INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS

MOVING FORWARD TO A TRUE AUTONOMOUS DRIVING: AVOIDING FATAL CRASHES USING METAMORPHIC TESTING

MAILSON DENNIS TRAJANO DE SOUZA

Cajazeiras
fevereiro de 2020

MAILSON DENNIS TRAJANO DE SOUZA

MOVING FORWARD TO A TRUE AUTONOMOUS DRIVING: AVOIDING FATAL CRASHES USING METAMORPHIC TESTING

Trabalho de Conclusão de Curso apresentado junto ao programa de **Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas** do **Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras**, como requisito parcial à obtenção do título de **Tecnólogo em Análise e Desenvolvimento de Sistemas**.

Orientador:

Prof. MSc. Diogo Dantas Moreira.

Cajazeiras
fevereiro de 2020

IFPB
Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Daniel Andrade CRB-15/593

S729m

Souza, Mailson Dennis Trajano de

Moving forward to a true autonomous driving: avoiding fatal crashes using metamorphic testingo / Mailson Dennis Trajano de Souza; orientador Diogo Dantas Moreira.- 2020.
61 f.: il.

Orientador: Diogo Dantas Moreira.
TCC (Tecnólogo em Análise e Desenvolvimento de Sistemas.) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2020.

1. Carros autônomos 2. Testes metamórficos 3. Testes de software I. Título

004.681.5 (0.067)

SERVIÇO PÚBLICO FEDERAL
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
COORDENAÇÃO DA UNIDADE ACADÊMICA DA ÁREA DE INFORMÁTICA - CAMPUS CAJAZEIRAS

Às **14:00** horas do dia **03** do mês de **fevereiro** do ano de **2021**, via Google Meet, compareceu para defesa pública do **Trabalho de Conclusão de Curso**, requisito obrigatório para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, o(a) aluno(a) **MAILSON DENNIS TRAJANO DE SOUZA**, matrícula **201712010028**, tendo como Título do Trabalho **MOVING FORWARD TO A TRUE AUTONOMOUS DRIVING: AVOIDING FATAL CRASHES USING METAMORPHIC TESTING**. Constituíram a Banca Examinadora os professores **DIOGO DANTAS MOREIRA** (orientador), **RICARDO DE SOUSA JOB** (examinador) e **PAULO EWERTON GOMES FRAGOSO** (examinador).

Após a apresentação e as observações dos membros da Banca Examinadora, ficou definido que o trabalho foi considerado **APROVADO** com nota **95,** com a condição de que o (a) aluno (a) entregue, no prazo máximo de 30 dias, a versão final do trabalho, via processo eletrônico à coordenação de curso. A versão deve conter a ficha catalográfica e atender às sugestões feitas pelos membros da banca. O código fonte desenvolvido no trabalho (caso haja) deve ser enviado para o e-mail da coordenação do curso (cads.cz@ifpb.edu.br).

Cajazeiras-PB, 8 de fevereiro de 2021.

Documento assinado eletronicamente por:

- **Mailson Dennis Trajano de Souza**, **ALUNO (201712010028) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 01/03/2021 16:47:48.
- **Ricardo de Sousa Job**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/02/2021 11:11:52.
- **Paulo Ewerton Gomes Fragoso**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 08/02/2021 11:57:27.
- **Diogo Dantas Moreira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 08/02/2021 09:51:07.

Este documento foi emitido pelo SUAP em 03/02/2021. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 154544
Código de Autenticação: 0d77d5aa5d

A meu pai **Marcondes Trajano** *(in memorian). Espero que de onde estiver, esteja comemorando este passo na minha jornada. Se estou aqui hoje, é graças a tudo que você me ensinou na vida. Obrigado por tudo*

A minha Mãe, Dalva Andrade.

AGRADECIMENTOS

A minha mãe, **Maria Dalva de Andrade Trajano**, por estar sempre ao meu lado se preocupando comigo durante toda esta jornada. A meu irmão, **Maysson David Trajano de Souza**, por me incentivar a continuar seguindo o meu caminho e me ajudar sempre que precisei. A meu pai, **Marcondes Trajano de Souza**, por ser alguém que transmitia pelo olhar o orgulho que sentia de mim, olhar esse que me guiou durantes os meus piores momentos.

A meu orientador, **Diogo Dantas Moreira**, por ter aceitado me guiar neste caminho de se tornar um profissional. Agradeço por ter aceitado seguir este caminho comigo com um tema tão pouco explorado. Sem dúvida, as diversas lições que aprendi durante o tempo em que fui seu orientando e aluno, irei levar comigo pelo resto da vida. Você me fez ser um profissional melhor, e me fez ser uma pessoa melhor também! Muito Obrigado!

A **Felipe Irnyo Bernardino Luna**, um irmão, que me acompanhou e me apoiou durante toda a jornada. Dos piores aos melhores momentos. Que me incentivou e sempre acreditou em mim. Que me fez crer, que até mesmo quando a esperança se esgota, existe alguém do seu lado. Sempre serei grato por ter alguém como você ao meu lado!

A **Maria Vitória Mendes Batista (Maví)**, por ser alguém capaz de sempre me fazer sorrir nos momentos em que eu mais precisava. Pelas inúmeras noites de companhia na Loopis durante o desenvolvimento deste trabalho. Por todos os abraços durante os momentos difíceis. Por fazer parte da minha vida. Muito obrigado, por cada pequeno momento!

A **Andriele Andrade de Freitas**, por me incentivar a seguir nesta área desde o ensino médio. Por ser a pessoa que mais acredita no meu potencial. Por estar sempre por perto quando eu preciso. Por nunca desistir de mim, mesmo sendo um amigo difícil as vezes. Muito obrigado!

Agradeço especialmente a **Alexa Lins** por acreditar em mim desde o início do curso, e por me dar um espaço na Loopis para que eu tivesse um computador para estudar. A **Ricardo Job**, por me emprestar um notebook que ajudou a chegar a este ponto. A **Luciana Ferreira**, por me aturar e me incentivar durante o final deste jornada. Obrigado a todos!

Durante os últimos 4 anos, diversas pessoas passaram pela minha vida. Sei que não posso manter todas elas por perto, mas de uma maneira ou de outra estas pessoas são importantes para mim. Em específico, o grupo dos **"Paulo Renjes"**. Juntos, conseguimos passar por todas as dificuldades desta jornada.

E por fim, agradeço ao meu eu de 17 anos, que não fazia ideia de que um dia, todo aquele esforço resultaria onde estamos hoje.

"Não se pode aprender nada de uma lição que não seja acompanhada por dor, já que não se pode conseguir nada sem um sacrifício. Mas quando você aguenta essa dor e a supera, as pessoas conseguem um coração forte que não perde para nada. Sim, um coração de aço."

Edward Elric, Fullmetal Alchemist: Brotherhood

RESUMO

O desenvolvimento de aplicações para carros autônomos tem se tornado cada vez mais comum nos últimos anos devido às diversas vantagens que este tipo de tecnologia pode nos proporcionar, como a diminuição de acidentes de trânsito e a acessibilidade para pessoas incapacitadas de dirigir. Porém, a realização de testes em sistemas autônomos acaba por se deparar com inúmeras barreiras, tais como o perigo de testar estes sistemas em um ambiente de mundo real e a dificuldade de validar se o comportamento do carro está correto ou não. Tais fatores influenciaram para a primeira fatalidade registrada com relação a um carro autônomo, ocorrida no ano de 2018, em Tempe, Arizona. Nos últimos anos, a abordagem de Testes Metamórficos tem se mostrado como uma ótima alternativa para a garantia da qualidade destes sistemas expondo falhas que poderiam resultar em demais fatalidades. Porém, ainda não é possível afirmar quais fatores podem ter maior influência na causa destas falhas pelos custos e perigos da realização destes testes em mundo real. Objetivando auxiliar no mapeamento desses fatores, esse trabalho utiliza um simulador para sistemas de carros autônomos, o CARLA, junto de um sistema de rede neural treinado, por meio da abordagem Learn by Cheating, e aplica um conjunto de Testes Metamórficos que possibilita analisar e evidenciar, por meio de várias execuções, quais fatores possuem maior influência na causa destas possíveis falhas. Pelos experimentos realizados é possível analisar, por meio de um conjunto de 6 relações metamórficas, que a chuva possui uma influência em falhas equivalente a 14,6% do total de execuções realizadas, além de mostrar influência absoluta na execução de uma das rotas propostas.

Palavras-chave: Carros Autônomos. Testes Metamórficos. Testes de Software. Simulador.

ABSTRACT

The development of applications for autonomous cars has become increasingly common in recent years due to the several advantages that this type of technology can provide us, such as the reduction of traffic accidents and accessibility for people unable to drive. However, conducting tests on autonomous systems ends up encountering numerous barriers, such as the danger of testing these systems in a real-world environment and the difficulty of validating whether the car's behavior is correct or not. These factors influenced the first recorded fatality concerning an autonomous car, which occurred in 2018, in Tempe, Arizona. In recent years, the Metamorphic Testing approach appears as a great alternative to guarantee the quality of these systems, exposing flaws that could result in other fatalities. However, it is not yet possible to state which factors may have the greatest influence on the cause of these failures due to the costs and dangers of performing these tests in the real world. To assist in the mapping of these factors, this work uses an autonomous car system simulator, CARLA, next to a trained neural network system, through use of the Learn by Cheating approach, and apply a set of Metamorphic Tests that allow us to anylize and show, through a variety of executions, which factors have the gratest influence in the cause of these possible failures. Through the experiments carried out, it's possible to analyze, by a set of 6 metamorphic reactions, that the rain has an influence on failures equivalent to 14.6% of the total of executions carried out, besides showing absolute influence in the execution of one of the proposed routes.

Keywords: Autonomous Cars. Metamorphic Testing. Software Testing. Simulator

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

TM	Teste(s) Metamórfico(s)
RM	Relação Metamórfica
IA	Inteligência Artificial
GCC	*GNU Compiler Collection* (Coleção de Compiladores GNU)
NHTSA	*National Highway Traffic Safety Administration* (Administração Nacional de Segurança Rodoviária dos Estados Unidos)
SUT	*System Under Test* (Sistema Sob Teste)
LiDAR	*Light Detection and Ranging* Light Detection and Ranging (Detecção e Variação de Luz)
RADAR	*Radio Detection And Ranging* (Detecção e Variação de Rádio)
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
NN	*Neural Network* (Rede Neural)
DNN	*Deep Neural Network* (Rede Neural Profunda)
API	*Application Programming Interface* (Interface de Programação de Aplicações)
NPCs	*Non-Player Character* (Personagem Não Jogável)
LBC	*Learning by Cheating* (Aprendizagem por Trapaça)
UPC	Unidade de processamento computacional
UPG	Unidade de processamento gráfico

SUMÁRIO

1 INTRODUÇÃO

Este capítulo expõe o contexto no qual esta pesquisa foi escrita, relatando conceitos a respeito de testes de *software*, do que se trata um oráculo e quais os problemas existentes em testes que utilizam oráculos. Neste cenário então é exposta a abordagem de testes de *software* que será utilizada neste trabalho. Logo após, é apresentada a motivação a respeito da relevância e importância de carros autônomos e das possíveis falhas existentes em seus sistemas. Com isso, é apresentada a problemática deste trabalho, expondo o impacto de uma falha em sistemas de carros autônomos. Após isso, são apresentados os objetivos deste trabalho. Logo após estão dispostas as atividades a serem realizadas no desenvolvimento deste trabalho junto do cronograma previsto. Por fim, é descrito o conteúdo da estrutura do trabalho.

1.1 CONTEXTO

Quando falamos sobre desenvolvimento de *softwares* devemos dar à devida importância a garantia da qualidade de software, para termos certeza de que o nosso software não é falho e atende aquilo que ele se propõe a fazer. Diversas abordagens são utilizadas para que possamos garantir a qualidade de software, porém, testes de *softwares* tem se mostrado a abordagem mais eficiente neste quesito.

No livro "The Art Of Software Testing" (MYERS et al., 2011, tradução nossa), testes de software é definido como "um processo, ou uma série de processos, para garantir que o código faça aquilo que ele é designado à fazer, e em contraponto, que não faça nada que não seja intencional [...]". Doravante adotada como principal definição nesta pesquisa.

Apesar de existirem diversas técnicas para a aplicação de Testes de *Software*, uma das abordagens mais conhecidas e utilizadas é o "Oráculo" (WEYUKER, 1982, tradução nossa), que se trata de analisar, dado um conjunto de entradas, no software em execução, a equivalência com as saídas previstas, para que assim possamos detectar falhas.

Segundo Myers et al. (2011, tradução nossa), *softwares* devem ser previsíveis e não devem apresentar nenhuma surpresa ao usuário, o que não se aplica hoje em dia a sistemas de **Inteligência artificial (IA)**, devido a grande maioria possuir uma natureza não determinística. Em casos como este a abordagem do oráculo se torna inviável, seja por custos ou pela complexidade do *software* sob teste. Este tipo de situação é conhecido como "O Problema do Oráculo", onde não é possível, de maneira simples, prever a saída de uma determinada execução. Alguns autores se referem a softwares que se encaixam neste cenário como "Não-testáveis" (SEGURA et al., 2018; CHEN et al., 1998; WEYUKER, 1982).

Uma alternativa que pode ser utilizada para contornar o problema do oráculo é chamada de **Testes Metamórficos (TM)** (CHEN et al., 1998). Diferente da abordagem utilizada com o oráculo, não observamos apenas uma única saída, mas sim a relação que existe entre diversas saídas e entradas para que possamos avaliar se o software possui falhas. Estas relações são chamadas de **Relações Metamórficas (RM)**.

Para que possamos compreender melhor o funcionamento de TM vamos utilizar um exemplo famoso de pesquisa em um motor de busca (ZHOU et al., 2015), como o *Google*, *Bing* e etc. Imagine que realizamos uma busca por "Testes metamórficos", que será chamada de *c1*, e nos traga um resultado com um número X de páginas. Se buscarmos agora por "Testes metamórficos em *webservice*", que será chamado de *c2*, obviamente o resultado dado por *c2* deve conter o mesmo número de páginas retornado por *c1* ou inferior. Caso o resultado de *c2* não cumpra este requisito podemos dizer que existe uma falha no motor de busca. A condição imposta para sabermos se o resultado de *c2* possuía alguma falha é o que chamamos de relação metamórfica, enquanto *c1* e *c2* seriam, respectivamente, os casos de teste original e complementar.

A partir do momento que a abordagem de TM foi proposta por Chen et al. (1998) ela já vem sendo utilizada para encontrar falhas em diversos tipos de sistemas, como é o caso das falhas encontradas no motor de busca do *Google*, o compilador *GCC* e até mesmo softwares de IA. Pelo fato da abordagem possuir um conceito de fácil aprendizagem e baixo custo para utilização (CHEN et al., 2018), ela tem se mostrado muito útil na descoberta de problemas que afetariam diretamente o cotidiano das pessoas ou até mesmo colocariam em risco a vida de pessoas, como é o caso se sistemas críticos.

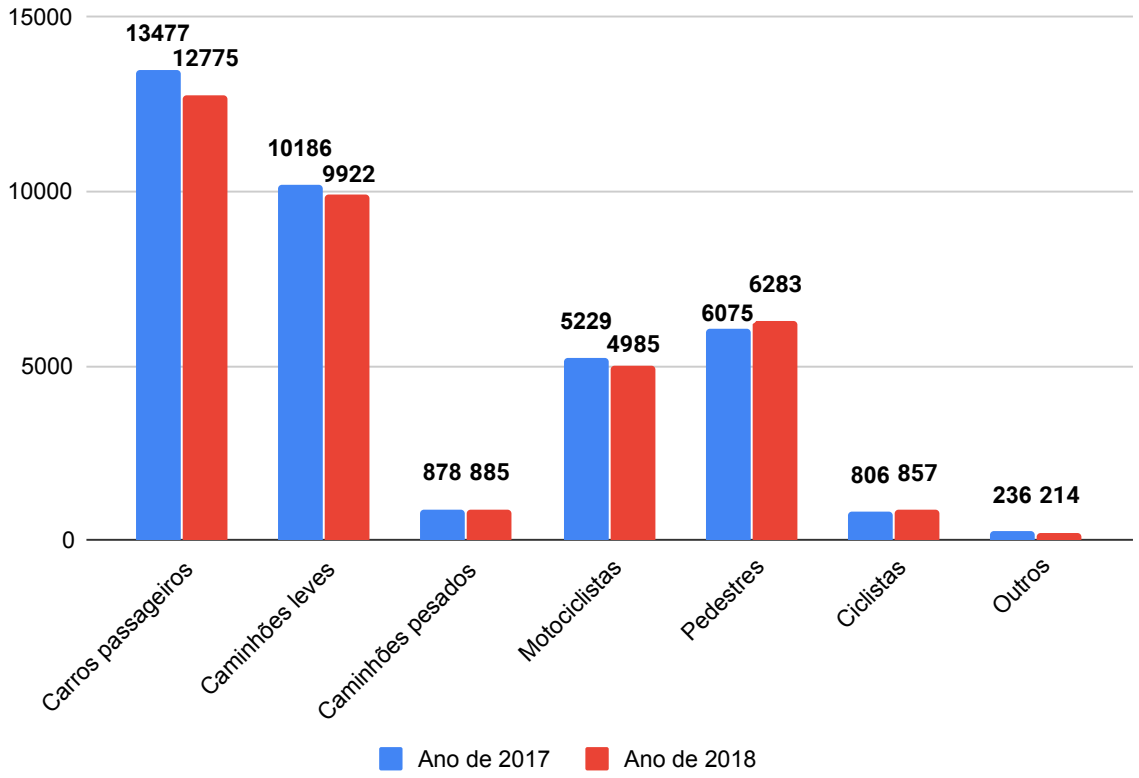
1.2 MOTIVAÇÃO

Nos últimos anos foram divulgadas diversas informações na mídia a respeito dos avanços tecnológicos em carros autônomos, além de questionar bastante sua confiabilidade e utilidade no nosso cotidiano (HEINEKE et al., 2018; MADRIGAL, 2018; TOPHAM, 2021). A respeito destes dois aspectos, a NHTSA (Administração Nacional de Segurança Rodoviária dos Estados Unidos) diz que a total automatização de veículos iria diminuir bastante o número de acidentes envolvendo veículos motorizados, já que "94% dos acidentes graves de trânsito são causados por falhas humanas" (NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION, 2019), além de trazer mais acessibilidade de locomoção para aqueles que não podem dirigir ou que possuem alguma deficiência física.

Em outubro do ano de 2019 foi publicada uma nota de pesquisa pela NHTSA que nos informa que no ano de 2018 houve um total de 36.560 **fatalidades** em acidentes envolvendo veículos motorizados nos Estados Unidos da América, enquanto destes, 6.283 eram pedestres e

857 eram ciclistas (NATIONAL CENTER FOR STATISTICS AND ANALYSIS, 2019). Os dados citados encontram-se na Figura 1.1, além de outros números sobre fatalidades em acidentes de trânsito.

Figura 1.1 – Fatalidades em acidentes de trânsito



Fonte: Adaptado de National Center for Statistics and Analysis (2019)

O número de fatalidades relacionados a veículos motorizados em zonas urbanas no ano de 2018 alcançou 19.498, 34% a mais se comparado com o ano de 2009, onde ocorreram 14.501 fatalidades. Já nas zonas rurais o número de fatalidades chegou a 16.411, 14% a menos do que no ano de 2009, onde houve 19.323 fatalidades (NATIONAL CENTER FOR STATISTICS AND ANALYSIS, 2019). Ainda no ano de 2018, foi levantado que ocorreram 10.511 fatalidades causadas por motoristas alcoolizados, dos quais 4.217 eram apenas passageiros nestes veículos (NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION, 2019).

A utilização de veículos autônomos busca trazer o benefício da diminuição do número de fatalidades envolvendo veículos motorizados, principalmente as causadas por motoristas alcoolizados ou sob efeitos de outras drogas. Porém devemos nos questionar, ainda irão existir fatalidades causadas por veículos motorizados, mesmo se os automatizarmos? E se existirem, o que faremos?

1.3 PROBLEMÁTICA

Em março de 2018 ocorreu o primeiro **acidente fatal** relacionado a um veículo autônomo. Elaine Herzberg se tornou a primeira pedestre a ser morta por um carro autônomo após ser atropelada por um carro da Uber na cidade de Tempe, Arizona (LEVIN, 2018). Devido ao ocorrido, foi levantado um questionamento: Será que existem cenários onde um software pudesse interpretar de maneira errônea os dados dos sensores para a detecção de obstáculos, o tornando incapaz de detectar um pedestre ou outros objetos no caminho? Infelizmente a resposta para essa pergunta é sim (ZHOU; SUN, 2019).

No mesmo ano foi realizada uma pesquisa para gerenciamento de testes no **Baidu Apollo**[1], um sistema utilizado para controle de diversos carros autônomos no mercado atualmente e também responsável pela interpretação de dados de saída de um **LiDAR** (Light Detection and Ranging) para tomadas de decisões. O LiDAR se trata de um aparelho utilizado para que carros autônomos possam realizar uma tarefa análoga a visão humana através de pulsos de luz capaz de detectar obstáculos a centenas de pés de distância (GAO et al., 2018).

Foram utilizados então TM automatizados para avaliar se o software possui ou não falhas, e como resultado foram descobertas diversas falhas na interpretação dos dados recebidos pelo LiDAR, o que ocasionava na não detecção de obstáculos que deveriam ser considerados pelo software, o que poderia resultar em inúmeros outros acidentes (ZHOU; SUN, 2019).

Contudo, na pesquisa realizada apenas uma RM foi utilizada para realizar os testes no sistema de interpretação dos dados do LiDAR, o que nos deixa questionar se outras RMs iriam expor outras falhas que também influenciariam em um cenário de detecção de pedestres. Ou até mesmo se outros módulos de interpretação de dados de entrada poderiam cometer falhas fatais com esta.

Diante dos questionamentos e dados evidenciados até o momento atual, compreendendo os benefícios de se utilizar veículos autônomos e o problema relacionado à segurança e corretude dos sistemas críticos de interpretação de dados para tomadas de decisões em veículos autônomos é que chegamos a seguinte indagação que representa o problema abordado neste trabalho:

Quais fatores podem ter maior influência na ocorrência de falhas em sistemas de carros autônomos?

[1] <http://apollo.auto/>

1.4 OBJETIVOS

Este trabalho tem como objetivo **Explorar o uso de técnicas de testes metamórficos para realizar verificação em sistemas de carros autônomos**. Para atingir este objetivo, propõe-se aqui os seguintes objetivos específicos:

- Estudar as técnicas e aplicações de testes metamórficos;
- Identificar abordagens existentes dentro do contexto de verificação de sistemas de carros autônomos;
- Propor um conjunto de casos de testes capazes de identificar fatores que causam falhas nos sistemas propostos;
- Avaliar a eficácia do conjunto de casos de testes propostos.

1.5 ESTRUTURA DO TRABALHO

Após este capítulo de introdução este trabalho se encontra organizado da seguinte maneira: No Capítulo 2, apresentamos a **Fundamentação teórica**, onde abordamos os principais conceitos a respeito de Testes, Oráculos e os demais fundamentos necessários para a total compreensão de Testes Metamórficos, além de uma seção dedicada ao funcionamento básico de carros autônomos e redes neurais; no Capítulo 3, apresentamos os **Trabalhos relacionados** e seus devidos critérios de busca e seleção; no Capítulo 4, descrevemos a **Metodologia** deste trabalho; e por fim, os capítulos 5 e 6 apresentam a **Ciclo de Design 1** apresentada para solucionar a pergunta de pesquisa levantada neste capítulo, dissertando a respeito do processo seguido para a execução dos experimentos proposto com seus devidos resultados, e por fim, expondo as devidas **Considerações finais** deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

No decorrer deste capítulo iremos apresentar a fundamentação teórica que serviu de base para este trabalho. Na seção 2.1 discorremos a respeito de Testes de Software, sua importância, seus principais conceitos e dos Níveis de Testes de Software, além de falarmos a respeito do que são Oráculos e o Problema do Oráculo. Logo após, na seção 2.2 apresentamos os conceitos básicos a respeito de Testes Metamórficos e Relações metamórficas. São apresentados então na seção 2.3 alguns conceitos básicos a respeito de Carros Autônomos. E por fim, ainda na seção 2.3 são expostas breves descrições a respeito de Detecção de Objetos e Redes Neurais, que por mais que não sejam o principal foco deste trabalho, ainda são necessários para um melhor entendimento da proposta deste trabalho.

2.1 TESTES DE SOFTWARE

Ao falarmos sobre testes de software, estamos nos referindo a uma das principais abordagens utilizadas no controle de qualidade de software. Para o aumento da confiabilidade e da segurança o "Software deve ser testado para cobrir o maior número de erros possíveis [...]" (PRESSMAN, 2015, tradução nossa). A existência de erros em nossos sistemas é **inevitável**, por tal motivo, utilizamos testes de softwares para mapear estes erros para que então possamos corrigi-los e prosseguir com este ciclo para a garantia ao máximo a **corretude** do nosso sistema.

Softwares são testados inúmeras vezes antes de entrarem em um ambiente real de uso, principalmente se estes softwares forem críticos e possuírem a necessidade de ter uma alta confiabilidade antes serem utilizados de fato. Realizamos isso para que possamos minimizar o risco de falhas serem encontradas durante o uso do sistema em produção, o que causaria um impacto negativo ao uso do software (ISO/IEC/IEEE 29119-1, 2013).

Como dito por Pressman (2015, tradução nossa, grifo nosso), "O teste de software é um elemento de um tópico mais amplo, geralmente chamado de **verificação** e **validação**". Quando testamos, não garantimos apenas um aumento na corretude do sistema, mas também conseguimos garantir que o sistema faz aquilo que se propõe a fazer (validação) e que ele atende aos requisitos definidos pelo usuário (verificação). Diversos outros tipos de falhas podem ser encontradas através de Testes de Software, como mal desempenho do sistema; interpretação errada dos requisitos do sistema, entre outros. Contudo, existem algumas estratégias diferentes que podemos utilizar para testar softwares, e que são separadas em alguns níveis.

2.1.1 Níveis de testes de software

Os testes de software podem ser realizados de duas diferentes perspectivas, sendo elas os **Testes de caixa branca** (White-box testing) e os **Testes de caixa preta** (Black-box testing). Na perspectiva de testes de caixa branca, temos todo o conhecimento a respeito da lógica da aplicação e de como os componentes interagem entre si, para que possamos testar se os componentes funcionam de maneira apropriada. Já na perspectiva de testes de caixa preta, temos apenas conhecimento da interface do sistema e de seus requisitos, para que possamos avaliar se ele atende a tais requisitos (PRESSMAN, 2015).

Quando realizamos testes em nosso sistema, é interessante que consigamos alcançar o sistema da maneira mais ampla possível para a melhor garantia de sua corretude e confiabilidade, por tal motivo os testes realizados em um sistema são divididos em alguns níveis, como: **Unidade**, **Integração** e **Sistema** (PRESSMAN, 2015). Tais níveis funcionam como uma evolução dos testes no sistema, desde o menor dos componentes existente lá dentro, até o sistema como um todo. Para isto, a criação destes testes podem ser derivados desde requisitos, especificações, design, artefatos e código fonte (AMMANN; OFFUTT, 2008).

A princípio, os testes são realizados geralmente pelos desenvolvedores em cada componente de maneira isolada, para garantir que cada componente consiga desempenhar a função para a qual foi designado. Para esta parte do processo é dado o nome de **Teste de Unidade** (PRESSMAN, 2015). Com testes de unidade podemos encontrar e tratar erros de maneira isolada em cada componente antes de realizarmos algum tipo de integração com os demais componentes.

Após a realização dos testes de unidade, passamos a realizar testes em cima de componentes interligados para que possamos garantir que suas funcionalidades, quando postas em conjunto, produzam o resultado desejado de uma funcionalidade maior (PRESSMAN, 2015). Para esta parte do processo damos o nome de **Testes de Integração**. Com os testes de integração podemos garantir que os sub sistemas, testados individualmente, possuem consistência quando postos em conjunto e se comunicam corretamente (AMMANN; OFFUTT, 2008).

Por fim, somos aptos a pôr em prática os testes no sistema como um todo, para que possamos avaliar se o sistema desenvolvido atende a suas especificações (AMMANN; OFFUTT, 2008). Para esta parte do processo damos o nome de **Testes de Sistema**. Nesta etapa também podemos verificar se o sistema atende ao que era esperado pelo usuário final, o que pode ser chamado de **Teste de Aceitação** (PRESSMAN, 2015).

Porém, a aplicação de todo este processo em diferentes níveis de testes a um software é bastante custoso e leva um longo tempo para ser concluído, tempo esse que pode ser muito

mais custoso ainda dependendo da complexidade do software sob teste. Por tal motivo é utilizada bastante a abordagem de automatização dos testes: estratégia que tem se mostrado como a solução mais apropriada para solucionar o problema de custo e tempo, e que pode ser utilizada nos diversos estágios de desenvolvimento (BERTOLINO, 2007). Para que possamos aplicar a automatização dos testes utilizamos uma abordagem chamada **Oráculo de Testes**.

2.1.2 Oráculos

Quando testamos nosso sistema, temos como principal objetivo encontrar falhas ou comportamentos inesperados de um componente ou módulo do nosso sistema, e para que possamos de fato realizar esta tarefa uma das abordagens mais bem aceitas é o Oráculo de **Testes**, que tem o objetivo de "checar quando o sistema sob teste se comportou da maneira correta a execuções particulares" (BARESI; YOUNG, 2001, tradução nossa).

O oráculo de testes consistem em um grupo de métodos de decisão para que possamos julgar e interpretar os resultados de um teste, e a partir disso afirmar se ele foi bem sucedido ou não (OLIVEIRA et al., 2014). O tipo mais comum de oráculo é aquele utilizado para que possamos avaliar a corretude de um conjunto de dados de saída, dado um determinado conjunto de dados de entrada (HOWDEN, 1978).

Contudo, em alguns casos a complexidade do sistema é tanta que nem sempre é viável o uso de um oráculo de testes, seja financeiramente ou por questões de tempo (CHEN et al., 1998; WEYUKER, 1982). Para este tipo de cenário é dado o nome de **Problema do Oráculo**.

2.1.2.1 Problema do oráculo

O problema do oráculo ocorre em cenários onde a viabilidade de se ter um oráculo é muito baixa, pelo fato de que se torna muito complexo e custoso tentar garantir a corretude do comportamento ou saídas de um sistema. Em muitos casos, sistemas que se encontram em cenários como este são chamados de Sistemas Não-Testáveis (SEGURA et al., 2018; CHEN et al., 1998; WEYUKER, 1982; OLIVEIRA et al., 2014; BARR et al., 2014).

Para demonstrar um caso onde o problema do oráculo se aplica, usaremos um exemplo bastante simples abordado por Zhou et al. (2015) de um motor de busca como o *Google*, *Bing*, etc. Imagine que realizamos uma busca X no motor de busca M. Devido a enorme quantidade de dados tratada por estes motores de busca, é muito difícil dizer se o resultado trazido por $M(X)$ está correto ou não.

Diversas alternativas foram propostas na literatura científica para aliviar ou resolver o problema do oráculo. Uma das principais alternativas propostas foi a dos **Oráculos Derivados**.

2.1.2.2 Oráculos derivados

Ao utilizarmos de oráculos derivados conseguimos garantir a corretude de um sistema através de informações derivadas de diversos artefatos, como documentação, propriedades do sistema ou até mesmo das demais execuções do sistema sob teste. Recorremos então a esta alternativa sempre que um Oráculo de Testes propriamente dito é considerado inviável para a situação (BARR et al., 2014).

2.2 TESTES METAMÓRFICOS

Quando realizamos Testes de Software em sua grande maioria utilizamos um Oráculo como principal ferramenta para que possamos validar os dados de saída de um componente ou seu comportamento, para que com isso possamos então detectar falhas. Contudo, de tal maneira não conseguimos garantir que não hajam outras falhas (SEGURA; ZHOU, 2018). Dado este problema, em casos onde não há um oráculo o que pode ser feito para que possamos garantir que existem erros?

Testes metamórficos é uma abordagem proposta por Chen et al. (1998) que é bastante utilizada para que possamos aliviar o problema do oráculo e expor a existência de falhas através de relações entre diferentes execuções e de seus dados de entrada e de saída. Além de aliviar o problema do oráculo, Testes metamórficos podem ser facilmente automatizados (SEGURA et al., 2016; CHEN, 2010). Dado o fato de que esta técnica independe do uso de um oráculo, ela também se encaixa na categoria de **Oráculo derivado**.

Em diversas situações nos deparamos com casos de testes bem sucedidos (aqueles que não acusam falhas), e tais casos de testes vêm sendo considerados como inúteis (pelo fato de não revelarem falhas) e são geralmente descartados (CHEN et al., 2018; SEGURA; ZHOU, 2018). Por outro lado, Chen et al. (1998) propôs que os casos de testes bem sucedidos não fossem descartados, e sim reutilizados para a criação de novos casos de testes baseados em informações úteis e possíveis falhas comumente associados com o tipo de aplicação sob teste. Com isso, é possível validar uma determinada relação entre estes dois casos de teste para detectar a existência de falhas. Para este tipo de ligação damos o nome de **Relação Metamórfica**.

2.2.1 Relações metamórficas

Para que possamos definir quando um caso de teste é bem sucedido ou falho ao utilizarmos de Testes Metamórficos é necessária a utilização de Relações Metamórficas (RM), que dizem respeito a propriedades necessárias das funcionalidades do sistema a ser testado entre os dados de entrada de diversas execuções e seus respectivos dados de saída esperados (SEGURA; ZHOU, 2018).

Uma relação metamórfica tem o papel de identificar quando ocorrerá uma determinada mudança no comportamento de um programa devido a uma alteração específica realizada nos dados de entrada (Kanewala; Yueh Chen, 2019). Podemos expor a existência de falhas em um programa *P* ao aplicar a um caso de teste bem sucedido *C*, alguma alteração nos seus dados de entrada, criando então *C'*, e então avaliar os resultados de *P(C)* e *P(C')* através de uma Relação Metamórfica entre as diversas execuções. Caso alguma das execuções viole a relação definida, é possível afirmar que o programa *P* possui alguma falha.

A partir deste momento para facilitar a compreensão deste trabalho iremos utilizar as definições de Moreira (2019) que estão descritas abaixo:

Caso de Teste Original Um caso de teste já existente ou gerado e que não detectou nenhum tipo de falha.

Caso de Teste Complementar Um caso de teste gerado a partir de uma relação metamórfica e tendo como base o caso de teste original.

Como dito por Chen (2010, Tradução nossa), "A seleção das relações metamórficas é uma questão fundamental na aplicação de testes metamórficos". Com o tempo, a busca por relações metamórficas tem se mostrado bastante eficiente em diversos domínios, como aprendizagem de máquina, *web services* e recentemente sistemas autônomos (CHEN, 2010; SEGURA et al., 2016; SEGURA et al., 2018). Contudo, para que possamos pôr as relações metamórficas em prática, devemos seguir alguns passos separados por alguns autores na literatura científica como o **Processo de Testes Metamórficos**.

2.2.2 Processo de teste metamórfico

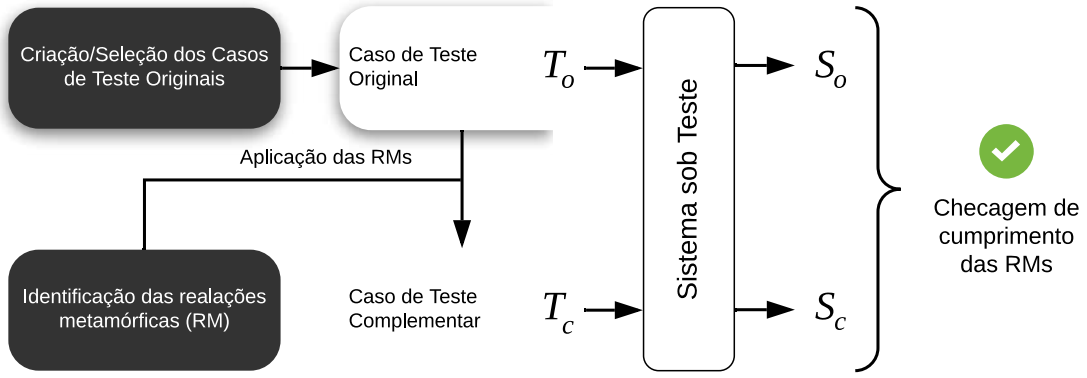
O processo de testes metamórficos trata-se de um conjunto de tarefas que devemos seguir para que possamos encontrar o maior número de falhas possíveis no *System under test* (SUT). Para tal processo, Kanewala e Yueh Chen (2019) definiram as seguintes tarefas a serem seguidas durante a aplicação de testes metamórficos e que estão dispostas na figura 2.1:

1. **Identificação das Relações Metamórficas** (RM), onde são propostas um conjunto de relações metamórficas para o SUT através de suas especificações. Para que possamos identificar estas relações podemos utilizar de propriedades únicas do sistema, além de necessitarmos de um conhecimento relativo a respeito do domínio do SUT (BARR et al., 2014; SEGURA et al., 2016).
2. **Criação e Execução de Casos de Teste Originais**, comumente gerados a partir de técnicas como randomização, geração de dados de entrada para testes baseados em

falhas, entre outros meios. Estes casos também podem ser casos de teste já existentes e que não encontraram nenhuma falha.

3. **Criação de Casos de Teste Complementares**, onde é utilizado o conjunto de relações metamórficas encontradas no passo 1 para que possamos transformar os casos de teste originais obtidos pelo passo 2 em casos de testes complementares
4. **Execução dos Casos de Teste Complementares**, para que possamos comparar as saídas entre o caso de teste original e o caso de teste complementar, e com isso, avaliarmos quais delas atendem às relações metamórficas definidas. Caso a relação metamórfica seja violada é exposto então que o SUT pode ser falho.

Figura 2.1 – Processo de testes metamórficos



Fonte: Elaborado pelo autor (2019)

Como proposto inicialmente por Chen et al. (1998), ao utilizamos o processo de testes metamórficos em nossa aplicação, aumentamos o número de casos de testes, dado o passo 3 citado anteriormente, pela geração dos casos de testes complementares. Pelo fato deste processo ser facilmente automatizado, o aumento na cobertura de falhas é bastante considerável.

Ao utilizarmos Testes metamórficos conseguimos **verificar** a corretude do SUT além de conseguirmos **validar** o mesmo, como já mostrado por Xie et al. (2011), onde foi demonstrado que com a utilização de testes metamórficos foi possível encontrar um conjunto de relações metamórficas baseadas, não em propriedades específicas do algoritmo testado, mas nas demais propriedades do sistema.

A utilização deste processo pode ser aplicado em diversas situações que apresentam ou não o problema do oráculo, além de ser aplicado em outros domínios, principalmente domínios de aplicações críticas (onde falhas podem causar danos irreversíveis).

2.3 CARROS AUTÔNOMOS

Com o decorrer das evoluções tecnológicas ao passar dos anos, diversas ideias que não passavam de ficção científica acabaram se tornando realidade, e algumas delas têm se mostrado cada vez mais presentes no nosso cotidiano. Um dos principais casos e mais comentados na atualidade é o de **Carros Autônomos**. Desde o ano de 2012[2] os carros autônomos já vem sendo utilizados ao redor do mundo em algumas cidades e estados.

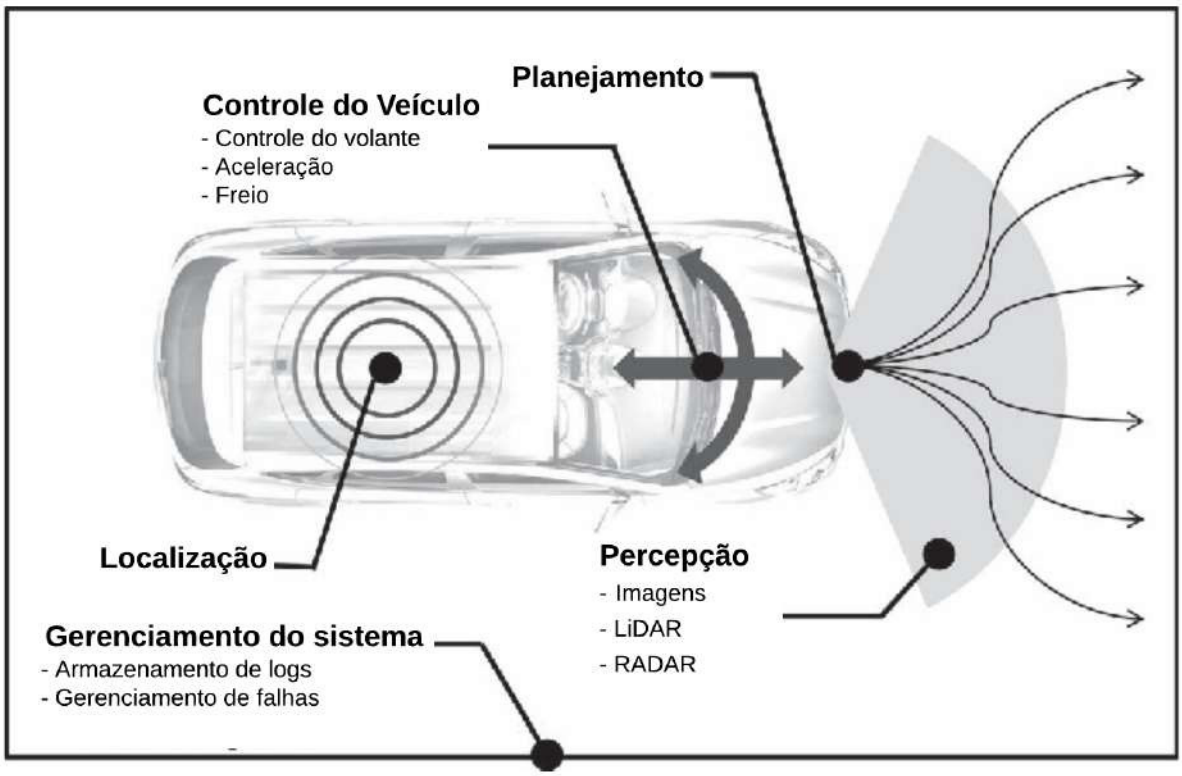
Um carro autônomo trata-se de um **híbrido** entre veículos e computadores. Estes híbridos são então operados por sistemas bastante complexos compostos de diversos sub-componentes, como: câmeras; sensores a laser; e sistemas de GPS; e Radares (DUFFY; HOPKINS, 2013). São este sub-componentes, junto das tomadas de decisão a partir de seus dados, que permitem que um carro autônomo cumpra seu propósito, que se trata de operar sem nenhuma intervenção humana (DUFFY; HOPKINS, 2013; JO et al., 2014; BIMBRAW, 2015).

A partir das responsabilidades destes sub-componentes é possível atender às cinco funcionalidades básicas definidas por Jo et al. (2014) para um carro autônomo, dispostas na Figura 2.2, que consistem em:

- **Percepção** trata-se do processo de identificação do ambiente ao redor do carro utilizando diversos tipos de técnicas sensoriais como RADAR, LiDAR e visão computacional. Esta é uma das funcionalidades mais críticas, pois qualquer falha pode resultar em acidentes fatais.
- **Localização**, onde é definida a localização do carro a partir de técnicas de GPS e mapas de estradas.
- **Planejamento** é a funcionalidade onde é definido o comportamento e o movimento do carro autônomo que são baseadas nas informações adquiridas das funcionalidades de percepção e de localização.
- **Controle**, fase onde são seguidos os comandos decididos a serem tomados pela funcionalidade de planejamento, como o movimento da direção, o controle da aceleração e de freios do carro.
- **Gerenciamento do sistema**, tem como principal função supervisionar o sistema de automação de modo geral, gerenciando possíveis falhas do sistema, registrando *logs* do sistema e dispondo uma interface humano-máquina.

[2] <https://edition.cnn.com/2012/05/07/tech/nevada-driveless-car>

Figura 2.2 – Funcionalidades básicas de um carro autônomo



Fonte: Adaptado de Jo et al. (2014)

Atualmente, acidentes de trânsito têm se mostrado como uma das maiores causas de mortes no mundo inteiro (BIMBRAW, 2015). Com a utilização de carros autônomos podemos diminuir consideravelmente este número de acidentes, em inúmeros fatores, além de diminuir congestionamentos, impactos e até mesmo poluição (no caso de carros elétricos). Com as tomadas de decisões autônomas do carro, a partir da percepção do ambiente ao seu redor em tempo real, é possível diminuir consideravelmente as preocupação com casos de motoristas sob efeitos de álcool ou outras drogas, distraídos, entre outras situações (XIE et al., 2011; EMANI et al., 2019). Mas para isso, é necessário que estes sistemas tenham capacidade de distinguir completamente cada objeto ao seu redor para não tomarem decisões erradas.

2.3.1 Detecção de objetos

Quando desenvolvemos um sistema para carros autônomos é necessário garantir um conjunto de funcionalidades, como descrito anteriormente. Mas uma delas é de importância crucial para o total funcionamento do sistema, trata-se da **Detecção de Objetos**. São inúmeros os fatores que podem influenciar no comportamento de um carro autônomo, desde a presença de outros carros próximos, até mesmo uma placa de trânsito.

A detecção de sinais e placas de trânsito são os responsáveis pelo comportamento

usual do carro nas estradas, o comportamento que nos é ensinado sobre regras de trânsito. Enquanto os outros fatores, como pedestres, ciclistas, carros próximos, entre outros, são os responsáveis pelas demais tomadas de decisão em tempo real. Ainda assim, mudanças climáticas e de luz podem acabar interferindo na detecção destes objetos.

Esta detecção é realizada a partir do diversos sensores existentes nos sub-componentes presentes em um carro autônomo, como o caso do LiDAR, RADAR, câmeras para a aplicação de visão computacional, entre outros (BIMBRAW, 2015; EMANI et al., 2019). Por tal motivo, é necessária uma maneira de se interpretar os dados de saída destes sensores para que seja possível tomar as decisões corretas, e para isto são utilizadas comumente **Redes Neurais** (NN).

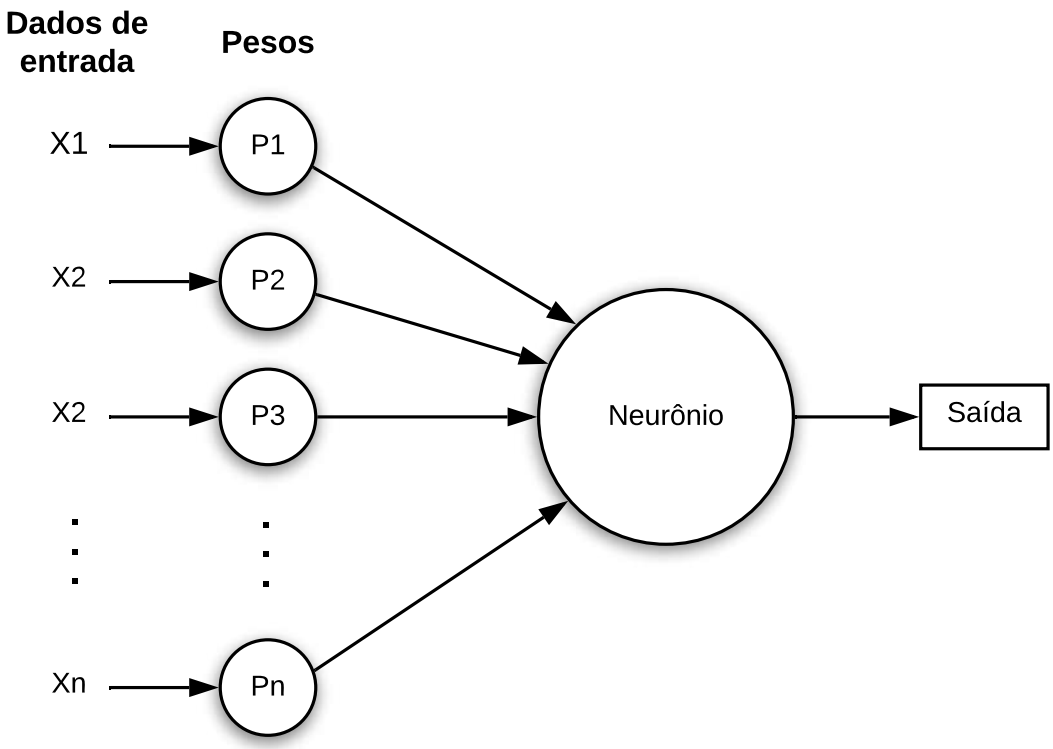
2.3.1.1 Redes neurais

Se analisarmos calmamente, o ato de tomar uma decisão a partir de um conjunto de informações é algo que se assemelha bastante ao funcionamento conhecido de um cérebro humano. Possuímos um conjunto de células em nosso sistema nervoso conectadas umas às outras as quais são chamadas de neurônios. Ao aplicarmos estímulos externos a estes neurônios os tornamos mais fortes, o que nos possibilita aprender constantemente. Na tecnologia, existe uma técnica focada em recriar esta estrutura para máquinas, a qual damos o nome de **Redes Neurais** (AGGARWAL, 2018).

Em uma rede neural utilizamos um conjunto de unidades computacionais que são chamadas de **Neurônios**. Cada neurônio é conectado a outro por uma ligação que chamamos de peso. Para cada dado de entrada que um neurônio recebe, é classificado com um determinado peso, o que afeta diretamente no comportamento da função computacional (AGGARWAL, 2018; NIELSEN, 2015). Cada neurônio então calcula os dados recebidos dos neurônios anteriores, utilizando os pesos como parâmetros intermediários, e então propaga o seu resultado para os neurônios seguintes, como demonstrado na Figura 2.3.

Assim como a aprendizagem humana ocorre através do fortalecimento dos ligamentos entre os nossos neurônios, para que possamos fazer uma rede neural aprender temos de fazer alterações nos pesos que interligam os neurônios. É necessário um conjunto de dados dispostos em pares entrada-saída da função para que possamos então treinar o algoritmo sobre como se comportar naquela situação (AGGARWAL, 2018). Ao realizarmos o treinamento alimentando a rede neural com o conjunto de dados, recebemos uma avaliação dos pesos a partir do quão bem o sistema se comporta com os dados recebidos considerando os treinamentos prévios. A partir de seus erros, o sistema modifica os pesos até se adaptar à resposta correta (NIELSEN, 2015).

Figura 2.3 – Funcionamento de um Neurônio em uma NN



Fonte: Elaborado pelo autor (2019)

Com uma abordagem como essa podemos treinar uma rede neural para diversos tipos de domínios e funcionalidades, como reconhecer objetos em uma imagem e classificar qual objeto é aquele. Identificar a distância de um objeto através de dados de um sensor a laser e ainda determinar padrões que identifiquem que tipo de objeto é aquele. Todos estes cenários se encontram presentes em um carro autônomo e necessitam de tomadas de decisões bastante precisas e críticas.

2.4 CONSIDERAÇÕES FINAIS

No decorrer deste capítulo foi apresentada a fundamentação teórica necessária para o melhor entendimento deste trabalho. Inicialmente foram apresentados conceitos básicos a respeito de **Testes de Software** e seus diferentes níveis, **Oráculos de Testes** e a respeito do **Problema do Oráculo**.

Foram apresentados ainda uma introdução a **Testes Metamórficos** e **Relações Metamórficas**, expondo seu funcionamento e suas principais aplicações como técnica de validação e verificação, além de funcionar como abordagem para geração de novos casos de

testes.

Por fim foram apresentados então o funcionamento básico de um **Carro Autônomo** e de **Redes Neurais**, que são de extrema importância para o total entendimento do objetivo deste trabalho.

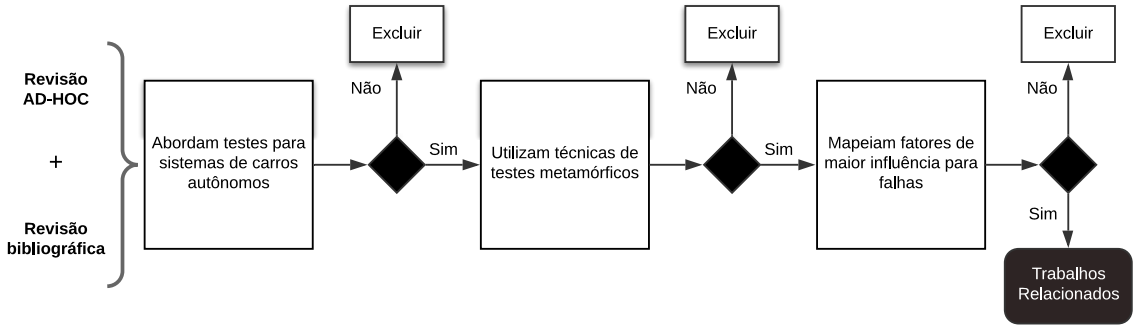
3 TRABALHOS RELACIONADOS

Este capítulo aponta trabalhos relacionados com esta pesquisa, discorrendo a respeito do método de seleção utilizado para escolher os trabalhos e da estratégia de busca, explicitando as bases onde foram realizadas as buscas e quais palavras chaves foram escolhidas para tal. Logo após, são relatados os trabalhos relacionados. E por fim é apresentada uma discussão a respeito da relevância destes trabalhos e da atualidade do tema abordado.

3.1 MÉTODO DE SELEÇÃO

Por este trabalho se tratar de uma pesquisa que envolve a utilização de Testes Metamórficos em sistemas de carros autônomos foram então levantados uma série de critérios utilizados no refinamento da busca pelos trabalhos relacionados. Critérios esses que estão especificados na estratégia de seleção na Figura 3.1.

Figura 3.1 – Estratégia de seleção de trabalhos relacionados



Fonte: Elaborado pelo autor (2019)

A princípio são selecionados os trabalhos que abordam algum tipo de teste em sistemas de carros autônomos. A seleção é então refinada para os trabalhos que se utilizam da técnica de Testes Metamórficos como sua abordagem principal. E por fim é aplicado um último filtro para selecionar os trabalhos que mapeiam algum fator que possua influência na causa de falhas. Caso o trabalho atenda a estes três critérios descritos ele é então considerado como um trabalho relacionado. Caso ele não atenda a algum dos três critérios ele é descartado.

3.2 ESTRATÉGIA DE BUSCA

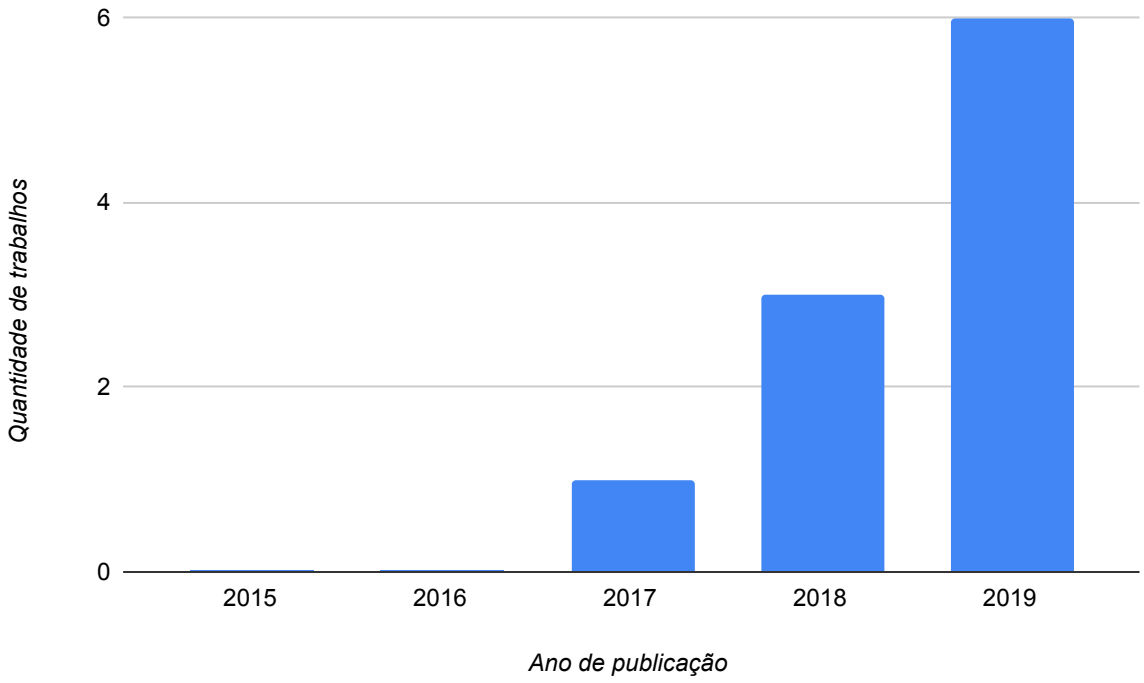
Para que pudéssemos escolher quais trabalhos passariam pelos critérios de seleção definidos na seção 3.1 foi criada então uma uma estratégia de busca por estes trabalhos, onde

foi realizada uma série de pesquisas em um conjunto dos principais repositórios de trabalhos técnicos, sendo eles: **ACM**, **IEEE Xplore**, **Springer** e o **ScienceDirect**.

Foi então realizada uma busca pelos trabalhos entre 1998 (Ano de publicação do primeiro trabalho sobre Testes Metamórficos (CHEN et al., 1998)) e 2019 e que possuam "Metamorphic Testing", "Metamorphic Test" ou "Metamorphic Relation" em sua abstract, título ou palavras chave, além de incluir "Autonomous Car", "Self-driving Car", "Driverless Car" ou "Autonomous system" também em sua abstract, título ou palavras chave.

Com isso foram encontrados um total de 10 (dez) candidatos a trabalhos relacionados, dispostos por ano na figura 3.2, que foram submetidos então aos critérios definidos na estratégia de seleção mencionada anteriormente para decidir quais trabalhos seriam de fato relacionados ou excluídos.

Figura 3.2 – Candidatos a trabalhos relacionados



Fonte: Elaborado pelo autor (2019)

Dos candidatos a trabalhos relacionados é possível notar que apesar do intervalo de tempo definido para a busca ser entre 1998 e 2019, foram encontrados trabalhos candidatos apenas entre os anos de 2017 e 2019, sendo mostrada uma crescente na quantidade publicações a respeito deste tema, o que remete a pertinência que este tema tem tomado nos dias atuais.

3.3 TRABALHOS RELACIONADOS

Após os processos de busca e seleção descritos anteriormente, foram selecionados 3 (três) trabalhos como relacionados, que estão listados a seguir.

- **DeepTest: automated testing of deep-neural-network-driven autonomous cars**, (TIAN et al., 2018)
- **DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems**, (ZHANG et al., 2018)
- **Metamorphic Testing of Driverless Cars**, (ZHOU; SUN, 2019)

O Trabalho de Tian et al. (2018) apresenta o DeepTest, que consiste em uma ferramenta para testes sistemáticos e automatizados de **Deep Neural Networks (DNNs)** para veículos autônomos. No trabalho é utilizada uma técnica para sistematização automática de casos de testes que cobrem o maior número possível de neurônios em uma DNN. Para isso é utilizada a câmera como dado de entrada escolhido para a realização dos testes na DNN do sistema realizando transformações realistas de possíveis cenários para um carro autônomo, como a adição de névoa, chuva e até mesmo contraste nas imagens, como demonstrado na Figura 3.3.

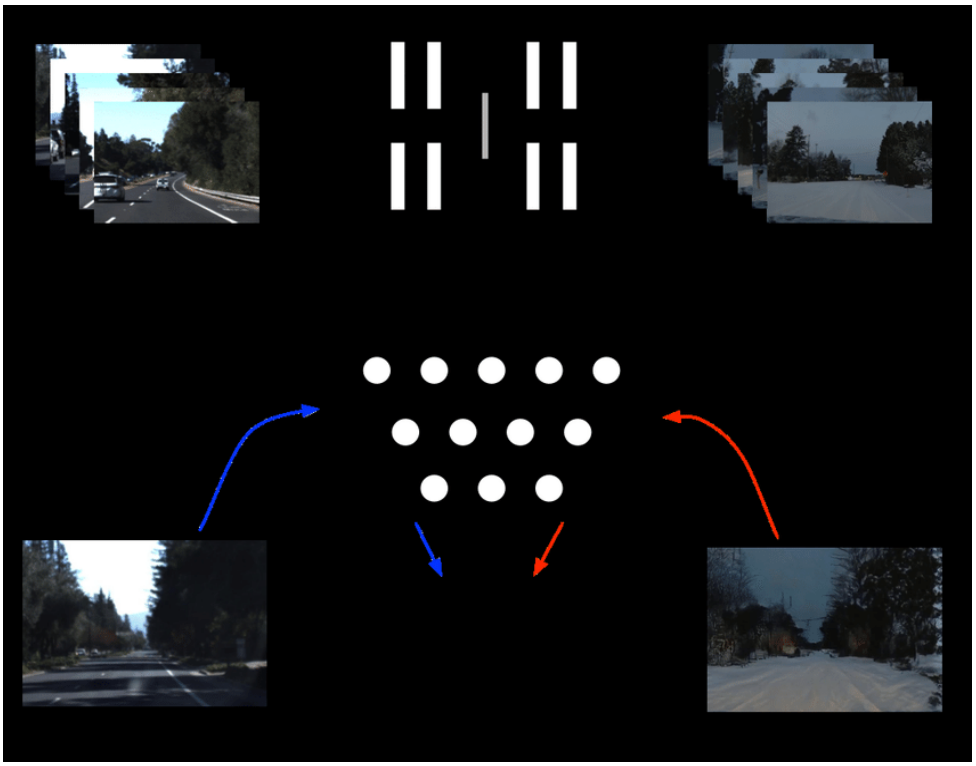
Figura 3.3 – Adição de névoa ao cenário demonstrando comportamento errôneo

Fonte: Tian et al. (2018)

Comparando o comportamento nos dois cenários é possível encontrar as falhas. Com isso, foram encontrados mais de 1000 (mil) comportamentos errôneos nas DNNs testadas. Além de ser demonstrado de maneira empírica que a quantidade de neurônios cobertos por um dado de entrada na DNN influencia estatisticamente no comportamento do carro, abrangendo assim uma maior quantidade de erros quando testado.

Semelhante ao trabalho anterior, Mengshi Zhang (ZHANG et al., 2018) utiliza de uma técnica de transformação de imagens para a criação de um framework chamado DeepRoad para o treinamento de DNNs de carros autônomos e validar a corretude de suas decisões em cenários distintos. Porém, é proposto no trabalho a utilização do framework **UNIT** (LIU et al., 2017), para as transformações de imagens altamente realistas de um ambiente para outro.

Figura 3.4 – Arquitetura do framework DeepRoad



Fonte: Zhang et al. (2018)

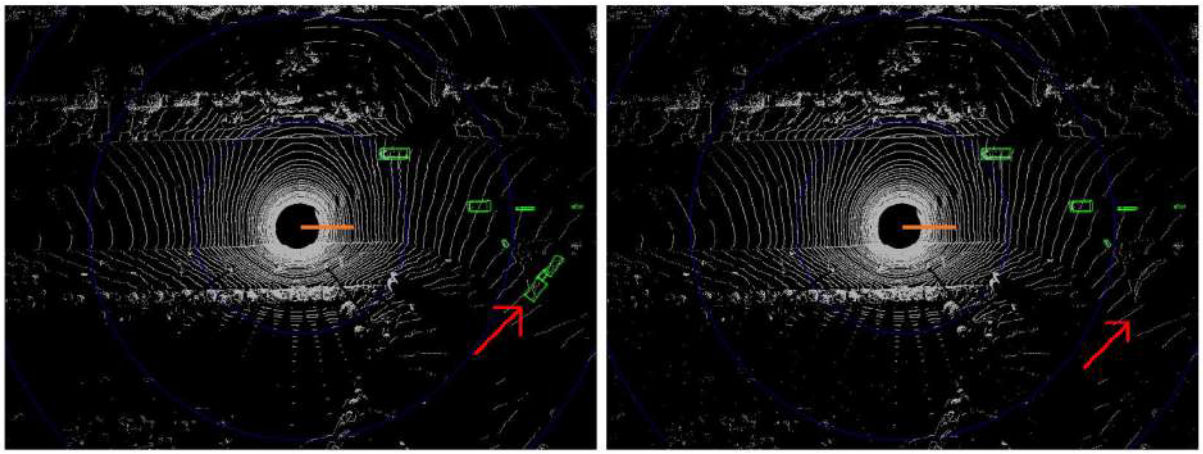
Como demonstrado na Figura 3.4, é utilizado um conjunto de imagens comuns para treinamento, com um ambiente favorável (ensolarado), e um conjunto de imagens que representa condições extremas (como neve, noite e chuva), e através do UNIT, é realizada a transformação de uma imagem comum, em diversas condições climáticas altamente realistas, como demonstrado na parte inferior da imagem. Todo este conjunto de imagens (as geradas e as sem alterações) são usadas como treinamento da DNN, e os Testes Metamórficos validam se o comportamento das duas corresponde igualmente, encontrando assim possíveis falhas em determinados tipos de condições climáticas. E vale destacar que foram encontrados centenas de comportamentos errôneos com esta técnica.

No trabalho de Zhi Q. Zhou (ZHOU; SUN, 2019) vemos a utilização da abordagem de Testes Metamórficos no módulo de percepção de obstáculos do sistema Apolo[3], no qual foi testado o subsistema de percepção de obstáculos por **LiDAR**. Para isso foi adicionado algum

[3] <http://apollo.auto/platform/perception.html>

ruído (pontos aleatórios) aos dados de entrada informados pelo LiDAR, porém fora da **ROI** (Região de interesse que define a área dirigível para o carro) de detecção, o que não deveria influenciar na detecção dos objetos dentro da ROI.

Figura 3.5 – Nuvem de pontos com ruído demonstrando falha na detecção de objetos



Fonte: Zhou e Sun (2019)

Como demonstrado na Figura 3.5, ao adicionarmos alguns pontos aleatórios aos dados de entrada, o módulo de percepção falha ao detectar os objetos que foram detectados previamente na primeira imagem, com a entrada de dados inalterada. Foi encontrada também uma relação direta entre a quantidade de pontos aleatórios adicionados e o número de detecções falhas pelo sistema.

3.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou e dissertou sobre as estratégias de seleção e de busca definidas para se encontrar os trabalhos relacionados, além de discorrer sobre estes trabalhos, apresentando então outras aplicações já existentes de Testes Metamórficos em sistemas de carros autônomos e seus resultados.

Através dos trabalhos abordados neste capítulo e da estratégia de busca utilizada, é possível notar que apesar da abordagem de Testes Metamórficos ter sido apresentada por Chen em 1998, a sua utilização em sistemas autônomos tem sido algo bastante recente, além de ter se mostrado bastante eficaz na detecção de falhas, encontrando centenas de falhas apenas nos trabalhos apresentados.

Contudo, a quantidade de trabalhos que abordam este tema ainda é bastante pequena, e por tal motivo ainda é possível abordar diversas outras aplicações de Testes Metamórficos em sistemas autônomos, para garantir ainda mais a melhoria e confiabilidade destas aplicações.

Tal confiabilidade é bastante necessária para evoluirmos cada vez mais no meio de carros autônomos e prevenir tragédias como a citada na problemática deste trabalho.

4 METODOLOGIA

Este capítulo tem como objetivo demonstrar a metodologia utilizada durante o desenvolvimento deste trabalho, bem como os métodos de pesquisa realizados para o levantamento da problemática na seção 1.3, para a busca dos trabalhos relacionados apresentados no Capítulo 3 e a solução de proposta apresentada durante o Capítulo 5.

O levantamento da problemática foi realizado de maneira exploratória a partir dos artigos (CHEN et al., 2018; Kanewala; Yueh Chen, 2019; SEGURA et al., 2016; SEGURA et al., 2018; ZHOU; SUN, 2019). A partir do levantamento da problemática, foi realizado uma revisão bibliográfica para o mapeamento dos trabalhos relacionados.

Ao final da revisão dos trabalhos relacionados, foi construída a proposta dessa pesquisa, selecionando as ferramentas necessárias e como os passos da mesma serão executados.

A execução dos experimentos propostos se darão por meio de ciclos de design como descrito em Wieringa (2014).

4.1 DESIGN SCIENCE

A utilização do método de **Design Science Research** prepara pesquisas cujas saídas comumente se tratam de artefatos ou um conjunto de sugestões a respeito de um determinado cenário. A partir de um dado problema são então construídas as saídas ou sugestões a serem avaliadas para que assim possamos, por meio de contribuição, realizar algum tipo de melhoria no cenário proposto ou até mesmo modificá-lo para um cenário desejado (DRESCH et al., 2015).

A aplicação de *Design Science* em uma pesquisa é composta por iterações que se dispõem em alguns passos a serem seguidos. Ao final destes passos temos o que chamamos de um **Ciclo de Design**, composto por:

1. **Investigação do problema**, nesta etapa analisamos um problema para que possamos definir quais pontos devem ser melhorados e por quais motivos. Com isso podemos então preparar o *design* através de um maior aprendizado a respeito do problema a ser tratado.
2. **Design de tratamento** é onde são especificados os requisitos necessários para o tratamento do problema proposto e então são construídos um ou mais artefatos que possam tratar do problema. A proposta aplicada deve ter como objetivo melhorar o cenário do problema a ser tratado.

3. A **Validação do tratamento** é a última etapa do ciclo, onde avaliamos se os artefatos identificados a serem construídos conseguem produzir algum efeito positivo para com o cenário do problema. Esta avaliação é então realizada em contextos mais "sensíveis" para que possamos ter uma experiência de como funcionará ao aplicarmos esta proposta a um contexto real.

Ao finalizarmos o ciclo de design e obtermos seus resultados é possível então dar início a novos ciclos de design a partir dos anteriores, identificando outros problemas, construindo uma solução e a validando assim como nos demais ciclos.

Além disso, é possível dar continuidade ao ciclo de design através do ciclo de engenharia, que acrescenta duas outras tarefas responsáveis pela "**Implementação da Proposta**" e pela "**Avaliação da Proposta**", aplicando os artefatos construídos para um contexto de mundo real e então avaliando sua eficiência neste contexto, respectivamente.

Ao realizarmos o processo presente no ciclo de engenharia, concluindo todas as tarefas descritas anteriormente, teremos um artefato construído como tratamento para o problema aplicado a um ambiente de mundo real e a avaliação do seu impacto no contexto avaliado com relação aos objetivos dos *stakeholders*.

Quando tratamos de um trabalho como uma pesquisa numa abordagem como *design science* não temos a obrigação de realizar todas as etapas do ciclo de engenharia, nos focando apenas nos passos do ciclo de design descrito anteriormente (WIERINGA, 2014). A implementação da proposta e sua avaliação em um ambiente real são tarefas que podem ser realizadas após a conclusão da pesquisa, por não fazerem parte do escopo definido no trabalho.

Sendo assim, neste trabalho ocorrerão apenas os ciclos de *design*, utilizando os resultados de cada ciclo para auxiliar na compreensão do problema e na construção de um solução (ou proposta de solução) que busque responder a pergunta de pesquisa. Uma vez que o aprendizado sobre o problema e o desenvolvimento da solução ocorrem durante a execução dos ciclos, não é possível estimar quantos ciclos serão necessários.

Como apontado no capítulo 6, é realizado apenas 1 ciclo *design* neste trabalho, devido a limitações e problemas encontrados. De todo modo, tal ciclo nos proporcionou um conjunto de contribuições evidenciadas no mesmo capítulo. Além disto, é possível continuar de maneira direta este ciclo, por meio dos trabalhos futuros propostos.

5 CICLO DE DESIGN 1

Este capítulo tem o objetivo de descrever a proposta de solução para a questão levantada na problemática deste trabalho, por meio da utilização de um simulador para o desenvolvimento e testes de sistemas de carros autônomos chamado de **CARLA**. Devido à grande quantidade de subsistemas existentes em um carro autônomo, este trabalho terá um maior foco nas **Redes Neurais** utilizadas para a detecção de objetos e de tomadas de decisão baseada em visão.

5.1 INVESTIGAÇÃO DO PROBLEMA

Como demonstrado no trabalho de Zhou e Sun (2019), falhas neste tipo de detecção podem acabar por tomar vidas. Porém, ainda é um enorme desafio mapear que tipo de fatores teriam maior influência na causa destas falhas. Alguns desses desafios se dão pelo fato de quão custosa é a implementação de um sistema de carro autônomo com todos os sensores utilizados nas detecções de ambiente e objetos para as tomadas de decisão. Além do custo, à complicações de se treinar e testar estes sistemas em ambientes reais acaba se mostrando como um outro problema. Estes desafios acabam por limitar as pesquisas neste meio.

5.2 DESIGN DA PROPOSTA

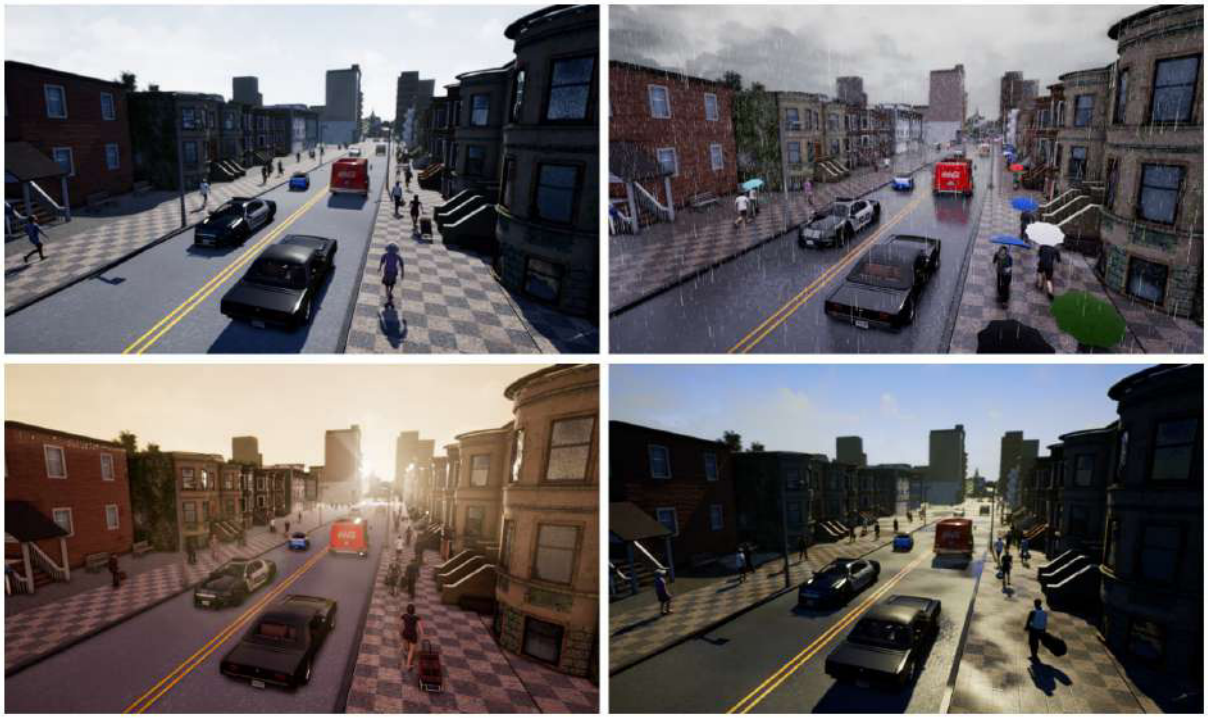
Para contornar esses desafios de custo e periculosidade optamos pela utilização de um sistema que simula virtualmente todo o ambiente de atuação de um carro autônomo, além de emular todos os sensores de detecção utilizados. Para isto, foi escolhido o sistema **CARLA**, um simulador para pesquisas em sistemas de carros autônomos (DOSOVITSKIY et al., 2017), como ambiente de realização dos testes que serão desenvolvidos.

5.2.1 Simulador CARLA

Diferente dos demais simuladores disponíveis até o momento de desenvolvimento deste trabalho, o CARLA disponibiliza ambientes urbanos complexos e com diferentes configurações climáticas, como demonstrado na Figura 5.1. Além disso, o sistema é de código aberto, o que possibilita a aplicação de mudanças com bastante facilidade. Devido a estes detalhes, sua utilização se mostra como uma ótima alternativa para a validação de sistemas de carros autônomos.

O CARLA disponibiliza uma interface de programação para aplicação (API) construída em *Python* para que um cliente possa controlar diversos fatores existentes no sistema, desde a quantidade de personagens controlados pela aplicação (NPCs), como carros, pedestres e

Figura 5.1 – Ambiente urbano disposto em quatro diferentes condições climáticas



Fonte: Dosovitskiy et al. (2017)

ciclistas existentes no mapa, como suas devidas localizações no mapa e até mesmo as condições climáticas, entre diversas outras funcionalidades. Todos estes fatores de manipulação nos permitem construir uma automatização dos testes para abranger o maior número de casos possíveis, o que possibilita aumentar a cobertura das possíveis falhas a serem encontradas.

5.2.2 Testes metamórficos no cenário proposto

Como foi dito anteriormente, devido a API disponibilizada pelo CARLA para a modificação de diversos fatores no ambiente do sistema, é possível construir uma automatização de testes. Porém, como estamos falando de testes em redes neurais nos deparamos com o Problema do Oráculo, como citado na subseção 2.1.2.1. Objetivando contornar este problema, iremos então utilizar da abordagem de **Testes Metamórficos**.

Para que possamos então realizar a aplicação desta abordagem, iremos nos dispor à um conjunto de rotas, cada uma com trajeto entre um ponto **inicial** e um ponto **final**, cujo ambiente seja controlável, para que possamos aplicar as relações metamórficas que estão descritas na sessão 5.2.6. Contudo, será necessário um sistema implementado para a automatização de veículos que possamos utilizar no CARLA e que disponha de um conjunto de dados para treinamento. Para isso, iremos utilizar de uma das abordagens criada e adaptada para o *CARLA Challenger*[3], chamada de *Learning By Cheating* (LBC) (CHEN et al., 2019).

[3] <https://carlachallenge.org/>

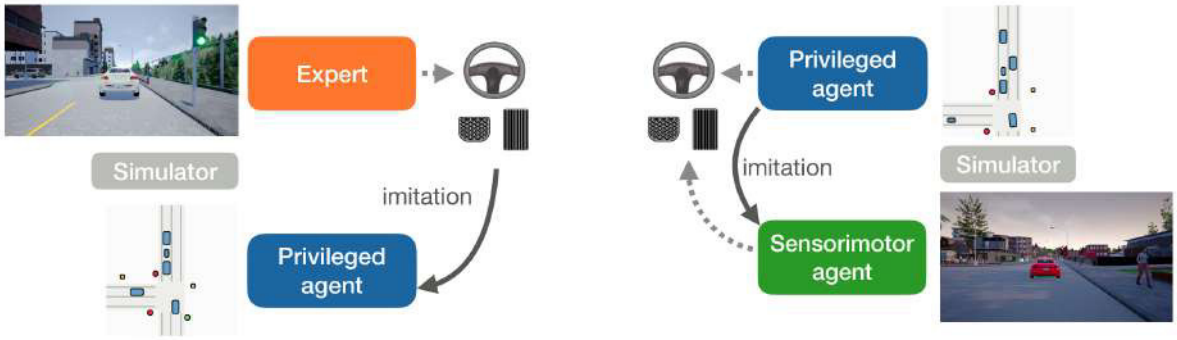
5.2.3 Sistema sob teste

O sistema então escolhido para a realização dos testes desta pesquisa foi o LBC, que se trata de uma abordagem criada para facilitar o processo de aprendizagem por imitação para sistemas de direção urbana baseada em visão (CHEN et al., 2019). Esta abordagem, por sua vez, foi desenvolvida completamente dentro do ambiente do CARLA, onde foram construídas diversas rotas de testes que são utilizadas no treinamento das redes neurais.

O conceito por traz do LBC consiste de quebrar o processo de aprendizagem por imitação das redes neurais utilizadas em dois passos. No primeiro passo, temos o treinamento de um agente privilegiado, o qual, durante o processo de aprendizagem nas chamadas "trajetórias de especialista", além do acesso as imagens do que acontece ao seu redor, ele irá "trapacear"ao ter acesso a informações a respeito do ambiente e de todos os outros participantes do trânsito, sejam eles pedestres, ciclista ou outros veículos (CHEN et al., 2019).

Após o treinamento deste agente privilegiado, é então iniciado o treinamento do agente sensorimotor, cujo comportamento será puramente baseado em visão, o qual passará por um processo de aprendizagem por imitações utilizando o agente privilegiado como "professor"nos trajetos de treinamento (CHEN et al., 2019). A representação deste processo pode ser observado na Figura 5.2.

Figura 5.2 – Processo de treinamento dos agentes privilegiado e sensorimotor na abordagem LBC



Fonte: Chen et al. (2019)

Após todo o processo de treinamento, o agente sensorimotor se torna o resultado final desta abordagem. Por tal motivo, um agente sensorimotor treinado será o sistema que iremos utilizar nesta pesquisa para aplicação dos testes com as relações metamórficas definidas na seção 5.2.6.

5.2.4 Ambiente de teste

Para que possamos realizar este experimento, algumas ferramentas e configurações se mostram necessárias. A principal ferramenta que será necessária é um ambiente CARLA executável, e para isso foi escolhida a versão 9.10.1[4], por se tratar de uma das versões mais recentes do CARLA até o momento de desenvolvimento deste experimento e ser uma das versões recomendadas para a execução do LBC[5].

Contudo, a execução de um ambiente CARLA é algo que demanda uma grande quantidade de recursos, principalmente recursos gráficos e computacionais. Por tal motivo, a escolha de uma máquina que possua uma unidade de processamento gráfico dedicada (UPG) e um unidade de processamento computacional (UPC) razoável torna-se algo inevitável.

5.2.4.1 Recursos Computacionais

Para que possamos suprir a necessidade aqui imposta por tais recursos, foi escolhido então, para a execução deste experimento, um Notebook com uma UPG *Geforce GTX1050* com 3GB de memória dedicada, uma UPC *Intel Core i5 9300H* e 16GB de memória RAM.

Além da necessidade de uma UPG para que possamos executar um ambiente CARLA sem muitas dificuldades, a UPG também se mostra extremamente necessária pelo fato das redes neurais existentes no LBC utilizarem também de uma grande quantidade de recursos gráficos.

A realização de uma mudança em tais fatores computacionais podem acabar por interferir, de maneira positiva ou negativa na replicação deste experimento. É sabido, entretanto, que a utilização de melhores recursos podem trazer resultados mais precisos.

5.2.4.2 Treinamento do Agente Sensorimotor

Os recursos computacionais se mostram como uma das partes fundamentais para um dos passos necessários para de fato executarmos o experimento aqui proposto, a qual se trata de todo o processo de treinamento da abordagem LBC para que tenhamos acesso a um agente sensorimotor funcional.

No repositório da implementação do LBC escolhida para este trabalho é possível encontrar um *dataset* de aproximadamente 9GB de dados, com mais de 70.000 (setenta mil) exemplos coletados em 75 rotas diferentes[6]. Este *dataset* é provido e recomendado para o processo de treinamento da abordagem LBC.

[4] <https://github.com/carla-simulator/carla/releases/tag/0.9.10.1>
[5] <https://github.com/bradyz/2020_CARLA_challenge>
[6] <https://github.com/bradyz/2020_CARLA_challenge#dataset>

Contudo, o processo de treinamento é bastante custoso em termos de recursos computacionais, bem além do que podemos oferecer com a máquina escolhida para a realização do experimento, além de ser bastante demorado e possuir diversas variáveis que podem ocasionar em um treinamento falho. Por tais motivos, neste trabalho iremos optar pelo uso de uma rede neural pré treinada com este *dataset* disponibilizado no repositório. O próprio LBC disponibiliza os arquivos de pesos para uma rede neural pré treinada e que pode ser encontrado no mesmo repositório.

O resultado obtido a partir deste treinamento prévio serão os pesos que serão utilizados pelos neurônios da rede neural do agente sensorimotor. Com isso, podemos então executar o agente sensorimotor treinado dentro do ambiente CARLA. Mas para que possamos realizar esta execução, será necessária uma rota de teste bem definida. Por tal motivo, selecionamos um conjunto de rotas que serão utilizadas neste experimento.

5.2.5 Rotas definidas

Para que se mostre possível a execução deste experimento, um requisito crucial deve ser atendido no planejamento dos casos de teste que iremos utilizar para a aplicação das RMs. Este requisito se trata de um conjunto de rotas bem definidas.

De tal forma, escolhemos arbitrariamente para este experimento 4 (quatro) entre 26 (vinte e seis) rotas de teste existentes e disponibilizadas no repositório do LBC. Por tanto, a estas quatro rotas doravante será atribuído o título de **conjunto de rotas**, e quando necessário, para que possamos nos referir de maneira específica a cada uma delas, iremos enumerá-las de maneira crescente entre 1 (um) e 4 (quatro).

Para cada uma das rotas, serão feitas quatro execuções. Duas delas em cenários onde não haverá nenhum tipo de alteração e outras duas com as alterações propostas nas RMs descritas na seção 5.2.6. As execuções sem alterações tratam-se dos nossos casos de teste originais, enquanto as demais se tratam dos casos de teste complementares.

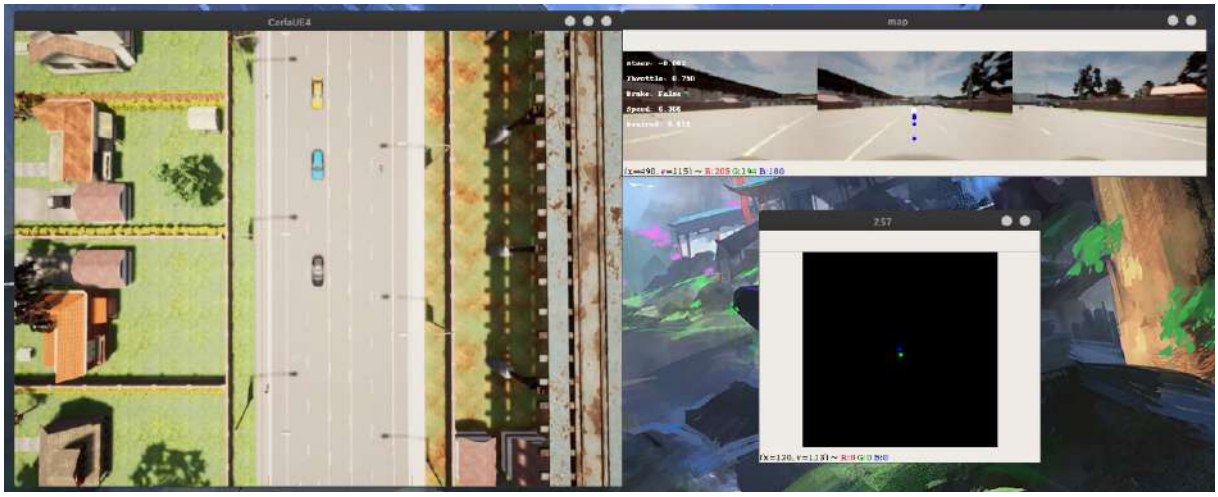
As múltiplas execuções de cada caso de teste se mostram necessárias para que tenhamos uma maior acurácia no momento de validação deste experimento, pois, devido ao enorme número de variáveis que envolve a simulação (pedestres, veículos, sinais de trânsito, etc), apenas uma execução torna-se insuficiente para uma validação do experimento.

5.2.5.1 Processo de execução de rotas

A implementação da execução das rotas dentro de um ambiente CARLA para que possamos executar um agente sensorimotor treinado também é disponibilizado no repositório do LBC em um *script Python* chamado de `leadboard_evaluator.py`. Este *script* tem

como objetivo iniciar a uma das rotas dentro do ambiente CARLA, com todas as variáveis necessárias, como pedestres, veículos e ciclistas, além de configurar um conjunto de visões para acompanhamento da execução do agente sensorimotor na rota selecionada, como pode ser visto na Figura 5.3.

Figura 5.3 – Execução em tempo real do agente sensorimotor treinado por meio do script `leadboard_evaluator.py`



Fonte: Registrado pelo autor

Após a conclusão da rota, seja pelo agente sensorimotor alcançar o seu destino ou por esgotar o tempo que é dado pela rota para a sua conclusão, este *script* irá nos retornar um arquivo *JSON*[7] com um conjunto de dados sumarizados a respeito da execução que acabou de ocorrer. Esta sumarização inclui informações a respeito de infrações de transito, colisões e médias referentes a execução da rota.

Por meio deste arquivo, após cada execução, podemos então realizar um processo de extração de dados para que tenhamos acesso a algumas variáveis que serão utilizadas para avaliar a adequação as relações metamórficas propostas por esta pesquisa.

5.2.6 Relações metamórficas propostas

Para facilitar a compreensão deste trabalho além de trazer uma melhor organização, a partir deste momento as relações metamórficas apresentadas irão seguir uma padronização definida por Segura et al. (2017).

- **RM-1 Influência da chuva em colisões com veículos**

  SE Um carro causa um número X de colisões com veículos em um cenário com clima ensolarado

[7] <https://www.json.org/json-en.html>

ENTÃO O mesmo carro deve manter o número de colisões com veículos igual ou menor do que X no mesmo cenário, independente da adição de chuva no clima.

- **RM-2 Influência da chuva em colisões com o cenário**

  SE Um carro causa um número X de colisões com o cenário em um cenário com clima ensolarado

  ENTÃO O mesmo carro deve manter o número de colisões com o cenário igual ou menor do que X no mesmo cenário, independente da adição de chuva no clima.

- **RM-3 Influência da chuva em colisões com o pedestres**

  SE Um carro causa um número X de colisões com o pedestres em um cenário com clima ensolarado

  ENTÃO O mesmo carro deve manter o número de colisões com pedestres igual ou menor do que X no mesmo cenário, independente da adição de chuva no clima.

- **RM-4 Influência da chuva em infrações de sinal vermelho**

  SE Um carro causa um número X de infrações por passar sinais vermelhos em um cenário com clima ensolarado

  ENTÃO O mesmo carro deve manter o número de infrações por passar em sinais vermelhos igual ou menor do que X no mesmo cenário, independente da adição de chuva no clima.

- **RM-5 Influência da chuva em infrações de placa de pare**

  SE Um carro causa um número X de infrações por passar placas de pare em um cenário com clima ensolarado

  ENTÃO O mesmo carro deve manter o número de infrações por placas de pare igual ou menor do que X no mesmo cenário, independente da adição de chuva no clima.

- **RM-6 Influência da chuva no seguimento da rota**

  SE Um carro foge da rota definida um número X de vezes em um cenário com clima ensolarado

  ENTÃO O mesmo carro deve manter o número de fugas da rota igual ou menor do que X no mesmo cenário, independente da adição de chuva no clima.

A partir das violação que ocorrerem na aplicação das RMs aqui propostas, será possível realizar uma análise quantitativa da influência de chuva neste experimento. Cada uma das das RMs aqui definidas será executado no nosso conjunto de rotas para cada caso de teste complementar que for executado.

Devido a flexibilidade que o CARLA nos propõe, a replicação deste experimento com outras redes neurais pode facilmente ser realizado, e de tal maneira, o reuso destas RMs para outros cenários torna-se viável. Porém, para isso, é necessário que possamos encapsular o código que será responsável pela a aplicação das modificações de ambiente descritas nas RMs.

5.2.7 Adaptações feitas no ambiente de simulação

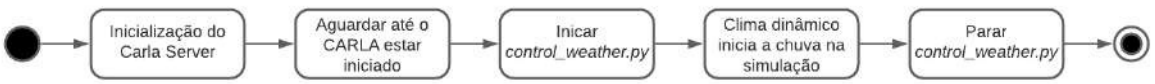
Para que possamos diferir as execuções dos casos de teste originais e complementares, devemos nos ater a algumas adaptações que devem ser feitas no ambiente CARLA durante a execução dos testes aqui propostos. Na seção 5.2.6 definimos que a principal alteração a ser aplicada pelas RMs deste ciclo de design será a **influência da chuva** na execução das rotas, e para isso, devemos construir um controle de chuva que simule e se aproxime de um ambiente de mundo real.

5.2.7.1 Um controle de chuva dinâmico e pseudo aleatório

Como dito na seção 5.2.1, o CARLA dispõe de uma API *Python* que nos permite ter um enorme controle sobre o mundo da simulação. Parte deste controle se encontra nas variáveis climáticas, que simula com alta competência um clima de mundo real. Por meio delas, podemos controlar variáveis como o vento, a densidade e quantidade de nuvens, a umidade do ar e a quantidade de chuva no cenário.

Para que possamos ter um alto controle sobre o clima de chuva que queremos construir para este experimento, foi criado então um *script Python* chamado de *control_weather.py*, que tem como objetivo, após ser iniciado, se conectar, por meio desta API disponibilizada pelo CARLA, ao mundo da simulação, que já deve estar iniciado, para então realizar a alteração climática. Este fluxo pode ser observado na Figura 5.4.

Figura 5.4 – Fluxo de execução do *script control_weather* no ambiente de simulação CARLA



Fonte: Elaborado pelo autor

Durante a execução do *script*, um conjunto de passos é seguido para que possamos ter uma simulação pseudo aleatória da chuva no nosso ambiente. Estes passos são:

- **Passo 1** A densidade e quantidade de nuvens cresce de maneira gradativa até cobrirem todo o céu, indo de 0% a 100% no ambiente de simulação.

- **Passo 2** De maneira dependente da quantidade de nuvens, o vento começa a aumentar no ambiente de simulação, indo de 0% a 100%.
- **Passo 3** Após a quantidade de nuvens no céu ultrapassar 50% a chuva se inicia de maneira leve e gradativa. Conforme as nuvens tomam o céu, a chuva vai crescendo seu valor na simulação, indo de 0% a 100%.
- **Passo 4** De maneira dependente da quantidade de nuvens e de chuva na simulação, a umidade do ar começa a crescer, indo de 0% a 100%. O 100% só é atingido quando a chuva e as nuvens estão em seu valor máximo.
- **Passo 5** A chuva entra em um laço pseudo aleatório, onde os valores de chuva, vento, nuvens e umidade começa a variar para cima até que atinjam 100%. Após isso, os mesmo valores começam a variar para baixo até que atinjam 50% (exceto pelo valor de chuva, que pode variar até 20%).

Tais passos fazem com que tenhamos um cenário onde a chuva irá variar entre picos e pontos onde a chuva estará quase cessando. Porém, para que a chuva cesse de fato é necessário parar o *script* de maneira explícita.

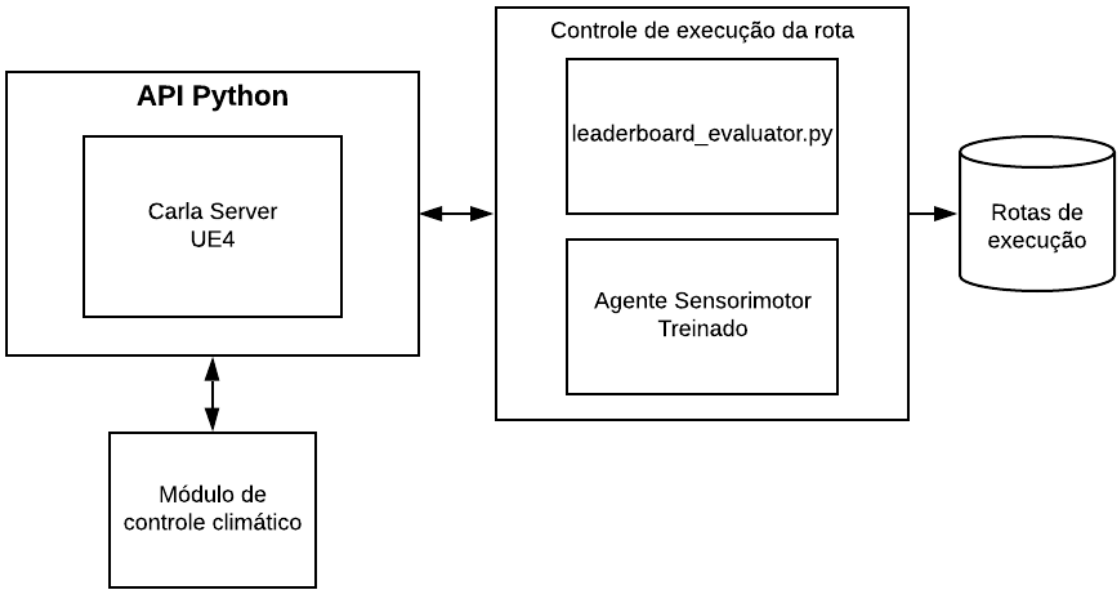
Pelo fato do *script control_weather.py* se conectar ao ambiente de simulação por meio da API disponibilizada pelo CARLA, o mesmo pode ser tratado como um módulo de controle climático, onde pode ser utilizado independente da rede neural que estamos testando ou até mesmo do conjunto de rotas escolhidas para a realização dos testes. Este processo de modularização pode ser observado na Figura 5.5, onde temos nosso módulo de controle climático conectado ao CARLA, de maneira independente dos demais componentes.

5.3 VALIDAÇÃO DA PROPOSTA

Como última etapa deste ciclo de design, foi realizada então a execução do experimento para que possamos validar a proposta desta pesquisa. Para auxiliar nesta processo iremos utilizar de um conjunto de perguntas que irão servir de validador final para este ciclo de design e de ponto de partida para possíveis novos ciclos.

- **P1** Como as relações metamórficas propostas neste ciclo podem auxiliar na descoberta de falhas para sistemas de carros autônomos?
- **P2** Como as relações metamórficas posposta neste ciclo podem apontar fatores que possuem maior influência na ocorrência de falhas em sistemas de carros autônomos?

Figura 5.5 – Arquitetura do processo de execução do módulo de controle de rotas de maneira independente ao módulo de controle climático



Fonte: Elaborado pelo autor

Com isso, podemos realizar processo de extração dos dados ao final da execução dos casos de teste propostos para este ciclo. Cada caso de teste, por sua vez, nos traz duas execuções em uma rota. Sejam estas, execuções de casos de teste originas ou complementares. Contudo, cada caso de teste irá nos dar apenas um arquivo *JSON* com os dados das duas execuções. Será então, por meio destes arquivos, que iremos nos ater aos resultados deste experimento.

5.3.1 Resultados

Todas as relações metamórficas propostas neste ciclo de design foram aplicadas para o conjunto de rotas definido neste mesmo ciclo. Toda RM foi aplicada para cada uma das duas execuções presentes em cada caso de teste proposto pelas 4 rotas do conjunto de teste. Isso resulta em **16 casos de teste** executados, onde temos **8 casos de teste originais** e **8 casos de teste complementares**.

Para a exposição destes resultados optamos por tratar as RMs como base de análise dos dados, como pode ser observado a seguir. Devido ao fato das RMs propostas neste ciclo seguirem o mesmo princípio, que se trata do fator de alteração climática, iremos nos ater, durante suas descrições, apenas às variáveis que estamos analisando em cada RM.

Para facilitar a compreensão, a partir deste ponto até o final desta seção, iremos nos

referir aos casos de teste originais como CT_o e aos casos de teste complementares por CT_c.

RM-1 Influência da chuva em colisões com veículos

Esta relação especifica que a adição de alteração climática por meio de chuva, em um cenário onde o resultado da variável *collisions_with_vehicles* (Colisões com veículos) dado por um número X, deve se manter igual ou menor na execução do CT_c. Do contrário, podemos afirmar que houve uma violação da relação. O resultado da execução dos testes da RM-1 aqui proposta podem ser observados na Tabela 5.1.

Tabela 5.1 – Resultados coletados para a RM-1

Rota	N° de colisões CT_o	N° de colisões CT_c	Violação
1 - execução 1	0	1	SIM
1 - execução 2	1	2	SIM
2 - execução 1	0	4	SIM
2 - execução 2	0	0	NÃO
3 - execução 1	0	0	NÃO
3 - execução 2	0	0	NÃO
4 - execução 1	1	0	NÃO
4 - execução 2	0	0	NÃO

Fonte: Elaborado pelo autor

RM-2 Influência da chuva em colisões com o cenário

Esta relação especifica que a adição de alteração climática por meio de chuva, em um cenário onde o resultado da variável *collisions_with_layout* (Colisões com o cenário) dado por um número X, deve se manter igual ou menor na execução do CT_c. Do contrário, podemos afirmar que houve uma violação da relação. O resultado da execução dos testes da RM-2 aqui proposta podem ser observados na Tabela 5.2

Tabela 5.2 – Resultados coletados para a RM-2

Rota	N° de colisões CT_o	N° de colisões CT_c	Violação
1 - execução 1	1	2	SIM
1 - execução 2	1	1	NÃO
2 - execução 1	0	1	SIM
2 - execução 2	0	0	NÃO
3 - execução 1	1	0	NÃO
3 - execução 2	0	0	NÃO
4 - execução 1	2	0	NÃO
4 - execução 2	0	0	NÃO

Fonte: Elaborado pelo autor

RM-3 Influência da chuva em colisões com o pedestres

Esta relação especifica que a adição de alteração climática por meio de chuva, em um cenário onde o resultado da variável *collisions_with_pedestrians* (Colisões com pedestres) dado por um número *X*, deve se manter igual ou menor na execução do CT_c. Do contrário, podemos afirmar que houve uma violação da relação. O resultado da execução dos testes da RM-3 aqui proposta podem ser observados na Tabela 5.3

Tabela 5.3 – Resultados coletados para a RM-3

Rota	Nº de colisões CT_o	Nº de colisões CT_c	Violação
1 - execução 1	0	0	NÃO
1 - execução 2	0	0	NÃO
2 - execução 1	0	0	NÃO
2 - execução 2	0	0	NÃO
3 - execução 1	0	0	NÃO
3 - execução 2	0	0	NÃO
4 - execução 1	0	0	NÃO
4 - execução 2	0	0	NÃO

Fonte: Elaborado pelo autor

RM-4 Influência da chuva em infrações de sinal vermelho

Esta relação especifica que a adição de alteração climática por meio de chuva, em um cenário onde o resultado da variável *red_lights_infractions* (Infrações de sinal vermelho) dado por um número *X*, deve se manter igual ou menor na execução do CT_c. Do contrário, podemos afirmar que houve uma violação da relação. O resultado da execução dos testes da RM-4 aqui proposta podem ser observados na Tabela 5.4

Tabela 5.4 – Resultados coletados para a RM-4

Rota	Nº de colisões CT_o	Nº de colisões CT_c	Violação
1 - execução 1	0	1	SIM
1 - execução 2	1	0	NÃO
2 - execução 1	2	2	NÃO
2 - execução 2	2	1	NÃO
3 - execução 1	2	0	NÃO
3 - execução 2	0	0	NÃO
4 - execução 1	3	0	NÃO
4 - execução 2	0	0	NÃO

Fonte: Elaborado pelo autor

RM-5 Influência da chuva em infrações de placa de pare

Esta relação especifica que a adição de alteração climática por meio de chuva, em um cenário onde o resultado da variável *stop_sign_infractions* (infrações de placa de pare) dado por um número X, deve se manter igual ou menor na execução do CT_c. Do contrário, podemos afirmar que houve uma violação da relação. O resultado da execução dos testes da RM-5 aqui proposta podem ser observados na Tabela 5.5

Tabela 5.5 – Resultados coletados para a RM-5

Rota	Nº de colisões CT_o	Nº de colisões CT_c	Violação
1 - execução 1	0	0	NÃO
1 - execução 2	0	0	NÃO
2 - execução 1	1	0	NÃO
2 - execução 2	1	0	NÃO
3 - execução 1	2	1	NÃO
3 - execução 2	0	1	SIM
4 - execução 1	1	0	NÃO
4 - execução 2	0	0	NÃO

Fonte: Elaborado pelo autor

RM-6 Influência da chuva no seguimento da rota

Esta relação especifica que a adição de alteração climática por meio de chuva, em um cenário onde o resultado da variável *route_deviations* (desvios da rota) dado por um número X, deve se manter igual ou menor na execução do CT_c. Do contrário, podemos afirmar que houve uma violação da relação. O resultado da execução dos testes da RM-6 aqui proposta podem ser observados na Tabela 5.6

Tabela 5.6 – Resultados coletados para a RM-6

Rota	Nº de desvios CT_o	Nº de desvios CT_c	Violação
1 - execução 1	0	0	NÃO
1 - execução 2	0	0	NÃO
2 - execução 1	0	0	NÃO
2 - execução 2	0	0	NÃO
3 - execução 1	0	0	NÃO
3 - execução 2	0	0	NÃO
4 - execução 1	0	0	NÃO
4 - execução 2	0	0	NÃO

Fonte: Elaborado pelo autor

5.4 AVALIAÇÃO

Por meio dos dados extraídos após a execução dos testes metamórficos propostos por este ciclo de design, iremos nos focar na tentativa de responder as perguntas levantadas na seção 5.3.

P1: Como as relações metamórficas propostas neste ciclo podem auxiliar na descoberta de falhas para sistemas de carros autônomos?

A utilização de relações metamórficas para exposição de falhas em sistemas de carros autônomos se mostrou algo bastante eficiente no trabalho de Zhou e Sun (2019). Dezenas de falhas podem ser detectadas com apenas uma única relação metamórfica. Entretanto, cenários de simulação para carros autônomos envolvem um alto número de variáveis, tais como os diversos veículos e pedestres presentes no ambiente.

Não é viável a utilização de um ambiente totalmente controlado, pois estaríamos fugindo dos cenários de uso típico de um sistema de carro autônomo, o qual também é influenciado por um altíssimo número de variáveis. Caso esta abordagem fosse utilizada, com um ambiente totalmente controlado, teríamos uma maior facilidade de apontar falhas específicas. Contudo, presenciaríamos o caos ao implantar este sistema em um ambiente realista. Por tais motivos, as relações metamórficas aqui apresentadas funcionam como uma espécie de gatilho para que possamos encontrar com mais facilidade a existência de falhas.

Como demonstrado nos resultados expostos através a execução dos testes no sistema implementado da abordagem LBC de Chen et al. (2019), algumas das relações metamórficas não apresentaram uma relevante quantidade de violações durante suas execuções. Entretanto, a relação **RM-1** apresentou um total de 3 execuções onde houve violação. Isso equivale a **37.5% de violação em todas as execuções** e a **100% de violação nas execuções da rota 1**.

Tais dados nos mostram que a **RM-1** expôs algum tipo de falha existente na execução do agente sensorimotor treinado no percurso da rota 1. A partir deste ponto, pode ser aplicado o mesmo conjunto de RMs na rota 1, porém, com um número maior de execuções, para que seja possível ter um quadro mais amplo a respeito das possíveis falhas existentes na execução deste percurso.

O fato das relações metamórficas aqui propostas estarem atreladas diretamente às variáveis de execução do CARLA, nos traz a possibilidade de reusá-las em qualquer rede neural implementada para o controle de um carro autônomo que também for implementada no CARLA. Além disso, pelo fato de estarmos executando este experimento em um cenário simulado, é

possível também reusar estar relações metamórficas em cenários de mundo real, apenas se atendo às variáveis existentes em um cenário desta natureza.

P2: Como as relações metamórficas propostas neste ciclo podem apontar fatores que possuem maior influência na ocorrência de falhas em sistemas de carros autônomos?

Como demonstrado na pesquisa de Zhou e Sun (2019), devido a ampla quantidade de variáveis que influenciam em uma execução de sistema de carro autônomo, fatores, que poderiam ser considerados de baixa relevância no processo de execução podem acabar por se mostrarem como algumas das principais influências para a causa de falhas neste tipo de sistema.

A exploração destes fatores aleatórios ou pseudo aleatórios podem nos proporcionar uma enorme quantidade de RMs para experimentar, principalmente em ambientes simulados como é o caso do CARLA.

Pelas RMs propostas neste ciclo, podemos então avaliar qual a o nível de influência da chuva na ocorrência de falhas. Para cada uma das **8 execuções** que foram realizadas, foram aplicadas **6 RMs**. Isso nos proporciona um total de 48 dados para análise, dos quais 7 tiveram algum tipo de violação. Esto equivale a **14,6% das execuções violadas por influência de chuva**.

Por meio desta análise, podemos inferir que existe uma influência da chuva na causa de falhas para o cenário proposto, mesmo que seja vista como uma pequena influência. Contudo, em sistemas críticos, nos quais temos risco de vida envolvidos, toda pequena influência deve ser levada em conta. Esta influência aqui exposta, poderia vir a ser o causador de um acidente capaz de tirar a vida de alguém.

Seguindo o princípio utilizado das relações metamórficas aqui propostas para a exploração da influência por fatores climáticos de chuva, é possível também, explorar outros fatores dentro do CARLA por meio da API *Python*. Fatores como a aplicação de névoa em um trajeto, diferenciação entre trajetos noturnos e diurnos, entre diversos outros. Se aplicarmos o conjunto de RMs aqui propostas, mudando o fator climático principal, como demonstrado por este experimento, é possível sim descobrir o nível de influência que estes fatores podem ter na ocorrência de falhas em sistemas de carros autônomos.

6 CONSIDERAÇÕES FINAIS

Durante o decorrer deste trabalho foi apresentada uma contextualização a respeito de Testes metamórficos, suas aplicações e carros autônomos. Foi também demonstrada a periculosidade de uma falha em um sistema de carro autônomo e como ocorreu a primeira fatalidade em seu uso. Com isso foi então levantada a pergunta de pesquisa deste trabalho.

Após isso, foi apresentada a fundamentação teórica deste trabalho, na qual foi dissertado a respeito de conceitos e teorias necessárias para a total compreensão deste trabalho, como Testes de Software e seus níveis, Oráculo e o Problema do Oráculo. Logo após foram descritos conceitos a respeito de Testes Metamórficos, Relações Metamórficas e o processo de Testes Metamórficos. Para finalizar a fundamentação teórica foi apresentado o básico a respeito de Carros Autônomos e Redes Neurais para detecção de objetos.

Durante o decorrer do trabalho foi realizada uma busca pelos trabalhos relacionados que estão dispostos no Capítulo 3, onde é descrito quais os critérios de seleção para os trabalhos relacionados e qual foi o processo de busca realizado para decidir quais trabalhos passariam pelos critérios de seleção. Foi então dissertado sobre os trabalhos escolhidos e discorrido a respeito da atualidade do tema e da pequena quantidade de trabalhos existentes nesta vertente.

Foi então descrita como foi e será realizada a metodologia deste trabalho, apresentando o uso da revisão bibliográfica de maneira exploratória para o levantamento da problemática deste trabalho além de apresentar a abordagem de Design Science como metodologia a ser utilizada na realização dos experimentos.

Após isso foi apresentada e executada uma proposta para a solução da pergunta de pesquisa levantada na Problemática, que se dá pela utilização de um simulador de sistemas de carros autônomos, o CARLA, como ferramenta para o desenvolvimento e aplicação do processo de testes metamórficos utilizando as relações metamórficas propostas, e as aplicando na implementação da abordagem LBC, para que assim seja possível apontar fatores que possam ter influência na causa de falhas.

Vale destacar que este trabalho apresenta apenas uma **proposta de relações metamórficas para serem aplicadas em sistemas de carros autônomos**, sendo possível ainda propor diversas outras relações metamórficas para a validação destes sistemas. Além disso, as relações metamórficas propostas para a validação destes sistemas podem também ser aplicadas em ambientes de mundo real, o que aumenta ainda mais a cobertura de falhas possíveis de serem encontradas por estas relações.

6.1 DIFICULDADES ENCONTRADAS

Durante todo o decorrer do desenvolvimento do experimento demonstrado no ciclo de design proposto no Capítulo 5, uma enorme gama de dificuldades foram se expondo pelo caminho. Tais dificuldades acabaram por interferir bastante no resultado final deste experimento. Nesta seção, iremos abordar algumas destas dificuldades.

6.1.1 Limitações computacionais

Como apresentado na subseção 5.2.4.1, foi necessário um conjunto relativamente elevado recursos computacionais para a realização deste experimento. Contudo, durante todo o decorrer do experimento, o uso destes recursos se mostrava quase sempre como um empecilho, pois os mesmos não estavam se mostrando suficientes.

O principal ponto onde estes recursos de mostraram como um problema grave de fato para a pesquisa foi durante o processo de treinamento do LBC. Os recursos de UPG simplesmente não eram suficientes durante o processo de treinamento, resultando sempre em falhas durante o mesmo. Por tal motivo, foi optado pelo uso de uma rede neural pré treinada, e que felizmente, era disponibilizado também pelo LBC.

Com isso, a realização do experimento foi possível, contudo, ainda que realizado, durante todo o experimento foi necessário a redução da qualidade gráfica na execução do CARLA, pois era notório que se executássemos o CARLA em sua qualidade padrão junto com o agente sensorimotor, a UPG mais uma vez não seria capaz de lidar com a demanda exigida.

Mensurar uma configuração mínima para a realização deste experimento é complicado, pois a quantidade de recursos consumidos pela rede neural pode variar drasticamente de máquina para máquina. Entretanto, é possível analisar a configuração mínima para o uso do CARLA em sua documentação[8]. Como é possível se observar por meio desta configuração apontada pelo CARLA, o hardware utilizado neste experimento ainda consegue ser inferior a configuração mínima sugerida.

6.1.2 Uma implementação prática

A escolha da implementação do LBC como sistema sob testes não foi feita por acaso. Diversas opções de sistemas para teste foram exploradas durante o decorrer deste trabalho. Contudo, praticamente todas as opções exploradas, exceto pelo LBC, apresentaram diversos problemas que nos impediram de utilizá-las neste experimento.

Várias alternativas de implementação encontradas não estavam bem encapsuladas

[8] <https://carla.readthedocs.io/en/latest/build_faq/#recommended-hardware-to-run-carla>

(em quesitos de especificação de dependências) para uso, e isso resultava em um enorme desastre de dependências não resolvidas para a execução do sistema. Mesmo tentando resolver as dependências manualmente, uma por uma, ainda se encontravam diversos problemas pelo fato das versões para estas dependências não estarem bem especificadas.

Devido ao fato desta área de desenvolvimento ser relativamente recente, é compreensível que os desenvolvedores ainda estejam se adaptando às melhores maneiras de encapsular este tipo de implementação. Algumas das implementações até foram encontradas com encapsulamento por meio de *containers* Docker[9]. Porém, a utilização de Docker para este tipo de aplicação mostra-se complexa, o *container* docker teria de ter acesso também à UPG, e a realização disso em um *kernel* Linux se mostra bastante problemática, pois em muitos casos, sistemas de *kernel* linux podem apresentar complicações para reconhecer o hardware da UPG para que seja possível utilizá-lo.

6.1.3 Complicações de tempo

É de nosso conhecimento que quanto maior a quantidade dados que possamos analisar, maior será a precisão do experimento. Isto não é diferente em experimentos com relações metamórficas. Porém, quando tratamos de experimentos com sistemas que envolvem redes neurais, uma grande complicação nos assola, e ela se trata do tempo.

O processo de treinamento para uma rede neural, dependendo da quantidade de dados utilizados durante o processo, pode vir a durar horas, ou até mesmo algumas dezenas de horas. Só este processo de treinamento por si só já é uma complicação enorme. Repetir este processo inúmeras vezes onde cada uma delas falha pela metade, se expõe como uma grande quantidade de tempo desperdiçado.

Além disso, o fato de estarmos tentando simular um ambiente próximo o da realidade para que estes testes sejam úteis na implantação de um sistema em ambiente real faz com que a execução das rotas de teste sejam demoradas igual uma rota de mundo real é. A quantidade de execuções realizadas antes do experimento final, só para que pudéssemos analisar quais tipos de dados seriam utilizados nas RMs já se convertem em dezenas de horas de execuções.

Com certeza se pudéssemos acrescentar mais execuções ao experimento proposto, o resultado seria bem mais preciso. Contudo, isso demandaria horas e mais horas somente de execuções. Quando se trabalha com este tipo de sistema, o tempo é um fator crucial.

[9] <https://www.docker.com/>

6.2 CONTRIBUIÇÕES PARA ACADEMIA

Como apontado no capítulo 3, a quantidade de pesquisas relacionadas a aplicação de testes metamórficos em sistemas de carros autônomos ainda é um assunto bastante recente. Em específico, a utilização de testes metamórficos com o auxílio de ambientes simulados como o CARLA se mostra quase inexistente até a data de desenvolvimento desta pesquisa.

Além disto, as ideias aqui apontadas e as relações metamórficas propostas podem servir de um ponto de partida para outras pesquisas, onde tais RMs podem ser derivadas para a criação de novas RMs.

A exploração do uso de ambientes simulados pode servir como um grande auxílio para futuras pesquisas envolvendo testes, que poderiam ser bastante custosas caso aplicadas em um ambiente de mundo real. Esperamos que, por meio desta pesquisa, seja possível notar o quão benéfico pode ser a aplicação de testes para sistemas de carros autônomos neste tipo de ambiente.

6.3 CONTRIBUIÇÕES PARA INDÚSTRIA

Por meio do conjunto de relações metamórficas aqui apresentado, é possível realizar uma replicação destas relações para aplicações de carros autônomos já consolidados no mercado, seja por meio do CARLA ou por meio da replicação destas RMs em outros ambientes.

Caso seja optado pelo uso destas relações metamórficas dentro do CARLA, é possível encontrar também o *script control_weather.py* hospedado em um repositório no github[9]. Além disso, é também possível encontrar o conjunto de rotas disponibilizado pelo LBC no mesmo repositório. Com isso, empresas que desejam entrar no ramo de desenvolvimento de sistemas para carros autônomos também pode utilizar desta pesquisa como meio de validação para seus sistemas.

6.4 LIMITAÇÕES E AMEAÇAS À VALIDADE

Como mencionado na seção 6.1, diversas limitações referentes a hardware se mostraram como um grande obstáculo para a realização desta pesquisa. Estes fatores de hardware podem também influenciar negativamente no funcionamento de processamento da rede neural, o que pode acabar por trazer algum tipo de comprometimento para a acurácia do experimento.

Além deste fator, devido à enorme quantidade de variáveis que ronda um ambiente de execução para um sistema de carro autônomo, é extremamente difícil isolar e apontar

[9] <https://github.com/MailsonD/Carla_MetamorphicTesting_LBC>

especificamente onde se encontra uma falha. Para isto, uma análise bem mais aprofundada e cautelosa é requerida.

Devido também a grande quantidade de dificuldades encontradas no caminho para o desenvolvimento desta pesquisa, a criação de novos ciclos de design que dessem continuidade ao ciclo inicial proposto acabaram por serem descartados. Por tais motivos, diversos outros fatores climáticos acabaram por não serem explorados. Contudo, isto abre possibilidades que podemos explorar em trabalhos futuros.

6.5 TRABALHOS FUTUROS

Devido a escolha da metodologia de Design Science Research para esta pesquisa, o uso do modelo de desenvolvimento por meio de ciclos de design nos dá a liberdade da criação de um ciclo que pode ser continuado a partir de uma nova pesquisa que também siga esta mesma metodologia, de maneira fácil e prática.

Com o fim do ciclo de design apresentado no capítulo 5, podemos analisar o nível de influência de chuva no ambiente proposto, e como essa influência nos ajuda a encontrar falhas. Contudo, durante o desenvolvimento desta pesquisa, alguns outros fatores existentes dentro do CARLA foram levados em conta para serem explorados em pesquisas futuras a respeito de sua influência na causa de falhas ou em outros aspectos. Alguns deles se encontram na lista abaixo.

- A influência da aplicação de névoa em uma rota
- A influência da aplicação de um clima noturno em uma rota
- A influência da variação de vento em uma rota
- A influência da umidade do ar em uma rota
- A influência da quantidade de pedestres existentes na calçada para com a rota
- A influência de um clima com altíssima luminosidade em uma rota

Tais fatores aqui levantados pode ser convertidos em dezenas ou até mesmo centenas de relações metamórficas para a descoberta de falhas em sistemas de carros autônomos. Além disto, como mencionado anteriormente, as relações metamórficas aqui propostas nesta pesquisa podem ser derivadas em inúmeras outras relações.

Outro ponto que pode ser interessante de se analisar para aumentar ainda mais a gama de possíveis execuções para com as relações metamórficas propostas, trata-se da inversão das

rotas definidas para as execuções. Com isso, seria possível dobrar o número de possibilidades durante a execução do experimento.

Por fim, caso a rede neural em análise seja uma baseada completamente em visão, como é o caso do LBC, é possível experimentar variações com os veículos existentes no cenário, para analisar como o agente treinado se comporta a isso. Variações como uma grande quantidade de modelos de veículos diferentes, ou até mesmo diferenciação na cor destes veículos, para que seja possível analisar se tais fatores também possuem algum de influência em falhas.

Em sistemas críticos de grande porte, como é o caso de uma rede neural para controle de sistemas de carros autônomos, são nos detalhes que consideramos irrelevantes que se escondem as falhas que podem vir a causar uma fatalidade.

REFERÊNCIAS

AGGARWAL, C. C. Neural networks and deep learning. **Springer**, Springer, v. 10, p. 978–3, 2018.

AMMANN, P.; OFFUTT, J. **Introduction to Software Testing**. 1. ed. New York, NY, USA: Cambridge University Press, 2008. ISBN 0521880386, 9780521880381.

BARESI, L.; YOUNG, M. **Test Oracles**. [S.l.], 2001.

BARR, E. T.; HARMAN, M.; MCMINN, P.; SHAHBAZ, M.; YOO, S. The Oracle Problem in Software Testing : A Survey. p. 1–30, 2014.

BERTOLINO, A. Software Testing Research: Achievements, Challenges, Dreams. In: **Future of Software Engineering (FOSE '07)**. [S.l.]: IEEE, 2007. p. 85–103. ISBN 0-7695-2829-5.

BIMBRAW, K. Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In: IEEE. **2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)**. [S.l.], 2015. v. 1, p. 191–198.

CHEN, D.; ZHOU, B.; KOLTUN, V.; KRÄHENBÜHL, P. Learning by cheating. In: **Conference on Robot Learning (CoRL)**. [S.l.: s.n.], 2019.

CHEN, T.; CHEUNG, S.; YIU, S. Metamorphic testing: a new approach for generating next test cases. **Dep. Comput. Sci. Hong Kong Univ. Sci. Technol.**, p. 1–11, 1998. Disponível em: <https://www.cse.ust.hk/{~}scc/publ/CS98-01-metamorphictesting.p>.

CHEN, T. Y. Metamorphic testing: A simple approach to alleviate the oracle problem. In: IEEE. **2010 Fifth IEEE International Symposium on Service Oriented System Engineering**. [S.l.], 2010. p. 1–2.

CHEN, T. Y.; KUO, F.-C.; LIU, H.; POON, P.-L.; TOWEY, D.; TSE, T. H.; ZHOU, Z. Q. Metamorphic Testing: : A Review of Challenges and Opportunities. **ACM Computing Surveys**, v. 51, n. 1, p. 1–27, jan 2018. ISSN 03600300.

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.l.: s.n.], 2017. p. 1–16.

DRESCH, A.; LACERDA, D. P.; Antunes Jr, J. A. V. **Design Science Research**. Cham: Springer International Publishing, 2015. ISSN 1063-8016. ISBN 978-3-319-07373-6. Disponível em: <http://link.springer.com/10.1007/978-3-319-07374-3>.

DUFFY, S. H.; HOPKINS, J. P. Sit, stay, drive: The future of autonomous car liability. **SMU Sci. & Tech. L. Rev.**, HeinOnline, v. 16, p. 453, 2013.

EMANI, S.; SOMAN, K.; VARIYAR, V. S.; ADARSH, S. Obstacle detection and distance estimation for autonomous electric vehicle using stereo vision and dnn. In: **Soft Computing and Signal Processing**. [S.l.]: Springer, 2019. p. 639–648.

GAO, H.; CHENG, B.; WANG, J.; LI, K.; ZHAO, J.; LI, D. Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment. **IEEE Transactions on Industrial Informatics**, IEEE, v. 14, n. 9, p. 4224–4231, 2018.

HEINEKE, K.; KAMPSHOFF, P.; MKRTCHYAN, A.; SHAO, E. **Self-driving car technology: When will the robots hit the road?** McKinsey Company, 2018. Disponível em: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/self-driving-car-technology-when-will-the-robots-hit-the-road/de-de#>.

HOWDEN, W. E. Theoretical and empirical studies of program testing. **IEEE Transactions on Software Engineering**, IEEE, n. 4, p. 293–298, 1978.

ISO/IEC/IEEE 29119-1. ISO/IEC/IEEE 29119-1:2013. v. 2013, 2013. Disponível em: <ieee.com>.

JO, K.; KIM, J.; KIM, D.; JANG, C.; SUNWOO, M. Development of autonomous car—part i: Distributed system architecture and development process. **IEEE Transactions on Industrial Electronics**, IEEE, v. 61, n. 12, p. 7131–7140, 2014.

Kanewala, U.; Yueh Chen, T. Metamorphic testing: A simple yet effective approach for testing scientific software. **Computing in Science Engineering**, v. 21, n. 1, p. 66–72, Jan 2019.

LEVIN, S. **Uber crash shows 'catastrophic failure' of self-driving technology, experts say**. Guardian News and Media, 2018. Disponível em: <https://www.theguardian.com/technology/2018/mar/22/self-driving-car-uber-death-woman-failure-fatal-crash-arizona>.

LIU, M.-Y.; BREUEL, T.; KAUTZ, J. Unsupervised image-to-image translation networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2017. p. 700–708.

MADRIGAL, A. C. **7 Arguments Against the Autonomous-Vehicle Utopia**. Atlantic Media Company, 2018. Disponível em: <https://www.theatlantic.com/technology/archive/2018/12/7-arguments-against-the-autonomous-vehicle-utopia/578638/>.

MOREIRA, D. D. Testes de sistemas de classificação de cenas acústicas utilizando relações metamórficas. **Cesar School**, dec 2019. Dissertação de Mestrado (Mestrado Profissional em engenharia de software).

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. 3rd. ed. [S.l.]: Wiley Publishing, 2011. ISBN 1118031962, 9781118031964.

NATIONAL CENTER FOR STATISTICS AND ANALYSIS. **2018 fatal motor vehicle crashes: Overview**. 2019. Disponível em: <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812826>. Acesso em: 05 fev. 2020.

NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION. **Automated Vehicles for Safety**. 2019. Disponível em: <https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety>. Acesso em: 05 fev. 2020.

NIELSEN, M. A. **Neural networks and deep learning**. [S.l.]: Determination press San Francisco, CA, USA:, 2015. v. 2018.

OLIVEIRA, R. A.; KANEWALA, U.; NARDI, P. A. Automated test oracles: State of the art, taxonomies, and trends. In: **Advances in computers**. [S.l.]: Elsevier, 2014. v. 95, p. 113–199.

PRESSMAN, R. S. **Software engineering: A practitioner's approach**. [S.l.]: McGraw-Hill Education, 2015.

SEGURA, S.; DURÁN, A.; TROYA, J.; CORTÉS, A. R. A template-based approach to describing metamorphic relations. In: IEEE. **2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)**. [S.l.], 2017. p. 3–9.

SEGURA, S.; FRASER, G.; SANCHEZ, A. B.; RUIZ-CORTES, A. A Survey on Metamorphic Testing. **IEEE Transactions on Software Engineering**, v. 42, n. 9, p. 805–824, 2016. ISSN 00985589.

SEGURA, S.; TOWEY, D.; ZHOU, Z. Q.; CHEN, T. Y. Metamorphic testing: Testing the untestable. **IEEE Software**, IEEE, 2018.

SEGURA, S.; ZHOU, Z. Q. Metamorphic testing 20 years later: A hands-on introduction. In: **Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings**. [S.l.: s.n.], 2018. p. 538–539.

TIAN, Y.; PEI, K.; JANA, S.; RAY, B. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: **Proceedings of the 40th international conference on software engineering**. [S.l.: s.n.], 2018. p. 303–314.

TOPHAM, G. **'Peak hype': why the driverless car revolution has stalled**. Guardian News and Media, 2021. Disponível em: <https://www.theguardian.com/technology/2021/jan/03/peak-hype-driverless-car-revolution-uber-robotaxis-autonomous-vehicle>.

WEYUKER, E. J. On Testing Non-Testable Programs. **Comput. J.**, v. 25, n. 4, p. 465–470, nov 1982. ISSN 0010-4620. Disponível em: <https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/25.4.465>.

WIERINGA, R. J. **Design Science Methodology for Information Systems and Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. 1–332 p. ISBN 978-3-662-43838-1. Disponível em: <http://link.springer.com/10.1007/978-3-662-43839-8>.

XIE, X.; HO, J. W.; MURPHY, C.; KAISER, G.; XU, B.; CHEN, T. Y. Testing and validating machine learning classifiers by metamorphic testing. **Journal of Systems and Software**, Elsevier, v. 84, n. 4, p. 544–558, 2011.

ZHANG, M.; ZHANG, Y.; ZHANG, L.; LIU, C.; KHURSHID, S. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In: **Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering**. [S.l.: s.n.], 2018. p. 132–142.

ZHOU, Z. Q.; SUN, L. Metamorphic testing of driverless cars. **Communications of the ACM**, ACM New York, NY, USA, v. 62, n. 3, p. 61–67, 2019.

ZHOU, Z. Q.; XIANG, S.; CHEN, T. Y. Metamorphic testing for software quality assessment: A study of search engines. **IEEE Transactions on Software Engineering**, IEEE, v. 42, n. 3, p. 264–284, 2015.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras
Rua José Antônio da Silva, 300 - Bairro Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Ostensivo (Público)

Trabalho de Conclusão de Curso

Assunto: Trabalho de Conclusão de Curso
Assinado por: Mailson Souza
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Mailson Dennis Trajano de Souza**, **ALUNO (201712010028) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 02/03/2021 13:15:00.

Este documento foi armazenado no SUAP em 02/03/2021. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 181797
Código de Autenticação: fff9b424d4