida repleta de bênçãos.

À minha atenciosa, amorosa e solidária família, Geysa, Domingos, Emílio, Fernanda, Rennan, Maju e Malu; minha mais profunda gratidão.

Eu dedico os meus sinceros agradecimentos ao meu orientador, Prof. Dr. Valdir Grassi Jr., por seu trabalho exemplar e por ter despertado meu interesse em veículos autônomos. Eu também estendo meus agradecimentos aos membros da banca, Prof. Dr. Denis Fernando Wolf e Prof. Dr. Danilo Alves de Lima, por valiosa atenção e comentários.

Ao CNPq (Conselho Nacional de Pesquisa e Desenvolvimento), pelo apoio financeiro; e à Escola de Engenharia de São Carlos e à Universidade de São Paulo, por tornar esta jornada possível.

Aos meus colegas do Laboratório de Sistemas Inteligentes, pela amizade e apoio.

Eu também dedico os meus sinceros agradecimentos aos meus amigos Nuno e Lucas, por todas as ideias em projetos de controle; e aos meus amigos Nicolas, Nilo e Anderson, cujas discussões sobre inteligência artificial tornaram este universo ainda mais fascinante.

A todas as pessoas que ajudaram a tornar esses dois anos inquietantes e únicos, em especial: Guilherme, Juliano, Arthur, Reinaldo, Felipe, Ariela, Will e Andrey.

“But how could you live and have no story to tell?”

Fyodor Dostoevsky, White Nights

RESUMO

de Moraes, G. A. P. **Aprendizagem profunda por reforço combinada com controlador recursivo robusto para controle lateral baseado em visão computacional**. 2020. 89p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Tecnologias na área de robótica móvel, como veículos autônomos, impactam diretamente na qualidade de vida e na mudança de comportamentos sociais, por isso, se mantêm como uma área promissora para a indústria. Mesmo com este cenário otimista, ainda existem muitos desafios relacionados à segurança que necessitam ser supridos para tornar essa tecnologia viável, como o desenvolvimento de técnicas de controle baseadas em visão para agentes autônomos. Algumas áreas de pesquisa estão propondo soluções a partir de métodos de aprendizagem de máquinas, como Aprendizagem Profunda por Reforço (APR), no entanto, algoritmos de APR ainda apresentam baixo desempenho em simuladores urbanos e realistas. Neste contexto, este trabalho propõe um modelo híbrido que combina APR com um Regulador Linear Quadrático (RLQ) padrão. O modelo definido utiliza algoritmos de aprendizagem de máquinas para extrair características de imagens e, na sequência, estas características são interpretadas como medidas de erro a serem minimizadas pelo controlador. Ademais, um segundo controlador híbrido é proposto, onde é aplicado robustez ao modelo anterior utilizando um Regulador Linear Quadrático Robusto (RLQR) e, devido à dinâmica desconhecida do sistema, o modelo realiza uma busca evolucionária para estimar as matrizes de incerteza do modelo. Desta forma, as entradas para cada sistema de controle são imagens de uma câmera RGB em tempo real, e suas saídas são ações contínuas de esterçamento do volante para manter o veículo com velocidade constante no centro da via. Os resultados experimentais obtidos mostram que o controle RLQ híbrido superou o controle realizado pelo algoritmo de APR implementado, que apresentou o maior erro de trajetória no cenário de treino e não foi capaz de generalizar suas ações no cenário de teste. Por fim, ao adicionar robustez ao controlador por meio do RLQR híbrido, obteve-se menos sensibilidade a distúrbios e, em todos os testes realizados, as colisões foram reduzidas a zero.

Palavras-chave: Controle baseado em visão. Veículos autônomos. Regulador Linear Quadrático. Regulador Linear Quadrático Robusto. Aprendizagem Profunda por Reforço. Algoritmos Evolucionários.

ABSTRACT

de Moraes, G. A. P. **Deep reinforcement learning combined with robust recursive controller for lateral control based on computer vision.** 2020. 89p.
Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Technologies in mobile robotics field, such as autonomous vehicles, have a direct impact on the quality of life and the change in social behaviors, therefore, they remain a promising area for the industry. Even with this optimistic scenario, there are still many security-related challenges that need to be addressed to make this technology feasible. One of these challenges is the development of vision-based control techniques for autonomous agents and, in this sense, some research areas are proposing solutions based on machine learning methods, such as Deep Reinforced Learning (DRL). However, DRL algorithms underperforms in urban and realistic simulators. In this context, this work proposes a hybrid model that merges DRL with a standard Linear Quadratic Regulator (RQL), applied for vision-based autonomous vehicles control. The proposed model uses machine learning algorithms to feature extraction and, subsequently, these values are interpreted as error measures to be minimized by the controller. In addition, a second hybrid controller is proposed, where it is applied robustness to the previous model using a Robust Quadratic Linear Regulator (RLQR) and, due to the unknown dynamics of the system, an evolutionary search was used to estimate the system uncertainty matrices. In this way, the inputs for each control system are real-time images from an RGB camera, and their outputs are continuous steering actions to keep the vehicle at constant speed in the center of the road. The experimental results obtained show that the hybrid RQL control outperformed the APR algorithm, which presented the highest trajectory error in the training scenario and it was not able to generalize its actions in the test scenario. Also, by adding robustness to the controller via the hybrid RLQR, it was obtained less sensitivity to disturbances and, in all tests performed, collisions were reduced to zero.

Keywords: Vision-based control. Autonomous vehicles. Linear Quadratic Regulator. Robust Linear Quadratic Regulator. Deep Reinforcement Learning. Evolutionary Algorithms.

LISTA DE FIGURAS

Figura 1 – Modelo de extração de características da imagem aplicados em controles padrões.	32
Figura 2 – Modelo de percepção de imagem para piloto automático em veículo Tesla.	33
Figura 3 – Aplicação de um algoritmo de APR para controle de um veículo real.	36
Figura 4 – Diagrama de operação dos métodos de AR.	40
Figura 5 – Estrutura de um neurônio artificial comparado com um neurônio biológico.	42
Figura 6 – Função objetivo para valores de $\hat{A}_t$ positivos e negativos.	44
Figura 7 – Operação de convolução em uma imagem de entrada 7×7 com um <i>kernel</i> 3×3	45
Figura 8 – Funções de ativação.	45
Figura 9 – Distribuição circular de valores amostrais.	48
Figura 10 – MBL para um problema de otimização com duas variáveis independentes.	51
Figura 11 – Fluxograma com a definição das três abordagens de controle apresentadas neste capítulo.	58
Figura 12 – Arquitetura do modelo neural proposto para o algoritmo de APR.	60
Figura 13 – Modelo híbrido: os canais de saída da CNN fornecem variáveis de entrada ao controlador.	63
Figura 14 – Simulador de código aberto CARLA em operação.	68
Figura 15 – Imagem de segmentação semântica usada como entrada para os modelos propostos.	68
Figura 16 – Ambientes de teste e treinamento no simulador CARLA.	69
Figura 17 – Curva de aprendizado do algoritmo de APR no processo de treinamento.	70
Figura 18 – Valores medidos dos canais de saída selecionados durante a condução manual em três situações diferentes. Em preto é representado a trajetória do veículo seguindo no centro da via, em vermelho o veículo é deslocado para a esquerda e em azul o veículo é deslocado para a direita.	71
Figura 19 – O critério de diferença honestamente significativa de Tukey, com base na distribuição da faixa Studentizada (nível de confiança de 95%), nos resultados.	75
Figura 20 – Seguimento de trajetória com aplicação de distúrbio externo na posição $x = 24\text{m}$ na Cidade 2.	75
Figura 21 – Seguimento de trajetória com aplicação de distúrbio externo na posição $x = 28\text{m}$ na Cidade 1.	76
Figura 22 – Ação de esterçamento e taxa de esterçamento no ambiente de treino.	76
Figura 23 – Ação de esterçamento e taxa de esterçamento no ambiente de teste.	77

LISTA DE TABELAS

- Tabela 1 – Desempenho dos três modelos de controle na Cidade 1 e na Cidade 2. É apresentado o número de colisões com variáveis estáticas, o valor médio da recompensa (Média) e o desvio padrão (σ) ao longo das simulações. 73

LISTA DE ABREVIATURAS E SIGLAS

APR	Aprendizagem Profunda por Reforço
AR	Aprendizagem por Reforço
CARLA	<i>Car Learning to Act</i>
CNN	<i>Convolution Neural Network</i>
CP	Conjunto de Pareto
CPI	<i>Conservative Policy Iteration</i>
CR	Condição de Restrição
DL	<i>Deep Learning</i>
EAND	<i>Evolutionary Algorithm with Numerical Differentiation</i>
FP	Frente de Pareto
GAE	<i>Generalized Advantage Estimator</i>
MBG	Método de Busca Global
MBL	Método de Busca Local
PDM	Processos de Decisão de Markov
PDMPO	Processos de Decisão de Markov Parcialmente Observáveis
PPO	<i>Proximal Policy Optimization</i>
ReLU	<i>Rectified Linear Units</i>
RLQ	Regulador Linear Quadrático
RLQR	Regulador Linear Quadrático Robusto
RNA	Redes Neurais Artificiais
RNP	Redes Neurais Profundas
SGD	<i>Stochastic Gradient Descendent</i>

LISTA DE SÍMBOLOS

F	matriz conhecida do modelo
G	matriz conhecida do modelo
H	matriz conhecida do modelo
$J(x)$	custo quadrático regularizado
$\bar{J}^\mu$	função de custo
K	ganho de realimentação
l	limite inferior
$\mathbf{m}$	indivíduo evolucionário mutado
r	valor de recompensa
t	instante de tempo
u	ação
u	limite superior
V	função de valor do estado
x	estado
$\mathbf{x}$	indivíduo evolucionário
$\hat{\mathbf{x}}$	indivíduo evolucionário para busca local
y	valor de avaliação do indivíduo evolucionário
$\hat{y}$	valor de avaliação do indivíduo evolucionário para busca local
$\hat{\mathcal{A}}$	função <i>Advantage</i>
$\mathcal{B}$	bias
$\mathcal{F}$	mapa de característica
$\mathcal{I}$	imagem de entrada
$\mathcal{K}$	<i>kernel</i>
$\mathcal{L}$	função objetivo

$\mathcal{S}$	bônus de entropia
$\mathbb{E}$	valor esperado
γ	fator de desconto
Δ	matriz arbitrária de contração
δF	parâmetros de incerteza
δG	parâmetros de incerteza
ϵ	hyperparâmetro <i>clip</i>
ε	parâmetro GAE
η	taxa de aprendizagem
θ	parâmetros da política comportamental
π	política comportamental
ω	pesos do modelo neural

SUMÁRIO

1	INTRODUÇÃO	25
1.1	Objetivos	28
1.2	Estrutura do trabalho	29
2	REVISÃO DA LITERATURA	31
2.1	Controle referenciado em visão para veículos autônomos em ambientes urbanos	31
2.2	Aprendizagem Profunda por Reforço para navegação autônoma	34
2.3	Aprendizagem por reforço para aplicações em controle	36
2.4	Contribuições	37
3	FUNDAMENTAÇÃO TEÓRICA EM APRENDIZAGEM DE MÁQUINAS	39
3.1	Aprendizagem Profunda por Reforço	39
3.1.1	<i>Proximal Policy Optimization</i>	42
3.1.2	<i>Convolutional Neural Networks</i>	44
3.2	<i>Evolutionary Algorithm with Numerical Differentiation</i>	46
3.2.1	Diferenciação numérica	47
3.2.2	Definição do algoritmo	48
3.2.2.1	Inicialização	48
3.2.2.2	Método de Busca Global	48
3.2.2.3	Método de Busca Local	49
4	FUNDAMENTAÇÃO TEÓRICA EM CONTROLE ROBUSTO	53
4.1	Formulação do Problema	53
4.2	Problema de mínimos quadrados regularizado	54
4.3	Problema de mínimos quadrados regularizado robusto	54
4.4	Regulador Linear Quadrático Recursivo Robusto	55
5	DEFINIÇÃO DO MODELO PROPOSTO	57
5.1	Aprendizagem Profunda por Reforço	57
5.1.1	Função de recompensas	58
5.1.2	Modelo neural	59
5.2	Determinação das referências de controle	59
5.2.1	Seleção dos canais de saída	60
5.2.2	Sinal de erro e ação de controle	62
5.3	Identificação do espaço de estados do sistema	63

5.4	Estimativa das matrizes de incerteza por busca evolutiva	64
6	RESULTADOS	67
6.1	Simulador CARLA	67
6.2	Implementação	69
6.3	Treinamento do algoritmo de Aprendizagem Profunda por Reforço .	70
6.4	Canais de saída	70
6.5	Definição do controlador RLQ híbrido	72
6.6	Definição do controlador RLQR híbrido	72
6.7	Comparação dos modelos	73
7	CONCLUSÃO	79
	REFERÊNCIAS	81

1 INTRODUÇÃO

Veículos autônomos em ambientes urbanos representam uma área de pesquisa com interesse crescente nas últimas décadas, visto que esta tecnologia está relacionada com várias melhorias, como a diminuição drástica do número de acidentes decorrentes de erros humanos, benefícios na mobilidade de pessoas com deficiência, otimização do consumo de combustível, entre outros benefícios (FAGNANT; KOCKELMAN, 2015). Mesmo com os recentes avanços em tecnologias associadas a veículos terrestres não tripulados, um veículo com total capacidade de decisão ainda é uma tecnologia difícil de ser aplicada em ambientes reais, considerando que os meios de transportes autônomos ainda precisam superar desafios relacionados a identificação de modelos, design de sistemas de controle para segmento de trajetória e algoritmos para prevenção de colisões, o que exige soluções inovadoras de pesquisadores e empresas. Estas dificuldades podem ser derivadas de fatores externos (como condições climáticas e imagens de baixa qualidade) ou fatores internos (como incertezas do modelo). Nesse sentido, modelos de controle para segmento de trajetória baseados em visão computacional são apresentados como soluções promissoras pois, por se tratar de e, por serem tecnologias que combinam atuadores com sensoramento, esses modelos apresentam menos necessidade de vários sensores e instrumentação complexa, conforme impostos por sistemas autônomos baseados em GPS (Kottath et al., 2017), (Reyes et al., 2018).

Indubitavelmente, as imagens capturadas pelas câmeras frontais e traseiras, junto com técnicas de processamento de imagem, fornecem informações valiosas para a condução autônoma de veículos. Recentemente, métodos de aprendizado de máquinas, como *Deep Learning* (DL) (GOODFELLOW; BENGIO; COURVILLE, 2016), estão em destaque como ferramentas para extrair características de imagens e, assim, poder utilizá-las em modelos autônomos. Aplicações bem-sucedidas de algoritmos para estimativa de posição e orientação, reconhecimento de semáforos, identificação de marcações na estrada, entre outras, são encontradas em (Shobha; Deepu, 2018), (Kim; Kum, 2019), (Chen et al., 2015), (Xuehong Mao et al., 2016) e (Wang; Zhou, 2019).

Ademais, um sistema autônomo pode usar imagens em tempo real processadas por técnicas de visão computacional, e usá-las como entradas para sistemas de controle laterais e longitudinais, sem nenhum conhecimento prévio sobre a dinâmica do sistema. Para essa tarefa, outros procedimentos de aprendizado de máquinas podem ser aplicados, como o Aprendizado por Reforço (AR) (KOBER; BAGNELL; PETERS, 2013). Este método permite que um agente autônomo descubra autonomamente um comportamento ideal por meio de interações de tentativa e erro com seu ambiente.

Nos métodos de AR, uma máquina com capacidade de decisão (agente autônomo) faz observações sobre a dinâmica externa do local e gera um estado no qual são atribuídas

todas as informações relevantes sobre a situação atual do agente. O modelo de AR pode, portanto, executar ações para alterar o estado do sistema e o algoritmo avalia se essas ações levam a interações positivas ou negativas com o ambiente. As experiências obtidas pelo agente permitem que um algoritmo de AR desenvolva e otimize uma política comportamental, representada pelo mapeamento de estados para ações, possibilitando que o algoritmo melhore seu desempenho independentemente da interferência humana.

A combinação de algoritmos de AR com técnicas de DL, resultou em algoritmos de Aprendizagem Profunda por Reforço (APR) (MNIH et al., 2015), que permitiram o uso de modelos de AR em problemas de alta dimensionalidade. A APR representa um campo de pesquisa promissor no domínio de aprendizado de máquinas, onde o estado da arte é impulsionado por modelos de AR que usam otimização de gradiente para o cálculo de políticas comportamentais, como o algoritmo PPO (do inglês *Proximal Policy Optimization*) (SCHULMAN et al., 2017), que promoveu um equilíbrio entre simplicidade na implementação e bons resultados para amostras complexas, e por métodos de DL para extração de recursos de imagens, como as CNNs (do inglês *Convolutional Neural Networks*) (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), que mostraram bom desempenho em várias aplicações, como classificação de imagens (LOPES et al., 2017), (NOGUEIRA; PENATTI; SANTOS, 2017), detecção de objetos (NGUYEN; LI; OGUNBONA, 2016), (PEDERSOLI et al., 2015), rastreamento (GU et al., 2018), (ZHANG et al., 2015), (ZHANG et al., 2013), etc.

Cenários urbanos e rodovias podem ser interpretados como problemas baseados em regras e, nesse contexto, os algoritmos de APR podem ser aplicados com eficiência (ISELE et al., 2018a). Devido às dificuldades logísticas de usar modelos autônomos em cenários reais, ambientes simulados e realistas permitiram o desenvolvimento, treinamento e validação destes modelos. Por exemplo, o simulador CARLA (do inglês *Car Learning to Act*) (DOSOVITSKIY et al., 2017), o simulador de carro autônomo desenvolvido pela Udacity¹, NVIDIA DRIVE Sim², etc. No entanto, os modelos de APR geralmente executam uma extensa busca por hiperparâmetro e, em um simulador realista, essa exploração se torna inviável, resultando em baixo desempenho (DOSOVITSKIY et al., 2017).

Os algoritmos de APR são conhecidos por serem sensíveis em aplicações cujo treinamento envolve uma grande quantidade de variáveis, o que torna esses métodos computacionalmente ineficientes em implementações como segmentos de trajetória em ambientes complexos e realistas a partir de imagens. Por outro lado, o procedimento de treinamento associado à APR pode parametrizar uma CNN de modo a destacar características de imagens, e utilizar estas características como referências de controle. Esse procedimento pode ser eficiente mesmo para um treinamento envolvendo uma pequena

---

¹ <<https://github.com/udacity/self-driving-car-sim>>.

² <<https://www.nvidia.com/en-us/self-driving-cars/>>.

quantidade de amostras de experiência.

Para fins de controle, a extração de dados robustos e relevantes das imagens ainda é uma preocupação. Em (ASSA; JANABI-SHARIFI, 2013), foi proposta uma abordagem de fusão sensorial baseada em filtro de Kalman para um algoritmo robusto de estimação de posição. Este processo, no entanto, apresentou a necessidade de várias câmeras, o que elevou os custos e limitações do experimento. Ademais, em (ABU-JBARA et al., 2015) foi usado um modelo baseado em modelagem física do ambiente para otimizar uma aterrissagem automática robusta de agente autônomo em uma pista. Contudo, o uso de uma abordagem matemática para gerar o modelo da pista exige um equilíbrio entre maximizar a diferenciação da pista do ambiente circundante e maximizar a robustez às perturbações locais. Especificamente no caso de veículos não tripulados, em (MAMMERI; LU; BOUKERCHE, 2015) propõe-se um algoritmo baseado em visão de três estágios para detecção e rastreamento de trajetórias, que envolvem várias etapas, como suavização de imagem, extração da região de interesse, mapa de perspectiva inversa, segmentação, extração de recursos e filtragem de dados.

Conforme apresentado, trabalhos recentes da literatura propuseram técnicas para o desenvolvimento de processos de controle baseados em imagens capturadas por câmeras. Normalmente, técnicas de processamento de imagem e projeto de controle são realizados em dois estágios diferentes. Essa modularidade pode simplificar o sistema geral, no entanto, a falta de acoplamento pode aumentar os ruídos e incertezas no modelo final definido para a dinâmica do sistema. Uma solução promissora seria trabalhar a robustez do modelo afim de superar os efeitos negativos de entradas exógenas e incertezas paramétricas e, assim, garantir a estabilidade geral do sistema.

Portanto, robustez é uma questão crucial para veículos autônomos, visto que o sistema de controle deve generalizar e funcionar adequadamente sob diferentes condições de operação. Por exemplo, (BARBOSA et al., 2019) aplicou um Regulador Linear Quadrático Robusto (RLQR) para seguimento de trajetória em um caminhão carregado com cargas variáveis; e (JI et al., 2018) usou um controle de movimento lateral robusto para veículos autônomos sujeitos a distúrbios externos desconhecidos. No caso do controle baseado na visão, surgem diferentes condições de operação, como ruído na imagem, luminosidade variável, diferentes ambientes, etc.

Nesse sentido, este projeto propõe um modelo híbrido usando APR e um Regulador Linear Quadrático (RLQ) padrão para aprimorar o desempenho de um controlador para segmento de trajetória baseado em visão. Nesta abordagem, um modelo treinado de APR composto por uma CNN e um algoritmo PPO infere sinais de erro de deslocamento lateral, a partir de operações de convolução nas imagens RGB da câmera frontal do veículo em ambiente simulado. Em seguida, para diminuir os efeitos negativos das incertezas paramétricas e melhorar a estabilização do veículo, o controlador RLQR é aplicado no

modelo anterior. Considerando que a dinâmica do sistema é desconhecida, as matrizes de incerteza foram estimadas por meio de busca evolutiva não-linear realizada pelo algoritmo EAND (do inglês *Evolutionary Algorithm with Numerical Differentiation*) (MORAIS et al., 2019). O EAND é um modelo com capacidade de evolução adaptativa, e que apresenta procedimentos de buscas globais e locais que promovem a diversidade da população baseados em técnicas de diferenciação numérica e otimização seletiva.

Entre as principais vantagens apresentadas pelo modelo proposto, destacam-se: (i) a CNN é capaz de extrair características visuais relevantes das imagens de uma única câmera, transformando-os em um conjunto de canais de saída; (ii) controlar o sistema do ponto de vista dos canais de saída da CNN não requer nenhum conhecimento dos parâmetros físicos do sistema; (iii) controle RLQ híbrido (resultado da combinação da rede CNN com o controlador RLQ) supera um controle independente baseado em APR para a aplicação em estudo; (iv) mais importante, o controle RLQR híbrido (resultado da combinação da rede CNN e do controlador RLQR) não apenas melhora o desempenho em um determinado cenário, mas também generaliza a abordagem de controle para cenários de teste distintos, nos quais o RLQ híbrido e o algoritmo de APR foram incapazes de realizar o controle de forma adequada.

1.1 Objetivos

Este trabalho propõe um controle lateral para segmento de trajetória baseado em visão computacional, aplicado em veículos autônomos em ambientes simulados. A entrada do modelo é uma imagem RGB em tempo real feita por câmera frontal que captura todas as informações na frente do veículo, e a saída é uma ação contínua de direção que mantém no centro da via um veículo dirigindo com velocidade constante.

Para este fim, este trabalho tem como primeiro objetivo definir um modelo híbrido entre o RLQ padrão e um algoritmo de APR. Durante o processo de treinamento, o modelo de APR parametriza os filtros da CNN para destacar as características das imagens que mais influenciam na resolução do problema. Na sequência, o controlador híbrido proposto minimiza os sinais de erro derivados a partir da convolução da imagem de entrada realizada pela CNN, ao selecionar uma nova ação contínua de esterçamento do volante.

O segundo objetivo do trabalho é aplicar robustez para aumentar o desempenho do modelo anterior, propondo um controlador RLQR híbrido. Devido à dinâmica desconhecida do sistema, as matrizes de incerteza de controle são definidas usando uma busca de otimização não-linear evolutiva realizada pelo algoritmo EAND.

1.2 Estrutura do trabalho

Esta dissertação está dividida em sete capítulos. O Capítulo 2 apresenta trabalhos relacionados com a proposta apresentada. O Capítulo 3 discute a teoria de aprendizado de máquinas na qual o trabalho se baseia, onde são introduzidos os conceitos de APR, abrangendo PPO e CNN, e o método de otimização proposto pelo algoritmo EAND. Uma revisão teórica de métodos de controle usados para esta abordagem é apresentada no Capítulo 4. As definições do algoritmo de APR junto com os modelos RLQ e RLQR híbridos são apresentadas no 5. O Capítulo 6 discute os resultados e, finalmente, o Capítulo 7 apresenta as conclusões e as propostas de trabalhos futuros.

2 REVISÃO DA LITERATURA

Este capítulo apresenta alguns trabalhos relacionados ao método proposto. Primeiro, a Seção 2.1 introduz trabalhos sobre veículos autônomos em ambientes urbanos, com controle referenciado em visão computacional. Na sequência, a Seção 2.2 apresenta trabalhos que fizeram uso de algoritmos de APR para navegação autônoma. A Seção 2.3 exemplifica trabalhos que combinam AR com métodos de controle padrão. Finalmente, a Seção 2.4 discute como o método proposto difere das demais obras apresentadas e destaca suas contribuições.

2.1 Controle referenciado em visão para veículos autônomos em ambientes urbanos

Algoritmos de visão computacional tornaram possível o desenvolvimento bem sucedido de modelos de veículos autônomos em ambientes não confinados e, à medida que foram apresentados resultados promissores, o interesse por grupos de pesquisas e empresas neste campo de pesquisa cresceu de forma ininterrupta. Um dos trabalhos pioneiros foi apresentado no projeto Eureka PROMETHEUS (NETWORK, 2013), desenvolvido na Europa entre os anos 1987 e 1995. Outros trabalhos foram apresentados na sequência, como o VITA II (ULMER, 1994) e o DARPA Grand Challenge (BUEHLER; IAGNEMMA; SINGH, 2007), proposto pelo departamento de defesa dos EUA como uma competição para veículos autônomos em um percurso *off-road*.

É evidente que para ambientes urbanos se faz necessário um modelo de processamento com maior complexidade. Neste sentido, a DARPA Urban Challenge (BUEHLER; IAGNEMMA; SINGH, 2009) propôs uma nova competição para equipes desenvolvedoras de veículos autônomos, com aplicação em contextos urbanos. O sucesso desta iniciativa induziu novas propostas de desafios, como apresentado em (BROGGI et al., 2012), (ENGLUND et al., 2016), (BROGGI et al., 2014).

Os modelos mais populares de trabalho envolvendo veículos autônomos referem-se a pipeline clássicos, resultantes da combinação de sensoramento com atuadores (MCALLISTER et al., 2017). Estes modelos tipicamente são representados por entrada de sensores de localização e detecção de objetos, seguido por modelo de predição e por fim um modelo de tomada de decisão (URMSON et al., 2008), (LEVINSON et al., 2011). Portanto, modelos de percepção baseados em visão computacional (JAIN; KASTURI; SCHUNCK, 1995) são essenciais para que um veículo autônomo atue com segurança em um ambiente urbano, o que faz com que esses modelos apresentem aplicações crescentes em cenários simulados e reais. Por exemplo, Lima e Victorino (2014) propuseram um controlador baseado em visão para evitar obstáculos e, na sequência, Lima e Victorino (2016) otimizaram o modelo anterior ao propor um modelo que controla o veículo e reduz a velocidade frente a

Figura 1 – Modelo de extração de características da imagem aplicados em controles padrões.

Fonte: (SAUER; SAVINOV; GEIGER, 2018)

obstáculos.

Em aplicações de controle referenciado em visão utilizando o simulador CARLA, Dosovitskiy et al. (2017) comparou o desempenho de três diferentes procedimentos de aprendizagem para condução autônoma, representados por um pipeline modular clássico baseado em percepção, planejamento e controle contínuo (PADEN et al., 2016), um método de RNP para aprendizagem por imitação (CODEVILLA et al., 2017) e um algoritmo de APR assíncrono (MNIH et al., 2016). Embora tenham sido usados métodos de aprendizagem que apresentaram bons desempenhos em ambientes simulados, nenhum dos três funcionou com total confiabilidade. Foi verificado que o desempenho diminuía ainda mais nas etapas de teste, indicando um problema de generalização em todos os modelos.

Seguindo o trabalho para condução autônoma a partir de processamento de imagem definido em (DOSOVITSKIY et al., 2017), Sauer, Savinov e Geiger (2018) apresentaram um modelo baseado em percepção direta, em que se obteve melhorias de até 68% comparado com os trabalhos anteriores no CARLA. Este trabalho possui semelhanças com o modelo proposto, ao considerar que uma imagem é processada por um modelo neural, e as informações abstraídas são usadas como entradas de um sistema de controle padrão, sendo o controle longitudinal representado por um controlador PID e o lateral representado por um controlador de Stanley (THRUN et al., 2006). Esta configuração é representada na Fig. 1. No entanto, para que o modelo neural fosse capaz de inferir um erro válido para o controlador em (SAUER; SAVINOV; GEIGER, 2018), foi necessário um completo conhecimento sobre a dinâmica do sistema. Deste modo, as RNAs foram treinadas utilizando vídeos que continham distâncias calculadas à objetos dinâmicos e estáticos, além de rótulos para placas de trânsito e semáforos.

Figura 2 – Modelo de percepção de imagem para piloto automático em veículo Tesla.

Fonte: <<https://www.autopilotreview.com/tesla-autonomous-full-self-driving/>>

Sistemas que oferecem diferentes níveis de automação, como controle de velocidade, prevenção de colisões, estacionamento automático ou assistência na manutenção de faixas, também são disponíveis em carros comerciais (OLIVARES-MENDEZ et al., 2016), como os veículos produzidos pela empresa Tesla. A Tesla dispõem aos seus clientes dois modos para piloto automático, onde são oferecidas assistências de direção ao motorista, como controle de distância e velocidade, centralidade com a via e alterações de faixa. Um modelo de percepção a partir de imagem em um veículo Tesla com sistema semiautônomo é apresentado na Fig. 2.

Outros trabalhos envolvendo métodos de visão aplicados para percepção e controle de veículos autônomos em ambientes urbanos podem ser encontradas em (CHEN; MIAO, 2016), (VITOR et al., 2013), (BERTOZZI; BROGGI; FASCIOLI, 2000), (BANSAL et al., 2008), (SOTELO; RODRIGUEZ; MAGDALENA, 2002). Estes trabalhos publicaram resultados eficientes para os problemas propostos, no entanto, algumas das principais desvantagens dos sistemas modulares baseados em percepção e ação, é que esses modelos são propensos à propagação de erros (MCALLISTER et al., 2017), necessita de um conhecimento prévio sobre a dinâmica do veículo e sua interação com o ambiente, e apresentam uma complexidade excessiva (YURTSEVER et al., 2020). Uma solução alternativa seria o uso de modelos de percepção direta (WEI et al., 2013), como APR, descrita a seguir.

2.2 Aprendizagem Profunda por Reforço para navegação autônoma

No fim de 2013, a empresa britânica DeepMind utilizou os conceitos de APR para treinar algoritmos de inteligência artificial para jogar vídeo games do tipo Atari 2600 (MNIH et al., 2013). No trabalho apresentado pela DeepMind, o algoritmo aprendia a interagir com os jogos apenas observando as variações de pixels na tela e como as ações combinadas aos estados do jogo conduziram a recompensas ou punições. Sete jogos foram treinados usando a mesma arquitetura de rede neural, e o modelo de APR obteve melhor desempenho do que humanos em três deles. Na sequência, foram propostas modificações nos métodos de APR, e o algoritmo foi novamente utilizado em 49 jogos de vídeo game Atari 2600, sendo todos com estruturas e objetivos diferentes (MNIH et al., 2015). O algoritmo de APR conseguiu obter desempenho semelhante a humanos na maioria deles e, em alguns casos, obteve desempenho superior.

Como apresentado anteriormente, o método proposto pela empresa britânica se aproximava da forma que humanos aprendem a executar ações: realizam uma sequência de observações, aprendem sobre a dinâmica de estados e ações, e tomam decisões que podem resultar em recompensa ou punição, a partir do tipo de reação obtido da ação realizada. Esses resultados motivaram diferentes grupos de pesquisa a melhorar ou propor novos algoritmos de APR em outros domínios, como condução autônoma de veículos (MARINA; SANDU, 2017).

Métodos probabilísticos como Processos de Decisão de Markov (PDM) (BELLMAN, 1957) e Processos de Decisão de Markov Parcialmente Observáveis (PDMPO) (MONAHAN, 1982), foram utilizados para modelar as incertezas provenientes de um ambiente dinâmico representado por um cenário urbano, onde foram apresentadas aplicações para veículos autônomos envolvendo planejamento e tomada de decisão (DING; PINTO; SURANA, 2013), geração de ações de direção semelhantes a humanos (GUO et al., 2018) e manobras em ambientes complexos (CHEN et al., 2014). Esse cenário ajuda a reduzir as limitações provocadas por imprecisões que, junto com o aperfeiçoamento de sensores, tornam preferível para as empresas automobilísticas trabalharem com aprendizagem supervisionada (MARINA; SANDU, 2017).

Algoritmos de APR abrangem a complexidade exigida por diferentes métodos de aprendizagem de máquinas, como aprendizagem supervisionada, pois exigem tanto uma predição interativa sequencial das possíveis ações quanto estruturas de recompensa complexas (KOBBER; BAGNELL; PETERS, 2013). Essa abrangência torna os problemas de APR computacionalmente difíceis de serem executados. No entanto, os avanços obtidos nesta área se mostraram promissores. Foram propostos algoritmos capazes de solucionar problemas de PDM e PDMPO com altas dimensões utilizando aproximação de funções. Algoritmos de aprendizagem de máquinas com capacidade de extrair informações de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) otimizaram os algoritmos de

APR, o que tornou este método capaz de identificar pedestres, veículos, linhas de divisão em rodovias e estimar distâncias em um cenário urbano. Devido às dificuldades de logística para utilizar modelos autônomos em cenários reais, ambientes simulados simples e realistas foram disponibilizados para uso e cooperação e permitiu o desenvolvimento, teste e validação de modelos (DOSOVITSKIY et al., 2017).

Considerando que ações motoras podem ser moduladas por métodos como o PDM, os algoritmos de APR podem ser apresentados como soluções competitivas para condução autônoma (HUSSEIN et al., 2018). Alguns trabalhos que validam esta aplicação foram apresentados em simuladores de carros de corrida autônomos (LILLICRAP et al., 2015), navegação 3D (ZHU et al., 2017), (MNIH et al., 2016), e para veículos aéreos não tripulados (ABBEEL et al., 2007), (NG et al., 2006), (ZHANG et al., 2016). No entanto, os modelos de APR ainda apresentam baixa eficiência para sistemas instáveis e exigem uma grande quantidade de amostras para treinamento, requerendo um tempo dispendioso. Este contexto torna os modelos de APR muitas vezes impraticáveis em cenários reais.

Dentre os métodos para navegação autônoma no simulador CARLA apresentados por Dosovitskiy et al. (2017), o algoritmo de APR obteve os piores resultados, com diferenças significativas nas etapas de treino e teste em relação aos outros modelos. Para otimizar o desempenho destes algoritmos, Jaafra et al. (2019) propôs uma abordagem para APR utilizando o simulador CARLA, onde a parametrização da política era executada de forma contínua por meio de aprendizagem por diferença temporal com retornos calculados em um conjunto maior de etapas. Este processo promoveu uma aprendizagem mais robusta nas etapas de treino, além de apresentar resultados melhores e obter um tempo de aprendizagem menor do que os métodos tradicionais de APR. No entanto, o modelo ainda não apresentou bons resultados em etapas de teste.

Já para aplicações de APR em cenário reais, Folkers, Rick e Büskens (2019) treinaram um algoritmo em um simulador que figura um estacionamento e, na sequência, aplicaram o modelo em um veículo real em ambiente confinado. Para a proposta apresentada, as posições inicial e final do veículo eram previamente definidas, e as informações sobre os arredores do automóvel eram inferidas por meio de scanners a laser. O algoritmo de APR interpretava as medidas feitas pelos sensores de obstáculos e selecionava uma ação de posição e velocidade de esterçamento do volante junto com valores de aceleração do carro. Este trabalho representou um dos primeiros modelos de APR implementado com sucesso em um veículo autônomo real, todavia o algoritmo foi treinado em um ambiente com baixa complexidade, e a aplicação do modelo foi limitada para o problema de estacionar com segurança. A Fig. 3 apresenta o experimento realizado, onde no lado esquerdo é mostrado o veículo em operação e no lado direito a interpretação do mesmo cenário em ambiente simulado, sendo as marcações em vermelho as medições do laser, as linhas pretas representam obstáculos e o caminho planejado é apresentado em amarelo.

Figura 3 – Aplicação de um algoritmo de APR para controle de um veículo real.

Fonte: (FOLKERS; RICK; BÜSKENS, 2019)

Algumas aplicações envolvendo APR também foram apresentadas para tomada de decisões estratégicas (ISELE et al., 2018b), e para definições de comandos de controle (SALLAB et al., 2017), (LILLICRAP et al., 2015). Esses trabalhos propuseram resultados aplicáveis em cenários reais, mas o problema de controle de um veículo utilizando APR a partir de visão computacional em um ambiente urbano ainda representa um grande desafio, considerando que estes modelos ainda não apresentam o desempenho adequado para uma aplicação urbana em ambiente não confinado.

2.3 Aprendizagem por reforço para aplicações em controle

Os sistemas de controle são modelos físicos que traduzem a dinâmica do sistema e, em geral, são representados por blocos que podem ser analisados matematicamente, enquanto os métodos de aprendizagem de máquinas são estruturas altamente flexíveis e adaptáveis, que geralmente não exigem conhecimento ou análise matemática do sistema (MOE; RUSTAD; HANSSEN, 2018). Mesmo considerando que o formalismo matemático exigido na definição de ambas as técnicas seja a princípio desassociado, vincular estes modelos pode resultar em otimização de projetos.

Os métodos clássicos de controle exigem do projetista conhecimento sobre o design do controlador e da modelagem dos processos, além de exigir o desenvolvimento de

modelos dinâmicos precisos (WANG; VELSWAMY; HUANG, 2018). Nesse contexto, alguns trabalhos foram conduzidos para combinar aprendizagem de máquinas com técnicas de controle, afim de evitar a dependência de definições físicas do modelo na formulação de plantas de controle.

Uma solução foi apresentada por Lewis e Vamvoudakis (2010), onde foi utilizado um método de AR chamado de programação dinâmica aproximada para projetar um controlador RLQ em tempo discreto, apenas observando os valores medidos de entrada e de saída. Essa integração representou uma solução para problemas de controle onde, em muitos casos, é infactível obter informações sobre a dinâmica interna do sistema.

Ademais, no trabalho apresentado por Rizvi e Lin (2019), foi utilizado um modelo de AR baseado em programação dinâmica adaptativa para definir os parâmetros ideais de um controlador RLQ para sistemas em tempo contínuo. Esse trabalho utilizou um algoritmo para observar as entradas e saídas da planta de controle e, a partir destas medidas, foram definidas equações de aprendizagem para o modelo de programação dinâmica adaptativa. Outra aplicação foi apresentada em (WANG; VELSWAMY; HUANG, 2018), onde foi apresentada uma abordagem de AR para aprender leis robustas de controle a partir dos dados processados de uma planta de um sistema de bobinas de aquecimento.

Para os trabalhos revisados nesta seção, foram utilizados métodos de otimização adaptativos por meio de AR para definir parâmetros internos ótimos de malhas de controle, considerando apenas os valores medidos de entrada e de saída da planta, sem ter nenhuma informação sobre a dinâmica interna do sistema. No entanto, nenhum dos artigos mencionados usou métodos AR para extrair características relevantes de uma imagem, e aproximar estas medidas como variáveis de estado do sistema, além de definir incertezas paramétricas derivadas do processamento de imagem.

2.4 Contribuições

Este capítulo apresentou métodos clássicos de controle baseados em visão para veículos autônomos. Os trabalhos revisados, geralmente, realizam o processamento de dados proveniente dos sensores e o cálculo de controle em etapas diferentes. Esta modularidade, além de ser dependente de informações previamente definidas sobre a dinâmica do sistema, promove uma falta de acoplamento que pode aumentar os ruídos e incertezas no modelo. Por outro lado, métodos de aprendizagem de máquinas, como APR, possibilitam este acoplamento entre percepção e tomadas de decisão, sem a necessidade de conhecimento prévio sobre a modelagem matemática do sistema e possibilitando uma parametrização automática das variáveis do sistema. Todavia, algoritmos de APR apresentam problemas para conduzir a aprendizagem de veículos autônomos, em especial devido à falta de generalização, tempo dispendioso para treinamento, baixa eficiência em sistemas instáveis ou realistas.

Com relação aos ambientes reais e controlados, este capítulo também apresentou procedimentos resultantes da combinação de métodos matemáticos com aprendizagem de máquinas, como a integração de controle clássico com algoritmos de AR. Estes algoritmos foram capazes de estimar o modelo matemático do sistema com eficiência, mas estas técnicas não foram exploradas em contextos complexos, como processamento de imagens.

Neste sentido, este trabalho propõe projetos de controle híbridos, onde a definição da dinâmica do sistema é feita a partir de algoritmos de APR aplicados em simuladores realistas para veículos autônomos, e a estabilidade geral do sistema é garantida a partir de técnicas de controle. Como apresentado nos resultados, o algoritmo desenvolvido neste trabalho foi capaz de definir variáveis de estado do sistema para uma tarefa específica de controle lateral a partir de uma imagem de entrada, usando um algoritmo de APR e um controlador recursivo. Ademais, o modelo proposto trabalha a robustez do sistema afim de superar os efeitos negativos de entradas exógenas e incertezas paramétricas.

3 FUNDAMENTAÇÃO TEÓRICA EM APRENDIZAGEM DE MÁQUINAS

Sistemas inteligentes vêm ganhando espaço cada vez maior no cenário de processos de automação e inovações no geral. É notório que eles estão liderando o caminho para otimização de todos os níveis de processos de produção, do chão de fábrica às vendas, até aspectos da vida cotidiana, como educação e lazer. Os sistemas inteligentes se baseiam no comportamento humano ou outras ações presentes na natureza, e as melhores e mais responsáveis decisões no cenário de automação são influenciadas por eles (CASTRO; CASTRO; TIMMIS, 2002).

Este capítulo introduz os métodos de inteligência computacional utilizados neste trabalho. Inicialmente, a base teórica sobre APR é descrita na Seção 3.1, onde são exploradas as técnicas para a definição do algoritmo utilizado, representadas pelo PPO, apresentado na Subseção 3.1.1, e pela CNN, apresentada na Subseção 3.1.2. Na sequência, a base teórica do algoritmo evolucionário EAND é apresentada na Seção 3.2.

3.1 Aprendizagem Profunda por Reforço

É possível distinguir três métodos principais para a aprendizagem de máquinas: aprendizagem supervisionada, não supervisionada e por reforço. Na aprendizagem supervisionada, os dados são rotulados antes de serem apresentados ao modelo. Durante o treinamento, o algoritmo encontra relações entre os dados de entrada com as saídas, e mantém essa relação nas etapas de teste. Para a aprendizagem não supervisionada, os dados não estão rotulados, sendo, então, objetivo do algoritmo aprender correlações entre os elementos e poder, por exemplo, separá-los em grupos com características semelhantes. AR, no entanto, representa um método intermediário, pois os dados são informados sem rótulos, mas o algoritmo recebe índices ou respostas que qualificam as decisões do algoritmo em função dos dados de entrada.

À vista disso, a AR pode ser entendida como uma área de inteligência artificial que tem como essência conduzir uma aprendizagem por meio de interações do tipo ação-consequência. Neste processo, um agente (responsável por realizar ações) interage com o seu ambiente, e aprende a alterar seu comportamento em função de observações de respostas adquiridas das suas ações, que podem ser recompensas ou punições (ARULKUMARAN et al., 2017).

O ambiente para AR representa o conjunto de informações disponíveis para o agente, onde, para cada instante de tempo t , o agente pode observar seu estado atual $x_t \in X$ e selecionar uma ação $u_t \in U$, resultando em uma transição para um novo estado x_{t+1} e um valor qualitativo para o par ação-estado $r_{t+1} \in R$. Nesta perspectiva, os modelos

Figura 4 – Diagrama de operação dos métodos de AR.

Fonte: Autor.

de AR também podem ser entendidos como Processos de Decisão de Markov (PDM) (BELLMAN, 1957), onde são apresentadas funções de probabilidades condicionais para ações que levam a recompensas e mudanças de estados e, diferente da aprendizagem supervisionada, a decisão atual do agente é consequência de seu par ação-estado anterior.

O PDM pode ser representado como a tupla (X, U, T, R, γ) , onde:

- X é uma lista finita de estados possíveis para o ambiente;
- U é uma lista finita de ações possíveis para o agente;
- $T \rightarrow [0, 1]$ é função de probabilidades de transição de estados, que informa a probabilidade do agente chegar ao estado x_{t+1} , partindo do estado x_t com a ação u_t ;
- $R \rightarrow \mathbb{R}$ é a função de recompensas, que indica o valor da recompensa que o agente recebe ao executar a ação u_t no estado x_t ;
- $\gamma \in [0, 1]$ é o fator de desconto, que faz um balanço entre recompensa imediata e futura.

O processo de atuação de um algoritmo de AR utilizando os princípios de PDM, é resumido na Fig. 4.

As consequências dos pares ação-estado são verificadas a partir das experiências passadas; portanto, os algoritmos AR visam reproduzir comportamentos para maximizar a expectativa de recompensas acumuladas, representada por:

$$V(x) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t | x_0 = x \right]. \quad (3.1)$$

Para alcançar este objetivo, o algoritmo de AR deve determinar uma política comportamental π_θ capaz de maximizar o retorno esperado das ações realizadas. Este processo apresenta o mesmo objetivo e formalismo matemático dos princípios de controle ótimo (ARULKUMARAN et al., 2017). Para esta operação, os parâmetros θ controlam a função

π_θ para selecionar uma ação u_t em função do estado x_t (CHEN; PENG; ZHANG, 2018). Portanto, o objetivo do algoritmo de AR pode ser resumido em definir os parâmetros ideais θ^* da política comportamental, como:

$$\theta^* = \arg \max_{\theta} V_{\pi_\theta}, \quad (3.2)$$

onde a função de valor do estado V_{π_θ} indica a expectativa de retorno para quando o agente segue a política π_θ .

Para aplicações reais ou em ambientes complexos, normalmente, a dinâmica das variáveis que compõem o sistema é desconhecida, o que, em muitos casos, inviabiliza a aplicação de modelos de PDM. No entanto, estes problemas podem ser resolvidos usando algoritmos de AR combinados com outras técnicas de aprendizado de máquina, como Redes Neurais Artificiais (RNAs).

As RNAs são sistemas massivamente paralelos e distribuídos, compostos por unidades de processamento simples que possuem uma capacidade natural de armazenar e utilizar conhecimento (HAYKIN et al., 2009). Por meio de uma estrutura conectada, na qual a saída de um conjunto de operações matemáticas é a entrada de outro conjunto, as RNAs podem aproximar qualquer função computacional com um número finito de descontinuidades a um modelo com precisão considerável (DEMUTH; BEALE; HAGAN, 2008).

Na Fig. 5, uma simplificação de um neurônio usado em uma RNA é comparada com um neurônio biológico. No exemplo tem-se N variáveis de entrada, sendo que cada uma está relacionada a um peso w . O conhecimento está armazenado nos pesos das conexões, e o processo de aprendizagem corresponde a identificar um conjunto apropriado de pesos de forma que a rede se comporte como desejado (CASTRO; ZUBEN, 2001). A soma de todas as entradas forma a entrada para a função de transferência f que, na sequência, forma a saída dos neurônios. Esse somatório é aplicado em um número finito de iterações, até que a RNA atinja um valor aceitável de diferença entre a saída informada pelo modelo neural e a saída esperada para o processo.

RNAs compostas por múltiplas camadas de processamento, com a capacidade de aprender representações de dados com várias camadas de abstração, são representadas pelas Redes Neurais Profundas (RNPs) (LECUN; BENGIO; HINTON, 2015). Os algoritmos de APR são definidos quando métodos de RNPs aproximam a função de valor do agente V_{π_θ} e a política π_θ , onde os parâmetros θ representam os pesos do modelo neural. Essa combinação permite o treinamento de algoritmos de AR com estados observáveis e ações em altas dimensões.

Este trabalho propõe o uso de um algoritmo de APR resultante da combinação do algoritmo PPO com CNNs. Para este procedimento, as CNNs são inicializadas com pesos aleatórios, representando um agente sem conhecimento sobre a dinâmica do ambiente.

Figura 5 – Estrutura de um neurônio artificial comparado com um neurônio biológico.

Fonte: Adaptado de (DEMUTH; BEALE; HAGAN, 2008).

À medida que o agente executa ações e acumula experiências, o algoritmo PPO define funções de custo aos modelos neurais, afim de ajustar os pesos das CNNs de modo a maximizar o retorno do par ação-estado do agente.

3.1.1 Proximal Policy Optimization

Os modelos de AR geram seus próprios dados de treinamento em função da sua política, isto faz com que as recompensas e observações mudem constantemente, sendo esta a principal causa de instabilidade nas etapas de treino. Ademais, esses modelos se mostram muito sensíveis a ajustes nos hiperparâmetros das RNAs. No entanto, o algoritmo PPO (SCHULMAN et al., 2017) foi proposto como um modelo de AR com implementação e ajustes simples, além de ser eficiente em gerar dados amostrais, o que soluciona algumas das limitações apresentadas.

Para o PPO, os parâmetros θ são ajustados por atualizações de gradiente e, assim, a política π_θ é definida diretamente pelo algoritmo. Esse método possui uma arquitetura *Actor-Critic*, formado por duas RNPs: a rede *Critic* realiza uma avaliação da política atual e atualiza os parâmetros da função de valor de estado V_{π_θ} , enquanto a rede *Actor* atualiza os parâmetros da política π_θ na direção sugerida pela rede *Critic*.

Em sua operação, o algoritmo PPO alterna entre coletar dados/amostras do ambiente e executar várias otimizações de uma função objetivo usando um estimador de amostragem (SCHULMAN et al., 2015a). Esse procedimento é proposto pela rede *Critic*, que usa o método GAE (do inglês *Generalized Advantage Estimator*) (SCHULMAN et al.,

2015b) para calcular a função *Advantage* ($\hat{\mathcal{A}}$) como:

$$\hat{\mathcal{A}}_t = \Lambda_t + (\gamma\varepsilon)\Lambda_{t+1} + \dots + (\gamma\varepsilon)^{N-t+1}\Lambda_{N-1}, \quad (3.3)$$

onde

$$\Lambda_t = r_t + \gamma V_\pi(x_{t+1}) - V_\pi(x_t), \quad (3.4)$$

$t = 0, \dots, N$, e γ e ε representam o fator de desconto e o parâmetro GAE, respectivamente. Portanto, $\hat{\mathcal{A}}_t$ pode ser interpretado como a qualidade da ação executada em função da expectativa do agente, ou seja, se a recompensa foi inferior, superior ou igual ao esperado.

No decorrer do processo, a política π_θ não retorna diretamente as ações do agente, mas sim uma média e um desvio padrão, que formam uma distribuição a partir da qual a ação é amostrada. Isso garante a exploração de diferentes ações. Usando a relação entre a política atual π_θ e a política antes da atualização dos parâmetros $\pi_{\theta_{old}}$, a função objetivo $\mathcal{L}^{CPI}$ (do inglês *Conservative Policy Iteration*) pode ser definida como:

$$\mathcal{L}_t^{CPI} = \mathbb{E}_t [\tau_t \hat{\mathcal{A}}_t], \quad (3.5)$$

onde

$$\tau_t = \frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)}. \quad (3.6)$$

Na sequência, o PPO restringe o espaço permitido para atualização dos parâmetros da política utilizando o componente *clip* da função objetivo, representado por:

$$\mathcal{L}_t^{CLIP} = \mathbb{E}_t [\min(\tau_t \hat{\mathcal{A}}_t; \text{clip}(\tau_t, 1 - \epsilon, 1 + \epsilon) \hat{\mathcal{A}}_t)], \quad (3.7)$$

onde ϵ é um hiperparâmetro. A Fig. 6 representa a otimização proposta pelo PPO para valores de $\hat{\mathcal{A}}_t$ positivos e negativos. É possível verificar que a função objetivo é limitada quando os valores de $\hat{\mathcal{A}}_t$ ultrapassam o limiar determinado por ϵ , impedindo qualquer aspecto caótico gerado por experiências ou valores extremos ou fora do padrão.

Por fim, para otimizar o desempenho do PPO, o modelo compartilha parâmetros entre as redes *Actor* e *Critic*, definindo a função de custo da rede *Actor* como:

$$\mathcal{L}_t = \mathbb{E}_t [\mathcal{L}_t^{CLIP} - c_1 \mathcal{L}_t^{VF} + c_2 \mathcal{S}[\pi_\theta](x_t)], \quad (3.8)$$

onde

$$\mathcal{L}_t^{VF} = [V_\pi(x_t) - r_t]^2 \quad (3.9)$$

representa a função de custo da rede *Critic*, c_1 e c_2 são coeficientes, e $\mathcal{S}$ é um bônus de entropia. O PPO requer apenas otimização de primeira ordem devido a suas taxas de probabilidade reduzidas, o que facilita a implementação e melhora o desempenho de vários problemas discretos e contínuos de alta dimensão. Nesse sentido, as CNNs foram consideradas como modelos de DL neste trabalho, representando as arquiteturas das redes *Actor* e *Critic*, e descritas a seguir.

Figura 6 – Função objetivo para valores de $\hat{A}_t$ positivos e negativos.

Fonte: Adaptado de (SCHULMAN et al., 2017).

3.1.2 Convolutional Neural Networks

As CNNs (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) são modelos de RNAs *feedforward* inspiradas em respostas biológicas do córtex visual. Estes modelos são computacionalmente eficientes e mostraram bons resultados ao processar e extrair características de pixels de imagens em problemas de aprendizado de máquinas. As CNNs são formadas por camadas convolucionais, camadas *pooling* e camadas totalmente conectadas. No conjunto convolucional, as CNNs executam operações de convolução nas imagens de entrada, deslizando uma matriz de números (chamada filtro ou *kernel*) e resultando em uma matriz de saída chamada de mapa de características, conforme definido por:

$$\mathcal{F}_{i_1 i_2} = (\mathcal{I} * \mathcal{K})_{i_1 i_2} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^3 \mathcal{K}_{m,n,c} \mathcal{I}_{i_1+m, i_2+n, c} + \mathcal{B}, \quad (3.10)$$

onde $\mathcal{F}$ representa o mapa de características de tamanho $i_1 \times i_2$, $\mathcal{I}$ é uma imagem de entrada com 3 canais (vermelho, azul e verde), $\mathcal{K}$ é um *kernel* de tamanho $k_1 \times k_2$ e $\mathcal{B}$ é o bias do modelo neural. Uma operação de convolução em uma imagem de entrada 7×7 com um *kernel* 3×3 é resumida na Fig. 7.

Na sequência, uma função de ativação não-linear é aplicada nos resultados de saída de cada camada. A Fig. 8 apresenta algumas funções de ativação usuais em RNAs, como: *i.* Função Degrau, que retorna os valores 0 ou 1; *ii.* Função Degrau Bipolar, que retorna 1 ou -1 ; *iii.* Função Log-Sigmoidal, que retorna saídas reais entre 0 e 1; *iv.* Função Tan-Sigmoidal, que expande o intervalo da Log-Sigmoidal para -1 e 1; *v.* Função Linear, que não altera as saídas dos neurônios; e a *vi.* Função ReLU (do inglês *Rectified Linear Units*), para retornar valor 0 quando $x < 0$ e retorna uma função linear quando $x \geq 0$.

As CNNs também usam estratégias para diminuir o custo computacional e a sensibilidade a ruídos. Por exemplo, um *stride* ou comprimento de passo pode ser aplicado nas camadas convolucionais, onde o *kernel* é deslocado em mais de um pixel em cada operação de convolução. Ademais, as camadas *pooling* são aplicadas para diminuir o número

Figura 7 – Operação de convolução em uma imagem de entrada 7×7 com um *kernel* 3×3 .

Fonte: Adaptado de <<https://medium.com/@bdhuma>>.

Figura 8 – Funções de ativação.

(e) Função Linear.

(f) Função ReLU.

Fonte: Adaptado de <<http://rasbt.github.io/mlxtend/>>.

de parâmetros para imagens de entrada com grandes dimensões. Este método reduz a dimensionalidade dos mapas de características, dividindo a matriz resultante em regiões e selecionando o valor máximo ou médio de cada região como o valor representativo de toda a divisão do mapa de características.

No fim dos processos de convolução, os resultados são passados como entradas para as camadas totalmente conectadas. Nesta etapa, um método de *dropout* (SRIVASTAVA et al., 2014) pode ser aplicado para evitar ajustes excessivos e diminuir o tempo de treinamento. No *dropout* um neurônio do modelo neural é escolhido com uma probabilidade fixa, e esse neurônio é temporariamente desativado, invalidando suas conexões de entrada e saída.

Após todo o processamento da imagem nas camadas da CNN, é iniciado o processo de treinamento. Para este procedimento, a CNN visa minimizar o erro associado à função de custo (E) a partir de algoritmos de otimização, como o SGD (do inglês *Stochastic Gradient Descent*) (BOTTOU, 2010). Esse algoritmo minimiza o erro ao parametrizar os erros na direção da saída desejada, como:

$$e_N = \frac{\partial E}{\partial \omega_{LN}}, \quad (3.11)$$

onde os N pesos das L camadas, ω_{LN} , produzem uma saída mais próxima do valor desejado ao serem ajustados na direção oposta ao gradiente.

Outro recurso utilizado na etapa de treinamento é o algoritmo *backpropagation* (LEUNG; HAYKIN, 1991), utilizado para definir o erro associado às camadas internas em função do erro calculado nas camadas externas. Esse processo de otimização ajusta os parâmetros dos *kernels*, de modo a extrair as melhores características da imagem para os mapas de características nas camadas convolucionais. Finalmente, o algoritmo SGD ajusta os pesos da camada totalmente conectada para melhor aproximar a função não-linear em análise.

3.2 Evolutionary Algorithm with Numerical Differentiation

O EAND foi apresentado como um algoritmo evolucionário, baseado em diferenciação numérica, para manter a diversidade em problemas de otimização não-lineares. Esse algoritmo ignora os parâmetros de cada indivíduo e considera os valores representativos derivados da função objetivo como referência para o processo de mutação. Desse modo, mesmo que os indivíduos possuam diferenças significativas em seus parâmetros, caso o valor associado à função objetivo de cada indivíduo seja próxima, esses elementos são considerados como pertencentes a população com pouca diversidade (MORAIS et al., 2019).

Ademais, o EAND propõe uma parametrização dinâmica, baseada nos princípios da diferenciação numérica, atualizada em cada processo de mutação. Esta parametrização

modifica o raio de busca em todas as iterações do algoritmo e, à medida que o processo de otimização evolui, o EAND realiza uma busca local em torno das melhores variáveis encontradas (MORAIS et al., 2019). O EAND foi inicialmente projetado para otimizar variáveis de entrada de sensores virtuais compostos por modelos caixa preta não-lineares, onde foi buscada as melhores combinações de defasagens de variáveis dentro de um intervalo de busca predefinido. O código-fonte do algoritmo pode ser encontrado em <<https://github.com/gprudencio>>, e a base teórica e implementação do algoritmo são apresentadas na sequência.

3.2.1 Diferenciação numérica

Diferenciação numérica é uma técnica analítica usada para encontrar uma derivada de curva por meio dos seus dados amostrados, sem explorar a função que a define. A derivada de um sinal pode apresentar detalhes sobre comportamento da função e a aproximação da derivada pode ser obtida por métodos de diferenciação numérica, como os procedimentos para pontos igualmente espaçados, representados pelas fórmulas de dois, três, quatro e cinco pontos (ZWILLINGER, 2003).

O método de cinco pontos, por exemplo, utiliza os valores da i -ésima abscissa a , igualmente espaçados por uma distância h , de modo que $h = a_{i+1} - a_i$. Na sequência, a função de avaliação no ponto i é definida por $y_i = f(a_0 + ih)$ e, como representado em (ZWILLINGER, 2003), a derivada do i -ésimo ponto pode ser aproximada a partir dos pontos consecutivos, como:

$$y'_i = \frac{1}{12h} (y_{i-2} - 8y_{i-1} + 8y_{i+1} - y_{i+2}), \quad (3.12)$$

sendo que os termos de erro na equação foram omitidos.

Em uma população $\mathbf{X}$, é possível associar cada indivíduo $\mathbf{x}_{ij} \in \mathbf{X}$, $i = 1, 2, \dots, N$, $j = 1, \dots, Dim$, com sua função de avaliação correspondente $\mathbf{y}_i \in \mathbf{Y}$, definida pela função objetivo do problema. Ademais, ao considerarmos que a função de avaliação dos indivíduos, representada por $\mathbf{y}_i$, esteja organizada de forma circular, podemos garantir que todos os pontos tenham vizinhos anteriores e posteriores, conforme apresentado na Fig. 9. Este arranjo possibilita a aplicação de métodos de derivação numérica em todos os valores de $\mathbf{y}_i$.

Ao aplicar a Eq. 3.12 (com $h = 1$) nos valores de $\mathbf{Y}$, é retornada a matriz $\mathbf{Y}'$, que mesmo não representando a derivada de $\mathbf{Y}$, uma vez que as variáveis independentes não estão ordenadas temporalmente, ela é capaz de definir uma relação entre indivíduos vizinhos no espaço de valores da função objetivo. Assim, a matriz $\mathbf{Y}'$ é usada no algoritmo EAND para determinar o comportamento das avaliações de indivíduos vizinhos à medida que a população evolui. Essa relação fornece informações sobre a convergência da população, ajudando o algoritmo a manter a diversidade.

Figura 9 – Distribuição circular de valores amostrais.

Fonte: Adaptado de (MORAIS et al., 2019).

3.2.2 Definição do algoritmo

O algoritmo EAND é formado por três processos: *i.* o primeiro é definido como inicialização, onde as variáveis são inicializadas dentro dos limites definidos para busca; *ii.* o segundo é o Método de Busca Global (MBG), onde operações de mutação e seleção são executados em todos os indivíduos da população; *iii.* o último é definido como Método de Busca Local (MBL), onde o processo evolutivo ocorre entre os 20% melhores indivíduos. Estas técnicas são descritas na sequência.

3.2.2.1 Inicialização

A população inicial $\mathbf{X}$ é criada aleatoriamente, com as variáveis independentes respeitando as restrições de limite exigidas pelo problema, de modo que:

$$\mathbf{x}_{ij} = \mathbf{l}_j - 1 + f_r (\mathbf{u}_j - \mathbf{l}_j), \quad (3.13)$$

onde $\mathbf{l}_j$ e $\mathbf{u}_j$ são os limites inferiores e superiores, respectivamente, de cada variável independente j do indivíduo $\mathbf{x}_i \in \mathbf{X}$, e f_r é uma função que gera um número aleatório entre 0 e o valor de entrada.

3.2.2.2 Método de Busca Global

Para este processo, o valor da função de avaliação $\mathbf{y}_i$ é calculado para todos os i indivíduos da população $\mathbf{X}$, e os valores associados a $\mathbf{y}'_i \in \mathbf{Y}'$ são calculados usando Eq. 3.12. Caso um dos i valores de $\mathbf{y}'_i$ seja igual a zero, ou todos os i valores sejam iguais, é aplicada a primeira condição de restrição (CR), representada por:

$$\begin{aligned} &\text{se } \mathbf{y}'_i \in \text{CR, então} \\ &\mathbf{y}'_i = r, \end{aligned} \quad (3.14)$$

onde r é um número real e aleatório entre 0 e 1. Na sequência, a variável $\mathbf{d}_i$ é definida para dimensionar o processo de mutação em função dos valores de $\mathbf{Y}'$, como apresentado em:

$$\mathbf{d}_i = \frac{\mathbf{y}'_i}{\frac{1}{N} \sum_{i=1}^N |\mathbf{y}'_i|}. \quad (3.15)$$

Em essência, a EAND modifica os valores das variáveis independentes dos indivíduos de acordo com os valores médios da população $\mathbf{X}$ e do inverso de $\mathbf{d}_i$. Assim, a função de mutação pode ser resumida por:

$$\mathbf{m}_{ij} = \mathbf{x}_{ij} + \frac{1}{d_i} \left(\frac{1}{N} \sum_{i=1}^N \mathbf{x}_{ij} \right), \quad (3.16)$$

onde $\mathbf{m}_{ij} \in \mathbf{M}$ é o i -ésimo indivíduo mutado.

Pode ser concluído, a partir da Eq. 3.16, que o fator $\mathbf{d}_i^{-1}$ intensifica o processo de mutação à medida que a população se tornar homogênea, evitando uma convergência prematura. Outra característica é que valores positivos ou negativos de $\mathbf{d}_i$ definem a direção da mutação individual, na qual ela poderá se aproximar ou se afastar dos valores médios da população.

Após a mutação, um método de restrições é implementado, onde é verificado se cada variável independente do indivíduo mutado é maior que o limite superior de busca $\mathbf{u}_j$ e, em caso de violação deste limite, a variável recebe um valor aleatório escolhido entre 75 e 100% do espaço factível de busca. Da mesma forma, se o valor alterado for menor que o limite inferior $\mathbf{l}_j$, este valor será substituído por um número aleatório escolhido de 0 a 25% do espaço de busca, como descrito abaixo:

$$\begin{aligned} &\text{se } \mathbf{m}_{ij} < \mathbf{l}_j, \text{ então} \\ &\quad \mathbf{m}_{ij} = f_r(0, 25\mathbf{u}_j) \\ &\text{se } \mathbf{m}_{ij} > \mathbf{u}_j, \text{ então} \\ &\quad \mathbf{m}_{ij} = f_r(0, 25\mathbf{u}_j) + 0,75\mathbf{u}_j. \end{aligned} \quad (3.17)$$

onde f_r é uma função que retorna um número aleatório entre 0 e o valor de entrada.

Para a seleção, o algoritmo EAND usa um método elitista para escolher os indivíduos da geração seguinte. Para esse processo, todos os indivíduos das populações $\mathbf{M}$ e $\mathbf{X}$ são classificados juntos de acordo com seus valores de avaliações. Na sequência, os N valores mais aptos são retornados aleatoriamente para a nova população $\mathbf{X}$. Após o processo de mutação e seleção, é inicializado o MBL, onde os processos de otimização são feitos considerando os melhores valores obtidos no MBG.

3.2.2.3 Método de Busca Local

Para o MBL, 20% dos melhores indivíduos da população $\mathbf{X}$ são armazenados ordenadamente em $\widehat{\mathbf{x}}_{hj} \in \widehat{\mathbf{X}}$, $h = 1, 2, \dots, q$, onde q é o número inteiro mais próximo

de $(0, 2 \times N)$ e $j = 1, \dots, Dim$; e os valores respectivos da função de avaliação são armazenados em $\hat{\mathbf{y}}_h \in \widehat{\mathbf{Y}}$. Nesse arranjo, o melhor indivíduo é armazenado na posição $\hat{\mathbf{x}}_{1j}$, sendo esse valor designado para liderar a busca no domínio local. Para esse procedimento, um novo operador, Δ_{kj} , em que $k = 1, 2, \dots, q - 1$, é usado para delimitar o novo espaço de busca, como representado por:

$$\Delta_{kj} = \hat{\mathbf{x}}_{1j} - \hat{\mathbf{x}}_{(k+1)j}. \quad (3.18)$$

No caso de algum valor de Δ_{kj} ser igual a 0, significa que as variáveis independentes dos indivíduos usados para definir a busca têm valores iguais, o que invalidaria o método apresentado nesta seção. Para evitar esse problema, a seguinte condição é aplicada:

$$\begin{aligned} \text{se } \Delta_{kj} = 0, \text{ então} \\ \Delta_{kj} = 1. \end{aligned} \quad (3.19)$$

O MBL calcula os indivíduos decorrentes da mutação, representados por $\widehat{\mathbf{m}}_{hj} \in \widehat{\mathbf{M}}$, em duas operações independentes, sendo a primeira envolvendo apenas o elemento $\widehat{\mathbf{m}}_{1j}$, e a segunda para os demais elementos $\widehat{\mathbf{m}}_{kj}$. Desse modo, o primeiro indivíduo (representado pelo melhor valor) realiza uma busca em função das suas próprias variáveis independentes, enquanto os outros indivíduos realizam uma busca em função da diferença entre eles e o melhor indivíduo. Esse processo é representado por:

$$\widehat{\mathbf{m}}_{1j} = \hat{\mathbf{x}}_{1j} + \hat{\mathbf{x}}_{1j} \times r \times ((-1)^{\hat{f}_r}), \quad (3.20)$$

$$\widehat{\mathbf{m}}_{kj} = \hat{\mathbf{x}}_{(k+1)j} + \Delta_{kj} \times r \times ((-1)^{\hat{f}_r}), \quad (3.21)$$

onde r é um número aleatório entre 0 e 1 e $\hat{f}_r$ é uma função que gera um número inteiro e aleatório entre 1 e o 2. Para concluir, é utilizada a estratégia elitista de seleção descrita para o MBG.

A [Fig. 10](#) representa o MBL para um problema de minimização com duas variáveis independentes, onde a matriz $\widehat{\mathbf{X}}$ é formada por três indivíduos, representados na [Fig. 10](#) por P_1 , P_2 e P_3 , cujos valores de função de avaliação são representados por $f(P_1)$, $f(P_2)$ e $f(P_3)$, respectivamente, sendo $f(P_1) < f(P_2) < f(P_3)$. A matriz Δ também é representada na figura, sendo Δ_{1j} e Δ_{2j} os valores de diferença dos indivíduos P_1 e P_2 , e P_1 e P_3 , respectivamente. Utilizando a [Eq. 3.20](#), é possível calcular o valor de $\widehat{\mathbf{m}}_1$ no espaço de busca A_1 ; e por meio da [Eq. 3.21](#), é possível calcular os valores de $\widehat{\mathbf{m}}_2$ e $\widehat{\mathbf{m}}_3$ nos espaços de busca A_2 e A_3 , respectivamente.

O [Algorithm 4](#) resume a implementação do EAND, o [Algorithm 2](#) descreve o MBG e o [Algorithm 3](#) descreve o MBL, sendo a variável t_{max} o número máximo de iterações definidas pelo usuário.

Figura 10 – MBL para um problema de otimização com duas variáveis independentes.

Fonte: (MORAIS et al., 2019).

Algorithm 1 *Evolutionary Algorithm with Numerical Differentiation*

- 1: Inicializa a população $\mathbf{X}$.
  - 2: While $t < t_{max}$
  - 3:   Avalia $\mathbf{Y} = f(\mathbf{X})$.
  - 4:   Executa o MBG.
  - 5:   Executa o MBL.
  - 6: $t = t + 1$
  - 7: End While
-

Algorithm 2 Método de Busca Global

- 1: Calcula $\mathbf{Y}'$ usando Eq. 3.12.
  - 2: Verifica a condição de restrição Eq. 3.14.
  - 3: Calcula $\mathbf{d}_i$ usando Eq. 3.15.
  - 4: Calcula $\mathbf{M}$ usando Eq. 3.16.
  - 5: Verifica a condição de restrição Eq. 3.17.
  - 6: Executa as técnicas de seleção.
  - 7: Atualiza as matrizes $\widehat{\mathbf{X}}$ e $\widehat{\mathbf{Y}}$.
-

Algorithm 3 Método de Busca Local

- 1: Calcula Δ_{kj} usando Eq. 3.18.
  - 2: Verifica a condição de restrição Eq. 3.19.
  - 3: Calcula $\widehat{\mathbf{m}}_{1j}$ usando Eq. 3.20.
  - 4: Calcula $\widehat{\mathbf{m}}_{(h+1)j}$ usando Eq. 3.21.
  - 5: Verifica a condição de restrição Eq. 3.17.
  - 6: Executa as técnicas de seleção.
  - 7: Atualiza a matriz $\mathbf{X}$.
-

4 FUNDAMENTAÇÃO TEÓRICA EM CONTROLE ROBUSTO

Este capítulo apresenta o problema de otimização do regulador RLQR e sua solução, conforme propostas por (TERRA; CERRI; ISHIHARA, 2014). Esse método tem como princípio uma otimização do tipo min-max de desempenho quadrático penalizado, onde dois objetivos opostos são avaliados de modo a definir a melhor ação de controle sujeita a máxima influência de incertezas. Para essa formulação, a função de custo é minimizada utilizando uma lei de controle ideal u_t de modo que $u_t = K_t x_t$, com $K_t \in \mathbb{R}^{n \times n}$ representando o ganho de realimentação do sistema.

4.1 Formulação do Problema

Considere o seguinte sistema de tempo discreto sujeito a incertezas paramétricas:

$$\begin{aligned} x_{t+1} &= (F_t + \delta F_t)x_t + (G_t + \delta G_t)u_t, \\ x_0 &= \text{constante}, \end{aligned} \quad (4.1)$$

onde $t = 0, \dots, N$, $x_t \in \mathbb{R}^n$ é o vetor de estados, $u_t \in \mathbb{R}^m$ é a entrada de controle, $F_t \in \mathbb{R}^{n \times n}$ e $G_t \in \mathbb{R}^{n \times m}$ são as matrizes conhecidas do modelo, e δF_t e δG_t representam incertezas paramétricas.

As matrizes de incerteza são definidas como:

$$\begin{bmatrix} \delta F_t & \delta G_t \end{bmatrix} = H_t \Delta_t \begin{bmatrix} E_{F_t} & E_{G_t} \end{bmatrix}, \quad (4.2)$$

onde $t = 0, \dots, N$ e $H_t \in \mathbb{R}^{n \times p}$, $E_{F_t} \in \mathbb{R}^{l \times n}$ e $E_{G_t} \in \mathbb{R}^{l \times m}$ são matrizes conhecidas, enquanto $\Delta_t \in \mathbb{R}^{p \times l}$ é uma matriz arbitrária, tal que $\|\Delta_t\| \leq 1$.

O controlador RLQR se origina da solução do problema de otimização de mínimos quadrados (TERRA; CERRI; ISHIHARA, 2014), representado por:

$$\min_{x_{t+1}, u_t} \max_{\delta F_t, \delta G_t} \bar{J}_t^\mu(x_{t+1}, u_t, \delta F_t, \delta G_t), \quad (4.3)$$

onde $\bar{J}_t^\mu$ é a função de custo:

$$\bar{J}_t^\mu(x_{t+1}, u_t, \delta F_t, \delta G_t) = \begin{bmatrix} x_{t+1} \\ u_t \end{bmatrix}^T \begin{bmatrix} P_{t+1} & 0 \\ 0 & R_t \end{bmatrix} \begin{bmatrix} x_{t+1} \\ u_t \end{bmatrix} + \Phi^T \begin{bmatrix} Q_t & 0 \\ 0 & \mu I \end{bmatrix} \Phi, \quad (4.4)$$

com penalidade $\mu > 0$ fixa, e as matrizes de ponderação $Q_t \succ 0$, $R_t \succ 0$, $P_{t+1} \succ 0$ e

$$\Phi = \left\{ \begin{bmatrix} 0 & 0 \\ I & -G_t - \delta G_t \end{bmatrix} \begin{bmatrix} x_{t+1} \\ u_t \end{bmatrix} - \begin{bmatrix} -I \\ F_t + \delta F_t \end{bmatrix} x_t \right\}.$$

É possível considerar que (4.3)-(4.4) pode ser reduzido ao Problema de minimização padrão em que o sistema não está sujeito a incertezas.

4.2 Problema de mínimos quadrados regularizado

Os seguintes resultados são baseados em (KAILATH ALI H. SAYED, 2000) para mínimos quadrados regularizados para sistemas sem incerteza. Considerando o problema de otimização representado por:

$$\min_{x \in \mathbb{R}^m} \{J(x)\}, \quad (4.5)$$

onde $J(x)$ é o custo quadrático regularizado:

$$J(x) = x^T Q x + (Ax - b)^T W (Ax - b), \quad (4.6)$$

com $A \in \mathbb{R}^{m \times m}$ e $b \in \mathbb{R}^n$ valores conhecidos, $x \in \mathbb{R}^m$ o vetor de estados, $Q \in \mathbb{R}^{m \times m}$ uma matriz simétrica positiva-definida e $W \in \mathbb{R}^{n \times n}$ uma matriz simétrica positiva semi-definida, a solução pode ser obtida pelo lema 4.2.1:

Lemma 4.2.1. *A solução ótima para o problema de otimização (4.5)-(4.6) é dada por:*

$$x^* = (Q + A^T W A)^{-1} A^T W b. \quad (4.7)$$

Para demonstração, consultar (KAILATH ALI H. SAYED, 2000).

4.3 Problema de mínimos quadrados regularizado robusto

Considere o Problema de otimização (4.3) definido por meio de uma expressão geral em que as informações de dados, fornecidos na matriz A , e as medidas, fornecidas no vetor b , estão sujeitos a incertezas:

$$\min_x \max_{\delta A, \delta b} \{J(x, \delta A, \delta b)\}, \quad (4.8)$$

onde

$$J(x) = x^T Q x + ((A + \delta A)x - (b + \delta b))^T W ((A + \delta A)x - (b + \delta b)), \quad (4.9)$$

cujas incertezas δA e δb foram modeladas como:

$$\begin{bmatrix} \delta A & \delta b \end{bmatrix} = H \Delta \begin{bmatrix} E_A & E_b \end{bmatrix}, \quad (4.10)$$

com A , b , H , E_A , E_b , Q e W matrizes conhecidas de dimensões apropriadas, Δ é uma matriz arbitrária ($\|\Delta\| \leq 1$) e x é um vetor desconhecido. O Lema 4.3.1 (SAYED; NASCIMENTO, 1999) apresenta a solução ótima para o Problema (4.8)-(4.10):

Lemma 4.3.1. *A solução para o Problema de otimização (4.8)-(4.10) é:*

$$x^* = (\hat{Q} + A^T \hat{W} A)^{-1} (A^T \hat{W} b + \hat{\lambda} E_A^T E_b),$$

com $\hat{Q}$ e $\hat{W}$ definidas como:

$$\begin{aligned}\hat{Q} &:= Q + \hat{\lambda} E_A^T E_A, \\ \hat{W} &:= W + WH(\hat{\lambda}I - H^TWH)^\dagger H^TW,\end{aligned}$$

onde $\hat{\lambda} > \|H^TWH\|$. Para demonstração, consultar ([SAYED; NASCIMENTO, 1999](#)).

Uma solução equivalente para o Problema (4.8)-(4.10), escrito em termos de vetores de matrizes, é mostrado no Lema 4.3.2.

Lemma 4.3.2. *Se $Q \succ 0$ e $W \succ 0$, então a solução x^* para o Problema (4.8)-(4.10) pode ser expresso como:*

$$\begin{bmatrix} x^* \\ J(x^*) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & b \\ 0 & E_b \\ I & 0 \end{bmatrix}^T \begin{bmatrix} Q^{-1} & 0 & 0 & I \\ 0 & \hat{W}^{-1} & 0 & A \\ 0 & 0 & \hat{\lambda}^{-1}I & E_A \\ I & A^T & E_A^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ b \\ E_b \\ 0 \end{bmatrix},$$

com $\hat{W}$ e $\hat{\lambda}$ como no Lema 4.3.1.

Para demonstração, consultar ([CERRI; TERRA; ISHIHARA, 2009](#)).

4.4 Regulador Linear Quadrático Recursivo Robusto

Nesta seção, é aplicado o lema 4.3.2 com $\mu > 0$ para resolver o problema de otimização (4.3)-(4.4), baseado em ([TERRA; CERRI; ISHIHARA, 2014](#)):

$$\begin{bmatrix} x_{t+1}^* \\ u_t^* \end{bmatrix} = \begin{bmatrix} L_t \\ K_t \end{bmatrix} x_t^*, \quad t = 0, \dots, N, \quad (4.11)$$

onde L_t e K_t resultam da recursão:

$$\begin{bmatrix} L_t \\ K_t \\ P_t \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -I \\ 0 & 0 & \hat{F}_t \\ I & 0 & 0 \\ 0 & I & 0 \end{bmatrix}^T \begin{bmatrix} P_{t+1}^{-1} & 0 & 0 & 0 & I & 0 \\ 0 & R_t^{-1} & 0 & 0 & 0 & I \\ 0 & 0 & Q_t^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \Sigma_t(\mu, \hat{\lambda}_t) & \hat{I} & -\hat{G}_t \\ I & 0 & 0 & \hat{I}^T & 0 & 0 \\ 0 & I & 0 & -\hat{G}^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ -I \\ \hat{F}_t \\ 0 \\ 0 \end{bmatrix}, \quad (4.12)$$

com

$$\Sigma_t = \begin{bmatrix} \mu^{-1}I - \hat{\lambda}_t^{-1}H_tH_t^T & 0 \\ 0 & \hat{\lambda}_t^{-1}I \end{bmatrix}, \quad \hat{I} = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad \hat{G}_t = \begin{bmatrix} G_t \\ E_{G_t} \end{bmatrix}, \quad \hat{F}_t = \begin{bmatrix} F_t \\ E_{F_t} \end{bmatrix}.$$

Para o problema (4.3)-(4.4), é considerado as seguintes identificações com (4.9)-(4.10):

$$\begin{aligned}
 Q &\leftarrow \begin{bmatrix} P_{t+1} & 0 \\ 0 & R_t \end{bmatrix}, \quad W \leftarrow \begin{bmatrix} Q_t & 0 \\ 0 & \mu I \end{bmatrix}, \quad x \leftarrow \begin{bmatrix} x_{t+1} \\ u_t \end{bmatrix}, \\
 A &\leftarrow \begin{bmatrix} 0 & 0 \\ I & -G_t \end{bmatrix}, \quad \delta A \leftarrow \begin{bmatrix} 0 & 0 \\ 0 & -\delta G_t \end{bmatrix}, \quad b \leftarrow \begin{bmatrix} -I \\ F_t \end{bmatrix} x_t, \quad \delta b \leftarrow \begin{bmatrix} 0 \\ \delta F_t \end{bmatrix} x_t, \\
 H &\leftarrow \begin{bmatrix} 0 \\ H_t \end{bmatrix}, \quad \Delta \leftarrow \Delta_i, \quad E_A \leftarrow \begin{bmatrix} 0 & -E_{G_t} \end{bmatrix}, \quad E_b \leftarrow E_{F_t} x_t.
 \end{aligned} \tag{4.13}$$

Em seguida, aplicando o resultado do lema 4.3.2 e realizando algumas manipulações algébricas, as Equações (4.14) e (4.15) são obtidas (TERRA; CERRI; ISHIHARA, 2014).

Quando as incertezas desaparecem, o RLQ nominal pode ser obtido por:

$$\begin{bmatrix} x_{t+1}^* \\ u_t^* \end{bmatrix} = \begin{bmatrix} L_t \\ K_t \end{bmatrix} x_t^*, \quad t = 0, \dots, N, \tag{4.14}$$

onde L_t e K_t resultam da recursão:

$$\begin{bmatrix} L_t \\ K_t \\ P_t \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -I \\ 0 & 0 & F_t \\ I & 0 & 0 \\ 0 & I & 0 \end{bmatrix}^T \begin{bmatrix} P_{t+1}^{-1} & 0 & 0 & 0 & I & 0 \\ 0 & R_t^{-1} & 0 & 0 & 0 & I \\ 0 & 0 & Q_t^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & -G_t \\ I & 0 & 0 & I^T & 0 & 0 \\ 0 & I & 0 & -G_t^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ -I \\ F_t \\ 0 \\ 0 \end{bmatrix}. \tag{4.15}$$

5 DEFINIÇÃO DO MODELO PROPOSTO

Este capítulo detalha a modelagem dos controladores propostos para condução autônoma de veículos, referenciados em visão computacional, para manter o veículo alinhado com o centro da via em um ambiente simulado. Para essas abordagens, três algoritmos de controle foram implementados, sendo que todos os modelos recebiam como entrada imagens RGB frontais do ambiente, e retornavam uma ação de direção contínua afim de manter o veículo dirigindo com velocidade constante no centro da via.

Na Seção 5.1 é apresentada a primeira abordagem de controle, definida como um algoritmo de APR. A seção define a equação de recompensa associada ao algoritmo e detalha as configurações do modelo neural e a implementação utilizada. Durante o processo de treinamento realizado pelo algoritmo de APR, os *kernels* das camadas convolucionais da CNN foram parametrizados de modo a intensificar as características da imagem que mais influenciam no controle lateral. Foram executados experimentos para avaliar os canais de saída dos mapas de características resultantes, e foram selecionados os canais que inferiam sensibilidade ao deslocamento lateral do veículo. A Seção 5.2 destaca este procedimento.

O segundo procedimento de controle, apresentado como o RLQ híbrido, é discutido na Seção 5.3. Para essa operação, o modelo neural executa operações de convolução nas imagens de entrada do algoritmo e os canais de saída selecionados são interpretados pelo controlador como variáveis de estado do sistema.

Finalmente, na Seção 5.4 é apresentada a terceira abordagem de controle, definida como o RLQR híbrido. Este modelo é apresentado como uma otimização do modelo híbrido anterior, visto que foi aplicado robustez à nova configuração. Este novo controlador híbrido considera o algoritmo evolucionário EAND (MORAIS et al., 2019) como método de busca e definição das melhores matrizes de incerteza recursiva adequadas à dinâmica do sistema. As abordagens de controle exploradas neste capítulo são resumidas na Fig. 11.

5.1 Aprendizagem Profunda por Reforço

Esta seção explora o modelo de aprendizado de máquinas, representado como um algoritmo de APR, e resultante da combinação do algoritmo PPO (SCHULMAN et al., 2017) com uma CNN (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Conforme apresentado no Capítulo 3, o PPO determina a política comportamental do agente usando métodos de otimização de gradiente, e se apresenta como um modelo com implementação simples e eficiente para amostras complexas. Da mesma forma, a CNN se apresenta como um modelo neural computacionalmente eficiente e que possui bons desempenhos envolvendo treinamento com imagens.

Figura 11 – Fluxograma com a definição das três abordagens de controle apresentadas neste capítulo.

Fonte: Autor.

Inicialmente, a função de recompensa para o algoritmo de APR é definida na Subseção 5.1.1. Esta função implica diretamente no comportamento do agente, o que torna sua modelagem uma etapa crítica na definição do algoritmo. Na sequência, a arquitetura de rede neural é apresentada em detalhes na Subseção 5.1.2.

5.1.1 Função de recompensas

Como apresentado anteriormente, o estado observável de um algoritmo de AR representa os sensores e medições que o agente utiliza para interpretar o ambiente. Utilizando as informações disponíveis do estado observável, um modelo de AR tem como objetivo maximizar uma função de recompensa qualificadora do par ação-estado e, consequentemente, aprender a mapear o estado para uma ação. Portanto, a função de recompensa r_t no tempo t foi definida como:

$$r_t = w_c r_{c_t} - w_b r_{b_t}. \quad (5.1)$$

A Eq. 5.1 é representado por uma variável associada a recompensas, r_{c_t} , que qualifica a posição do veículo em relação a posição desejada, e uma variável associada a punições, r_{b_t} , que avalia colisões com variáveis estáticas, como postes e muros.

A variável r_{c_t} é definida como uma função Gaussiana que relacionada a medida atual da distância Euclidiana (d_t) entre o centro da pista e a posição do veículo, de modo

que:

$$\begin{cases} r_{c_t} = 1 \text{ se } d_t = 0, \\ r_{c_t} \text{ diminui quando } d_t \text{ aumenta,} \\ r_{c_t} = 0 \text{ se } d_t > W_{st}/2, \end{cases} \quad (5.2)$$

onde W_{st} é a largura da via navegável. Nesse sentido, $0 \leq r_{c_t} \leq 1$ quando o veículo estiver dentro dos limites permitidos para o deslocamento, sendo $r_{c_t} = 1$ quando o veículo estiver exatamente no centro da via, e $r_{c_t} = 0$ quando estiver fora da via navegável. Já a variável r_{b_t} é definida por meio de um sinal normalizado do sensor de colisão, fornecido pelo simulador, e que mede a ocorrência e a intensidade de colisões. O valor de r_{b_t} também varia de 0 a 1.

O objetivo de um agente de APR é maximizar sua função de recompensas e, neste sentido, recompensas positivas encorajam o agente a realizar mais ações. Por outro lado, recompensas negativas ou punições representam uma resistência em desempenhar atividades e uma tentativa de retorno a situações de respostas benéficas. Afim de intensificar as ações, foi atribuído valores 100 e 10 aos pesos w_c e w_b , respectivamente. As recompensas e os pesos associados mostram sentido ao movimento, e a sequência detalhada nesta seção direcionam o comportamento de forma a não permitir ambiguidades. Por fim, a função de recompensas também define a função de custo do modelo neural descrito a seguir.

5.1.2 Modelo neural

O modelo de CNN considerado foi formado por três camadas convolucionais, cada uma com dois *kernels*, e duas camadas totalmente conectadas. Esta configuração de camadas convolucionais resulta em dois mapas de características distintos, cada um com três canais (vermelho, azul e verde). Os *kernels* são parametrizados no processo de treinamento, com o intuito de destacar as características da imagem que mais influenciam no problema de controle de segmento de trajetória. A entrada do modelo neural é o estado observável do agente, definido pelas três últimas imagens RGB do ambiente, representadas na Fig. 15, sendo que estas imagens foram concatenadas para promover uma percepção temporal do movimento do agente durante o processo de aprendizagem.

Como resultado, as camadas convolucionais retornam 778 detecções de características distintas, resultantes de 6 mapas de características, cada um com 128 canais de saída. As camadas totalmente conectadas processam esses mapas, e retornam uma ação contínua de direção que varia de -1 a 1 , representada por u_t . Este design é resumido na Fig. 12.

5.2 Determinação das referências de controle

A seção anterior apresenta uma CNN cujas operações convolucionais resultam em 768 canais de saída distintos. Esta seção discute procedimentos para selecionar quais

Figura 12 – Arquitetura do modelo neural proposto para o algoritmo de APR.

Fonte: Autor.

canais de saída são capazes de inferir características suficientes da imagem para manter o veículo no centro da via e, assim, serem utilizados como referências de controle. Para esta seleção, foi utilizado um procedimento de comparação multiobjetivo, onde foi avaliada a sensibilidade ao deslocamento lateral de cada saída convolucional medida.

5.2.1 Seleção dos canais de saída

Para a comparação multiobjetivo, o veículo foi conduzido manualmente em três situações diferentes no simulador, e os valores dos canais de saída foram avaliados em cada situação. Na primeira simulação, o veículo seguiu a trajetória no centro da rua e as saídas das camadas convolucionais foram armazenadas no vetor O_C . Em seguida, na segunda simulação, o veículo seguiu a trajetória deslocado para a direita em relação ao experimento anterior e as respostas de saída foram armazenadas no vetor O_R . Finalmente, na última simulação, o veículo foi deslocado para o lado esquerdo, e as respostas dos canais de saída foram armazenados no vetor O_L . Nas três situações o veículo se manteve nos limites da via permitidos para deslocamento.

A simulação coletou m_x amostras em cada operação e, para cada canal de saída n_x , foi definida uma matriz de decisão $\mathbf{a}_i = (O_{C_{ij}}, O_{R_{ij}}, O_{L_{ij}})^T$, $i = \{1, \dots, n_x\}$, $j = \{1, \dots, m_x\}$. Os dados da matriz $\mathbf{a}$ também foram normalizados entre valores 0 e 1 para tornar a comparação factível. O procedimento de comparação multiobjetivo considerado para este trabalho visou selecionar os melhores canais de saída em três funções objetivos conflitantes, descritas como segue:

1. A primeira função objetivo compara a simetria entre os três deslocamentos. Para essa métrica, é desejada a norma mínima entre os deslocamentos laterais e a referência central, representada por O_C . Essa função é definida como:

$$\arg \min_i (f_1(\mathbf{a}_i) = \|(\|O_{C_i} - O_{R_i}\| - \|O_{C_i} - O_{L_i}\|)\|). \quad (5.3)$$

2. Na sequência, a segunda função objetivo avalia a sensibilidade ao deslocamento lateral. Para esta função, é desejado o máximo erro quadrático médio entre o veículo

percorrendo o caminho central O_C , e as medidas obtidas quando o veículo realizou os deslocamentos laterais O_R e O_L :

$$\arg \max_i \left(f_2(\mathbf{a}_i) = \frac{1}{m} \sum_{j=1}^m \left((O_{C_{ij}} - O_{R_{ij}})^2 + (O_{C_{ij}} - O_{L_{ij}})^2 \right) \right). \quad (5.4)$$

3. Finalmente, o terceiro objetivo avalia o espalhamento ou perturbações dos resultados medidos. É desejado valores menos ruidosos ou sem *outliers* e, portanto, são buscados canais de saídas com medidas com o menor desvio padrão:

$$\arg \min_i \left(f_3(\mathbf{a}_i) = \sqrt{\frac{1}{m-1} \sum_{i=1}^m \left((O_{C_{ij}} - \bar{O}_{C_i})^2 + (O_{R_{ij}} - \bar{O}_{R_i})^2 + (O_{L_{ij}} - \bar{O}_{L_i})^2 \right)} \right). \quad (5.5)$$

Na sequência, o conceito de dominância é aplicado para selecionar os melhores resultados no problema multiobjetivo proposto. Para este conceito de comparação, ao se avaliar dois vetores de decisão $\mathbf{a}_a$ e $\mathbf{a}_b$, pode-se concluir que $\mathbf{a}_a$ domina $\mathbf{a}_b$ se e somente se:

$$\begin{aligned} f_1(\mathbf{a}_a) &\leq f_1(\mathbf{a}_b) \\ f_2(\mathbf{a}_a) &\geq f_2(\mathbf{a}_b) \\ f_3(\mathbf{a}_a) &\leq f_3(\mathbf{a}_b), \end{aligned} \quad (5.6)$$

e para pelo menos uma função objetivo:

$$\begin{aligned} \exists f_1(\mathbf{a}_a) &< f_1(\mathbf{a}_b) \\ \exists f_2(\mathbf{a}_a) &> f_2(\mathbf{a}_b) \\ \exists f_3(\mathbf{a}_a) &< f_3(\mathbf{a}_b). \end{aligned} \quad (5.7)$$

Um vetor de decisão $\mathbf{a}_i^*$ é definido como um valor ótimo de Pareto se nenhuma solução no espaço de busca dominar $\mathbf{a}_i^*$. Todas as soluções ótimo de Pareto compõem o Conjunto Pareto (CP), como:

$$\text{CP} := \{\mathbf{a}_i | \mathbf{a}_i \text{ é valor ótimo de Pareto}\}. \quad (5.8)$$

A composição das funções objetivo do CP definem a Frente de Pareto (FP), como:

$$\text{FP} := \{f_1(\mathbf{a}_i), f_2(\mathbf{a}_i), f_3(\mathbf{a}_i) | \mathbf{a}_i \in \text{CP}\}. \quad (5.9)$$

A FP é composta por canais de saída não dominados e sensíveis ao deslocamento lateral e, portanto, estes valores podem ser aplicados como referências de controle. No entanto, uma FP pode ser composto por mais de um único canal. Desse modo, é necessário definir um número mínimo de entradas para diminuir os custos computacionais e promover sistemas de controle menos complexos.

Portanto, este trabalho considera como referência de controle, o canal de saída mais centralizado na FP, representado por $\mathbf{a}_c$, e que pode promover o melhor equilíbrio entre as três funções objetivo. Caso o controlador não atinja um bom desempenho utilizando somente a variável $\mathbf{a}_c$ como entrada, é válido adicionar os κ vizinhos mais próximos de $\mathbf{a}_c$ à entrada do sistema, sendo κ definido pelo projetista do controlador.

5.2.2 Sinal de erro e ação de controle

Após a seleção do canal de saída normalizado, representado por $\mathbf{a}_c$, é definido um medidor de centralidade do veículo com a via e, assim, é possível inferir um valor de erro a ser minimizado por uma ação de direção. Inicialmente, foi considerado que valores negativos de direção significam curva à esquerda, enquanto valores positivos significam curva à direita. Para determinar o valor do erro em relação a referência, primeiro é avaliado se o valor médio dos dados obtidos na simulação deslocada para a direita ($\overline{O}_R$) é maior ou menor que o valor médio dos dados da simulação realizada no centro da via ($\overline{O}_C$). Dessa forma, considerando que o vetor O_C permite uma representação do caminho desejado, o valor de $\overline{O}_C$ é adotado como a referência de controle, tornando possível a determinação do erro de deslocamento (e), como:

$$\begin{aligned} &\text{Se } \overline{O}_C > \overline{O}_R, \text{ então} \\ &\quad e = \mathbf{a}_c - \overline{O}_C \\ &\text{Senão} \\ &\quad e = \overline{O}_C - \mathbf{a}_c. \end{aligned} \tag{5.10}$$

Devido à operação de convolução da imagem, o vetor de referência O_C pode possuir valores que variam consideravelmente em torno do valor médio $\overline{O}_C$, o que pode levar ações de direção oscilatórias. Para suavizar o movimento de direção, o sinal de erro foi atenuado quando sua medição estiver dentro de um intervalo de saturação. Foi considerado o desvio padrão (σ) do caminho desejado como delimitador do intervalo de saturação. Com essa abordagem, foi aplicada a condição de restrição para o sinal de erro de controle saturado:

$$\begin{aligned} &\text{Se } -\sigma(O_C) \leq e \leq \sigma(O_C), \text{ então} \\ &\quad e = 0, \end{aligned} \tag{5.11}$$

de modo que a influência de variações de erro com baixa magnitude em torno do valor desejado sejam diminuídas.

A [Fig. 13](#) resume o modelo híbrido proposto entre aprendizagem de máquinas e controle de malha fechada. As três últimas imagens RGB do ambiente são processadas por operações de convolução de uma CNN treinada, cujos canais de saída selecionados fornecem sinais de erro a serem minimizados pelo modelo de controle.

Figura 13 – Modelo híbrido: os canais de saída da CNN fornecem variáveis de entrada ao controlador.

Fonte: Autor.

5.3 Identificação do espaço de estados do sistema

O primeiro controlador a ser combinado com a CNN treinada foi o RLQ. Considerando que o ganho de realimentação K é calculado usando as matrizes que representam a dinâmica do sistema, F e G , é necessário realizar a estimação destes modelos. Entre os métodos aplicáveis para esta etapa, este trabalho considerou o método de estimativa de erro quadrático médio apresentado em (YOUNG; GARNIER; GILSON, 2008).

Inicialmente, foi considerado uma análise experimental para a identificação do sistema, onde o veículo foi dirigido manualmente no simulador, e as informações de variações de ações, que incluíam diferentes valores de esterçamento do volante, junto com as variações de estados correspondentes foram medidas. Para este procedimento, o modelo do sistema foi descrito no formato:

$$y_t = \varphi_t^T \Theta_t + v_t, \quad (5.12)$$

onde y_t e φ_t são vetores de valores medidos para a ação e os estados, respectivamente, v_t representa medidas de erros e perturbações e Θ_t é o vetor cujos parâmetros serão determinados, representados pelas matrizes F e G , sendo todas variáveis avaliadas em cada instante de tempo t .

Sendo o preditor de parâmetro definido como:

$$\hat{y}(t, \Theta) = \varphi_t^T \Theta_t, \quad (5.13)$$

a estimativa por mínimos quadrados $\hat{\Theta}_{LS}$ (YOUNG; GARNIER; GILSON, 2008) pode ser obtida como:

$$\hat{\Theta}_{LS} = \left[\frac{1}{N} \sum_{t=1}^N \varphi_t \varphi_t^T \right]^{-1} \frac{1}{N} \sum_{t=1}^N \varphi_t y_t. \quad (5.14)$$

Considerando que essa identificação de parâmetros é adequada para vetores, o

modelo em espaço de estados foi reorganizado como:

$$\underbrace{x_{j,t+1}}_{y_t} = \underbrace{\begin{bmatrix} x_t^T & u_t^T \end{bmatrix}}_{\varphi_t^T} \underbrace{\begin{bmatrix} F_j^T \\ G_j^T \end{bmatrix}}_{\Theta_t}, \quad (5.15)$$

onde $j = 1, 2$ representa a j -ésima linha das respectivas matrizes. Este procedimento tornou possível identificar diretamente cada uma das linhas das matrizes em espaço de estados.

5.4 Estimativa das matrizes de incerteza por busca evolutiva

Uma outra parte crítica deste trabalho foi estimar as matrizes de incerteza E_F e E_G para fornecer um design de controle robusto. Considerando que os parâmetros físicos do sistema foram derivados a partir de um método de aprendizagem de máquinas e, portanto, são desconhecidos, foi escolhido um método de busca evolucionária realizada pelo algoritmo EAND (MORAIS et al., 2019) para otimizar e determinar os melhores valores para as incertezas.

O algoritmo EAND é composto por dois processos principais de mutação: o primeiro conduz o processo de otimização de acordo com métodos de parametrização dinâmicas, com base nos princípios de diferenciação numérica entre os indivíduos; enquanto o segundo realiza uma busca em torno dos melhores indivíduos da população. Ademais, o EAND representa um algoritmo evolucionário com fácil implementação computacional, alta capacidade de evolução adaptativa e apresenta um novo método de parametrização, que intensifica o processo de mutação à medida que a população se torna homogênea, evitando uma convergência prematura e mantendo um equilíbrio entre a busca local e global (MORAIS et al., 2019).

Para o procedimento de otimização proposto, uma população aleatória foi inicializada com n indivíduos, sendo cada indivíduo representando por um conjunto de matrizes de incertezas, de modo que $\hat{\mathbf{X}}_i = [E_{F_i}, E_{G_i}]$, $i = \{1, \dots, n\}$. Para cada nova interação, o algoritmo definiu um modelo de controle RLQR híbrido, sendo as entradas do sistema as matrizes F e G , estimadas a partir do método apresentado na seção anterior, e os valores de incerteza $\hat{\mathbf{X}}_i$. No decorrer do processo, o algoritmo retornava um ganho de realimentação robusto K_{R_i} , e o EAND ajustava os valores de $\hat{\mathbf{X}}_i$ de modo a otimizar o ganho obtido.

O simulador CARLA foi utilizado para o método de otimização e busca das matrizes de incerteza promovido pelo EAND, medindo a eficiência de todos os ganhos de realimentação (K_{R_i}) retornados para o controlador RLQR híbrido. Nesse processo, uma iteração do algoritmo EAND terminava quando o veículo contabiliza 15.000 instantes de tempo em uma simulação ou colide e fica preso por mais de 100 segundos. A função

de avaliação utilizada em cada iteração, $\hat{\mathbf{Y}}_i$, representou a soma total das recompensas obtidas durante a simulação com o ganho de realimentação definido por K_{R_i} , sobre o número máximo de dados coletados em uma iteração i . Esta métrica é representada por:

$$\hat{\mathbf{Y}}_i = \frac{\sum_{t=1}^m r_t}{15,000}, \quad (5.16)$$

onde r_t é definido na Eq. 5.1 e m é o número de instantes de tempo acumulados na iteração i .

Em sua operação, o algoritmo EAND visa otimizar a matriz $\hat{\mathbf{X}}$ para aumentar os valores de $\hat{\mathbf{Y}}$. Este procedimento é resumido no Algorithm 4.

Algorithm 4 Regulador recursivo com estimativa de incertezas por busca evolutiva

---

- 1: Inicializa a população $\hat{\mathbf{X}}$.
  - 2: While $c < c_{max}$
  - 3:   Calcula o ganho de realimentação K_R de acordo com Eq. 4.15.
  - 4:   Executa uma iteração no simulador CARLA.
  - 5:   Calcula a função de avaliação $\hat{\mathbf{Y}}$ (Eq. 5.16).
  - 6:   Otimiza $\hat{\mathbf{X}}$ usando o algoritmo EAND, como proposto em (MORAIS et al., 2019).
  - 7: $c = c + 1$
  - 8: End While
-

Como apresentado anteriormente, os modelos de controle propostos têm como objetivo manter um veículo dirigindo com velocidade constante no centro da via em ambiente simulado, sendo a entrada do sistema uma imagem de câmera RGB segmentada semanticamente e em tempo real, e uma ação de direção contínua como saída. O desempenho dos modelos foi comparado e os resultados experimentais são apresentados no próximo capítulo.

6 RESULTADOS

Neste capítulo, os parâmetros internos das arquiteturas de controle são definidas e seus desempenhos são comparados. Os três modelos de controle em estudo são representados por:

- APR: como método de aprendizado de máquinas;
- RLQ híbrido: como modelo que combina APR com controle RLQ padrão;
- RLQR híbrido: como um modelo que adiciona robustez ao RLQ híbrido.

Ambientes simulados são frequentemente explorados para treinamento e teste de veículos autônomos devido às dificuldades associadas aos experimentos em ambientes reais (SAUER; SAVINOV; GEIGER, 2018). Nesse sentido, todas as implementações foram realizadas no simulador de código aberto CARLA (DOSOVITSKIY et al., 2017), desenvolvido para pesquisas em condução autônoma de veículos, e discutido na Seção 6.1.

Conforme descrito anteriormente, os controles possuem como entrada a concatenação das três últimas imagens RGB feitas em cada instante de tempo t do ambiente em frente ao veículo. Os controladores retornam como saída uma ação de direção contínua que varia de -1 (volante totalmente esterçado para a esquerda) a 1 (volante totalmente esterçado para a direita). Para os experimentos realizados, o veículo manteve uma velocidade constante de 6 km/h. A implementação utilizada é discutida na Subseção 6.2.

Na sequência, o treinamento do modelo de APR é detalhado na Seção 6.3. A Seção 6.4 apresenta os canais de saída selecionados a partir do treinamento do modelo de APR. Na sequência, as Seção 6.5 e Seção 6.6 discutem os parâmetros de design dos modelos RLQ e RLQR híbridos, respectivamente. Finalmente, na Seção 6.7 é feita uma comparação de desempenho dos controladores apresentados.

6.1 Simulador CARLA

Os projetos de controle propostos por este trabalho foram desenvolvidos no simulador realista e de código aberto para condução autônoma em ambientes urbanos, CARLA (DOSOVITSKIY et al., 2017). O simulador se apresenta como uma ferramenta eficaz para desenvolvimento, treinamento e validação de algoritmos para veículos autônomos, incluindo ações envolvendo percepção e controle. O CARLA figura um ambiente urbano, com pedestres, outros veículos, edifícios, semáforos, placas de sinalização, etc. O simulador em operação é ilustrado na Fig. 14.

Figura 14 – Simulador de código aberto CARLA em operação.

Fonte: (DOSOVITSKIY et al., 2017).

Figura 15 – Imagem de segmentação semântica usada como entrada para os modelos propostos.

Fonte: Simulador CARLA.

O CARLA oferece sinais de sensores que podem ser usados para treinar ou definir modelos de direção autônoma, como câmera RGB, coordenadas de GPS, posição e orientação do carro, posição do centro da pista (definida pelos pontos de referência ou *waypoints*), velocímetro, detecção de colisão e outras violações de trânsito. O CARLA também fornece uma imagem RGB de segmentação semântica com 12 classes semânticas que incluem vegetação, cerca, poste, muro, prédio, estrada, marcação de faixa, calçada e outros objetos estáticos. Um exemplo de imagem 200×180 segmentada semanticamente é apresentado na Fig. 15, a qual foi utilizada para treinamento e teste do algoritmo APR e como informação de entrada para os controladores híbridos.

O simulador CARLA também fornece duas cidades diferentes, Cidade 1 e Cidade 2, onde os mapas e vistas representativas são mostrados na Fig. 16. O modelo de aprendizado de máquinas considerado neste trabalho foi treinado na Cidade 2, e os três modelos de controle foram testados em ambas as cidades.

Figura 16 – Ambientes de teste e treinamento no simulador CARLA.

Fonte: Adaptado de (CODEVILLA et al., 2018).

6.2 Implementação

Para treinar o modelo neural usando o simulador CARLA, foi utilizado o Tensorblock¹ como *framework* de desenvolvimento. O Tensorblock é uma API do Tensorflow (ABADI et al., 2016) que encapsula todas as variáveis de um gráfico em uma única classe, facilitando o acesso, modificações e adições de novos códigos ao código base. Para aplicações de APR, o Tensorblock também fornece uma interface com duas classes distintas: a primeira aplicada ao simulador e a segunda ao algoritmo de aprendizado de máquina, permitindo alterações mínimas no código-fonte principal.

Para a arquitetura de APR considerada, os hiperparâmetros do PPO foram definidos como $\epsilon = 0,12$, $\gamma = 0,99$ e $\varepsilon = 0,95$, enquanto a CNN foi configurada sem *dropout*, função de ativação ReLU, otimizador Adam e taxa de aprendizagem $\eta = 5e^{-4}$. Todos os códigos foram implementados utilizando a linguagem de programação Python 2.7, e para o treinamento foi utilizado uma NVIDIA GeForce GTX 1050 Ti com 4 GB de memória, um *batch size* de 1024. Por fim, todos os códigos foram implementados e executados em um SO Ubuntu 16.04.

¹ Proposto por V. C. Guizilini, TensorBlock - Modular API for Tensorflow, (2017) como um framework de desenvolvimento. <<https://bitbucket.org/vguizilini/tensorblock/src/master/>>.

Figura 17 – Curva de aprendizado do algoritmo de APR no processo de treinamento.

Fonte: Autor.

6.3 Treinamento do algoritmo de Aprendizagem Profunda por Reforço

O algoritmo de APR representa o primeiro procedimento de controle. Para o processo de treinamento, uma CNN inicializada com pesos aleatórios executa ações no simulador CARLA, e a condução de exploração proposta pelo algoritmo PPO coleta informações para o processo de otimização e, assim, parametrizar os pesos da CNN para aproximar a função do ambiente a um comportamento desejado.

O processo de treinamento foi dividido em episódios, sendo um episódio formado por 1.000 instantes de tempo t acumulados. Para garantir que o agente não colida e fique preso, a simulação é reiniciada caso o carro permaneça com velocidade zero por mais de 100 segundos.

O treinamento ocorreu durante 14 horas na Cidade 2, onde foram coletados 333 episódios. A Fig. 17 apresenta a curva de aprendizado com os valores de recompensa acumulados em cada episódio. O modelo neural que apresentou o episódio com maior recompensa durante todo o processo de treinamento foi considerado como o modelo de APR. Pode ser observado na Fig. 17 que o algoritmo possuiu um comportamento ascendente na curva de aprendizado, e mesmo com a presença de um *outlier*, o modelo foi eficaz para a etapa de treinamento.

6.4 Canais de saída

Após o treinamento, foram selecionados os canais de saída que melhor representavam os sinais de erro para o controlador por meio do método de comparação multiobjetivo descrito na Subseção 5.2.1. Foi obtido como resultado uma FP com 10 diferentes soluções. Foi verificado que o canal de saída 529 era o valor mais centralizado na FP, seguido pelo canal de saída 568.

Figura 18 – Valores medidos dos canais de saída selecionados durante a condução manual em três situações diferentes. Em preto é representado a trajetória do veículo seguindo no centro da via, em vermelho o veículo é deslocado para a esquerda e em azul o veículo é deslocado para a direita.

Fonte: Autor.

Na primeira análise experimental realizada, foi considerado apenas o canal de saída a_{529} . Apesar do controlador seguir uma trajetória adequada na maior parte do ambiente de treino, o modelo não foi capaz de realizar uma curva onde foram apresentadas variáveis estáticas atípicas, representadas por grades de proteção de uma ponte. Ao adicionar o segundo canal de saída mais centralizado, representado por a_{568} , este problema foi solucionado.

É importante destacar que todas as soluções apresentadas na FP podem ser usadas para inferir um sinal de erro válido para o problema de controle. Entretanto, foi escolhido experimentalmente os dois valores mais centralizados na FP, a_{529} e a_{568} , para definição do estado, uma vez que estes valores representavam o conjunto mínimo de entradas de controle capaz de realizar com eficiência a tarefa de seguir a trajetória desejada.

Os valores medidos dos dois canais de saída são mostrados na Fig. 18, onde são apresentados em preto, vermelho e azul, os valores das trajetórias simuladas quando o carro seguiu no centro da via, deslocado para a esquerda e deslocado para a direita, respectivamente. Para todas as simulações, o carro permaneceu nos limites permitidos para condução na via. Na Fig. 18, é possível observar que os valores medidos se interceptavam quando o veículo realizava uma curva, e medidas de saída foram mais ruidosas quando a imagem processou variáveis estáticas próximas à via, como árvores, cercas de proteção, etc. Com base nas medidas apresentadas na Fig. 18, é possível justificar o uso de métodos de controle robusto para referência inferidas a partir de imagens, considerando à natureza ruidosa dos valores.

6.5 Definição do controlador RLQ híbrido

A partir da seleção dos canais de saída, um estado do ambiente s_t representado por uma imagem RGB foi reduzido por operações convolucionais para apenas duas variáveis de entrada, x_{1t} e x_{2t} . A próxima etapa incluiu a identificação do sistema usando os dados mensuráveis de entrada e saída. Para essa tarefa, foram utilizados comandos de direção manuais e foram coletadas informações de mudança de estado associados às ações, totalizando 5.000 dados experimentais. Na sequência, foi aplicado o método de identificação de sistemas resumido na Subseção 5.2.2, as matrizes F e G do sistema foram identificadas como:

$$F = \begin{bmatrix} 1,0054 & 0,0089 \\ 0,0058 & 0,9985 \end{bmatrix}, \quad G = \begin{bmatrix} -0,0361 \\ -0,0374 \end{bmatrix}.$$

Com a definição das matrizes F e G e as entradas de controle, foi possível formalizar uma arquitetura de controle que mescla uma CNN treinada e um RLQ padrão, em que o controlador regula o erro medido entre $[x_{1t} \ x_{2t}]^T$ e o valor de referência resultante de operações convolucionais. Por fim, para calcular a estrutura do controlador RLQ, as matrizes de ponderação Q e R foram definidas como:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad R = \begin{bmatrix} 1 \end{bmatrix},$$

E, finalmente, foi possível obter o ganho final do controlador:

$$K = \begin{bmatrix} -1,1143 & -0,8640 \end{bmatrix}.$$

6.6 Definição do controlador RLQR híbrido

O terceiro design de controle representa uma adaptação do modelo anterior e é apresentado como um controlador RLQR híbrido com estimativa das matrizes de incerteza por otimização evolutiva. A dinâmica do sistema obtido na seção anterior foi preservada e, seguindo o procedimento de controle apresentado no Capítulo 4, as variáveis de ponderação do controlador RLQR associadas a robustez foram consideradas como:

$$H = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad P_{N+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$\mu = 10^8$ e $N = 100$.

Para a obtenção das matrizes de incerteza, foi utilizado o algoritmo EAND, onde cada indivíduo da população representava uma combinação das matrizes E_F e E_G , como

descrito na Seção 5.3. Os procedimentos de mutação e seleção ocorreram durante 2.000 avaliações e o melhor conjunto de incertezas foi apresentado como:

$$E_F = \begin{bmatrix} -0,7112 & -0,8118 \end{bmatrix}, \quad E_G = \begin{bmatrix} -0,4554 \end{bmatrix}.$$

Com estes parâmetros foi possível calcular o ganho de realimentação do controlador RLQR híbrido como:

$$K_R = \begin{bmatrix} 0,8921 & 0,9039 \end{bmatrix}.$$

6.7 Comparação dos modelos

Para comparação de desempenho, os três modelos de controle foram executados no simulador CARLA em dois ambientes diferentes (Cidade 1 e Cidade 2), onde apenas a Cidade 2 foi utilizada nas etapas de treinamento. Cada simulação foi executada durante 100.000 instantes de tempo (totalizando 100 episódios) em cada cidade.

A avaliação de desempenho é mostrada na Tabela 1, onde são apresentados o número total de colisões durante a simulação, e a média e o desvio padrão (σ) do total de recompensas (apresentada na Eq. 5.1) acumuladas em cada episódio. É desejado que os modelos obtenham o menor número de ocorrência de colisões, o maior valor médio de recompensas e o menor σ , visto que σ pequeno representa menos desvios do centro da via durante a execução da simulação.

Tabela 1 – Desempenho dos três modelos de controle na Cidade 1 e na Cidade 2. É apresentado o número de colisões com variáveis estáticas, o valor médio da recompensa (Média) e o desvio padrão (σ) ao longo das simulações.

Ambiente	Modelo	Colisões	Média	σ
Cidade 2	APR	92	87853.85	3344.03
	RLQ híbrido	8	89602.61	5134.71
	RLQR híbrido	0	90460.82	1302.57
Cidade 1	APR	-	-	-
	RLQ híbrido	10	82102.53	3638.20
	RLQR híbrido	0	85867.64	1666.60

A partir da Tabela 1, pode-se concluir que o algoritmo de APR teve o pior desempenho, com número significativo de colisões e menor valor médio de recompensa na Cidade 2 (ambiente de treino). Ademais, este modelo mostrou um problema de generalização da rede neural, não sendo capaz de interpretar um novo ambiente apresentado na Cidade 1.

Como argumentado por (DOSOVITSKIY et al., 2017), a APR pode ter um desempenho inferior devido à inviabilidade de realizar uma busca extensiva por hiperparâmetros em simuladores realistas. Os algoritmos de APR também possuem uma característica sensível a uma grande quantidade de variáveis (HENDERSON et al., 2018), o que pode tornar esse modelo inviável para conduzir aprendizagem autônoma em ambientes urbanos. Um outro fator considerável, é que o tempo de treinamento do modelo de APR implementado foi apenas de 14 horas, em contraste com trabalhos anteriores que levaram 12 dias (DOSOVITSKIY et al., 2017). Este processo de treinamento durante um curto período de tempo também pode justificar a não generalização do modelo neural.

O controle RLQ híbrido mostrou uma melhoria significativa no desempenho em comparação com a APR, sendo capaz de executar ações em ambientes de treinamento e teste, não mostrou problemas de generalização e obteve maiores recompensas e menos colisões, mesmo utilizando uma CNN treinada com uma pequena quantidade de dados. No entanto, o aumento no valor do σ indica que o controlador RLQ híbrido oscila mais em torno dos valores de referência.

Por outro lado, a robustez fornecida pelo controlador RLQR híbrido garantiu estabilidade e manteve o veículo sempre no centro da via, com erros de trajetória significativamente pequenos e, como pode ser visto na Tabela 1, apresentou maiores recompensas acumuladas, menores valores de σ e prevenção total de colisões.

Na sequência, para validar a eficiência do modelo híbrido robusto proposto, uma análise de variância unidirecional (ANOVA) para o procedimento de comparação múltipla foi implementada usando o critério de diferença honestamente significativa de Tukey, com base na distribuição da faixa de Studentized (nível de confiança de 95%). Conforme mostrado em Fig. 19, é evidente que os resultados obtidos pelo controlador RLQR híbrido são significativamente diferentes daqueles alcançados pelos modelos RLQ híbrido e APR.

Os três modelos propostos também foram avaliados em termos de segmento de trajetória após a aplicação de uma perturbação externa, que consistia em esterçar o volante 30 graus para a esquerda durante um intervalo de tempo definido. O primeiro teste, mostrado na Fig. 20, é realizado na Cidade, 2 onde a perturbação é aplicada na posição $x = 24$ m. É possível observar que no ambiente de treinamento, os controladores RLQ híbrido e o RLQR híbrido tiveram desempenhos semelhantes, enquanto o APR teve o tempo de resposta mais lento e as oscilações mais altas em torno da referência.

Na sequência, a mesma perturbação foi aplicada na Cidade 1 na posição $x = 28$ m, como mostrado em Fig. 21. Neste caso, o modelo RLQR híbrido teve uma resposta mais rápida à correção da perturbação e manteve o carro mais próximo do valor de referência durante o percurso.

Por fim, Fig. 22 e Fig. 23 mostram a evolução do comando de direção (u_t) e taxa

Figura 19 – O critério de diferença honestamente significativa de Tukey, com base na distribuição da faixa Studentizada (nível de confiança de 95%), nos resultados.

(a) Recompensa média sobre a Cidade 2 (ambiente de treinamento). (b) Recompensa média sobre a Cidade 1 (ambiente de teste).

Fonte: Autor.

Figura 20 – Seguimento de trajetória com aplicação de distúrbio externo na posição $x = 24\text{m}$ na Cidade 2.

Fonte: Autor.

Figura 21 – Seguimento de trajetória com aplicação de distúrbio externo na posição $x = 28\text{m}$ na Cidade 1.

Fonte: Autor.

Figura 22 – Ação de esterçamento e taxa de esterçamento no ambiente de treino.

- (a) Esterçamento do volante no ambiente de treino. (b) Taxa de esterçamento do volante no ambiente de treino.

Fonte: Autor.

de variação de esterçamento do modelo DRL + EAND + RLQR nos ambientes de treino e teste, respectivamente. Como apresentado na figuras, a abordagem proposta é adequada em uma operação típica de condução de veículo.

Figura 23 – Ação de esterçamento e taxa de esterçamento no ambiente de teste.

- (a) Esterçamento do volante no ambiente de teste. (b) Taxa de esterçamento do volante no ambiente de teste.

Fonte: Autor.

7 CONCLUSÃO

Carros autônomos necessitam de um processamento de informações complexo para atuarem com segurança em um ambiente urbano. Mesmo considerando que o veículo esteja instrumentado com câmeras e lasers, ele deve ser capaz de interagir com dados incertos e não lineares, como intenção de pedestres e outros motoristas, e processamento de ruídos. Também é necessário associar essas variáveis a ações cotidianas de condução, como a conversão em um cruzamento, atravessar uma rotatória, realizar uma ultrapassagem ou entrar em uma via expressa. Para que essas tarefas sejam realizadas com eficiência e segurança, se faz necessário um modelo autônomo capaz de realizar o controle lateral do veículo referenciado em visão computacional.

Entretanto, para tornar essa tecnologia segura e viável economicamente, é necessário expandir essa área de pesquisa, propor novos algoritmos capazes de relacionar diferentes técnicas matemáticas e explorar ambientes em simulações realistas. Portanto, este trabalho propôs modelos híbridos que combina técnicas de aprendizagem de máquinas com um controlador recursivo robusto, aplicado para controle lateral baseado em visão computacional. Na abordagem apresentada, um modelo de APR resultante da combinação do algoritmo PPO e uma CNN, treina o modelo neural de modo a destacar as características da imagem que têm maior influência em manter o veículo no centro da via.

Na sequência, foi realizada uma comparação multiobjetivo para selecionar as informações extraídas dos canais de saída da CNN. Estas medidas foram interpretadas como sinais de erro de deslocamento lateral a serem minimizados por malhas de controle. Desta forma, foram implementados três processos de controle lateral baseados em visão, representados por um algoritmo de APR, um modelo RLQ híbrido e um modelo RLQR híbrido. Para comparar o desempenho dos três designs, experimentos foram conduzidos no simulador CARLA.

Foi possível concluir que a presença de incertezas do modelo degradou significativamente o desempenho geral do sistema, considerando que o modelo de APR não foi capaz de estabilizar o veículo em ambiente de teste e o modelo RLQ híbrido apresentou elevadas oscilações em torno do centro da via. No entanto, a abordagem RLQR híbrida proposta apresentou oscilações muito menores em torno do caminho de referência e todas as colisões desapareceram durante o experimento, mesmo na presença de incertezas do modelo, o que valida o potencial de um controlador robusto desenvolvido a partir de uma rede neural.

Para trabalhos futuros, a influência das técnicas de aprendizado de máquina no modelo RLQR híbrido será explorada, onde, além dos resultados encontrados neste trabalho, o modelo neural definirá os canais de saída e os parâmetros de controle ideais para o

agente autônomo. Ademais, aplicações envolvendo a topologia do RLQR híbrido baseada em visão computacional poderá ser investigada em outros tipos de veículos terrestres e aéreos não tripulados.

REFERÊNCIAS

- ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M. et al. Tensorflow: A system for large-scale machine learning. In: **12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)**. [s.n.], 2016. p. 265–283. Disponível em: <<https://research.google/pubs/pub45381/>>.
- ABBEEL, P.; COATES, A.; QUIGLEY, M.; NG, A. Y. An application of reinforcement learning to aerobatic helicopter flight. In: **Advances in neural information processing systems**. [s.n.], 2007. p. 1–8. Disponível em: <<http://papers.nips.cc/paper/3151-an-application-of-reinforcement-learning-to-aerobatic-helicopter-flight.pdf>>.
- ABU-JBARA, K.; ALHEADARY, W.; SUNDARAMORTHY, G.; CLAUDEL, C. A robust vision-based runway detection and tracking algorithm for automatic uav landing. In: IEEE. **2015 International conference on unmanned aircraft systems (ICUAS)**. 2015. p. 1148–1157. Disponível em: <<https://ieeexplore.ieee.org/document/7152407/keywords#keywords>>.
- ARULKUMARAN, K.; DEISENROTH, M. P.; BRUNDAGE, M.; BHARATH, A. A. A brief survey of deep reinforcement learning. **arXiv preprint arXiv:1708.05866**, 2017. Disponível em: <<https://arxiv.org/abs/1708.05866>>.
- ASSA, A.; JANABI-SHARIFI, F. A robust vision-based sensor fusion approach for real-time pose estimation. **IEEE transactions on cybernetics**, IEEE, v. 44, n. 2, p. 217–227, 2013. ISSN 2168-2275. Disponível em: <<https://ieeexplore.ieee.org/document/6506990>>.
- BANSAL, M.; DAS, A.; KREUTZER, G.; ELEDATH, J.; KUMAR, R.; SAWHNEY, H. Vision-based perception for autonomous urban navigation. In: IEEE. **2008 11th International IEEE Conference on Intelligent Transportation Systems**. 2008. p. 434–440. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/4732699>>.
- BARBOSA, F. M.; MARCOS, L. B.; SILVA, M. M. da; TERRA, M. H.; JUNIOR, V. G. Robust path-following control for articulated heavy-duty vehicles. **Control Engineering Practice**, Elsevier, v. 85, p. 246–256, 2019. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0967066118303216>>.
- BELLMAN, R. A markovian decision process. **Journal of Mathematics and Mechanics**, JSTOR, p. 679–684, 1957. Disponível em: <<https://www.jstor.org/stable/24900506?seq=1>>.
- BERTOZZI, M.; BROGGI, A.; FASCIOLI, A. Vision-based intelligent vehicles: State of the art and perspectives. **Robotics and Autonomous systems**, Elsevier, v. 32, n. 1, p. 1–16, 2000. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0921889099001256>>.
- BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: **Proceedings of COMPSTAT'2010**. Springer, 2010. p. 177–186. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-7908-2604-3_16>.

BROGGI, A.; CERRI, P.; DEBATTISTI, S.; LAGHI, M. C.; MEDICI, P.; PANCIOLOI, M.; PRIOLETTI, A. Proud-public road urban driverless test: Architecture and results. In: IEEE. **2014 IEEE Intelligent Vehicles Symposium Proceedings**. 2014. p. 648–654. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/6856478>>.

BROGGI, A.; CERRI, P.; FELISA, M.; LAGHI, M. C.; MAZZEI, L.; PORTA, P. P. The vislab intercontinental autonomous challenge: an extensive test for a platoon of intelligent vehicles. **International Journal of Vehicle Autonomous Systems**, Inderscience Publishers Ltd., PO Box 735 Olney Bucks MK 46 5 WB United Kingdom, v. 10, n. 3, p. 147–164, 2012.

BUEHLER, M.; IAGNEMMA, K.; SINGH, S. **The 2005 DARPA grand challenge: the great robot race**. [S.l.]: Springer, 2007. v. 36.

_____. **The DARPA urban challenge: autonomous vehicles in city traffic**. [S.l.]: springer, 2009. v. 56.

CASTRO, L. N.; CASTRO, L. N. D.; TIMMIS, J. **Artificial immune systems: a new computational intelligence approach**. Springer Science & Business Media, 2002. Disponível em: <<https://www.springer.com/gp/book/9781852335946>>.

CASTRO, L. N. de; ZUBEN, F. J. V. Immune and neural network models: theoretical and empirical comparisons. **International Journal of Computational Intelligence and Applications**, World Scientific, v. 1, n. 03, p. 239–257, 2001. Disponível em: <<https://www.worldscientific.com/doi/abs/10.1142/S1469026801000238>>.

CERRI, J. P.; TERRA, M. H.; ISHIHARA, J. Y. Recursive robust regulator for discrete-time state-space systems. In: IEEE. **2009 American Control Conference**. 2009. p. 3077–3082. Disponível em: <<https://ieeexplore.ieee.org/document/5160553>>.

CHEN, G.; PENG, Y.; ZHANG, M. An adaptive clipping approach for proximal policy optimization. **arXiv preprint arXiv:1804.06461**, 2018. Disponível em: <<https://arxiv.org/abs/1804.06461>>.

CHEN, J.; ZHAO, P.; LIANG, H.; MEI, T. A multiple attribute-based decision making model for autonomous vehicle in urban environment. In: IEEE. **2014 IEEE Intelligent Vehicles Symposium Proceedings**. 2014. p. 480–485. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/6856470>>.

Chen, T.; Chen, Z.; Shi, Q.; Huang, X. Road marking detection and classification using machine learning algorithms. In: **2015 IEEE Intelligent Vehicles Symposium (IV)**. [s.n.], 2015. p. 617–621. ISSN 1931-0587. Disponível em: <<https://ieeexplore.ieee.org/document/7225753>>.

CHEN, X.-M.; MIAO, Y.-S. Driving decision-making analysis of car-following for autonomous vehicle under complex urban environment. In: IEEE. **2016 9th International Symposium on Computational Intelligence and Design (ISCID)**. 2016. v. 1, p. 315–319. Disponível em: <<https://ieeexplore.ieee.org/document/7830353>>.

CODEVILLA, F.; MILLER, M.; LÓPEZ, A.; KOLTUN, V.; DOSOVITSKIY, A. End-to-end driving via conditional imitation learning. In: IEEE. **2018 IEEE International Conference on Robotics and Automation (ICRA)**. 2018. p. 1–9. Disponível em: <<https://ieeexplore.ieee.org/document/8460487>>.

CODEVILLA, F.; MÜLLER, M.; DOSOVITSKIY, A.; LÓPEZ, A.; KOLTUN, V. End-to-end driving via conditional imitation learning. **arXiv preprint arXiv:1710.02410**, 2017. Disponível em: <<https://arxiv.org/abs/1710.02410>>.

DEMUTH, H.; BEALE, M.; HAGAN, M. Neural network toolbox™ 6. **User's guide**, v. 37, 2008. Disponível em: <http://cda.psych.uiuc.edu/matlab_pdf/nnet.pdf>.

DING, X. C.; PINTO, A.; SURANA, A. Strategic planning under uncertainties via constrained markov decision processes. In: IEEE. **2013 IEEE International Conference on Robotics and Automation**. 2013. p. 4568–4575. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/6631226>>.

DOSOVITSKIY, A.; ROS, G.; CODEVILLA, F.; LOPEZ, A.; KOLTUN, V. CARLA: An open urban driving simulator. In: **Proceedings of the 1st Annual Conference on Robot Learning**. [s.n.], 2017. p. 1–16. Disponível em: <<http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf>>.

ENGLUND, C.; CHEN, L.; PLOEG, J.; SEMSAR-KAZEROONI, E.; VORONOV, A.; BENGTSSON, H. H.; DIDOFF, J. The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. **IEEE Wireless Communications**, IEEE, v. 23, n. 4, p. 146–152, 2016. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/7553038>>.

FAGNANT, D. J.; KOCKELMAN, K. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. **Transportation Research Part A: Policy and Practice**, v. 77, p. 167 – 181, 2015. ISSN 0965-8564. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0965856415000804>>.

FOLKERS, A.; RICK, M.; BÜSKENS, C. Controlling an autonomous vehicle with deep reinforcement learning. In: IEEE. **2019 IEEE Intelligent Vehicles Symposium (IV)**. 2019. p. 2025–2031. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/8814124>>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. MIT press, 2016. Disponível em: <<http://www.deeplearningbook.org>>.

GU, J.; WANG, Z.; KUEN, J.; MA, L.; SHAHROUDY, A.; SHUAI, B.; LIU, T.; WANG, X.; WANG, G.; CAI, J. et al. Recent advances in convolutional neural networks. **Pattern Recognition**, Elsevier, v. 77, p. 354–377, 2018. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0031320317304120>>.

GUO, C.; KIDONO, K.; TERASHIMA, R.; KOJIMA, Y. Toward human-like behavior generation in urban environment based on markov decision process with hybrid potential maps. In: IEEE. **2018 IEEE Intelligent Vehicles Symposium (IV)**. 2018. p. 2209–2215. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/8500439>>.

HAYKIN, S. S.; HAYKIN, S. S.; HAYKIN, S. S.; ELEKTROINGENIEUR, K.; HAYKIN, S. S. **Neural networks and learning machines**. Pearson Upper Saddle River, 2009. v. 3. Disponível em: <<https://www.pearson.com/us/higher-education/program/Haykin-Neural-Networks-and-Learning-Machines-3rd-Edition/PGM320370.html>>.

HENDERSON, P.; ISLAM, R.; BACHMAN, P.; PINEAU, J.; PRECUP, D.; MEGER, D. Deep reinforcement learning that matters. In: **Thirty-Second AAAI Conference on Artificial Intelligence**. [s.n.], 2018. Disponível em: <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669>.

HUSSEIN, A.; ELYAN, E.; GABER, M. M.; JAYNE, C. Deep imitation learning for 3d navigation tasks. **Neural computing and applications**, Springer, v. 29, n. 7, p. 389–404, 2018. Disponível em: <https://link.springer.com/article/10.1007/s00521-017-3241-z>.

ISELE, D.; RAHIMI, R.; COSGUN, A.; SUBRAMANIAN, K.; FUJIMURA, K. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In: IEEE. **2018 IEEE International Conference on Robotics and Automation (ICRA)**. 2018. p. 2034–2039. Disponível em: <https://ieeexplore.ieee.org/document/8461233/keywords#keywords>.

_____. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In: IEEE. **2018 IEEE International Conference on Robotics and Automation (ICRA)**. 2018. p. 2034–2039. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8461233>.

JAAFRA, Y.; LAURENT, J. L.; DERUYVER, A.; NACEUR, M. S. Robust reinforcement learning for autonomous driving. 2019. Disponível em: <https://openreview.net/forum?id=Bke03G85DN>.

JAIN, R.; KASTURI, R.; SCHUNCK, B. G. **Machine vision**. [S.l.]: McGraw-Hill New York, 1995. v. 5.

JI, X.; HE, X.; LV, C.; LIU, Y.; WU, J. Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits. **Control Engineering Practice**, Elsevier, v. 76, p. 41–53, 2018. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0967066118300881>.

KAILATH ALI H. SAYED, B. H. T. **Linear estimation**. 1. ed. [S.l.]: Prentice Hall, 2000. (Prentice Hall information and system sciences series). ISBN 0130224642,9780130224644.

Kim, Y.; Kum, D. Deep learning based vehicle position and orientation estimation via inverse perspective mapping image. In: **2019 IEEE Intelligent Vehicles Symposium (IV)**. [s.n.], 2019. p. 317–323. ISSN 1931-0587. Disponível em: <https://ieeexplore.ieee.org/document/8814050>.

KOBER, J.; BAGNELL, J. A.; PETERS, J. Reinforcement learning in robotics: A survey. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 32, n. 11, p. 1238–1274, 2013. Disponível em: <https://journals.sagepub.com/doi/abs/10.1177/0278364913495721>.

Kottath, R.; Yalamandala, D. P.; Poddar, S.; Bhondekar, A. P.; Karar, V. Inertia constrained visual odometry for navigational applications. In: **2017 Fourth International Conference on Image Information Processing (ICIIP)**. [s.n.], 2017. p. 1–4. ISSN null. Disponível em: <https://ieeexplore.ieee.org/document/8313714/keywords#keywords>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [s.n.], 2012. p. 1097–1105. Disponível em: <<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015. Disponível em: <<https://www.nature.com/articles/nature14539>>.

LEUNG, H.; HAYKIN, S. The complex backpropagation algorithm. **IEEE Transactions on signal processing**, IEEE, v. 39, n. 9, p. 2101–2104, 1991. Disponível em: <<https://ieeexplore.ieee.org/document/134446>>.

LEVINSON, J.; ASKELAND, J.; BECKER, J.; DOLSON, J.; HELD, D.; KAMMEL, S.; KOLTER, J. Z.; LANGER, D.; PINK, O.; PRATT, V. et al. Towards fully autonomous driving: Systems and algorithms. In: IEEE. **2011 IEEE Intelligent Vehicles Symposium (IV)**. 2011. p. 163–168. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/5940562>>.

LEWIS, F. L.; VAMVOUDAKIS, K. G. Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 41, n. 1, p. 14–25, 2010. Disponível em: <<https://ieeexplore.ieee.org/document/5439950>>.

LILLICRAP, T. P.; HUNT, J. J.; PRITZEL, A.; HEESS, N.; EREZ, T.; TASSA, Y.; SILVER, D.; WIERSTRA, D. Continuous control with deep reinforcement learning. **arXiv preprint arXiv:1509.02971**, 2015. Disponível em: <<https://arxiv.org/abs/1509.02971>>.

LIMA, D. A. de; VICTORINO, A. C. An image based dynamic window approach for local navigation of an autonomous vehicle in urban environments. In: . [s.n.], 2014. Disponível em: <<https://hal.archives-ouvertes.fr/hal-01088389/>>.

_____. A hybrid controller for vision-based navigation of autonomous vehicles in urban environments. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 17, n. 8, p. 2310–2323, 2016. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/7442130>>.

LOPES, A. T.; AGUIAR, E. de; SOUZA, A. F. D.; OLIVEIRA-SANTOS, T. Facial expression recognition with convolutional neural networks: coping with few data and the training sample order. **Pattern Recognition**, Elsevier, v. 61, p. 610–628, 2017. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0031320316301753>>.

MAMMERI, A.; LU, G.; BOUKERCHE, A. Design of lane keeping assist system for autonomous vehicles. In: IEEE. **2015 7th International Conference on New Technologies, Mobility and Security (NTMS)**. 2015. p. 1–5. Disponível em: <<https://ieeexplore.ieee.org/document/7266483>>.

MARINA, L.; SANDU, A. Deep reinforcement learning for autonomous vehicles-state of the art. **Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I**, Transilvania University of Brasov, v. 10, n. 2, p. 195–202, 2017. Disponível em: <<https://pdfs.semanticscholar.org/bd0e/9e5833e9de7a315a0ee085288fb34853634c.pdf>>.

MCALLISTER, R.; GAL, Y.; KENDALL, A.; WILK, M. V. D.; SHAH, A.; CIPOLLA, R.; WELLER, A. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In: INTERNATIONAL JOINT CONFERENCES ON ARTIFICIAL INTELLIGENCE, INC. 2017. Disponível em: <<https://www.repository.cam.ac.uk/handle/1810/266683>>.

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. In: **International conference on machine learning**. [s.n.], 2016. p. 1928–1937. Disponível em: <<http://proceedings.mlr.press/v48/mniha16.pdf>>.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. Playing atari with deep reinforcement learning. **arXiv preprint arXiv:1312.5602**, 2013. Disponível em: <<https://arxiv.org/abs/1312.5602>>.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G. et al. Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, v. 518, n. 7540, p. 529, 2015. Disponível em: <<https://www.nature.com/articles/nature14236>>.

MOE, S.; RUSTAD, A. M.; HANSSEN, K. G. Machine learning in control systems: An overview of the state of the art. In: SPRINGER. **International Conference on Innovative Techniques and Applications of Artificial Intelligence**. 2018. p. 250–265. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-04191-5_23>.

MONAHAN, G. E. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. **Management Science**, INFORMS, v. 28, n. 1, p. 1–16, 1982. Disponível em: <<https://pubsonline.informs.org/doi/10.1287/mnsc.28.1.1>>.

MORAIS, G. A. de; BARBOSA, B. H.; FERREIRA, D. D.; PAIVA, L. S. Soft sensors design in a petrochemical process using an evolutionary algorithm. **Measurement**, Elsevier, v. 148, p. 106920, 2019. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0263224119307778>>.

NETWORK, E. **Programme for a European traffic system with highest efficiency and unprecedented safety**. [S.l.]: PROMETHEUS), Brussels, Belgium. <http://www.eurekanetwork.org/project/-/id...>, 2013.

NG, A. Y.; COATES, A.; DIEL, M.; GANAPATHI, V.; SCHULTE, J.; TSE, B.; BERGER, E.; LIANG, E. Autonomous inverted helicopter flight via reinforcement learning. In: **Experimental robotics IX**. Springer, 2006. p. 363–372. Disponível em: <https://link.springer.com/chapter/10.1007/11552246_35>.

NGUYEN, D. T.; LI, W.; OGUNBONA, P. O. Human detection from images and videos: A survey. **Pattern Recognition**, Elsevier, v. 51, p. 148–175, 2016. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0031320315003179>>.

NOGUEIRA, K.; PENATTI, O. A.; SANTOS, J. A. D. Towards better exploiting convolutional neural networks for remote sensing scene classification. **Pattern Recognition**, Elsevier, v. 61, p. 539–556, 2017. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0031320316301509>>.

OLIVARES-MENDEZ, M. A.; SANCHEZ-LOPEZ, J. L.; JIMENEZ, F.; CAMPOY, P.; SAJADI-ALAMDARI, S. A.; VOOS, H. Vision-based steering control, speed assistance and localization for inner-city vehicles. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 3, p. 362, 2016. Disponível em: <<https://www.mdpi.com/1424-8220/16/3/362>>.

PADEN, B.; ČÁP, M.; YONG, S. Z.; YERSHOV, D.; FRAZZOLI, E. A survey of motion planning and control techniques for self-driving urban vehicles. **IEEE Transactions on Intelligent Vehicles**, IEEE, v. 1, n. 1, p. 33–55, 2016. Disponível em: <<https://ieeexplore.ieee.org/document/7490340>>.

PEDERSOLI, M.; VEDALDI, A.; GONZALEZ, J.; ROCA, X. A coarse-to-fine approach for fast deformable object detection. **Pattern Recognition**, Elsevier, v. 48, n. 5, p. 1844–1853, 2015. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0031320314004567>>.

Reyes, S.; Tokunaga, O. F.; Espinoza, E. S.; Salazar, S.; Romero, H.; Lozano, R. Autonomous ground vehicle navigation using a novel visual positioning system. In: **2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)**. [s.n.], 2018. p. 342–347. ISSN 2372-1618. Disponível em: <<https://ieeexplore.ieee.org/document/8540752>>.

RIZVI, S. A. A.; LIN, Z. Reinforcement learning-based linear quadratic regulation of continuous-time systems using dynamic output feedback. **IEEE transactions on cybernetics**, IEEE, 2019. Disponível em: <<https://ieeexplore.ieee.org/document/8600378>>.

SALLAB, A. E.; ABDOL, M.; PEROT, E.; YOGAMANI, S. Deep reinforcement learning framework for autonomous driving. **Electronic Imaging**, Society for Imaging Science and Technology, v. 2017, n. 19, p. 70–76, 2017. Disponível em: <<https://www.ingentaconnect.com/content/ist/ei/2017/00002017/00000019/art00012>>.

SAUER, A.; SAVINOV, N.; GEIGER, A. Conditional affordance learning for driving in urban environments. **arXiv preprint arXiv:1806.06498**, 2018. Disponível em: <<https://arxiv.org/abs/1806.06498>>.

SAYED, A. H.; NASCIMENTO, V. H. Design criteria for uncertain models with structured and unstructured uncertainties. In: GARULLI, A.; TESI, A. (Ed.). **Robustness in identification and control**. London: Springer London, 1999. p. 159–173. ISBN 978-1-84628-538-7. Disponível em: <<https://link.springer.com/chapter/10.1007/BFb0109867>>.

SCHULMAN, J.; LEVINE, S.; ABBEEL, P.; JORDAN, M.; MORITZ, P. Trust region policy optimization. In: **International conference on machine learning**. [s.n.], 2015. p. 1889–1897. Disponível em: <<http://proceedings.mlr.press/v37/schulman15.html>>.

SCHULMAN, J.; MORITZ, P.; LEVINE, S.; JORDAN, M.; ABBEEL, P. High-dimensional continuous control using generalized advantage estimation. **arXiv preprint arXiv:1506.02438**, 2015. Disponível em: <<https://arxiv.org/abs/1506.02438>>.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. **arXiv preprint arXiv:1707.06347**, 2017. Disponível em: <<https://arxiv.org/abs/1707.06347>>.

Shobha, B. S.; Deepu, R. A review on video based vehicle detection, recognition and tracking. In: **2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)**. [s.n.], 2018. p. 183–186. ISSN null. Disponível em: <https://ieeexplore.ieee.org/document/8768743>.

SOTELO, M.; RODRIGUEZ, F.; MAGDALENA, L. Vision-based navigation system for autonomous urban transport vehicles in outdoor environments. In: **IEEE. Intelligent Vehicle Symposium, 2002. IEEE**. 2002. v. 1, p. 52–57. Disponível em: <https://ieeexplore.ieee.org/document/1187927>.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Disponível em: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>.

TERRA, M. H.; CERRI, J. P.; ISHIHARA, J. Y. Optimal robust linear quadratic regulator for systems subject to uncertainties. **Transactions On Automatic Control**, IEEE, set. 2014. Disponível em: <https://ieeexplore.ieee.org/document/6754186>.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G. et al. Stanley: The robot that won the darpa grand challenge. **Journal of field Robotics**, Wiley Online Library, v. 23, n. 9, p. 661–692, 2006. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20147>.

ULMER, B. Vita ii-active collision avoidance in real traffic. In: **IEEE. Proceedings of the Intelligent Vehicles' 94 Symposium**. 1994. p. 1–6. Disponível em: <https://sci-hub.im/https://ieeexplore.ieee.org/abstract/document/639460/>.

URMSON, C.; ANHALT, J.; BAGNELL, D.; BAKER, C.; BITTNER, R.; CLARK, M.; DOLAN, J.; DUGGINS, D.; GALATALI, T.; GEYER, C. et al. Autonomous driving in urban environments: Boss and the urban challenge. **Journal of Field Robotics**, Wiley Online Library, v. 25, n. 8, p. 425–466, 2008. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20255>.

VITOR, G. B.; LIMA, D. A.; VICTORINO, A. C.; FERREIRA, J. V. A 2d/3d vision based approach applied to road detection in urban environments. In: **IEEE. 2013 IEEE Intelligent Vehicles Symposium (IV)**. 2013. p. 952–957. Disponível em: <https://ieeexplore.ieee.org/abstract/document/6629589>.

Wang, J.; Zhou, L. Traffic light recognition with high dynamic range imaging and deep learning. **IEEE Transactions on Intelligent Transportation Systems**, v. 20, n. 4, p. 1341–1352, April 2019. ISSN 1558-0016. Disponível em: <https://ieeexplore.ieee.org/document/8419782>.

WANG, Y.; VELSWAMY, K.; HUANG, B. A novel approach to feedback control with deep reinforcement learning. **IFAC-PapersOnLine**, Elsevier, v. 51, n. 18, p. 31–36, 2018. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405896318319177>.

WEI, J.; SNIDER, J. M.; KIM, J.; DOLAN, J. M.; RAJKUMAR, R.; LITKOUHL, B. Towards a viable autonomous driving research platform. In: **IEEE. 2013**

IEEE Intelligent Vehicles Symposium (IV). 2013. p. 763–770. Disponível em: <https://ieeexplore.ieee.org/abstract/document/6629559>.

Xuehong Mao; Hijazi, S.; Casas, R.; Kaul, P.; Kumar, R.; Rowen, C. Hierarchical cnn for traffic sign recognition. In: **2016 IEEE Intelligent Vehicles Symposium (IV)**. [s.n.], 2016. p. 130–135. ISSN null. Disponível em: <https://ieeexplore.ieee.org/document/7535376>.

YOUNG, P.; GARNIER, H.; GILSON, M. Identification of continuous-time models from sampled data. **Refined instrumental variable identification of continuous-time hybrid Box-Jenkins models**. H. Garnier and L. Wang (editor), p. 91–131, 2008. Disponível em: <https://www.springer.com/gp/book/9781848001602>.

YURTSEVER, E.; LAMBERT, J.; CARBALLO, A.; TAKEDA, K. A survey of autonomous driving: Common practices and emerging technologies. **IEEE Access**, IEEE, v. 8, p. 58443–58469, 2020. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9046805>.

ZHANG, S.; WANG, J.; WANG, Z.; GONG, Y.; LIU, Y. Multi-target tracking by learning local-to-global trajectory models. **Pattern Recognition**, Elsevier, v. 48, n. 2, p. 580–590, 2015. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0031320314003276>.

ZHANG, S.; YAO, H.; SUN, X.; LU, X. Sparse coding based visual tracking: Review and experimental comparison. **Pattern Recognition**, Elsevier, v. 46, n. 7, p. 1772–1788, 2013. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0031320312004396>.

ZHANG, T.; KAHN, G.; LEVINE, S.; ABBEEL, P. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In: IEEE. **2016 IEEE international conference on robotics and automation (ICRA)**. 2016. p. 528–535. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7487175>.

ZHU, Y.; MOTTAGHI, R.; KOLVE, E.; LIM, J. J.; GUPTA, A.; FEI-FEI, L.; FARHADI, A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: IEEE. **2017 IEEE international conference on robotics and automation (ICRA)**. 2017. p. 3357–3364. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7989381>.

ZWILLINGER, D. **CRC Standard Mathematical Tables and Formulae**. 31. ed. Chapman & Hall/CRC, 2003. Disponível em: <https://www.crcpress.com/CRC-Standard-Mathematical-Tables-and-Formulas/Zwillinger/p/book/9781498777803>.