Uma proposta de arquitetura de software para a
simulação e experimentação de veículos autônomos

Tiago Cesar dos Santos

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Tiago Cesar dos Santos

Uma proposta de arquitetura de software para a simulação e experimentação de veículos autônomos

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Mestre em Ciências – Ciências de Computação e
Matemática Computacional. VERSÃO REVISADA

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientador: Prof. Dr. Denis Fernando Wolf

USP – São Carlos
Junho de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

d237p dos Santos, Tiago Cesar
Uma proposta de arquitetura de software para a
simulação e experimentação de veículos autônomos /
Tiago Cesar dos Santos; orientador Denis Fernando
Wolf. -- São Carlos, 2016.
95 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2016.

1. Simulação. 2. Arquitetura de Software. 3.
CaRINA. 4. MORSE. 5. Blender. I. Wolf, Denis
Fernando, orient. II. Título.

Tiago Cesar dos Santos

A software architecture for simulation and experimentation of autonomous vehicles

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. FINAL VERSION

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Denis Fernando Wolf

USP – São Carlos
June 2016

AGRADECIMENTOS

Primeiramente gostaria de agradecer à minha família por me guiarem até este momento.

Agradeço,

aos funcionários do ICMC e da USP pelo auxílio e cooperação em diversas situações

às agências de fomento CAPES, CNPq e a FIPAI em cooperação com a empresa Scania pelo auxílio financeiro em determinadas etapas deste trabalho e também a participação em eventos

ao grupo do Laboratório de Robótica Móvel pelo imenso apoio para realização de diversos experimentos e contribuições, também pelos vários momentos de descontração no laboratório

aos professores Fernando Osório e Valdir Grassi que contribuíram com este trabalho.

especialmente ao meu orientador Denis Wolf pela oportunidade que me deu para participar de projetos grandiosos e de fazer parte de um grupo altamente qualificado. Também pela confiança em deixar inúmeros equipamentos aos meus cuidados. Por fim, pela imensurável ajuda para conclusão deste trabalho

e finalmente, à Marcela, cujo agradecimento é impossível de ser exposto em palavras.

“ Se enxerguei mais longe,
foi porque estava sobre os ombros de gigantes”
(Isaac Newton)

RESUM O

DOS SANTOS, T. C.. Uma proposta de arquitetura de software para a simulação e experimentação de veículos autônomos. 2016. 95 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Sistemas de Transportes Inteligentes representam um imenso impacto social e aos poucos tem modificado o paradigma de mobilidade atual. Desde a década de 80, veículos autônomos vêm sendo desenvolvidos e estudados pela comunidade científica e hoje atrai o interesse de grandes empresas automobilísticas. Esses sistemas têm como objetivo a redução do número de acidentes de trânsito, aumento da eficiência dos transportes e inclusão social, sendo que neste contexto surge o projeto CaRINA. Através do desenvolvimento de uma plataforma robótica móvel pretende-se realizar a navegação completamente autônoma em ambiente urbano. Contudo, os experimentos realizados com a plataforma real são custosos, demorados e perigosos. A logística dos testes é complexa, uma vez que necessitam de local apropriado e disponibilidade de recursos. Portanto, o objetivo deste trabalho é desenvolver uma arquitetura de simulação para veículos autônomos que seja capaz de realizar experimentos em laboratório e facilite a portabilidade dos programas desenvolvidos em simulação para o veículo real. A flexibilidade da arquitetura do simulador também permite realizar experimentos utilizando múltiplos veículos.

Palavras-chave: Simulação, Arquitetura de Software, CaRINA, MORSE, Blender.

ABSTRACT

DOS SANTOS, T. C.. Uma proposta de arquitetura de software para a simulação e experimentação de veículos autônomos. 2016. 95 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Intelligent Transportation Systems represent a huge social impact and has gradually modified the current mobility paradigm. Since the 1980s, autonomous vehicles have been developed and investigated by the scientific community and today attracts the interest of major automotive companies. These systems aims to reduce the number of traffic accidents, increase the transports efficiency and social inclusion, in this context the CaRINA project started. Through the development of a mobile robotic platform is intended to perform a fully autonomous navigation in urban environments. However, the experiments with the platform are costly, time-consuming and dangerous. The logistics of the tests are complex, since that require the appropriate location and availability of resources. Therefore, the purpose of this work is to develop a simulation architecture for autonomous vehicles to be able to perform experiments in the laboratory and to facilitate the portability of programs developed in simulation to the real vehicle. The flexibility of the simulator architecture also allows to perform experiments using multiple vehicles.

Key-words: Simulation, Software Architecture, CaRINA, MORSE, Blender.

LISTA DE ILUSTRAÇÕES

Figura 1 – DA PPA Grand Challenge	24
Figura 2 – Plataforma CaRINA 2	25
Figura 3 – Veículo Stanley vencedor do DARPA Grand Challenge 2005	28
Figura 4 – Terreno desafiador do DARPA Grand Challenge	28
Figura 5 – Veículos do DARPA Urban Challenge	29
Figura 6 – Veículos desenvolvidos pelo VisLab	30
Figura 7 – Veículos Piaggio Porter utilizados na expedição	31
Figura 8 – Cenário em que o veículo líder é visível pelo seguidor	31
Figura 9 – Cenário em que o veículo líder não é visível pelo seguidor. Coordenadas GPS são transmitidas do líder para o seguidor	32
Figura 10 – Destaque da cor dos carros do VIAC	33
Figura 11 – Participantes do GCDC 2011 na Rodovia A270	34
Figura 12 – Equipamentos da Rodovia A270	34
Figura 13 – Fase urbana do evento	35
Figura 14 – Fase rodovia do evento.	36
Figura 15 – Plataforma CaRINA 2	39
Figura 16 – Experimentos realizados em 2013 nas ruas da cidade de São Carlos - SP com autorização e colaboração Secretarias Municipais de Transporte e Trânsito, e de Desenvolvimento Sustentável, Ciência e Tecnologia de São Carlos	39
Figura 17 – Ciclo Percepção-Planejamento-Atuação	42
Figura 18 – Arquitetura do veículo Stanley	43
Figura 19 – Arquitetura simplificada do veículo Boss	46
Figura 20 – Funcionalidades da camada de Execução de Comportamento do veículo Boss	46
Figura 21 – Exemplo de grafo no ROS	54
Figura 22 – Exemplo de publicação e assinatura de tópicos	54
Figura 23 – Exemplo de requisição e resposta através de serviços	55
Figura 24 – Diagrama do fluxo de dados entre o Blender e um middleware	56
Figura 25 – Alguns dos cenários e robôs disponíveis no MORSE.	57
Figura 26 – Utilização do simulador para interação robô e humano	57
Figura 27 – Diagrama de fluxo de dados da arquitetura geral composta basicamente por localização e mapa, plataforma robótica, controle, percepção e diagnóstico	60

Figura 28 – Diagrama de fluxo de dados da arquitetura do CaRINA 2. Cada módulo, em verde, representa um conjunto de funcionalidades (pacote). As mensagens, em cinza, mostram os tipos de dados e tópicos específicos do ROS	62
Figura 29 – Veículo 3D simplificado	65
Figura 30 – Ferramenta drag and drop	66
Figura 31 – Propriedades de objetos	67
Figura 32 – Comparação entre o veículo real e o simulado	67
Figura 33 – Cenários utilizados para simulação	68
Figura 34 – Texturização dos objetos	69
Figura 35 – Arquivos necessários para criação de um robô no MORSE	69
Figura 36 – Arquitetura de Simulação do CaRINA 2	70
Figura 37 – Arquitetura do CaRINA 2	73
Figura 38 – Trajeto 1. A linha em negrito representa os dados registrados com o GPS RTK e transformados para UTM.	76
Figura 39 – Trajeto 2. A linha em negrito representa os dados registrados com o GPS RTK e transformados para UTM.	76
Figura 40 – Trajeto 1. A linha em negrito representa a rota original previamente gravada, em verde o trajeto autônomo realizado pelo CaRINA 2 e em vermelho o trajeto autônomo realizado em simulação	77
Figura 41 – Erro lateral no Trajeto 1. Em verde é mostrado o erro lateral do veículo CaRINA 2 e em vermelho o erro lateral do veículo simulado	78
Figura 42 – Trajeto 2. A linha em negrito representa a rota original previamente gravada, em verde o trajeto autônomo realizado pelo CaRINA 2 e em vermelho o trajeto autônomo realizado em simulação	79
Figura 43 – Erro lateral no Trajeto 2. Em verde é mostrado o erro lateral do veículo CaRINA 2 e em vermelho o erro lateral do veículo simulado.	79
Figura 44 – Construção do sensor NIC	81
Figura 45 – Arquitetura da plataforma de simulação de múltiplos veículos	82
Figura 46 – Perfil de estabilidade do sistema	82
Figura 47 – Disposição dos veículos simulados. Cada veículo comunica-se imediatamente com o antecessor, sendo que o líder controlado por script e os seguidores são conduzidos em modo autônomo	83
Figura 48 – Velocidade dos veículos durante o experimento. Velocidade do veículo Líder (L, preto), Seguidor 1(F1, vermelho), Seguidor 2 (F2, verde), Seguidor 3 (F3, amarelo)	84
Figura 49 – Distância entre os veículos durante o experimento. L e F1(L para F1, preto), F1 e F2(F1 para F2, verde), F2 e F3(F2 para F3, vermelho)	84

Figura 50 – Histograma 1: Diferença da Velocidade entre L e F1. Histograma 2: Diferença da Velocidade entre L e F2. Histograma 3: Diferença da Velocidade entre L e F3	85
Figura 51 – Arquitetura do Caminhão Autônomo	87
Figura 52 – Apresentação publica do projeto do Caminhão Autônomo	88

LISTA DE TABELAS

Tabela 1 – Exemplo de simuladores utilizado em robótica ou no desenvolvimento veicular 50

LISTA DE ABREVIATURAS E SIGLAS

ACC		Adaptive Cruise Control
ADAS		Advanced Driver Assistance Systems
CACC		Cooperative Adaptive Cruise Control
CAN		Controller Area Network
CaRINA	..	Carro Robótico Inteligente para Navegação Autônoma
DARPA	..	Defense Advanced Research Projects Agency
GCDC	...	Grand Cooperative Driving Challenge
GPS		Global Positioning System
IMU		Inertial Measurement Unit
LIDAR	...	Light Detection And Ranging
NTRIP	...	Networked Transport of RTCM via Internet Protocol
ROS		Robot Operating System
RTK		Real Time Kinematic
TB		Terabyte
UTM		Universal Transverse Mercator
V2I		Vehicle-to-Infrastructure
V2V		Vehicle-to-Vehicle
VIAC		VisLab Intercontinental Autonomous Challenge

SUMÁRIO

1	INTRODUÇÃO	23
2	VEÍCULOS AUTÔNOMOS	27
2.1	Eventos promovidos pelo DARPA	27
2.2	VIAC (VisLab Intercontinental Autonomous Challenge)	29
2.3	GCDC (Grand Cooperative Driving Challenge)	32
2.4	Trabalhos Recentes	37
2.5	CaRINA 2	38
2.6	Considerações Finais	40
3	TRABALHOS RELACIONADOS	41
3.1	Arquitetura Clássica de Robôs Móveis	41
3.2	Arquitetura de Veículos Autônomos	42
3.3	Simuladores	47
3.4	Considerações Finais	51
4	FERRAMENTAS COMPUTACIONAIS E SIMULADORES	53
4.1	Robot Operating System	53
4.2	Simulador MORSE	55
5	ARQUITETURA DE SOFTWARE	59
5.1	Arquitetura Geral	59
5.2	Arquitetura CaRINA 2	60
5.3	Modelo Simulado do Veículo	65
5.3.1	Simulação	65
5.3.2	Arquitetura Simulador	70
5.4	Considerações Finais	72
6	RESULTADOS EXPERIMENTAIS	75
6.1	Validação Experimental	75
6.1.1	Trajeto 1	77
6.1.2	Trajeto 2	78
6.2	Simulação de Múltiplos Veículos	80
6.3	Experimentos com Veículos Pesados	85

6.4	Considerações Finais	86
7	CONCLUSÃO	89
7.1	Conclusão	89
	REFERÊNCIAS	91

INTRODUÇÃO

O avanço tecnológico da eletrônica e computação permitiu o desenvolvimento das primeiras plataformas e manipuladores robóticos, os quais invadiram os processos de manufatura industrial. Robôs são capazes de realizar tarefas repetitivas e exaustivas com maior precisão do que quando praticadas por pessoas. Isso também gera benefícios para a sociedade, uma vez que as pessoas podem ser poupadas de trabalhos insalubres e perigosos.

Nas últimas décadas a robótica tem expandido o horizonte de aplicações. A necessidade dos novos mercados, impulsionados pela demanda social, requer sistemas mais flexíveis, dinâmicos e inteligentes. Hoje é possível encontrar sistemas robóticos na construção civil, agricultura, mineração, medicina e etc. Particularmente, a área de transportes tem sido intensamente cercada por sistemas inteligentes, tanto em termos de infraestrutura, quanto os meios de transporte. Desde a década de 80 é possível notar iniciativas relacionadas a veículos autônomos (THORPE et al., 1988) (CRISMAN; THORPE, 1993) (BROGGI et al., 1999). Ao longo dos anos esses trabalhos continuaram e culminaram em um dos principais eventos de veículos autônomos, o Grand Challenge (Figura 1), promovido pela agência de desenvolvimento tecnológico militar dos EUA conhecido como DARPA. Apesar de interesses militares, o evento acelerou o desenvolvimento da tecnologia e despertou o interesse das empresas automotivas, que hoje já incorporaram algumas dessas tecnologias em seus veículos comerciais para aumentar a segurança e o conforto dos passageiros.

Outras vertentes de pesquisa da área de Sistemas de Transporte Inteligentes defendem o aperfeiçoamento da infraestrutura de transportes tornando possível realizar interações entre diversos elementos através da comunicação. De acordo com P. Tientrakool (2011) pode-se melhorar a eficiência do trânsito em 273% com o uso da comunicação entre veículos, sendo que a capacidade das vias aumentaria através da redução do espaçamento horizontal e lateral entre os veículos.

Figura 1 – DARPA Grand Challenge

Fonte: <<http://archive.defense.gov/>>.

Segundo dados da Organização Mundial de Saúde¹, acidentes de trânsito estão entre as dez maiores causadoras² de morte no mundo, em que dezenas de milhares de pessoas são mortas por ano, não sendo diferente no Brasil³. Grande parte dos acidentes de trânsito são atribuídos a falha humana em virtude da condução imprudente dos veículos, outros fatores como condições climáticas e mecânicas também são notadas em alguns casos.

Neste contexto, surge uma das iniciativas nacionais de mobilidade, o projeto CaRI NA⁴ (Carro Robótico Inteligente para Navegação Autônoma). O projeto tem como objetivo o desenvolvimento de uma plataforma robótica móvel (Figura 2) capaz de se locomover de forma autônoma em ambiente urbano. Como benefícios o projeto visa a diminuição dos acidentes e melhorando a eficiência do trânsito, o que pode contribuir para redução dos congestionamentos e consequentemente reduzir a emissão de poluentes.

O desenvolvimento de veículos autônomos requer constante experimentação em cenários diversificados para que seja possível identificar os problemas e as limitações dos métodos utilizados. Uma vez que o veículo é submetido a testes em cenário real ele está sujeito a diversas situações não controladas que podem causar acidentes. Da mesma forma, testes reais podem ser custosos e consumir muito tempo. A logística complexa e a disponibilidade de recursos, pessoas e espaço para testes também atrasa ou impossibilita diversos experimentos. Atualmente a plataforma CaRI NA 2 possui uma série de programas de computador que devem ser executados para realizar a navegação autônoma e também devem ser monitorados para garantir o bom funcionamento. A alta complexidade da arquitetura da plataforma composta de vários módulos atrasam os experimentos ou mesmo gerar acidentes caso alguma falha não seja notada pelo

¹ <http://www.who.int/violence_injury_prevention/road_traffic/en/>

² <<http://apps.who.int/gho/data/>>

³ <http://www.who.int/violence_injury_prevention/road_safety_status/2013/country_profiles/brazil.pdf?ua=1>

⁴ <<http://www.lrm.icmc.usp.br/>>

Figura 2 – Plataforma CaRINA 2

Fonte: <<http://www.lrm.icmc.usp.br/>>.

usuário.

Para melhorar e acelerar o processo de desenvolvimento da plataforma torna-se imprescindível o uso de uma ferramenta de simulação. Em vista disso, o objetivo principal deste trabalho é o desenvolvimento de um simulador que possa realizar experimentos, desenvolver métodos e algoritmos em laboratório. Portanto o conhecimento da arquitetura da plataforma CaRINA 2 é de vital importância para realização deste trabalho. A arquitetura do simulador deve ser totalmente compatível com a arquitetura do veículo real, possibilitando a adaptação e portabilidade dos programas desenvolvidos em simulação para a plataforma real. A flexibilidade da arquitetura de simulação permite que sejam conduzidos experimentos com múltiplos veículos, tópico de grande interesse da comunidade científica.

O estudo arquitetural da plataforma CaRINA 2 revelou alguns módulos suscetíveis a falhas, que podem vir a ocorrer durante testes em campo. A fim de aumentar a segurança dos experimentos um sistema de diagnóstico foi desenvolvido para monitorar o funcionamento dos principais módulos do veículo e até mesmo a qualidade das informações geradas por sensores. A análise arquitetural do CaRINA 2 também permitiu realizar a sua organização e documentação. Isso facilitou a adaptação da arquitetura para veículos pesados, sendo que esses também são contribuições deste trabalho.

Por fim, esta dissertação foi dividida em sete capítulos. No Capítulo 2 são discutidos alguns dos principais trabalhos relacionados a veículos autônomos, sendo que os mais relevantes da literatura são analisados detalhadamente. As especificações de métodos e algoritmos podem ser encontrada no Capítulo 3, onde são estudadas as principais arquiteturas dos veículos vencedores dos eventos patrocinados pelo DARPA que são fundamentais para a análise da arquitetura do

projeto CaRINA 2. Também são discutidos alguns dos principais simuladores utilizados por pesquisadores e empresas.

O Capítulo 4 mostra as ferramentas que foram utilizadas no desenvolvimento desse trabalho. A arquitetura do CaRINA 2 é analisada no Capítulo 5, onde são levantados os requisitos necessários para desenvolver a arquitetura de simulação. Também é apresentado o desenvolvimento do modelo simulado do CaRINA 2 e a adaptação do simulador MORSE que compõe a arquitetura de simulação. Os resultados da validação experimental são apresentados no Capítulo 6, onde também é discutida a adaptação do simulador para múltiplos veículos e sua validação. Ao final deste capítulo é apresentada a portabilidade do sistema para veículos pesados. Por fim, a conclusão deste trabalho é discutido no Capítulo 7.

VEÍCULOS AUTÔNOMOS

Este capítulo reúne e discute alguns dos principais eventos e desafios relacionados a veículos autônomos e cooperativos. Os eventos promovidos pelo DARPA, além de contextualizarem este trabalho, permitiram o avanço acelerado da tecnologia e a construção de uma rica literatura. O VIAC e o GCDC são alguns dos eventos mais conhecidos na área de múltiplos veículos e utilizam diretamente a comunicação entre veículos e de veículos entre infraestrutura (V2V e V2I respectivamente). Aqui também é apresentado o projeto CaRINA, do qual faz parte o presente trabalho.

2.1 Eventos promovidos pelo DARPA

A evolução da robótica móvel tem proporcionado melhoria na vida das pessoas. Hoje, pode-se notar o interesse de empresas automobilísticas em tecnologias desenvolvidas em pesquisas acadêmicas. É sempre importante ressaltar os eventos realizados pela agência militar de pesquisa norte-americana (DARPA) que estabeleceram um marco no desenvolvimento da robótica móvel aplicado diretamente em veículos, utilizando novas tecnologias e algoritmos com abordagens probabilísticas.

Em 2004, o DARPA realizou a primeira grande competição relacionada a veículos autônomos. Essa primeira edição ficou conhecida como DARPA Grand Challenge ([THRUN et al., 2006](#)), no qual foi oferecido um prêmio de \$1 milhão à equipe que conseguisse percorrer 240 km em terreno desértico (off-road) e sem intervenção humana. Apesar de nenhuma equipe ter atingido o objetivo, a edição representou um marco no desenvolvimento de carros autônomos no qual as experiências obtidas por cada equipe amadureceram novas ideias que foram levadas para as próximas edições. Prontamente, no ano seguinte, foi proposto um desafio similar com percurso de 212 km e premiação de \$2 milhões, 23 equipes se qualificaram para participar do evento. O veículo Stanley (Figura 3), da equipe Stanford Racing Team foi o vencedor da competição, graças as abordagens probabilísticas e aos métodos de aprendizado de máquina que ajudaram a

equipe a superar os desafios (THRUN et al., 2006).

Figura 3 – Veículo Stanley vencedor do DARPA Grand Challenge 2005

Fonte: Thrun et al. (2006).

O sistema de navegação do Stanley era composto por cinco sensores LIDAR montados no topo do veículo e inclinados para baixo. O conjunto de sensores gera uma nuvem de pontos que foi utilizada para analisar a presença de obstáculos próximos ao veículo, também foi utilizado uma câmera para avaliar a navegabilidade do terreno (THRUN et al., 2006), principalmente em longas distâncias. Assim, a navegação do Stanley era baseada na fusão de sensores com o objetivo principal de encontrar regiões navegáveis.

Um dos principais desafios do Grand Challenge consistia fundamentalmente em manter o veículo dentro da área navegável, isso devido ao terreno acidentado (Figuras 4) o qual impôs muitas dificuldades e qualquer erro poderia facilmente danificar os veículos ou até mesmo retirá-los da competição. Segundo Thrun et al. (2006), Stanley não era capaz de navegar autonomamente em ambiente urbano, pois ele foi desenvolvido e adaptado exclusivamente para esse cenário.

Figura 4 – Terreno desafiador do DARPA Grand Challenge

Fonte: Thrun et al. (2006).

Em 2007 foi proposto o Urban Challenge, promovido pela mesma agência DARPA. Agora em ambiente urbano, além de navegar nas ruas de uma cidade simplificada, os competidores tinham que lidar com as leis de trânsito e sinalizações. Também incumbia ao veículo autônomo a função de ultrapassagem e estacionamento. O evento qualificou 11 equipes para competir na final e também ofereceu \$2 milhões ao vencedor.

Nessa edição do evento pode-se notar o avanço de algumas tecnologias, por exemplo, o sensor LIDAR 360° da Velodyne ([URMSON et al., 2007](#)) ([MONTEMERLO et al., 2008](#)), utilizado nos veículos, Boss (Figura 5a) da Universidade Carnegie Mellon em parceria com a General Motors e Junior (Figura 5b) da Universidade de Stanford. Esse novo sensor difere-se do LIDAR planar, muito utilizado na edição anterior, ao fazer uma varredura em 360° o que proporciona um mapeamento 3D ao redor do mesmo.

Figura 5 – Veículos do DARPA Urban Challenge

(a) Veículo Boss

(b) Veículo Junior

Fonte: [Baker e Dolan \(2009\)](#) e [Montemerlo et al. \(2008\)](#) respectivamente.

Além dos sensores LIDAR as equipes também utilizaram câmeras que foram empregadas na detecção de áreas navegáveis, agregando as informações contextuais das marcações na pista, utilizadas na detecção de obstáculos estáticos ([URMSON et al., 2007](#)).

Apesar de ter sido mais um grande avanço na robótica móvel, o Urban Challenge ainda deixou algumas lacunas. Na cidade simplificada não haviam sinalizações através de placas ou semáforos e mesmo o tráfego era manipulado pela equipe do DARPA de forma a evitar congestionamento ([MONTEMERLO et al., 2008](#)). À vista disso, não foi possível esclarecer se os veículos agiriam com sensatez no meio de congestionamento real.

2.2 VIAC (VisLab Intercontinental Autonomous Challenge)

O grupo italiano da Universidade de Parma (VisLab) atua na área de veículos robóticos a mais de 17 anos. Grande parte dos estudos consiste no desenvolvimento de visão computacional, visão artificial, processamento de imagens e aprendizado de máquina para reconhecimento de padrões. O primeiro trabalho do grupo consistiu no desenvolvimento de um projeto chamado ARGO ([BROGGI et al., 1999](#)) (Automatic Vehicle Guidance), que percorreu mais de 2.000 km nas rodovias italianas. Em 2004 e 2005, o grupo fez parte do desenvolvimento do veículo TerraMax que participou das edições do DARPA Grand Challenge, embora tenha ultrapassado o tempo limite de 10 horas, o veículo conseguiu finalizar a prova dirigindo todo o percurso de forma autônoma ([BRAID; BROGGI; SCHMIEDEL, 2006](#)). A equipe também teve participação no DARPA Urban Challenge ([CHEN et al., 2009](#)) com o mesmo time desenvolvedor do veículo

TerraMax. Mais tarde em 2009 a equipe investiu no desenvolvimento do veículo BRAiVE, um carro de passeio adaptado para navegação autônoma dentro de ambientes urbanos, em que grande parte de seus algoritmos de inteligência são viabilizados com auxílio de visão computacional. Os veículos podem ser vistos nas Figuras 6.

Figura 6 – Veículos desenvolvidos pelo VisLab

(a) ARGO

(b) TerraMax

(c) BRAiVE

Fonte: <<http://vislab.it/automotive/>>.

Com as experiências obtidas no desenvolvimento do ARGO e nos eventos DARPA, em 2010, o Vislab propôs o VIAC (VisLab Intercontinental Autonomous Challenge), um desafio que envolvia diretamente um comboio de veículos autônomos. O desafio consistia em navegar 13.000 km em uma travessia intercontinental entre a Europa e a Ásia, partindo da Itália com destino a China (BROGGI et al., 2010). Não por coincidência, a equipe almejava chegar em Shanghai na data prevista para participar do The World Expo 2010, cujo tema era Better cities, better life. Além de mostrar os avanços na área da robótica móvel, a equipe também dispôs quatro veículos elétricos com painéis solares para realizar a expedição, e com isso, enaltecer a eficiência de um carro elétrico.

VisLab equipou quatro veículos elétricos com diversos tipos de sensores como LIDAR, câmeras e GPS, devido a longa viagem proposta apenas dois veículos eram utilizados nos experimentos e outros eram sobressalentes. Os veículos são mostrados na Figura 7. A proposta deste extenso experimento foi realizar todo o trajeto em comboio com o veículo líder parcialmente autônomo (quando alguma decisão tinha que ser tomada o veículo era dirigido manualmente) e o veículo seguidor 100% autônomo. De acordo com Bertozzi et al. (2011a), devido a problemas técnicos ou de logística, somente 8.244 km dos 13.000 km foram percorridos de forma autônoma.

Figura 7 – Veículos Piaggio Porter utilizados na expedição

Fonte: <<http://vislab.it/automotive/>>.

Conforme mencionado por Broggi et al. (2010), o sistema de comboio foi dividido em dois cenários. O primeiro cenário (Figura 8) acontece quando o veículo líder é visível pelo veículo seguidor, que o detecta através dos métodos de visão computacional e imagens obtidas pela câmera. O segundo cenário (Figura 9) ocorre quando o seguidor perde a visada do líder, então os dados da posição GPS do líder são enviados ao seguidor. Em ambos, o veículo seguidor é autossuficiente para manter a condução autônoma na pista, ou seja, ele possui algoritmos de detecção de áreas navegáveis e obstáculos.

Figura 8 – Cenário em que o veículo líder é visível pelo seguidor

Fonte: Broggi et al. (2010).

De acordo com Bertozzi et al. (2013), seis algoritmos foram fundamentais para realizar a expedição que representam a parte sensorial dos veículos:

- Conservador de pista: as faixas longitudinais da pista são detectadas através da câmera frontal do veículo. O algoritmo é capaz de discriminar linhas contínuas e linhas tracejadas.
- Detecção de Obstáculos Baseado em Câmera: obstáculos são encontrados utilizando um mapa de profundidade fornecido por um conjunto de câmeras estéreo.

Figura 9 – Cenário em que o veículo líder não é visível pelo seguidor. Coordenadas GPS são transmitidas do líder para o seguidor

Fonte: Broggi et al. (2010).

- Detecção de Obstáculos Baseado em Laser: o algoritmo processa dados dos LIDARs de feixe simples e de múltiplos feixes. Além de permitir uma detecção mais apurada dos obstáculos, ele auxilia a encontrar terrenos navegáveis.
- Seguidor de Veículo: a detecção é feita associando dados do LIDAR com dados da câmera. O primeiro é utilizado para selecionar veículos candidatos e o segundo para validação dos candidatos.
- Localização de Área Navegável: como mencionado anteriormente, um dos LIDARs auxilia na detecção de relevos, acostamentos e depressões na pista, principalmente em terreno off-road.
- Planejador de Trajetória: o planejador utiliza GPS para navegação, além disso, todos os algoritmos citados acima são utilizados para validação da trajetória.

De acordo com Bertozzi et al. (2011a), por volta de 40 TB foram arquivados durante toda a expedição. Esses dados são importantes pois podem ser utilizados para o desenvolvimento de novos métodos em laboratório, já que a expedição passou por diferentes cenários e enfrentou climas variados.

Vale ressaltar que a maior parte do percurso em que o veículo líder dirigiu autonomamente foi em rodovias (BERTOZZI et al., 2011b). Também é possível ser feita uma análise relacionada a cor dos veículos, uma vez que eles possuem cor alaranjada e se destacam em um cenário tornando mais fácil a sua detecção (Figura 10).

2.3 GCDC (Grand Cooperative Driving Challenge)

Como discutido na seção anterior, há um grande interesse tanto da comunidade científica quanto das empresas automobilísticas em investigar tecnologias voltadas a comunicação entre

Figura 10 – Destaque da cor dos carros do VIAC

Fonte: <<http://viac.vislab.it/>>.

veículos e também entre veículos e infraestrutura. Vários sistemas como anti-colisão, mudança de faixa e controle podem ser melhorados com a aplicação da comunicação como mencionado por P. Tientrakool (2011).

Com essa motivação, em maio de 2011, em um trecho da estrada A270 na Holanda, ocorreu um dos maiores eventos de veículos cooperativos que ficou conhecido como GCDC. A proposta da competição, que reuniu esforços de várias empresas, era acelerar o desenvolvimento das tecnologias de transporte inteligente, tanto de veículos autônomos quanto da infraestrutura das vias, tentando resolver o impasse entre os governos e as empresas automobilísticas. De acordo com Nunen et al. (2012), "governos esperam pelo desenvolvimento de sistemas cooperativos e carros autônomos por parte das indústrias que, por sua vez, esperam que os governos informem uma descrição técnica de uma infraestrutura mais sofisticada".

O evento foi organizado por dois grupos de pesquisa da Holanda, HTAS¹ e TNO². Alguns dos objetivos do GCDC foram desenvolver tecnologias de direção cooperativa baseadas em comunicação veículo-veículo ou veículo-infraestrutura que podem diminuir os congestionamentos nas grandes cidades, também demonstrar a possibilidade da diminuição na emissão de CO₂ e outros poluentes apostando na formação de comboios, o que leva a redução do consumo de combustíveis. Finalmente, melhorar a segurança e o conforto da pessoas (COMMISSION, 2003) (CHEON; CENTER, 2002).

A rodovia A270 (Figura 11) nas proximidades de Helmond na Holanda foi especialmente aparelhada e fechada ao público nos dias da competição e testes. Como pode ser visto na Figura 12, a rodovia foi equipada para comunicação veículo-infraestrutura dispondo de 48 câmeras fixas, 12 gateways (portais de acesso para comunicação wireless) e um GPS de alta precisão (RTK, real-time kinematic) que garantia uma boa localização dos veículos. Todos os sensores da pista e participantes foram monitorados e gerenciados pelos organizadores em uma central.

¹ <www.htas.nl>

² <www.tno.nl>

Figura 11 – Participantes do GCDC 2011 na Rodovia A270

Fonte: <<http://ieeexplore.ieee.org/>>.

Figura 12 – Equipamentos da Rodovia A270

Fonte: Nunen et al. (2012).

O GCDC desenvolveu um ambiente realista e forneceu aos times participantes uma infraestrutura para testar suas novas tecnologias e soluções inovadoras em ambiente cooperativo. Neste evento o GCDC preparou dois tipos de cenários, urbano e rodoviário. Na própria rodovia A270 os organizadores colocaram dois semáforos (além da sinalização visual tradicional os semáforos enviavam sinal em broadcast, ou seja, para todos os participantes conectados) com a finalidade de representar uma cidade inteligente. No segundo cenário foi utilizada a própria rodovia equipada com sensores.

Primeiramente os competidores foram divididos em dois times, cada time foi enfileirado

longitudinalmente separados pela faixa central da pista. No cenário urbano, os times foram separados em dois pelotões e posicionados nos faróis (Figura 13), sendo o primeiro farol liderado pelo veículo do GCDC e o segundo, apenas com veículos dos competidores. O pelotão do primeiro farol foi liberado após o do segundo, assim, o pelotão do segundo farol tenta se acomodar ao comboio da frente de forma a avaliar o comportamento e a estabilidade entre comboios. O carro do GCDC permaneceu em uma velocidade preestabelecida e os competidores tentaram manter a comunicação entre todo sistema para realizar a tarefa. A prova terminou quando o último veículo cruzou a linha de chegada.

Figura 13 – Fase urbana do evento

Fonte: Adaptada de Nunen et al. (2012).

O trecho de rodovia se iniciou após o término do urbano, enquanto isso o carro do GCDC permaneceu em velocidade constante esperando o grupo se organizar. O veículo do GCDC inseriu perturbações no comboio para que fosse avaliada a capacidade de suavizar o efeito das ondas de perturbação ou string stability (Figura 14). Portanto, a tarefa do pelotão era manter a estabilidade do comboio, minimizando o comprimento do mesmo e evidentemente garantindo a segurança dos pilotos.

As provas supracitadas foram realizadas diversas vezes a fim de coletar dados representativos para definir um vencedor dessa edição. Desta forma, os times eram sempre modificados e escolhidos aleatoriamente. As equipes foram avaliadas tanto individualmente quanto coletivamente.

Figura 14 – Fase rodovia do evento.

Fonte: Adaptada de [Nunen et al. \(2012\)](#).

vamente a partir dos times de que fizeram parte. O GCDC avaliou principalmente os seguintes quesitos ([NUNEN et al., 2012](#)):

- Comprimento do pelotão (Coletivo): medido no cenário urbano no momento em que o último veículo passa na linha de chegada (o tamanho dos veículos é levado em consideração).
- Espaçamento máximo do pelotão (Coletivo): a cada dois veículos no pelotão existe um espaço, o maior espaço é registrado no cenário rodovia.
- Estabilidade da fila de veículos (Individual): é a capacidade de suavizar as ondas de perturbação (string stability).

Com os dados coletados, o GCDC classificou a equipe AnnieWAY (Instituto Tecnológico de Karlsruhe) na primeira colocação. A equipe se destacou no quesito individual, sendo que seu controle de estabilidade fez maior pontuação ([NUNEN et al., 2012](#)). Segundo [Lauer \(2011\)](#) a classificação final foi: AnnieWAY, Halmstad, Chalmers, Scoop, AUTOPIA, ATeam, Mekar, FUTURUM e LATVIA.

Durante o evento o protocolo CALM (Communications access for land mobiles)³ foi utilizado para realizar a comunicação. Esse protocolo foi idealizado para operar em vários tipos de cenários, de curto a longo alcance. É uma iniciativa de arquitetura de comunicação entre veículos, veículos e infraestrutura e entre infraestruturas. Ele funciona nos mais variados dispositivos e na maior parte dos aparelhos com tecnologia IEEE 802.11 (WIFI). Entretanto o GCDL (TNO) também providenciou um programa que traduzia mensagens CALM para mensagens de protocolo IP (GEIGER et al., 2012), assim os participantes não precisavam se preocupar com o protocolo de comunicação.

Apesar do protocolo CALM ter sido adequado para o evento, ele ainda requer muito desenvolvimento principalmente no quesito segurança. Segundo Geiger et al. (2012), o protocolo não possuía um bom nível de encriptação e verificação das fontes de dados o que deixava todo sistema vulnerável a ataques.

De acordo com Geiger et al. (2012), nem todos os participantes forneceram informações corretas a respeito dos seus veículos. Os competidores já esperavam que por motivo de oclusões o sinal da comunicação seria afetado. Quando os veículos passavam embaixo de pontes o sinal da comunicação diminuía e a posição dos carros era congelada, causando anomalias no sistema, tornando as informações não confiáveis. O time AnnieWAY colocou toda confiança no radar frontal do veículo que possuía um alcance de 50 metros.

Assim como o Urban Challenge, o GCDL foi uma competição que construiu um ambiente totalmente controlado para que fossem realizados os experimentos. Porém, vale ressaltar que a infraestrutura fazia parte da competição. O GCDL buscou o desenvolvimento do controle longitudinal baseado no Cooperative Adaptive Cruise Control, sendo que o controle vertical (manter o veículo dentro da área navegável) era feito manualmente pelos motoristas das equipes (NUNEN et al., 2012) (KIANFAR et al., 2012). Ruídos nos sensores e informações incorretas estavam presente a todo instante, isso poderia causar algumas perturbações de forma que se um veículo que estivesse posicionado na pista esquerda poderia enviar informações errôneas afirmando que ele estava na pista direita (ou vice-versa), isso poderia formar um cenário de comportamentos instáveis no comboio e de frenagem de emergência. Assim, quase todas as equipes ignoraram as informações de origem do lado oposto ao qual estava no momento, apesar de ser permitido nas regras do GCDL, o que não caracteriza um ambiente realístico (GEIGER et al., 2012).

2.4 Trabalhos Recentes

A rota histórica de Mannheim a Pforzheim realizada pela Bertha Benz completou 125 anos em 2013. Ziegler et al. (2014) concretizaram uma homenagem ao memorial ao realizar a

³ <http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees/iso_technical_committee.htm?commid=54706>

rota de modo completamente autônomo com um Mercedes Benz S-Class. A rota possui cerca de 103 km e vários cenários desafiadores como estradas urbanas e rurais. A rota também cruza o centro da cidade de Heidelberg, no qual o veículo teve que lidar com trânsito e pedestres.

Um dos veículos autônomos mais famosos da atualidade é Google Self-Driving Car. O projeto foi iniciado em 2009 e a equipe de desenvolvimento conta com alguns dos participantes dos eventos promovidos pelo DARPA. Segundo o relatório⁴ de Janeiro de 2016 do Google, os veículos autônomos da empresa já percorreram cerca de 2.3 milhões de quilômetros no total. A empresa ressalta que os testes experimentais que estão sendo realizados em vias públicas ajudam o desenvolvimento da tecnologia, já que diferentes ambientes e situações trazem novos desafios à equipe.

O relatório da Google também enfatiza que o simulador utilizado pela equipe contribui consideravelmente com o desenvolvimento do veículo. Quaisquer alterações realizadas nos softwares são exaustivamente testadas em simulação antes de serem incorporadas nos veículos. De acordo com o relatório, é percorrido aproximadamente 5 milhões de quilômetros por dia em simulação.

2.5 CaRINA 2

O projeto CaRINA, Carro Robótico Inteligente para Navegação Autônoma, tem como principais objetivos a diminuição dos acidentes de trânsito causados principalmente por falhas humanas, aumento da eficiência dos sistemas de transportes e até mesmo promover a inclusão social. A multidisciplinaridade do projeto permite contribuir tecnologicamente no âmbito de Sistemas de Transportes Inteligentes utilizando recursos da Robótica Móvel, Computação e Engenharia. Por fim, o projeto visa o desenvolvimento de uma plataforma de navegação autônoma, bem como sistemas de auxílio ao motorista.

O projeto CaRINA 1 foi iniciado em 2010 com um veículo elétrico de pequeno porte (Carryall 232), mas com características similares a um carro comum. Ele foi modificado para condução autônoma e equipado com vários sensores. Ao final de 2010, foram feitos os primeiros experimentos de navegação totalmente autônoma em ambiente restrito dentro do Campus da USP em São Carlos.

Aproveitando o conhecimento adquirido com o veículo supracitado, o grupo iniciou em 2011 o desenvolvimento de uma nova plataforma que permitiu experimentos em trânsito real. O veículo de passeio Fiat Palio Adventure Dualogic foi escolhido por apresentar espaço para acomodar os equipamentos, um rack onde é possível posicionar diversos sensores com diferentes organizações e câmbio automatizado. O CaRINA 2 (Figura 15), como foi nomeado,

⁴ <<https://static.googleusercontent.com/media/www.google.com/en/selfdrivingcar/files/reports/report-0116.pdf>>

apresenta maiores possibilidades experimentais de navegação urbana, uma vez que o Fiat Palio é um veículo de porte comum e consegue alcançar maiores velocidades.

Figura 15 – Plataforma CaRINA 2

Fonte: <http://www.lrm.icmc.usp.br/>.

Em meados de 2012, foram realizados os primeiros testes totalmente autônomo em ambiente restrito no Campus da USP em São Carlos. Em 2013, com autorização e colaboração das Secretarias Municipais de Transporte e Trânsito, e de Desenvolvimento Sustentável, Ciência e Tecnologia de São Carlos, foram realizados os primeiros experimentos de navegação autônoma em trânsito urbano real, mostrados na Figura 16.

Figura 16 – Experimentos realizados em 2013 nas ruas da cidade de São Carlos - SP com autorização e colaboração Secretarias Municipais de Transporte e Trânsito, e de Desenvolvimento Sustentável, Ciência e Tecnologia de São Carlos

Fonte: <http://www.lrm.icmc.usp.br/>.

2.6 Considerações Finais

Este capítulo tratou de alguns dos principais eventos relacionados a veículos autônomos. Analisando os eventos promovidos pelo DARPA, pode-se notar a evolução dos equipamentos, o amadurecimento dos sistemas e também da comunidade científica onde vale ressaltar que em 2004 nenhum participante foi capaz de finalizar o percurso, já no evento de 2007, 6 participantes conseguiram completar todas as missões. Hoje as linhas de pesquisa envolvem o estudo de sistemas cooperativos baseados em comunicação V2V e V2I.

TRABALHOS RELACIONADOS

Este capítulo tem como objetivo analisar alguns conceitos básicos relacionados a arquitetura de robôs móveis, bem como são detalhadas as arquiteturas dos veículos vencedores dos eventos patrocinados pelo DARPA. O estudo dessas arquiteturas proporcionam o embasamento teórico necessário para a compreensão dos métodos para realizar a navegação autônoma, sendo que muitos desses são utilizados na plataforma CaRINA 2. Em seguida, são apresentados alguns dos simuladores mais famosos utilizados no mundo da robótica, bem como suas características e licenças de uso.

3.1 Arquitetura Clássica de Robôs Móveis

A Robótica é a ciência que estuda a teoria e aplicação dos robôs, é uma área multidisciplinar que abrange conhecimentos da Ciência da Computação, Engenharia Mecânica e Elétrica, entre outros. O campo da robótica é vasto e tem sido fortemente aplicado nos processos de produção industrial em linhas de montagem, principalmente por meio de manipuladores e braços robóticos que podem executar tarefas mais precisas e rápidas (SIEGWART; NOURBAKHSH, 2004). Como mencionado por Siegwart e Nourbakhsh (2004), esses robôs possuem uma característica em comum, são incapazes de se locomover. Isso impossibilita a prática de uma infinidade de tarefas. A robótica móvel, por sua vez, estuda lidar com os desafios de locomoção em ambientes não supervisionados, tanto estruturados como não estruturados. Para tanto existem pesquisas que tratam de robôs terrestres, aéreos e aquáticos.

Robôs móveis terrestres, objeto de estudo neste trabalho, devem ter a capacidade de navegar em ambientes desconhecidos e não estruturados, utilizando suas próprias faculdades para controlar os meios de locomoção. Usualmente um robô móvel é constituído de partes mecânicas responsáveis pela locomoção e interação com o ambiente, chamados de atuadores. Além disso, uma funcionalidade primordial para navegação em ambientes não supervisionados é a capacidade de percepção, já que o ambiente pode ser alterado dinamicamente. A percepção

constitui a interpretação dos dados do ambiente que são obtidos por sensores.

O sistema de um robô móvel pode ser resumido em uma arquitetura, na qual os objetivos são, em geral, previamente especificados como o desvio de obstáculos, mapeamento e localização. As capacidades citadas anteriormente compõem três elementos críticos que são fundamentais de uma arquitetura: a percepção, o planejamento e a atuação. A percepção é responsável por identificar elementos ao redor do robô através do sensoriamento, sendo possível dessa forma realizar o planejamento necessário para completar uma tarefa. Já a atuação diz respeito ao sistema de controle direto dos atuadores do robô. Esses elementos compreendem o ciclo Percepção-Planejamento-Atuação, mostrado na Figura 17, o fluxo de dados entre esses componentes é linear e unidirecional (GAT, 1998).

Figura 17 – Ciclo Percepção-Planejamento-Atuação

Arquiteturas de robôs podem ser classificadas levando em consideração os níveis de decisão. Quando para atingir um objetivo o robô age puramente através de um planejamento prévio para tomar decisões, a arquitetura é chamada de deliberativa. Por outro lado, quando são tomadas decisões puramente baseada em sensoriamento local, a estrutura é chamada de reativa. A arquitetura híbrida, por sua vez, tenta utilizar os dois tipos de comportamento para conduzir o robô através de um planejamento global ou local, no qual o sensoriamento pode requerer mudanças locais ou mesmo um replanejamento global.

Segundo Gat (1998), em meados da década de 80, vários pesquisadores se depararam com inúmeras ineficiências no ciclo de fluxo linear Percepção-Planejamento-Atuação, onde os esforços eram baseados em planejamento e modelagem do mundo. O desenvolvimento dessas arquiteturas culminou na Arquitetura de Três Camadas, que apesar de ser uma arquitetura híbrida, a sua principal característica é possuir um módulo intermediário entre a camada reativa e deliberativa que coordena a execução das mesmas, dependendo do estado do robô. A execução dos módulos é feita separadamente, característica que permite o paralelismo.

3.2 Arquitetura de Veículos Autônomos

Stanley, o veículo autônomo da Universidade de Stanford, deixou sua marca na história ao vencer o DARPA Grand Challenge de 2005. Trata-se de um veículo Volkswagen Touareg 2004 que foi devidamente equipado com proteções para enfrentar o cenário off-road. Para realizar as manobras de forma autônoma um motor DC foi acoplado a barra de direção para realizar o esterçamento e um atuador linear foi devidamente posicionado no câmbio para efetuar as trocas

de marcha entre drive, parking e ré. O sensoriamento proprioceptivo foi feito através da rede CAN do próprio veículo, no qual era possível obter dados como a velocidade de cada roda.

Um conjunto de sensores para percepção foi acomodado no bagageiro superior assim como antenas de GPS e E-STOP, o qual poderia ser utilizado para frear o veículo remotamente em caso de emergência. No interior um conjunto de seis computadores foi instalado, sendo que dois operavam todo o sistema, um era totalmente dedicado a visão computacional, um gerava os registro de toda corrida e os outros dois computadores eram sobressalentes.

A arquitetura de software do Stanley é baseada na Arquitetura de Três Camadas, não possuindo processo principal centralizado (GAT, 1998). O sistema possui por volta de 30 módulos que se comunicam de maneira assíncrona, ou seja, de forma paralela. O sistema completo é apresentado na Figura 18.

Figura 18 – Arquitetura do veículo Stanley

Fonte: Adaptada de Thrun et al. (2006).

Em geral, o sistema pode ser dividido em seis grandes grupos: camada de sensores, percepção, controle, interface veículo, interface com usuário e serviços globais.

A camada de interface com os sensores compreende todos os dispositivos utilizados, tratando da aquisição dos dados bem como da marcação do tempo (time stamped). Como cada sensor trabalha em uma frequência diferente era necessário marcar o tempo de cada mensagem de dados dos sensores para uma futura sincronização. A camada de percepção utiliza os dados da camada anterior sendo possível notar duas grandes funcionalidades nela, localização e reconhecimento de área navegável juntamente à detecção de obstáculos.

As informações de posicionamento e orientação globais do GPS, velocidade das rodas e IMU são fundidas em um Filtro de Kalman (UKF, Unscented Kalman Filter). O filtro é utilizado para estimar a posição do veículo e melhorar a precisão, principalmente quando o sinal do GPS é perdido. Foram utilizados dois modelos matemáticos para descrever o movimento do veículo, um para quando há sinal de GPS e outro para sua ausência. No primeiro modelo, o veículo pode mover-se livremente em qualquer direção, levando em consideração o terreno escorregadio onde as rodas podem derrapar facilmente. No segundo caso, há uma restrição de movimentação que utiliza a orientação como parâmetro, já que a velocidade do veículo era reduzida quando o sinal do GPS era perdido.

O método utilizado pelo Stanley para o reconhecimento da área navegável era baseado na geração de um mapa de duas dimensões baseados no sensoriamento dos LIDARs, câmeras e radares. O mapa derivado do LIDAR era responsável por encontrar as bordas da rua para que o veículo pudesse navegar centralizado na pista. Era feita então uma classificação de área em espaço ocupado, livre e desconhecido. A primeira classificação era feita comparando os pontos adquiridos diretamente do sensoriamento do LIDAR, verificando a diferença de altura e quando maior que um certo limiar, a região era considerado obstáculo. Porém só essa análise não era suficiente, pois algumas regiões eram classificadas erroneamente devido a erros de rotação e inclinação do veículo que poderiam fazer com que ele alterasse a rota desnecessariamente. Esses erros ocorriam geralmente quando a distância de tempo de leitura dos dados era muito grande, em vista disso, para modelar esse erro, foi utilizada uma Cadeia de Markov de primeira ordem a fim de realizar testes probabilísticos para inferir a ocorrência de obstáculos.

Tendo em vista que a distância que um LIDAR consegue detectar é por volta de 25 metros a frente do veículo, a equipe utilizou recursos da visão computacional para controlar a velocidade baseando-se na capacidade das câmeras, que podem detectar mais longe do que o LIDAR, entretanto as câmeras não foram utilizadas para tomar decisões de esterçamento.

Uma classe de algoritmos de aprendizado de máquina conhecido como boosting and decision stumps foi utilizada para treinar um classificador de área navegável. O método armazenava na memória uma mistura de gaussianas referentes aos três canais de cores (vermelho, verde e azul) para efetuar uma adaptação aos diferentes cenários ou iluminação. Finalmente, um módulo de avaliação de superfície determinava uma velocidade segura. Os radares seriam utilizados para

mesma finalidade, porém problema técnicos com USB levaram a equipe a abandonar o sensor.

A camada de controle era responsável por atuar diretamente no esterçamento, aceleração e frenagem. O módulo chave era o planejador de trajetória que utilizava os dados do arquivo RDDF (Route Data Definition File) e dos mapeamentos dos sensores para gerar uma trajetória e também transformar em dados de velocidade e esterçamento para alimentar o controle. Esse é um modelo matemático que apresenta grande dificuldade de modelagem por conta do solo escorregadio o qual dificulta a navegação.

A interface de direção drive-by-wire, freio e acelerador foram tratados na camada do veículo, assim como o regulador de energia. A camada do usuário cuidava da interface interativa com tela touch-screen para manipular os softwares e o E-STOP. Por último, a camada de serviços globais realizava o monitoramento do sistema, a geração de registros de dados, a comunicação entre processos e a sincronização do tempo do servidor.

No DARPA Urban Challenge, em 2007, a equipe desenvolveu uma nova plataforma autônoma batizada de Junior, cujo veículo era um modelo Volkswagen Passat Wagon. A arquitetura do software baseada na troca de mensagens não era muito diferente do antecessor Stanley. No entanto, novos módulos foram adicionados, como rastreamento de obstáculos e o sistema para definir as missões. Nesta competição, um novo arquivo de descrição do mapa foi entregue aos competidores. O RNDF (Route Network Definition File) continha as informações do posicionamento geométrico das faixas, sinalizações horizontais e verticais, estacionamentos e pontos especiais (checkpoints), visto que as missões eram baseadas em uma sequência de checkpoints. Os veículos tinham que completar as missões obedecendo as regras de trânsito.

O principal sensor utilizado pela equipe foi o LIDAR de 360° da Velodyne de 64 feixes, que realiza uma varredura de 360° ao redor do veículo. Neste evento, foram utilizadas técnicas de rastreamento de objetos utilizando filtro de partículas. A localização também usou um Filtro de Kalman UKF que funcionava basicamente como um histograma de uma dimensão. Apesar das inovações da equipe de Stanford, a equipe vitoriosa foi a Tartan Racing da Universidade Carnegie Mellon, com o veículo Boss, apoiada pela General Motors. O veículo completou as missões com aproximadamente 20 minutos de diferença em relação ao Junior, que ficou com a segunda colocação.

A equipe Tartan Racing utilizou um veículo Chevrolet Tahoe, equipado com vários sensores LIDAR de feixe único e também 360°, câmeras, GPS e IMU, ambos de alta precisão. Uma das principais vantagens da equipe foi o sistema chamado de TROCS (Tartan Racing Operator Control System), trata-se de uma interface gráfica que centraliza todas as operações do veículo, tais como gerenciar as missões, telemetria e registro de dados. A Figura 19 mostra a arquitetura simplificada do veículo Boss baseada em quatro camadas: percepção, planejador de missão, execução de comportamento e planejador de movimento.

A percepção processa dados dos sensores do veículo e fornece os recursos para outros

Figura 19 – Arquitetura simplificada do veículo Boss

Fonte: Adaptada de Baker e Dolan (2009).

módulos do sistema. As principais medições dos sensores são a posição e status do veículo, localização de obstáculos que gera um mapa local e o modelo da rua. O planejador da missão contém todas as metas parciais que se deseja alcançar, dessa forma ele calcula a rota mais rápida para atingir o próximo objetivo da missão. A execução de comportamento combina os dados dos dois módulos anteriores para gerar objetivos menores que são executados pelo planejador de movimentos. A Figura 20 apresenta o fluxo de dados dos módulos utilizados pelo veículo Boss.

Figura 20 – Funcionalidades da camada de Execução de Comportamento do veículo Boss

Fonte: Adaptada de Baker e Dolan (2009).

3.3 Simuladores

A simulação é um processo fundamental para o desenvolvimento de projetos, ela permite compreender o comportamento de um sistema, avaliar métodos e otimizá-los. Segundo [Wolf et al. \(2009\)](#) e [Banks et al. \(2004\)](#), dentre as principais vantagens da simulação destacam-se a economia de recursos e de tempo, além da capacidade de repetibilidade e reprodutibilidade de experimentos. Sistemas complexos podem ser testados exaustivamente a fim de revelar falhas e assegurar a robustez do sistema.

Os dados gerados por um simulador devem corresponder adequadamente aos dados que podem ser obtidos a partir de um sistema real ([BANKS et al., 2004](#)). Simuladores podem ser classificados como restritos, os quais simulam algum componente específico, ou abrangentes que podem simular uma situação envolvendo diversos elementos ([ECHEVERRIA et al., 2011](#)). Os simuladores do tipo abrangente são os mais utilizados em robótica móvel, pois permitem realizar experimentos de diversos componentes de um robô (sensores e atuadores) ou múltiplos robôs em diferentes cenários e situações, garantindo simular adequadamente as propriedades físicas como inércia, gravidade e atrito. No entanto, simular eventos do mundo real é um grande desafio pois o processo pode envolver inúmeras variáveis, sendo que modelos matemáticos podem não ser suficientes para representá-los. Portanto, para se criar um simulador, deve-se considerar um modelo sintetizado ou aproximado da realidade ([MILTON et al., 2006](#)).

No âmbito da Robótica móvel, três componentes fundamentais devem ser analisados: sensores, atuadores e o comportamento físico ([WOLF et al., 2009](#)). Os sensores representam equipamentos como LIDAR, radar, sonar, câmeras, GPS, unidade inercial, entre outros. Um dos principais problemas dos sensores simulados é a fidelidade dos dados, que em geral são perfeitos e não servem para validar experimentos. Por esse motivo, pode-se aplicar alguma função que adicione erros aleatórios de forma a inserir ruído nas informações dos sensores. O atuador pode ser visto como a própria plataforma robótica, ou dividido em partes como manipuladores robóticos. O comportamento físico está relacionado a simulação de forças físicas.

Existem diferentes simuladores disponíveis para atingir um mesmo propósito. Cada simulador possui características diferentes e apresentam vantagens e desvantagens em relação ao desempenho computacional, precisão, qualidade gráfica, dimensionalidade (2D ou 3D), tipos de sensores disponíveis, plataforma de sistema operacional e muitas outras. Em geral, não é possível otimizar todas essas características em um mesmo simulador, sendo que a escolha mais adequada envolve vários fatores referentes aos requisitos específicos de um sistema ([QUIGLEY; GERKEY; SMART, 2015](#)).

Atualmente, existem muitos simuladores, tanto sob licença comercial quanto em código aberto. Alguns dos simuladores de robôs mais conhecidos são Player/Stage, Gazebo, MORSE, Webots e Microsoft Robotics Studio. Simuladores específicos de veículos são mais restritos, sendo poucos de código aberto como o simulador TORCS. As grandes companhias automo-

tivas desenvolveram ou utilizam ferramentas que melhor simulam a dinâmica veicular como MATLAB/Simulink e o CarSim/TruckSim.

MATLAB¹ e Simulink², desenvolvidos pela MathWorks, estão entre as ferramentas mais famosas utilizadas por pesquisadores e engenheiros e possuem uma linguagem própria e bastante conhecida pela manipulação de matrizes. Uma das principais características do MATLAB é a infinidade de complementos chamados de toolbox que podem ser adquiridos separadamente. Simulink é uma ferramenta de simulação baseada na diagramação de blocos, muito utilizada em sistemas dinâmicos. CarSim e TruckSim são simuladores veiculares utilizados por diversas empresas, possuem conexão com o MATLAB e podem ser utilizados para Hardware-in-the-Loop. O PreScan é uma ferramenta de simulação com alto potencial para simular ADAS e sistemas cooperativos que utilizam comunicação V2V e V2I.

Webots³ é um simulador de robôs comercial utilizado por muitos centros acadêmicos e empresariais. Uma das vantagens é a ferramenta de prototipagem rápida que permite ao usuário representar o seu problema facilmente, construindo um mundo tridimensional sem que seja necessário habilidade em modelagem gráfica. Ele possui várias linguagens de programação (C, C++, Java, Python e Matlab) e a possibilidade de acelerar ou reduzir a velocidade do tempo de simulação. A ODE garante a simulação física de inércia, fricção e colisão, possuindo também recursos para comunicação com middlewares como ROS.

O simulador da Microsoft Robotics Studio⁴ pode ser obtido gratuitamente para fins não lucrativos, possuindo suporte para várias plataformas robóticas e sensores do mercado. Com o Microsoft Visual Studio (ambiente de desenvolvimento integrado, IDE) o MRS pode desenvolver projetos extensos, onde a linguagem principal é C Sharp e o .NET. O software possui interface amigável e inclui uma ferramenta drag and drop blocks onde o usuário cria seu ambiente de simulação e determina o funcionamento do seu robô apenas arrastando blocos que representam a codificação. Além disso, o simulador desfruta do PhysXTM (motor de simulação física) da NvidiaTM, que é utilizado em vários jogos e programas de modelagem 3D. Isso proporciona ao MRS criar ambientes extremamente realísticos e interações físicas com alta fidelidade. A equipe da Universidade de Princeton⁵ utilizou esse simulador no evento DARPA Urban Challenge.

O projeto Player/Stage, criado no ano 2000 por pesquisadores da Universidade do Sul da Califórnia, foi um dos simuladores mais utilizados em projetos acadêmicos e um dos primeiros a ser chamado de framework para programação de robôs (COLLETT; MACDONALD, 2005). O Player consiste em um paradigma cliente/servidor que utiliza TCP sockets para realizar a comunicação entre os processos e equipamentos envolvidos, com isso, consegue controlar diferentes equipamentos como sensores e atuadores ou mesmo plataformas robóticas completas,

¹ <<http://www.mathworks.com/products/matlab/>>

² <<http://www.mathworks.com/products/simulink/>>

³ <<https://www.cyberbotics.com/overview>>

⁴ <<https://www.microsoft.com/en-us/download/details.aspx?id=29081>>

⁵ <http://archive.darpa.mil/grandchallenge/TechPapers/Princeton_University.pdf>

como o ActivMedia Pioneer 2 que foi a plataforma original utilizada pelos desenvolvedores do Player. A estrutura distribuída do Player garante a conexão de vários clientes a um servidor, que podem realizar requisições de dados aos sensores ou mesmo escrever comandos nos atuadores. Várias linguagens de programação são utilizadas para criar programas no Player, isto se deve ao fato dele utilizar a estrutura de comunicação por sockets, onde C e C++ são as principais linguagens utilizadas. Desta forma, Player é uma interface de controle de componentes, tais componentes podem ser substituídos por componentes virtuais dentro do simulador Stage.

O Stage é um ambiente de simulação muito robusto e pode simular interações entre uma equipe de robôs trabalhando cooperativamente, apesar de ser um simulador 2D. Mapas podem ser criados a partir de imagens simples uma vez que são interpretadas pelo Stage como obstáculos ou paredes. Esse simulador é fundamentado em plataformas robóticas da Pioneer, ainda assim é possível criar e especificar robôs diferentes e posicionar os sensores e atuadores de forma adequar-se aos problemas específicos dos usuários. Uma das suas principais limitações, além de ser um simulador 2D, é ausência de simulação física.

O Gazebo é um dos simuladores de código aberto mais conhecidos entre os pesquisadores. Atualmente é o simulador padrão do ROS, o que facilita a sua integração com todo o framework. Foi desenvolvido com a biblioteca gráfica OGRE e inicialmente com a biblioteca física ODE, porém hoje é possível utilizar a Bullet. É um simulador bastante completo em termos de sensores e plataformas robóticas e customizável, ao passo que pode ser adaptado para simular ambiente como o DARPA Urban Challenge (OZGUNER et al., 2008) e desenvolvimento de sistemas do Hardware-in-the-Loop (SWANSON et al., 2013).

O projeto de simulação MORSE (Modular OpenRobot Simulation Engine) foi desenvolvido em 2011 nos laboratórios de pesquisa LAAS-CNRS (Laboratory for Analysis and Architecture of Systems - Centre national de la recherche scientifique) e ONERA (Office national d'études et de recherches aérospatiales). A principal proposta do simulador é proporcionar aos usuários um ambiente de simulação de alta complexidade e realismo, aproveitando o potencial disponibilizado pelo Blender (ECHEVERRIA et al., 2011). O simulador conta atualmente com diversos sensores, atuadores e robôs, além de alguns cenários prontos. Possui também a flexibilidade de se comunicar com vários middlewares, incluindo o ROS, o que permite o desenvolvimento rápido de Software-in-the-Loop. A Tabela 1 resume todas essas informações.

Nos eventos como o do DARPA Challenge a simulação de toda arquitetura dos veículos torna-se muito importante. A análise de vários artigos referentes ao evento revelou o uso de simuladores para o desenvolvimento dos veículos (BACHA et al., 2008a) (BOHREN et al., 2008) (PATZ et al., 2008). Porém não foi possível encontrar informações detalhadas e específicas dos simuladores utilizados.

A equipe VictorTango de Virginia Tech, desenvolvedores do veículo Odin, relatam o uso intenso de simulação durante o desenvolvimento do software, garantindo a eles a terceira colocação. O simulador foi utilizado durante todas as fases de desenvolvimento e incluiu tes-

Tabela 1 – Exemplo de simuladores utilizado em robótica ou no desenvolvimento veicular

Simulador	Licença	Tipo	Característica
MATLAB/Simulink	Proprietário	Geral	<ul style="list-style-type: none"> - baseado em modelos - alta precisão na simulação de dinâmica - possibilidade de controlar base móvel - pouco customizável - funcionalidade SIL e HIL
CarSim/TruckSim	Proprietário	Veicular	<ul style="list-style-type: none"> - baseado em modelos - alta precisão na simulação de dinâmica - possibilidade de controlar base móvel - pouco customizável - funcionalidade SIL e HIL
PreScan	Proprietário	Veicular	<ul style="list-style-type: none"> - diversidade de sensores - simulação ADAS - simulação V2V e V2I - funcionalidade SIL e HIL
Webots	Proprietário	Geral	<ul style="list-style-type: none"> - diversidade de robôs e sensores - comunica-se com vários middlewares
Microsoft Robotics Studio	Proprietário	Geral	<ul style="list-style-type: none"> - alta qualidade gráfica - última versão estável 2012
TORCS	GNU GPL	Veicular	<ul style="list-style-type: none"> - precisão da simulação de dinâmica - dificuldade na aquisição de dados
Player/Stage	GNU GPL	Geral	<ul style="list-style-type: none"> - facilidade de uso - fácil integração com alguns robôs - ambiente 2D - última versão estável 2010
Gazebo	Apache 2.0	Geral	<ul style="list-style-type: none"> - alta qualidade de simulação física e gráfica - diversidade de sensores e robôs - difícil criação de cenários - difícil criação de robôs - funcionalidade SIL através de middlewares
MORSE	BSD	Geral	<ul style="list-style-type: none"> - permite desenvolver novos sensores - alta qualidade de simulação física e gráfica - biblioteca pronta para desenvolver veículos - diversidade de sensores e robôs - difícil criação de cenários - difícil criação de robôs - funcionalidade SIL através de middlewares

tes de validação da arquitetura de software, simulação de situações e até mesmo simulação hardware-in-the-loop, sendo esse último também utilizado como software de visualização. De acordo com [Bacha et al. \(2008b\)](#) a facilidade do uso do simulador que permite que diferentes desenvolvedores tenham rápido retorno dos seus sistemas. Segundo [Patz et al. \(2008\)](#) o fato de vários desenvolvedores não terem disponibilidade de tempo para testar diferentes módulos no veículo autônomo real, porém uma ferramenta de simulação foi disponibilizada para testes simulados por várias pessoas do time, sanando este problema.

3.4 Considerações Finais

Este capítulo introduziu alguns conceitos chave relacionados a arquitetura de robôs móveis, bem como detalhou a arquitetura dos veículos vitoriosos do eventos promovidos pelo DARPA. O estudo da arquitetura permite compreender o funcionamento dos veículos autônomos.

Ao final do capítulo, são apresentados alguns dos simuladores mais utilizados no mundo da robótica. Alguns requisitos foram avaliados para a escolha do simulador mais adequado para este trabalho. Neste trabalho, almeja-se desenvolver uma arquitetura de simulação para veículos autônomos, bem como para múltiplos veículos. Portanto, inicialmente, foi avaliada a flexibilidade de adaptação dos simuladores, sendo que os simuladores de licença de código aberto são mais propensos à expansões do que proprietários. Também é desejado um simulador com boa qualidade de geração gráfica e física. Uma importante característica é a possuir comunicação com diversos middlewares para facilitar a integração e portabilidade de códigos. A disponibilidade de ferramentas prontas para desenvolver veículos é de altíssimo interesse, sendo que o simulador MORSE atende a todos esses requisitos, portanto, foi o simulador escolhido para desenvolver a arquitetura de simulação da plataforma CaRINA 2, e também o simulador de múltiplos veículos.

FERRAMENTAS COMPUTACIONAIS E SIMULADORES

Este capítulo tem por objetivo apresentar as ferramentas e o simulador utilizado neste trabalho. O funcionamento do sistema ROS é analisado em detalhes, assim como simulador MORSE.

4.1 Robot Operating System

O projeto iniciado pelo Laboratório de Inteligência Artificial de Stanford em 2007 é hoje um dos softwares mais utilizados pelos pesquisadores da área da robótica. Em 2010 foi lançada a primeira versão estável do sistema e de código fonte aberto sob licença BSD (Berkeley Software Distribution). O sistema é utilizado por importantes centros de pesquisa ou mesmo na indústria onde uma série de plataformas, sensores e atuadores de diversos fabricantes foram adaptados e padronizados no sistema ROS. Dentre vários grupos de pesquisas e fabricantes que utilizam o sistema, pode-se citar: UC Berkeley, Rethink Robotics, Pi Robot, UT Austin, Bosch Research and Technology Center e Oregon State University.

O sistema ROS é considerado um framework de desenvolvimento que facilita a modularização e reutilização de código, além de fornecer diversas ferramentas para visualização, simulação e gravação de dados para análise futura (MARTINEZ; FERNANDEZ, 2013). Sua estrutura foi baseada em sistemas operacionais como Unix e por isso apresenta vantagens como a camada de abstração de hardware que permite ao usuário utilizar vários dispositivos sem ter que lidar com o software o qual faz interface direta com o hardware. A troca de mensagens entre processos é baseada em uma arquitetura de grafos com topologia centralizada (MARTINEZ; FERNANDEZ, 2013), portanto, os processos são tratados como nós de um grafo, como mostrado na Figura 21. Existe um processo mestre (principal) chamado de roscore que gerencia os sub-processos.

Figura 21 – Exemplo de grafo no ROS

ROS foi desenvolvido para ser modular e com granularidade fina, onde os nós representam processos ou programas no sistema, quando o sistema todo é analisado durante a execução, nota-se a formação de um grafo. Para realizar a comunicação entre os nós são utilizadas mensagens. Uma mensagem é tipicamente descrita como um tipo de dado primitivo (inteiro, decimal, booleano, entre outros) ou uma estrutura de dados complexa. A troca de mensagens entre os nós pode ser feita através de tópicos ou serviços. Um nó pode publicar uma mensagem através de um tópico ficando disponível para outros nós subscrever-las, portanto, tópicos enviam mensagens em broadcast. Tópicos são definidos como `msg` e possuem a característica de ser assíncrono. Já no caso dos Serviços, esses são definidos como `srv` e possuem característica de ser síncronos, baseados na estrutura pedido/resposta.

A Figura 22 exemplifica a troca de mensagens por meio de tópicos. O nó A faz a publicação da mensagem em uma determinada frequência. Os nós B e C subscrevem a publicação e podem ser executados em diferentes frequências, caracterizando o método como assíncrono. Em determinadas situações é viável configurar uma fila de dados para que não seja descartada nenhuma mensagem da publicação, garantindo que os nós B e C recebam todas as mensagens.

Figura 22 – Exemplo de publicação e assinatura de tópicos

Os serviços, apresentados na Figura 23, possuem funcionamento similar ao de um servidor web services. O nó A fica aguardando uma chamada e quando os nós B ou C geram

uma requisição ele são imediatamente atendidos com uma resposta do nó A, garantindo assim o sincronismo da transação.

Figura 23 – Exemplo de requisição e resposta através de serviços

Os termos publicação e subscrição de mensagens serão comumente utilizados daqui em diante neste trabalho. Além disso, os diagramas apresentados nas Figuras 22 e 23 quando representarem a união de diversos módulos e coerente fluxo de dados serão referenciados como arquitetura.

4.2 Simulador MORSE

Simulações são essenciais na pesquisa e desenvolvimento em geral, elas permitem avaliar e validar métodos ou estratégias de maneira rápida, econômica e segura. Na robótica, as simulações devem proporcionar a interação de diversos elementos com um mundo virtual compreendido de características físicas. A proposta do simulador MORSE é oferecer maneiras de testar e avaliar um conjunto de processos ou missões complexas, utilizando para esse fim, o motor de jogo para trazer o realismo necessário às simulações (LEMAIGNAN et al., 2012).

Blender é um software de modelagem de objetos tridimensionais que também possibilita a renderização de imagem e animações. Ele possui alta sofisticação gráfica e viabiliza a modelagem de malhas 3D extremamente detalhadas com texturização e iluminação em tempo real, sendo muito utilizado por projetistas gráficos. Ao longo dos anos Blender incorporou suas próprias ferramentas para o desenvolvimento de jogos que ficou conhecido como Blender Game Engine ou BGE, utilizando o estado da arte das bibliotecas de interações físicas da Bullet e gráficos do OpenGL.

MORSE foi projetado para lidar com ambientes variados e se adaptar a diversas arquiteturas existentes, utilizando as ferramentas fornecidas pelo Blender. A estrutura do MORSE separa a simulação dos componentes (sensores e atuadores), da interface com os middlewares (como o ROS), característica que facilita sua incorporação nos projetos. Essa versatilidade permite realizar experimentos com característica software-in-the-loop, no qual um programa pode ser testado no simulador através da comunicação de algum middleware sem grandes, ou

nenhuma, alteração nos códigos. No entanto, MORSE não está vinculado a nenhum middleware ou arquitetura específica.

A inicialização de uma simulação ocorre da mesma forma que um jogo, cujo módulo principal é a BGE, e é dada através de um script de montagem. MORSE organiza a simulação através de um núcleo principal que coordena os eventos da BGE e uma coleção de componentes utilizados para a montagem dos robôs. Os componentes também podem ser conectados aos middlewares, como especificado no script de montagem, permitindo a comunicação com elementos externos ao simulador.

Alguns componentes como sensores e atuadores possuem a versatilidade de trabalhar em diferentes níveis de realismo e abstração. Os dados dos componentes podem ser alterados através dos modificadores para adicionar uma perturbação ou ruído nos dados ou mesmo alterá-los conforme desejado. O fluxo de dados do MORSE pode ser visto no diagrama da Figura 24.

Figura 24 – Diagrama do fluxo de dados entre o Blender e um middleware

O simulador MORSE fornece uma variedade de modelos de sensores, atuadores e robôs para uso imediato. Alguns dos sensores implementados são: Sick, Velodyne, câmeras, GPS, IMU e até mesmo sensor térmico. Alguns modelos de atuadores também estão disponíveis como o KUKA LWR e Mitsubishi PA-10. Também pode-se utilizar o Blender como modelador 3D, onde qualquer tipo de robô pode ser modelado e implementado diretamente no simulador. Algumas imagens do simulador podem ser vistas na Figura 25 e 26.

Figura 25 – Alguns dos cenários e robôs disponíveis no MORSE.

Fonte: <<https://www.openrobots.org/morse>>.

Figura 26 – Utilização do simulador para interação robô e humano .

Fonte: <<https://www.openrobots.org/morse>>.

ARQUITETURA DE SOFTWARE

Este capítulo apresenta uma discussão sobre a arquitetura da plataforma CaRINA e o desenvolvimento do modelo simulado, bem como sua adaptação para a arquitetura do CaRINA 2. Na Seção 5.1 os módulos da arquitetura do CaRINA 2 são apresentados de modo genérico, permitindo ter uma visão geral. Em seguida, na Seção 5.2, é apresentada a arquitetura específica do CaRINA 2, no qual os módulos são detalhados em profundidade, a fim de se identificar quais foram substituídos por simulação e também compreender o funcionamento de toda plataforma. O desenvolvimento do modelo simulado é apresentado na Seção 5.3, onde também são citadas as estratégias para adaptação do simulador na arquitetura do CaRINA 2.

5.1 Arquitetura Geral

A arquitetura da plataforma CaRINA 2 foi baseada no framework ROS. A sua utilização facilita a comunicação de processos, prototipagem rápida e interface com diversos sensores disponíveis no mercado. Além disso a modularidade do sistema permite que a arquitetura seja facilmente adaptada para se utilizar em outras plataformas, como mostrado na Seção 6.3.

A comunicação entre processos ou módulos, feita através de assinatura e subscrição de mensagens, se assemelha ao sistema utilizado pela equipe de Stanford nos veículos Stanley e Junior, no qual os processos são executados separadamente e em paralelo. Os principais elementos da arquitetura são mostrados na Figura 27, onde é possível notar seis camadas fundamentais como localização, mapa, plataforma (veículo), sistema de controle, percepção e diagnóstico.

A camada da localização é indispensável para navegação em ambiente urbano, ela representa a capacidade do veículo (robô) estimar seu posicionamento global ou local. Porém, em geral, a localização está altamente associada a um mapa ou algum tipo de representação do ambiente, o qual é utilizado pelo veículo como referência para navegação. Nos eventos DARPA, um mapa foi previamente construído e entregue aos participantes.

Localização e mapa são informações de entrada para o sistema de controle do veículo, que utiliza diferentes métodos para verificar o posicionamento, corrigindo-o se necessário. Pode-se ainda subdividir o controle em longitudinal (ajuste de velocidade) e lateral (o qual realiza o esterçamento).

A tomada de decisões proveniente da camada de percepção representa a inteligência do veículo, que utiliza sensoriamento exteroceptivo que é utilizado para detectar os elementos ao redor do veículo. A camada do veículo compreende principalmente os atuadores e o sensoriamento proprioceptivo. As mensagens de atuação chegam até a camada do veículo no qual acelerador, freio e direção são acionados conforme a necessidade.

O sistema de auto diagnóstico, presente em vários casos mostrados na Seção 3.2, monitora o funcionamento do veículo e dos sensores e também verifica a qualidade do sinal do GPS. O módulo gera alertas em caso de mal funcionamento dos atuadores do veículo, sensores e baixo nível de sinal do GPS

Figura 27 – Diagrama de fluxo de dados da arquitetura geral composta basicamente por localização e mapa, plataforma robótica, controle, percepção e diagnóstico

5.2 Arquitetura CaRINA 2

A estudo arquitetural do CaRINA 2 torna-se importante para a compreensão de todos os componentes da plataforma e permite levantar os requisitos necessários para o desenvolvimento dos módulos que devem ser adaptados para a arquitetura de simulação para o CaRINA 2. Dessa maneira, inicialmente, será feita uma breve discussão do funcionamento atual da arquitetura e

seu entendimento leva à uma visão geral dos procedimentos para realizar a navegação autônoma do veículo.

A princípio, como mostrado no Capítulo 3.2, a construção de mapas é de extrema importância para navegação em ambiente urbano. Além de ser utilizado para localização e planejamento de rotas, também é possível incluir vários tipos de informações semânticas referentes as sinalizações verticais e horizontais das vias. Sendo assim, o passo inicial para navegação autônoma é dado através da construção de um mapa.

Os dados para geração do mapa são obtidos por meio do armazenamento das coordenadas de GPS de um percurso realizado pela navegação de um condutor humano habilitado. As coordenadas de GPS, que são convertidas para UTM em tempo de execução, são armazenadas em um arquivo. A geração do mapa é feita de modo off-line, ou seja, é necessário um pré-processamento onde calcula-se curvas Clothoid¹ e a velocidade permitida. Também são mantidos os dados de posicionamento e orientação, uma vez que também são utilizados pelo controle (FILHO; WOLF, 2014).

De posse do mapa, posiciona-se o veículo com orientação adequada e assim pode-se inicializar a navegação autônoma através do controle, esse que utiliza os dados do mapa e da localização para corrigir o posicionamento e a velocidade do veículo. A camada de percepção utiliza os sensores disponíveis para verificar se um dado objeto pode ser considerado um obstáculo, ou seja, se irá interferir na rota do veículo. Para isso são comparados o posicionamento do obstáculo em relação ao mapa e sua distância em relação ao veículo. As informações de aceleração e frenagem passam por um módulo juiz, que trata de diminuir a velocidade ou parar o veículo, caso o resultado da camada de percepção indique a presença de um obstáculo. A arquitetura do CaRINA 2 é mostrada na Figura 28, os módulos são detalhados a seguir.

- Localização

O sistema de localização do CaRINA 2 é atualmente baseado principalmente no GPS. O pacote é responsável por inicializar o receptor e publicar as mensagens obtidas pelo mesmo. Essas que representam dados de posicionamento global em coordenadas geodésicas e orientação (bússola). Também é disponibilizada a qualidade do sinal e número de satélites.

A qualidade e precisão dos dados do GPS pode interferir criticamente na navegação do veículo. A fim de melhorar a precisão do posicionamento, o módulo RTK do receptor GPS deve ser ativado. A correção de posicionamento é disponibilizada por uma base fixa através de um serviço NTRIP, acessado por rede. Uma vez que os dados de correção estão disponíveis são enviados para o receptor do GPS por uma interface serial.

Após o GPS receber as devidas correções e melhorar a acurácia do posicionamento, é necessário converter as coordenadas geodésicas para um sistema baseado em coordenadas

¹ <<http://msemac.redwoods.edu/~darnold/math50c/CalcProj/Fall07/MollyRyan/Paper.pdf>>

Figura 28 – Diagrama de fluxo de dados da arquitetura do CaRINA 2. Cada módulo, em verde, representa um conjunto de funcionalidades (pacote). As mensagens, em cinza, mostram os tipos de dados e tópicos específicos do ROS

cartesianas conhecido com UTM. O pacote é responsável por converter as coordenadas geodésicas em UTM e disponibilizar a localização corrente e orientação do veículo para vários outros módulos.

- Mapa O mapa é construído através da navegação prévia de um condutor, no qual são armazenados os dados das coordenadas UTM e orientação. Curvas do tipo Clothoid são geradas em um pré-processamento, em que também são calculadas as velocidades. O mapa então é constituído de uma lista de coordenadas UTM, orientação, curvaturas de clothoid e velocidade.

- CaRINA 2 - Veículo

Esse pacote é constituído de três grandes módulos que caracterizam as funções de atuação do CaRINA 2. Eles se comunicam diretamente com os dispositivos através de drivers com a finalidade de esterçar, frear e acelerar. Vale mencionar que os módulos podem ser acionados individualmente.

Atualmente, a maioria dos veículos fabricados pelas montadoras apresentam uma Unidade de Controle Eletrônica, o qual se comunica dos diversos dispositivos do veículo através de uma rede CAN. Uma das funcionalidades da rede é transmitir o diagnóstico do veículo. Uma infinidade de dados são transmitidos pela rede e examinados pelo módulo `can_node`, que procura separar dados relevantes como velocidade, posição do câmbio, status dos freios e giros do motor. Esses dados são utilizados para compor uma mensagem de status do veículo, `Vehicle State`.

O esterçamento é realizado por um motor DC acoplado à barra de direção do próprio veículo, onde também se encontra um encoder que monitora os giros do motor, esses elementos são conectados ao controlador de motor. Esse último também é responsável por acionar o atuador linear que está ligado diretamente ao pedal do freio. Ambos podem ser ligados e desligados por questões de segurança, ou mesmo conforme a necessidade do experimento.

O sistema de aceleração, específico deste modelo de automóvel, é formado basicamente por um pedal que se interconecta a dois potenciômetros. Quando um motorista aciona o pedal do acelerador ele está na verdade movendo os potenciômetros, os quais geram dois sinais diferentes, criando-se uma redundância que torna o sistema tolerante a falhas. O sistema do veículo não foi alterado, para tanto, um sinal é gerado por uma placa de entrada/saída do computador do CaRINA 2 e antes de finalmente chegar ao pedal, o sinal é replicado e devidamente calibrado para se assemelhar ao gerado pelo par pedal-potenciômetro do veículo.

- Sistema de Controle

O Sistema de Controle constitui o pacote `cari_nav_cruise_control` e é dividido em duas partes, controle longitudinal e lateral, `control_node` e `steering_node` respectivamente. O módulo do controle lateral é responsável pelo esterçamento do veículo. Ele faz uma busca na lista de pontos do mapa para encontrar o mais próximo, que deve ser seguido. Para isso, utiliza dados do posicionamento corrente e também do status do veículo, esses que são parâmetros de entrada para o calcular o esterçamento (FILHO; WOLF, 2014). Ao final da operação uma mensagem `AckermannDriveStamped` é gerada, sendo composta de esterçamento e velocidade.

O módulo de controle longitudinal subscreve a mensagem gerada pelo módulo anterior e realiza cálculos para tentar manter a velocidade desejada. O valor obtido é subdividido em termos de aceleração e frenagem dependendo da situação em que o veículo se encontra.

- Percepção

A camada de percepção é constituída em utilizar sensores para detectar possíveis obstáculos ao redor do veículo. Embora existam vários estudos que utilizam câmeras já em fase de teste, o principal dispositivo para realizar a percepção do CaRINA 2 é o sensor LIDAR 360° da Velodyne.

O ROS possui interfaces prontas pra se comunicar com o Velodyne e publicar a nuvem de pontos gerada pelo sensor em um mensagem padronizada conhecida como `PointCloud2`. A nuvem de pontos tridimensional é planificada e publicada no formato de um LIDAR planar, `LaserScan`. Esse processo permite evitar alguns falsos positivos na classificação de obstáculos. A classificação de uma detecção considera um objeto como obstáculo quando esse interferir na rota do veículo. Para isso são comparados o posicionamento do objeto em relação ao mapa e sua distância em relação ao veículo. As informações de aceleração e frenagem passam por um módulo juiz, que trata de diminuir a velocidade ou parar o veículo, caso o resultado da camada de percepção indique a presença de um obstáculo.

- Diagnóstico

Todos os equipamentos estão sujeitos a problemas de funcionamento devido a uma série de fatores como interferências, má conexão de cabos ou mesmo por falha humana na operação de todo software. Para isso, foi desenvolvido um módulo específico de diagnóstico que monitora o funcionamento dos sensores e atuadores.

Como a navegação do CaRINA 2 é baseada em GPS, o sistema de diagnóstico também realiza um monitoramento da qualidade do sinal e avalia a precisão do posicionamento. O sistema gera alertas ao operador do veículo autônomo e facilita a busca dos problemas.

5.3 Modelo Simulado do Veículo

5.3.1 Simulação

Blender, como mencionado anteriormente, é uma ferramenta de modelagem tridimensional que permite modelar diferentes tipos de robôs, braços mecânicos e até mesmo veículos. A Game Engine fornece ferramentas diversas para modelar veículos aproveitando a simulação física proporcionada pela Bullet, tornando possível a simulação do efeito da aceleração, gravidade, suspensão, colisão e até mesmo atrito entre o pneu e vários tipos de solo. Logo, o MORSE pode desfrutar de todos esses recursos além da alta capacidade de detalhamento gráfico em tempo real.

Apesar do simulador MORSE disponibilizar vários robôs, foi necessária a criação de um robô específico no Blender que se adéqua ao projeto CaRINA 2. A partir de um veículo previamente modelado em 3D (semelhante ao CaRINA 2) composto basicamente de carroçaria e rodas, como mostrado na Figura 29, foi utilizada uma biblioteca de funções (scrips) conhecida como Vehicle Wrapper que permitiu implementar o modelo veicular aproximado do veículo CaRINA 2. A função `addWheel` é a mais importante da biblioteca, pois prepara o modelo para que mais tarde ele possa receber comandos de aceleração, frenagem e esterçamento. Essa função recebe parâmetros relacionados ao posicionamento das rodas, força e ângulo da suspensão, sendo possível também configurar individualmente o esterçamento de cada roda. O sistema de suspensão do veículo ainda pôde ser melhorado através de funções específicas que configuram a rigidez do amortecimento e atrito do pneu.

Figura 29 – Veículo 3D simplificado

Com a intenção de simplificar o desenvolvimento de jogos, Blender traz uma ferramenta no estilo drag and drop chamada de Logic Bricks que é capaz de conectar scripts aos objetos 3D. Na Figura 30 existem três colunas, sensores, controladores e atuadores. Os sensores são elementos utilizados para inicializar um script, podendo ser automáticos ou através de alguma interface com o usuário. Os controladores são utilizados para conectar os sensores aos atuadores ou somente executar um script. Por último, os atuadores podem ser utilizados para diversos propósitos como, fazer interação direta com o objeto 3D. Essa ferramenta foi utilizada para

conectar os scripts ao modelo tridimensional, deixando o modelo preparado para receber as funcionalidades de um veículo.

As três funcionalidades básicas do veículo, aceleração, frenagem e esterçamento, são incorporadas através de um script que traduz às funções `applyEngineForce`, `applyBraking` e `setSteeringValue`, respectivamente. As funcionalidades ficam disponíveis durante toda execução da simulação, sendo eventualmente utilizadas quando necessário através de códigos externos ou mesmo utilizando o teclado que está associado às funções.

Figura 30 – Ferramenta drag and drop

A manipulação de variáveis em tempo de execução na simulação é dada através do uso de propriedades dos objetos, chamada de Logic Properties. Propriedades são variáveis associadas a objetos e instanciadas no início da execução da Game Engine, ou seja, no início da simulação e ficam disponíveis durante toda sua execução. A Figura 31 mostra as propriedades que foram adicionadas ao modelo, elas são utilizadas para controlar as funcionalidades básicas do veículo além de determinar se o veículo será controlado de maneira autônoma ou por meio de comandos manuais do usuário. Isso facilita a comunicação indireta entre MORSE e ROS através de sockets, que será usada posteriormente.

A texturização e iluminação dos objetos e cenários também tem um papel importante na simulação, principalmente em aplicações que envolvem o desenvolvimento de métodos relacionados à visão computacional ou mesmo quando o usuário é inserido no simulador como um jogo. Dessa forma, o modelo tridimensional do CaRINA 2 foi texturizado, a comparação entre o veículo real e o simulado pode ser vista na Figura 32. Alguns cenários também foram modelados e texturizados como mostrado na Figura 33 e na Figura 34, onde é possível ver o veículo em simulação.

Como não foi fornecido o modelo dinâmico do veículo pelo fabricante, os parâmetros de suspensão, aceleração e frenagem foram obtidos empiricamente com o propósito de obter um modelo adequado ao da realidade.

A arquitetura do MORSE ((ECHEVERRIA et al., 2011)) segue a filosofia Software-in-

Figura 31 – Propriedades de objetos

Figura 32 – Comparação entre o veículo real e o simulado

the-Loop, similar ao Hardware-in-the-Loop, esse método possibilita testar softwares na simulação e futuramente no hardware real sem quaisquer alterações em virtude do uso de middlewares como o ROS, YARP, Pocolibs além de permitir troca de dados por meio de sockets. Nesse caso o middleware ROS foi escolhido para manter a compatibilidade com a arquitetura do CaRINA 2.

O fluxo de dados do MORSE ocorre dentro da execução do jogo no Blender e estabelece os canais de comunicação específicos para troca de dados com os middlewares. Para inicializar uma simulação no MORSE é necessário um script que contenha a especificação de um cenário e os robôs que são manipulados.

Existem três componentes robóticos disponíveis: sensores, atuadores e robôs. Os sensores

Figura 33 – Cenários utilizados para simulação

fazem a aquisição de dados proporcionados pela Game Engine e torna-os acessíveis externamente. Os atuadores podem exercer algum movimento através de comandos requisitados externamente. Por último, os robôs são as plataformas nas quais são conectados sensores e atuadores.

Os componentes sensor e atuador podem registrar serviços que são disponibilizados publicamente para acesso externo através de canais específicos de middleware ou mesmo por sockets e texto. Os serviços são funções que ficam disponíveis durante toda execução da simulação e podem ser utilizados para alterar o comportamento de algum componente.

Para inserir o modelo do veículo no simulador MORSE foi necessário seguir a estrutura de codificação orientada a objetos do próprio MORSE. Para definir o comportamento do robô deve-se herdar a classe `core.robot.Robot` e implementar principalmente o método `default action`, neste caso foi preciso definir os serviços disponibilizados para acesso externo na simulação.

Finalizada a definição do comportamento do robô, agora é necessário escolher o modelo

Figura 34 – Texturização dos objetos

tridimensional que terá o papel visual e físico na simulação, bem como agregar os sensores que são utilizados. Isso é feito através de outra classe responsável por montar o robô, chamada de `builder.morsebuilder.Robot`. O veículo pode ser equipado com vários sensores como GPS, IMU, sensor LIDAR, câmera e velocímetro. A Figura 35 demonstra a estrutura a ser seguida para a criação de um novo robô.

Figura 35 – Arquivos necessários para criação de um robô no MORSE

A arquitetura de simulação do CaRINA 2 é ilustrada na Figura 36. A princípio, nota-se o relacionamento entre MORSE e a Blender Game Engine seguindo o fluxo de dados mencionado anteriormente, sendo que a simulação é executada internamente no Blender. Portanto, ao se inicializar a simulação, a Game Engine é acionada e os modelos tridimensionais são carregados

com suas respectivas propriedades. O MORSE se encarrega de montar o robô com os sensores, disponibilizar os serviços e a publicar dados através do middleware ROS. Os dados dos sensores ficam disponíveis no ROS nos seus respectivos tópicos. Os serviços, vistos na Figura 36, exercem a função dos atuadores no MORSE já que estão diretamente relacionados com as propriedades do modelo tridimensional e suas funcionalidades básicas como ativar o modo autônomo, acelerar, frear e esterçar. Eles são acionados por meio de uma mensagem enviada por sockets.

Desta forma, agora é possível utilizar um software para ler os dados dos sensores e processá-los para controlar o veículo quando necessário. O Capítulo 5.1 descreve a arquitetura do simulador que se adapta para receber a arquitetura do CaRINA 2 baseada no ROS.

Figura 36 – Arquitetura de Simulação do CaRINA 2

5.3.2 Arquitetura Simulador

A Seção 5.2 apresentou as camadas que compõe a atual plataforma do CaRINA 2. A arquitetura é composta por diversas conexões físicas entre o computador, sensores e atuadores. A análise permitiu diferenciar os módulos de software que fazem interface direta com os sensores e atuadores dos módulos pertinentes à inteligência e navegação do veículo.

A arquitetura de simulação foi criada com o propósito de realizar experimentos completos em laboratório antes de levar os métodos ou programas ao veículo real. Isso permite realizar experimentos que envolvem riscos ou uma logística complexa, poupando recursos e tempo. Para

isso, o simulador foi adaptado para ter o mesmo papel da plataforma real, ou seja, fornecer e receber dados nos mesmos padrões dos módulos de inteligência e de navegação.

A padronização dos tipos de mensagem do ROS, utilizados na implementação da arquitetura do veículo real, facilita a adaptação do simulador. Uma vez que os módulos produzem um tipo de mensagem específica e podem ser substituídos por módulos que simulam tais mensagens. Desse forma, identificando-se os componentes da plataforma CaRINA 2 e categorizando-os como os sensores e atuadores, é possível substituí-los por módulos simulados.

Como discorrido na Seção 5.2 e utilizando o diagrama arquitetural da Figura 28 pode-se identificar três importantes camadas da arquitetura do CaRINA 2, localização, percepção e a própria plataforma veicular. Essas camadas estão relacionadas diretamente aos sensores e atuadores, sendo que essas são substituídas por módulos simulados.

A camada da localização do veículo real pode ser resumida a um receptor de GPS e a correção RTK obtida pela internet que melhora a precisão do posicionamento global. Ao final desse processo, a camada de localização publica o posicionamento e orientação atuais do veículo através do módulo `global_localization`, que como mencionado anteriormente, na Seção 5.2, faz a conversão entre coordenadas geodésicas para coordenadas UTM. A mensagem publicada por esse módulo é do tipo `PoseStamped` e possui dois componentes, posição e orientação.

Na simulação da localização, são utilizados dois sensores para compor a mensagem `PoseStamped`. O veículo simulado possui um GPS e uma IMU, ambos sem adição de ruídos. O GPS publica coordenadas cartesianas obtidas diretamente do mundo simulado, assim como a IMU publica orientação baseado no seu sistema local. Para manter o mesmo padrão de mensagem, substitui-se o módulo `global_localization` original por um módulo simulado que utiliza subscrive tópicos do GPS e IMU para montar e publicar a mensagem `PoseStamped` no mesmo tópico `/carina2/currentPose`.

A plataforma veicular possui três componentes básicos para efetuar a navegação, aceleração, frenagem e esterçamento. Como mencionado anteriormente a aceleração é feita eletronicamente. Já o esterçamento e a frenagem são efetuados por atuadores. No veículo real uma mensagem do tipo `AckermannDriveStamped` é dividida em esterçamento e velocidade. O esterçamento é enviado diretamente ao RoboteQ que controla o motor acoplado à barra de direção e velocidade passa por um controle longitudinal sendo subdividida em aceleração e frenagem. Por fim, as informações de status do veículo são obtidos pelo barramento CAN utilizada para gerar uma mensagem do tipo `VehicleState`.

Como mencionado na Seção 5.3.1, no simulador a mensagem `AckermannDriveStamped` é enviada através de socket para um serviço do MORSE que atualiza as propriedades do veículo simulado. A mensagem também é subdividida em aceleração, frenagem e esterçamento. O esterçamento é feito instantaneamente através de uma simples atribuição utilizando a função `setSteeringValue` e a aceleração e frenagem são transformadas nos parâmetros utilizado pelas

funções `applyEngineForce` e `applyBraking` respectivamente. Além disso uma mensagem equivalente ao status do veículo real (`VehicleState`) é gerada com os dados do veículo simulado e externada por um serviço MORSE através de socket e, finalmente, publicada no ROS.

Apesar do MORSE possuir vários sensores, o sensor LIDAR de 360° da Velodyne não possui bom desempenho computacional na simulação pelo fato dele reproduzir uma nuvem de pontos contendo cerca de 700.000 pontos obtidos a cada segundo, portanto um sensor LIDAR planar similar ao Sick foi utilizado. A substituição justifica-se pois na percepção do veículo real a nuvem de pontos tridimensionais obtida pelo Velodyne é planificada. Nesse processo é gerada uma mensagem do tipo `LaserScan` compatível com a mensagem do LIDAR planar Sick. No caso da simulação, não se faz necessário o módulo de Diagnóstico, pois esse é utilizado para verificar a qualidade e o bom funcionamento dos sensores. A arquitetura de simulação é apresentada na Figura 37.

5.4 Considerações Finais

Neste capítulo foram apresentadas as arquiteturas pertinentes ao CaRINA 2, bem como sua adaptação para simulação. Inicialmente foi apresentada a arquitetura geral dividida em módulos básicos para se ter uma visão geral da arquitetura. Em seguida os módulos foram detalhados e analisados individualmente. Cada módulo foi especificado a fim de ressaltar sua utilidade para realizar a navegação autônoma e também identificar os módulos que podem ser substituídos por simulação. Por fim, é apresentado o desenvolvimento do modelo simulado e sua adaptação para a arquitetura do CaRINA 2.

Figura 37 – Arquitetura do CaRINA 2

RESULTADOS EXPERIMENTAIS

Este capítulo trata de apresentar e discutir os resultados obtidos através de testes experimentais envolvendo a arquitetura e a plataforma CaRINA 2 e o simulador. Na Seção 6.1 é apresentada a validação experimental da arquitetura de simulação. Na Seção 6.2 é mostrado a extensão do simulador para múltiplos veículos, bem como a sua validação. Finalmente, na Seção 6.3, é apresentada a adaptação e implementação da arquitetura em um veículo pesado, mostrando sua portabilidade e flexibilidade.

6.1 Validação Experimental

A fim de avaliar a arquitetura de simulação proposta na Seção 5.3.2, bem como a qualidade e desempenho, os dois veículos (real e simulado) foram submetidos a testes de navegação autônoma utilizando o mesmo mapa e o mesmo sistema de controle. Dessa forma foi possível avaliar e comparar os trajetos feitos por ambos em relação a um mapa comum.

Dois mapas foram usados como referência utilizando os métodos de localização apresentados na Seção 5.2. Os dados foram gravados com boas condições climáticas a fim de diminuir erros do GPS. O resultado do mapeamento chamado de Trajeto 1 e Trajeto 2 são apresentados respectivamente na Figura 38 e 39. A imagem do satélite é apenas uma referência ao trajeto real percorrido e identificado por uma linha em negrito, esses dados foram obtidos pelo GPS do CaRINA 2 que navegou sob controle de um condutor habilitado. Vale ressaltar que a escala do gráfico releva o posicionamento global em coordenadas UTM no eixo X e Y, valores que foram obtidos pelo do módulo Localização Global (global localization).

De posse dos mapas, o CaRINA 2 foi então submetido a percorrer os dois trajetos de modo autônomo, além disso os dados de erro lateral no sistema de controle e posicionamento global foram armazenados para comparação. Utilizando o mesmo mapa o veículo simulado foi também submetido a realizar os trajetos de maneira autônoma. No caso da simulação foi

Figura 38 – Trajeto 1. A linha em negrito representa os dados registrados com o GPS RTK e transformados para UTM.

Fonte: Adaptada de <https://maps.google.com.br/> utilizando dados reais.

Figura 39 – Trajeto 2. A linha em negrito representa os dados registrados com o GPS RTK e transformados para UTM.

Fonte: Adaptada de <https://maps.google.com.br/> utilizando dados reais.

utilizado um cenário aberto e plano em condições ideais, sem erros e falhas de GPS. Para uma análise justa o veículo foi posicionado e orientado aproximadamente nas mesmas condições que o CaRINA 2 no início da navegação autônoma, dados de erro lateral e de posicionamento global também foram armazenados para estudo. Por último, vale lembrar que o CaRINA 2 utilizou a arquitetura apresentada na Seção 5.2 e a simulação utilizou a arquitetura proposta na Seção 5.3.2, os algoritmos e parâmetros de controle foram mantidos iguais. No entanto, houve uma remoção de um parâmetro no controle lateral relacionada a um vício de orientação do volante o qual não se aplica à simulação.

6.1.1 Trajeto 1

O resultado da navegação no Trajeto 1 é mostrado na Figura 40 onde as cores preto, verde e vermelho identificam respectivamente a rota original, o trajeto autônomo realizado pelo CaRINA 2 e o simulado. Inicialmente é possível notar a coerência entre os dados da simulação e do veículo real, ambos possuem trajetórias equivalentes.

Figura 40 – Trajeto 1. A linha em negrito representa a rota original previamente gravada, em verde o trajeto autônomo realizado pelo CaRINA 2 e em vermelho o trajeto autônomo realizado em simulação

Fonte: Adaptada de <https://maps.google.com.br/> utilizando dados reais.

Uma análise mais detalhada do Trajeto 1 permite visualizar os pontos de maior instabilidade, para isso, o trajeto foi subdividido em dois setores de curvas, já que os dois veículos apresentam bom desempenho em trechos de retas. Os setores representam curvas acentuadas a esquerda e mostram a tendência do veículo simulado realizar curvas mais fechadas ao contrário do veículo real.

É possível comparar o perfil dos histogramas apresentados na Figura 41. A alta concentração do erro lateral próximo a zero por parte dos dois veículos mostra grandes trechos de retas.

A simulação teve a tendência de realizar curvas mais fechadas o que mostrou uma quantidade de erros negativos, já o veículo real teve a tendência de abrir a curva e possuir erros positivos.

Figura 41 – Erro lateral no Trajeto 1. Em verde é mostrado o erro lateral do veículo CaRINA 2 e em vermelho o erro lateral do veículo simulado

Fonte: Dados da pesquisa.

6.1.2 Trajeto 2

O Trajeto 2 apresenta um cenários mais desafiador, longo e com maior quantidade de curvas. O resultado da navegação no Trajeto 2 é mostrado na Figura 42 onde as cores preto, verde e vermelho identificam respectivamente a rota original, o trajeto autônomo realizado pelo CaRINA 2 e o simulado. Aqui também é possível notar a coerência dos dados de navegação entre os dois veículos, sendo que ambos possuem trajetórias equivalentes.

O Trajeto 2 mais longo e com maior quantidade de curvas mostrou uma similaridade maior no perfil do histograma (Figura 43). Embora também tenha uma maior concentração de erro próximo a zero, devido a maior quantidade de retas, ambos histogramas ficam deslocados para o lado positivo esboçando a semelhança do perfil.

Os resultados apresentados na Seção 6.1.1 e 6.1.1 mostram pequenas divergências de posicionamento entre o veículo simulado e o veículo real. Porém este se mostrou eficaz nos experimentos, uma vez que o veículo simulado foi capaz de realizar o mesmo percurso que o veículo real. Assim, a arquitetura do simulador se mostrou eficaz em simular a plataforma CaRINA 2. Vale ressaltar que nos experimentos foram utilizados os mesmos mapas, métodos,

Figura 42 – Trajeto 2. A linha em negrito representa a rota original previamente gravada, em verde o trajeto autônomo realizado pelo CaRINA 2 e em vermelho o trajeto autônomo realizado em simulação

Fonte: Adaptada de <https://maps.google.com.br/> utilizando dados reais.

Figura 43 – Erro lateral no Trajeto 2. Em verde é mostrado o erro lateral do veículo CaRINA 2 e em vermelho o erro lateral do veículo simulado.

Fonte: Dados da pesquisa.

algoritmos e parâmetros de controle adaptados para o veículo real. Isso mostra a compatibilidade da arquitetura simulada com a da plataforma real.

6.2 Simulação de Múltiplos Veículos

Tal como mencionado anteriormente no Capítulo 2, pesquisas envolvendo múltiplos veículos tem despertado o interesse da comunidade científica. A condução cooperativa através de tecnologias de comunicação entre veículos (V2V) e até mesmo entre infraestrutura (V2I) pode trazer benefícios e diminuir significativamente os congestionamentos, emissões de CO₂, melhorando a segurança e o conforto dos passageiros (NUNEN et al., 2012).

Em geral, ensaios envolvendo múltiplos veículos são tarefas árduas devido ao número de equipamentos que é necessário operar durante o experimento, além da logística para configuração e manuseio, deve-se de alguma forma garantir a comunicação entre os elementos. A disponibilidade de recursos e espaço adequado também pode inviabilizar experimentos práticos, de forma que a segurança das pessoas e dos equipamentos também deve ser levada em consideração. Isso torna a pesquisa e o amadurecimento dessas tecnologias mais demorado e custoso.

As ferramentas de simulação tem sido uma opção conveniente para realização de tais experimentos, sendo que o modelo dinâmico dos veículos, sensores e cenários podem ser criados de forma a poupar tempo e os custos do projeto. Além disso é possível simular a comunicação entre veículos, uma vez que os protocolos, camadas de rede, efeitos das características físicas e atenuação de sinal podem igualmente reproduzidos.

Essas motivações levaram ao desenvolvimento de uma plataforma de simulação que possa acomodar experiências relacionadas a múltiplos veículos. A abordagem apresentada aqui pode ter ampla utilização, tanto com comunicação entre veículos quanto entre veículos e infraestrutura. Ainda é possível criar cenários complexos e diferentes situações.

A fim de simular um cenário com comunicação foi necessário integrar à plataforma discutida na Seção 5.3, um simulador de redes. Como mostrado por (PEREIRA; ROSSETTI, 2012) a integração de simuladores pode contribuir para desenvolvimento e avanço das áreas de veículos e transporte inteligentes. Para esse trabalho foi escolhido o NS-3 (ENDERSON et al., 2006) que é um simulador de redes muito utilizada em pesquisas científicas e ensino didático. Existem vários estudos sobre VANETs (Veicular Ad Hoc Networks) que fazem uso do NS-3 para simular situações que envolvem comunicação de muitos módulos.

Como proposto por Pereira e Rossetti (2012), o principal objetivo aqui é integrar dois simuladores, neste caso, MORSE como simulador físico e NS-3 como simulador de redes. Com a proposta de desenvolver uma plataforma de simulação para investigar situações que envolvem múltiplos veículos e comunicação.

Além do MORSE fornecer meios de criar novos robôs, como feito anteriormente com

veículo CaRINA 2, também é possível criar novos componentes como sensores e atuadores. Neste caso, um novo sensor foi criado para fazer o papel de uma placa de rede, o sensor foi chamado de NIC (Network Interface Card). A lógica do sensor foi criada, similarmente ao veículo CaRINA 2, através de uma classe chamada `morse.core.sensor.Sensor`. A lógica do sensor apenas criar serviços que disponibilizam dados dos robôs (veículos) como velocidade, aceleração, posicionamento global e orientação global. Para que o sensor possa ser utilizado ele ainda precisa ser criado, isso é feito através da classe `morse.builder.creator.SensorCreator` que junta a lógica do sensor com um modelo tridimensional simples e disponibiliza o componente para ser utilizado por qualquer robô, a Figura 44 exemplifica como o sensor NIC foi criado.

Figura 44 – Construção do sensor NIC

A Figura 45 mostra as instâncias dos robôs utilizados neste trabalho. Dentro do NS3 foram criadas instâncias de robôs, cada robô acessa o respectivo serviço no MORSE, trazendo os dados do veículo. Como a requisição dos dados é feita através de sockets no mesmo computador, ou seja, localhost o atraso na aquisição dos dados é muito pequeno e não influencia o desempenho da simulação. Com os dados prontos no simulador NS3 pode-se iniciar uma série de simulações de troca de pacotes de dados entre as instâncias dos robôs podendo ser simulado sendo que o principal interesse da simulação da rede neste está no delay entre a troca de pacotes. Após a simulação da rede os dados estão prontos para serem analisados. Outras simulações e efeitos de rede podem ser analisados como a perda de pacotes devido a distância, bem como simulações envolvendo comboio de veículos.

Para avaliar a plataforma de simulação de múltiplos veículos, foi realizado um experimento utilizando um controle do tipo CACC (Controle de Cruzeiro Adaptativo Cooperativo), baseado nas equações de controle propostas por (Arem; Driel; Visser, 2006). O CACC é uma extensão do controle ACC (Controle de Cruzeiro Adaptativo) pra lidar com múltiplos veículos, no qual além do veículo ter a capacidade própria de sensoriamento, ele também compartilha suas informações com outros veículos através da comunicação.

Figura 45 – Arquitetura da plataforma de simulação de múltiplos veículos

Pretende-se analisar nesse trabalho a "estabilidade de corda" que é a atenuação do erro de espaçamento entre veículos e também da diferença de velocidade (YANAKIEV; KANELLAPOULOS, 1996). A estabilidade do sistema, neste caso se tratando de um sistema autônomo e cooperativo, descarta a habilidade dos condutores e pode ser atribuída a diversos fatores como o projeto do controle, dinâmica do veículo e condições da comunicação. (PUEBOOBPAPHAN; AREM, 2010) mostra a diferença de um sistema estável Figura 46a e um sistema instável Figura 46b.

Figura 46 – Perfil de estabilidade do sistema

A fim de provar a integração dos simuladores através da comparação do perfil de estabilidade do sistema, como apresentado na Figura 46a, o efeito da comunicação foi analisado. O experimento consiste em dispor quatro veículos simulados de maneira enfileirada, como mostrado na Figura 47 com as mesmas características físicas, dinâmicas e mesmo projeto de controle. Cada veículo comunica-se imediatamente com o de antecessor.

Os quatro veículos, nomeados de Líder L e seguidores F1, F2, e F3, são suficientes para validar a plataforma e as interações entre os veículos. O atraso da comunicação foi definido como

Figura 47 – Disposição dos veículos simulados. Cada veículo comunica-se imediatamente com o antecessor, sendo que o líder controlado por script e os seguidores são conduzidos em modo autônomo

0.01 segundos. Os veículos seguidores tem acesso às informações do veículo correspondente que segue imediatamente a sua frente. A informações, posição, velocidade e aceleração somente são transmitidas após a simulação da rede e do atraso definido, ao passo que cada veículo tem acesso aos seus próprios sensores.

O veículo L é controlado através de um script que faz o papel de um condutor humano, o script permite a repetibilidade dos ensaios. Os veículos seguidores são completamente autônomos, no qual é feito o controle de aceleração e frenagem. O experimento envolve quatro etapas: fase inicial de aceleração, frenagem parcial, retomada da aceleração e frenagem até a parada. O dados de velocidade e distância dos veículos foram registrados e são apresentados nas Figuras 48 e Figura 49.

O tempo total desse experimentos foi de 33 segundos, na primeira seção do experimento (de 0 a 10 segundos) o veículo líder acelera de 0 até 37m/s. Na segunda seção do teste (10 a 15 segundos), a velocidade do veículo líder diminui para 8 m/s. De 15 a 25 segundos, na terceira seção, a velocidade aumenta novamente até 37 m/s. Finalmente no último trecho (25 a 33 segundos), o líder desacelera até 3 m/s na quarta seção.

A Figure 48 apresenta a velocidade de cada veículo durante todo experimento. É possível notar que quando o líder está na fase de aceleração, sua velocidade é maior do que a dos seguidores e na desaceleração esse comportamento é invertido. Isso acontece devido ao atraso da comunicação e também a resposta do controle devido a inércia dos veículos. Comparando os gráficos Figure 48 e Figura 46a pode-se notar o mesmo perfil, mostrando que a estabilidade do sistema foi atingida através da comunicação.

Os histogramas apresentados na Figuras 50 mostram o erro da velocidade e a diferença

Figura 48 – Velocidade dos veículos durante o experimento. Velocidade do veículo Líder (L, preto), Seguidor 1 (F1, vermelho), Seguidor 2 (F2, verde), Seguidor 3 (F3, amarelo)

Figura 49 – Distância entre os veículos durante o experimento. L e F1(L para F1, preto), F1 e F2(F1 para F2, verde), F2 e F3(F2 para F3, vermelho)

de velocidade do líder com relação a cada seguidor (F1, F2, F3). Pode-se notar que o erro aumenta para cada seguidor de acordo com a distância entre os veículos devido aos efeitos do atraso da rede.

Figura 50 – Histograma 1: Diferença da Velocidade entre L e F1. Histograma 2: Diferença da Velocidade entre L e F2. Histograma 3: Diferença da Velocidade entre L e F3

6.3 Experimentos com Veículos Pesados

Em 2013 iniciou-se o projeto do Caminhão Autônomo, uma parceria entre a empresa Scania, ICMC (Instituto de Ciências Matemáticas e de Computação) e a EESC (Escola de Engenharia de São Carlos) para desenvolver o primeiro protótipo de caminhão autônomo no Brasil. O veículo G360 6x4, doado pela Scania, foi inicialmente equipado para controle eletrônico do volante, freio e acelerador, e assim ser capaz de realizar um percurso pré-definido de modo autônomo.

Em meados de 2014 já foi possível fazer os primeiros testes em modo autônomo. Em 2015, alcançou-se a maturidade do projeto na qual foi fixada a arquitetura apresentada na Figura 51. A arquitetura do Caminhão Autônomo segue o mesmo modelo do CaRINA 2, apresentada na Seção 5.2, com as devidas adaptações.

As alterações no veículo, igualmente ao CaRINA 2, foram mínimas o que manteve a estética do veículo. Um motor DC foi acoplado à barra de direção com um encoder para se ter um retorno do movimento do motor e assim realizar o esterçamento do volante. Um atuador linear foi preso ao pedal do freio para realizar a frenagem eletrônica. Os dois módulos são conectados ao controlador de motor Roboteq.

O veículo possui câmbio automático, portanto o procedimento de aceleração funciona de forma similar ao CaRINA 2. Um sinal, gerado pelo computador, é duplicado e calibrado para ser enviado diretamente ao pedal do acelerador. O caminhão também possui um receptor de GPS que permite correção RTK de alta precisão e duas antenas, que facilita obter informação da orientação.

No caminhão, optou-se por utilizar o Radar automotivo da Delphi como principal sensor para percepção, pois devida as dimensões do caminhão, o próprio poderia gerar oclusão se fosse utilizado o sensor Laser 360° da Velodyne. Ainda assim, o Radar possui um custo muito mais baixo quando comparado com o sensor da Velodyne. Os módulos de percepção foram adaptados para uso do Radar, esses que mostraram a robustez através da fácil portabilidade.

A conclusão desse trabalho culminou em uma apresentação pública, através da parceria entre a empresa Scania e a USP. O veículo foi operado em modo autônomo durante cerca de 6 horas e exposto para diversas mídias. A Figura 52 mostra o caminhão navegando em modo autônomo em dois dias de apresentação.

6.4 Considerações Finais

Este capítulo constatou que o modelo simulado, bem como a arquitetura adaptada para simulação do CaRINA 2 apresentada na Seção 5.3, se mostrou capaz de simular a plataforma CaRINA 2 sem alterações críticas nos códigos do veículo real. Também foi apresentada a flexibilidade da arquitetura para realizar experimentos com múltiplos veículos. A arquitetura do CaRINA 2 também mostrou a sua portabilidade ao ser adaptada à um veículo pesado.

Figura 51 – Arquitetura do Caminhão Autônomo

Figura 52 – Apresentação pública do projeto do Caminhão Autônomo

Fonte: <<http://www.lrm.icmc.usp.br/>>.

CONCLUSÃO

7.1 Conclusão

Sistemas de Transportes Inteligentes representam um imenso impacto na sociedade atual. Interferem diretamente no paradigma de mobilidade social e também afetam o modelo de negócio da indústria automobilística. Como visto no Capítulo 2, atualmente há um grande interesse por parte das empresas, essas que buscam parcerias com pesquisadores e também participam de grandes eventos.

Neste contexto, o projeto CaRINA vem contribuindo com o desenvolvimento da área de transportes inteligentes utilizando diversos artifícios da robótica móvel. A análise arquitetural dos veículos Stanley, Junior e Boss foi de grande importância para compreensão dos métodos utilizados na navegação autônoma e revelou algumas defasagens no projeto CaRINA 2 como sistema de diagnóstico, planejador trajetória local e a localização.

Neste trabalho também foi implementado um sistema de diagnóstico que gera alertas ao usuário do veículo através do monitoramento dos sensores, atuadores e GPS. Futuramente o sistema também deverá fazer previsões a fim de detectar possíveis falhas. A localização depende exclusivamente do receptor GPS que está propenso a quedas de sinal, experimentos já estão sendo realizados a fim de integrar o GPS com IMU, similar aos métodos apresentados pelos veículos Stanley e Junior. Por fim, o planejador de trajetória torna-se importante para navegação em ambientes complexos, podendo ser utilizado para ultrapassagens e replanejamento global.

Há uma grande necessidade de realizar experimentos com a plataforma, sendo que esse processo pode ser custoso e demorado. A fim de contornar esses problemas, este trabalho teve como principal contribuição o desenvolvimento de uma plataforma de simulação compatível com o CaRINA 2, para ser utilizada em diversos contextos. O simulador é versátil e também pode ser utilizado para experimentos que envolvem múltiplos veículos como mostrado na Seção 6.2.

Atualmente o simulador tem ajudado no desenvolvimento de uma diversidade de métodos,

algoritmos e propostas de mapas. Os quais podem ser testados antes de serem levados ao veículo real. Projetos relacionados a estacionamento autônomo, planejador de trajetória e rotas, tem utilizado o simulador para desenvolvimento e experimentação, bem como novas propostas e extensões da arquitetura. O estudo da dinâmica veicular do CaRINA 2 também pode trazer melhorias para o modelo dinâmico da simulação.

No futuro, pretende-se construir cenários mais dinâmicos, com a adição de vários elementos de cena como veículos, pessoas e sinalizações de trânsito. Deseja-se utilizar um joystick pra realizar experimentos de sistemas de auxílio ao motorista (ADAS) e também estudos comportamentais do motorista. O simulador pode permitir a introdução de um condutor humano juntamente com veículos autônomos.

Finalmente, a análise feita no Capítulo 2 sugere uma grande inclinação a pesquisas envolvendo sistemas cooperativos, utilizando comunicação V2V e V2I. Essa área em expansão requer experimentos mais complexos, onde a simulação pode acelerar o desenvolvimento.

REFERÊNCIAS

Arem, B. van; Driel, C. J. van; Visser, R. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 7, n. 4, p. 429–436, December 2006. Disponível em: <<http://doc.utwente.nl/58157/>>. Citado na página 81.

BACHA, A.; BAUMAN, C.; FARUQUE, R.; FLEMING, M.; TERWELP, C.; REINHOLTZ, C. F.; HONG, D.; WICKS, A.; ALBERI, T.; ANDERSON, D.; CACCIOLA, S.; CURRIER, P.; DALTON, A.; FARMER, J.; HURDUS, J.; KIMMEL, S.; KING, P.; TAYLOR, A.; COVERN, D. V.; WEBSTER, M. Odin: Team victortango's entry in the DARPA urban challenge. *J. Field Robotics*, v. 25, n. 8, p. 467–492, 2008. Disponível em: <<http://dx.doi.org/10.1002/rob.20248>>. Citado na página 49.

BACHA, A.; BAUMAN, C.; FARUQUE, R.; FLEMING, M.; TERWELP, C.; REINHOLTZ, C.; HONG, D.; WICKS, A.; ALBERI, T.; ANDERSON, D.; CACCIOLA, S.; CURRIER, P.; DALTON, A.; FARMER, J.; HURDUS, J.; KIMMEL, S.; KING, P.; TAYLOR, A.; COVERN, D. V.; WEBSTER, M. Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics*, Wiley Subscription Services, Inc., A Wiley Company, v. 25, n. 8, p. 467–492, 2008. ISSN 1556-4967. Disponível em: <<http://dx.doi.org/10.1002/rob.20248>>. Citado na página 51.

BAKER, C.; DOLAN, J. Street smarts for boss. *Robotics Automation Magazine*, IEEE, v. 16, n. 1, p. 78–87, March 2009. ISSN 1070-9932. Citado 2 vezes nas páginas 29 e 46.

BANKS, J.; CARSON, J.; NELSON, B. L.; NICOL, D. *Discrete-Event System Simulation* (4th Edition). 4. ed. [S.l.]: Prentice Hall, 2004. Paperback. ISBN 0131446797. Citado na página 47.

BERTOZZI, M.; BOMBINI, L.; BROGGI, A.; BUZZONI, M.; CARDARELLI, E.; CATTANI, S.; CERRI, P.; COATI, A.; DEBATTISTI, S.; FALZONI andrea; FEDRIGA, R. I.; FELISA, M.; GATTI, L.; GIACOMAZZO, A.; GRISLERI, P.; LAGHI, M. C.; MAZZEI, L.; MEDICI, P.; PANCIROLI, M.; PORTA, P. P.; ZANI, P.; VERSARI, P. VIAC: an Out of Ordinary Experiment. In: *IEEE Intelligent Vehicles Symposium 2011*. Baden Baden, Germany: [s.n.], 2011. p. 175–180. ISSN: 1931-0587. Citado 2 vezes nas páginas 30 e 32.

BERTOZZI, M.; BROGGI, A.; CARDARELLI, E.; FEDRIGA, R. I.; MAZZEI, L.; PORTA, P. P. VIAC Expedition Toward Autonomous Mobility. *Robotics and Automation Magazine*, IEEE, v. 18, n. 3, p. 120–124, set. 2011. ISSN: 1070-9932. Citado na página 32.

BERTOZZI, M.; BROGGI, A.; COATI, A.; FEDRIGA, R. I. A 13,000 km Intercontinental Trip with Driverless Vehicles: The VIAC Experiment. *IEEE Intelligent Transportation System Magazine*, Piscataway, USA, v. 5, n. 1, p. 28–41, 2013. ISSN: 1939-1390. Citado na página 31.

BOHREN, J.; FOOTE, T.; KELLER, J.; KUSHLEYEV, A.; LEE, D. D.; STEWART, A.; VERNAZA, P.; DERENICK, J. C.; SPLETZER, J. R.; SATTERFIELD, B. Little ben: The ben

franklin racing team's entry in the 2007 DARPA urban challenge. *J. Field Robotics*, v. 25, n. 9, p. 598–614, 2008. Disponível em: <<http://dx.doi.org/10.1002/rob.20260>>. Citado na página 49.

BRAID, D.; BROGGI, A.; SCHMIEDEL, G. The terramax autonomous vehicle. *J. Field Robotics*, v. 23, n. 9, p. 693–708, 2006. Citado na página 29.

BROGGI, A.; BERTOZZI, M.; FASCIOLI, A.; CONTE, G. *Automatic Vehicle Guidance: the Experience of the ARGO Vehicle*. Singapore: World Scientific, 1999. ISBN 9810237200. Citado 2 vezes nas páginas 23 e 29.

BROGGI, A.; BOMBINI, L.; STEFANO, C.; CERRI, P.; FEDRIGA, R. I. Sensing requirements for a 13,000 km intercontinental autonomous drive. In: *Procs. IEEE Intelligent Vehicles Symposium 2010*. La Jolla, CA, USA: [s.n.], 2010. p. 500–505. Citado 3 vezes nas páginas 30, 31 e 32.

CHEN, Y.-L.; SUNDARESWARAN, V.; ANDERSON, C.; BROGGI, A.; GRISLERI, P.; PORTA, P. P.; ZANI, P.; BECK, J. TerraMax: Team Oshkosh Urban Robot. In: BUEHLER, M.; IAGNEMMA, K.; SINGH, S. (Ed.). *The DARPA Urban Challenge, Autonomous Vehicles in City Traffic*. [S.l.]: Springer-Verlag Berlin Heidelberg, 2009, (Springer Tracts in Advanced Robotics). p. 595–622. ISBN: 978-3-642-03990-4. Citado na página 29.

CHEON, S.; CENTER, U. of C. S. T. *An Overview of Automated Highway Systems (AHS) and the Social Institutional Challenges They Face*. University of California Transportation Center, University of California, 2002. Disponível em: <<http://books.google.com.br/books?id=-QQhtwAACAAJ>>. Citado na página 33.

COLLETT, T. H. J.; MACDONALD, B. A. Player 2.0: Toward a practical robot programming framework. In: *in Proc. of the Australasian Conference on Robotics and Automation (ACRA)*. [S.l.: s.n.], 2005. Citado na página 48.

COMMISSION, E. *Information and Communications Technologies for Safe and Intelligent Vehicles: Communication from the Commission to the Council and the European Parliament*. Office for Official Publications of the European Communities, 2003. (EDC collection). Disponível em: <http://books.google.com.br/books?id=XW7_SAAACAAJ>. Citado na página 33.

CRISMAN, J.; THORPE, C. Scarf: A color vision system that tracks roads and intersections. *IEEE Trans. on Robotics and Automation*, v. 9, n. 1, p. 49 – 58, February 1993. Citado na página 23.

ECHEVERRIA, G.; LASSABE, N.; DEGROOTE, A.; LEMAIGNAN, S. Modular openrobots simulation engine: Morse. In: *Proceedings of the IEEE ICRA*. [S.l.: s.n.], 2011. Citado 3 vezes nas páginas 47, 49 e 66.

FILHO, C. M.; WOLF, D. Dynamic inversion-based control for front wheel drive autonomous ground vehicles near the limits of handling. In: *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. [S.l.: s.n.], 2014. p. 2138–2143. Citado 2 vezes nas páginas 61 e 64.

GAT, E. *On three-layer architectures*. In: *Artificial Intelligence and Mobile Robots*. [S.l.]: MIT Press, 1998. Citado 2 vezes nas páginas 42 e 43.

GEIGER, A.; LAUER, M.; MOOSMANN, F.; RANFT, B.; RAPP, H.; STILLER, C.; ZIEGLER, J. Team annieway's entry to the 2011 grand cooperative driving challenge. *Intelligent Transportation Systems, IEEE Transactions on*, v. 13, n. 3, p. 1008–1017, 2012. Citado na página 37.

HENDERSON, T. R.; ROY, S.; FLOYD, S.; RILEY, G. F. Ns-3 project goals. In: *Proceeding from the 2006 Workshop on Ns-2: The IP Network Simulator*. New York, NY, USA: ACM, 2006. (WNS2 '06). ISBN 1-59593-508-8. Disponível em: <<http://doi.acm.org/10.1145/1190455.1190468>>. Citado na página 80.

KIANFAR, R.; AUGUSTO, B.; EBADIGHAJARI, A.; HAKEEM, U.; NILSSON, J.; RAZA, A.; TABAR, R.; IRUKULAPATI, N.; ENGLUND, C.; FALCONE, P.; PAPANASTASIOU, S.; SVENSSON, L.; WYMEERSCH, H. Design and experimental validation of a cooperative driving system in the grand cooperative driving challenge. *Intelligent Transportation Systems, IEEE Transactions on*, v. 13, n. 3, p. 994–1007, 2012. Citado na página 37.

LAUER, M. Grand cooperative driving challenge 2011 [its events]. *Intelligent Transportation Systems Magazine, IEEE*, v. 3, n. 3, p. 38–40, 2011. Citado na página 36.

LEMAIGNAN, S.; ECHEVERRIA, G.; KARG, M.; MAINPRICE, J.; KIRSCH, A.; ALAMI, R. Human-robot interaction in the morse simulator. In: *ACM. Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. [S.l.], 2012. p. 181–182. Citado na página 55.

MARTINEZ, A.; FERNANDEZ, E. *Learning ROS for Robotics Programming*. [S.l.]: Packt Publishing, 2013. ISBN 1782161449, 9781782161448. Citado na página 53.

MILTON, H.; OSÓRIO, F.; HEINEN, F.; KELBER, C. SEVA3D: Using Artificial Neural Networks to Autonomous Vehicle Parking Control. [S.l.]: *IEEE WCCI-IJCNN-International Joint Conference on Neural Networks*, 2006. Citado na página 47.

MONTEMERLO, M.; BECKER, J.; BHAT, S.; DAHLKAMP, H.; DOLGOV, D.; ETTINGER, S.; HAEHNEL, D.; HILDEN, T.; HOFFMANN, G.; HUHNKE, B.; JOHNSTON, D.; KLUMPP, S.; LANGER, D.; LEVANDOWSKI, A.; LEVINSON, J.; MARCIL, J.; ORENSTEIN, D.; PAEFGEN, J.; PENNY, I.; PETROVSKAYA, A.; PFLUEGER, M.; STANEK, G.; STAVENS, D.; VOGT, A.; THRUN, S. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008. Citado na página 29.

NUNEN, E. van; KWAKKERNAT, R.; PLOEG, J.; NETTEN, B. Cooperative competition for future mobility. *Intelligent Transportation Systems, IEEE Transactions on*, v. 13, n. 3, p. 1018–1025, 2012. Citado 6 vezes nas páginas 33, 34, 35, 36, 37 e 80.

OZGUNER, U.; REDMILL, K.; BIDDLESTONE, S.; HSIEH, M. F.; YAZICI, A.; TOTH, C. Simulation and testing environments for the darpa urban challenge. In: *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*. [S.l.: s.n.], 2008. p. 222–226. Citado na página 49.

P. TIENTRAKOOL, Y.-C. H. N. M. Highway capacity benefits from using vehicle-to-vehicle communication and sensors for collision avoidance. *Vehicular Technology Conference*, p. 1–5, 2011. Citado 2 vezes nas páginas 23 e 33.

PATZ, B. J.; PAPELIS, Y.; PILLAT, R.; STEIN, G.; HARPER, D. A practical approach to robotic design for the DARPA urban challenge. *J. Field Robotics*, v. 25, n. 8, p. 528–566, 2008. Disponível em: <<http://dx.doi.org/10.1002/rob.20251>>. Citado 2 vezes nas páginas 49 e 51.

PEREIRA, J. L. F.; ROSSETTI, R. J. F. An integrated architecture for autonomous vehicles simulation. In: OSSOWSKI, S.; LECCA, P. (Ed.). *SAC. ACM*, 2012. p. 286–292. ISBN 978-1-4503-0857-1. Disponível em: <<http://dblp.uni-trier.de/db/conf/sac/sac2012.html#PereiraR12>>. Citado na página 80.

PUEBOOBPAPHAN, R.; AREM, B. van. Driver and vehicle characteristics and platoon and traffic flow stability: Understanding the relationship for design and assessment of cooperative adaptive cruise control. *Transportation Research Record: Journal of the Transportation Research Board*, Transportation Research Board of the National Academies, n. 2189, p. 89–97, 2010. Citado na página 82.

QUIGLEY, M.; GERKEY, B.; SMART, W. D. *Programming Robots with ROS*. [S.l.]: O'Reilly Media, 2015. ISBN 978-1-4493-2389-9. Citado na página 47.

SIEGWARD, R.; NOURBAKHSH, I. R. *Introduction to Autonomous Mobile Robots*. Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X. Citado na página 41.

SWANSON, K.; BROWN, A.; BRENNAN, S.; LAJAMBE, C. Extending driving simulator capabilities toward hardware-in-the-loop testbeds and remote vehicle interfaces. In: *Intelligent Vehicles Symposium (IV)*, 2013 IEEE. [S.l.: s.n.], 2013. p. 122–127. ISSN 1931-0587. Citado na página 49.

THORPE, C.; HEBERT, M.; KANADE, T.; SHAFER, S. Vision and navigation for the carnegiemellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 10, n. 3, p. 362 – 373, May 1988. Citado na página 23.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L.-E.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; NIEKERK, J. van; JENSEN, E.; ALESSANDRINI, P.; BRADSKI, G.; DAVIES, B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, Wiley Subscription Services, Inc., A Wiley Company, v. 23, n. 9, p. 661–692, 2006. ISSN 1556-4967. Disponível em: <<http://dx.doi.org/10.1002/rob.20147>>. Citado 3 vezes nas páginas 27, 28 e 43.

URMSON, C.; ANHALT, J.; BAGNELL, D.; BAKER, C.; BITTNER, R.; DOLAN, J.; DUGGINS, D.; FERGUSON, D.; GALATALI, T.; GEYER, C.; GITTLEMAN, M.; HARBAUGH, S.; HEBERT, M.; HOWARD, T.; KELLY, A.; KOHANBASH, D.; LIKHACHEV, M.; MILLER, N.; PETERSON, K.; RAJKUMAR, R.; RYBSKI, P.; SALESKY, B.; SCHERER, S.; WOO-SEO, Y.; SIMMONS, R.; SINGH, S.; SNIDER, J.; STENTZ, A.; WHITTAKER, W. R.; ZIGLAR, J.; BAE, H.; LITKOUHI, B.; NICKOLAOU, J.; SADEKAR, V.; ZENG, S.; STRUBLE, J.; TAYLOR, M.; DARMS, M. Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge. 2007. Citado na página 29.

WOLF, D. F.; oES, E. do V. S.; OSÓRIO, F. S.; JUNIOR, O. T. *Robótica Móvel Inteligente: Da Simulação às Aplicações no Mundo Real*. [S.l.]: JAI2009, 2009. Citado na página 47.

YANAKIEV, D.; KANELAKOPOULOS, I. A simplified framework for string stability analysis in ahs. In: IN PROCEEDINGS OF THE 13TH IFAC WORLD CONGRESS. [S.l.: s.n.], 1996. p. 177–182. Citado na página [82](#).

ZIEGLER, J.; BENDER, P.; SCHREIBER, M.; LATEGAHN, H.; STRAUSS, T.; STILLER, C.; DANG, T.; FRANKE, U.; APPENRODT, N.; KELLER, C.; KAUS, E.; HERRTWICH, R.; RABE, C.; PFEIFFER, D.; LINDNER, F.; STEIN, F.; ERBS, F.; ENZWEILER, M.; KNOPPEL, C.; HIPPE, J.; HAUEIS, M.; TREPTE, M.; BRENK, C.; TAMKE, A.; GHANAAT, M.; BRAUN, M.; JOOS, A.; FRITZ, H.; MOCK, H.; HEIN, M.; ZEEB, E. Making bertha drive - an autonomous journey on a historic route. Intelligent Transportation Systems Magazine, IEEE, 2014. Citado na página [37](#).