Navegação autônoma para robôs móveis usando aprendizado supervisionado

Jefferson Rodrigo de Souza

Navegação autônoma para robôs móveis usando aprendizado supervisionado

Jefferson Rodrigo de Souza

Orientador: **Prof. Dr. Denis Fernando Wolf**

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

USP – São Carlos
Maio de 2014

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

Souza, Jefferson Rodrigo de
S729n Navegação autônoma para robôs móveis usando aprendizado supervisionado / Jefferson Rodrigo de Souza; orientador Denis Fernando Wolf. -- São Carlos, 2014.
81 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2014.

1. ROBÓTICA. 2. ROBÔS. 3. REDES NEURAIS. 4. PROCESSOS GAUSSIANOS. I. Wolf, Denis Fernando, orient. II. Título.

”*Se algum de vocês tem falta de sabedoria,*
peça-a a Deus, que a todos dá livremente,
de boa vontade; e lhe será concedida.”

(Tiago 1:5)
Bíblia Sagrada

Agradecimentos

Em primeiro lugar, agradeço a Deus. Sem Ele esta tese não teria saído. Toda honra, glória e poder a Deus. Agradeço a minha esposa, a minha mãe e irmão pelo amor, apoio e dedicação durante todos os momentos.

Agradeço ao meu orientador Prof. Denis Fernando Wolf pelo convite realizado, compreensão, apoio, orientação, companhia, dedicação, disponibilidade e por ter confiado em mim o desenvolvimento dessa tese.

Agradeço ao Prof. Fabio Tozeto Ramos pela atenção e co-orientação no período sanduíche na *The University of Sydney* (Austrália), além dos amigos conquistados Vitor Guizilini, Roman Marchant e Lionel Ott.

Agradeço aos professores Fernando Osório e Valdir Grassi pela inspiração e ajuda nesses anos. Também a todos do LRM do Instituto de Ciências Matemáticas de Computação da Universidade de São Paulo pelas discussões e descontração. Agradeço à FAPESP pelo apoio financeiro.

Ao grande amigo Gustavo Pessin, que tive a oportunidade de conhecer e com o qual aprendi muito através de suas pesquisas, agradeço pela ajuda, motivação e força nos estudos. Muito obrigado, meu amigo Pessin!

Agradeço a minha amada esposa pela abdicação, paciência, força, amor e coragem em todos os momentos. Meus sinceros agradecimentos a ela e a Sarah que estamos esperando. Eu te amo!

Agradeço a minha família, mesmo distante, mas sempre tão presente, mãe, irmão, avó, tia Zezé, Martinha, Magda, Anteogens e Nilda sempre ajudando e encorajando para ter novas conquistas.

Agradeço ao meu amigo Thomaz Bonato pelas palavras que tive na Austrália, para que em tudo que eu fizesse fosse intenso, violento e agressivo, no bom sentido. Também a sua esposa pelas palavras de Deus.

Agradeço ao meu pai, pois sem ele não estaria onde estou agora (*in memorian*). Esse sonho realizado foi fruto do seu trabalho. Obrigado!

Resumo

A navegação autônoma é um dos problemas fundamentais na área da robótica móvel. Algoritmos capazes de conduzir um robô até o seu destino de maneira segura e eficiente são um pré-requisito para que robôs móveis possam executar as mais diversas tarefas que são atribuídas a eles com sucesso. Dependendo da complexidade do ambiente e da tarefa que deve ser executada, a programação de algoritmos de navegação não é um problema de solução trivial.

Esta tese trata do desenvolvimento de sistemas de navegação autônoma baseados em técnicas de aprendizado supervisionado. Mais especificamente, foram abordados dois problemas distintos: a navegação de robôs/-veículos em ambientes urbanos e a navegação de robôs em ambientes não estruturados. No primeiro caso, o robô/veículo deve evitar obstáculos e se manter na via navegável, a partir de exemplos fornecidos por um motorista humano. No segundo caso, o robô deve identificar e evitar áreas irregulares (maior vibração), reduzindo o consumo de energia. Nesse caso, o aprendizado foi realizado a partir de informações obtidas por sensores. Em ambos os casos, algoritmos de aprendizado supervisionado foram capazes de permitir que os robôs navegassem de maneira segura e eficiente durante os testes experimentais realizados.

Palavras-chave: Navegação Autônoma, Robôs Móveis, Redes Neurais Artificiais, Processos Gaussianos e Ambientes Externos.

Abstract

Autonomous navigation is a fundamental problem in the field of mobile robotics. Algorithms capable of driving a robot to its destination safely and efficiently are a prerequisite for mobile robots to successfully perform different tasks that may be assigned to them. Depending on the complexity of the environment and the task to be executed, programming of navigation algorithms is not a trivial problem.

This thesis approaches the development of autonomous navigation systems based on supervised learning techniques. More specifically, two distinct problems have been addressed: a robot/vehicle navigation in urban environments and robot navigation in unstructured environments. In the first case, the robot/vehicle must avoid obstacles and keep itself in the road based on examples provided by a human driver. In the second case, the robot should identify and avoid unstructured areas (higher vibration), reducing energy consumption. In this case, learning was based on information obtained by sensors. In either case, supervised learning algorithms have been capable of allowing the robots to navigate in a safe and efficient manner during the experimental tests.

Keywords: Autonomous Navigation, Mobile Robots, Artificial Neural Networks, Gaussian Processes and Outdoor Environments.

Sumário

Agradecimentos **i**

Resumo **iii**

Abstract **v**

Lista de Figuras **ix**

Lista de Tabelas **xiii**

Lista de Siglas **xv**

1 Introdução **1**
1.1 Motivação 1
1.2 Justificativa 4
1.3 Objetivos 6
1.3.1 Objetivo Geral 6
1.3.2 Objetivos Específicos 6
1.4 Contribuições 6
1.5 Organização da Tese 7

2 Trabalhos Relacionados **9**
2.1 Navegação utilizando aprendizado supervisionado baseada na condução humana 9
2.2 Navegação utilizando aprendizado supervisionado baseada na irregularidade do terreno 13
2.3 Considerações Finais 16

3 Aprendizado Supervisionado **17**
3.1 Redes Neurais Artificiais . . . 18
3.2 Processos Gaussianos . . . 25
3.3 Otimização Bayesiana . . . 31

4 Navegação Autônoma Baseada no Condutor Humano **35**
4.1 Metodologia . . . 35
4.1.1 Identificação da rua . . . 37
4.1.2 Identificação de geometrias . . . 39
4.1.3 Navegação por aprendizado . . . 41
4.2 Resultados . . . 42
4.2.1 Cenário 1 . . . 42
4.2.2 Cenário 2 . . . 47
4.3 Considerações Finais . . . 50

5 Navegação Autônoma Baseada na Irregularidade do Terreno **53**
5.1 Metodologia . . . 53
5.1.1 Pré-processamento . . . 55
5.1.2 Processo Gaussiano para Modelagem de Vibração . . . 56
5.1.3 Otimização Bayesiana . . . 57
5.1.4 Otimização Bayesiana para Planejamento Informativo . . . 58
5.2 Resultados . . . 59
5.2.1 Modelo . . . 60
5.2.2 Planejamento de Trajetória . . . 62
5.3 Considerações Finais . . . 64

6 Conclusão **67**
6.1 Trabalhos Futuros . . . 68

Referências Bibliográficas **71**

A Artigos Publicados **79**
A.1 Publicados em periódicos . . . 79
A.2 Publicados em conferências . . . 80

Lista de Figuras

1.1 Robô móvel autônomo: percepção-decisão-ação (Wolf *et al.*, 2009). 1
1.2 Stanley, vencedor do *Grand Challenge* em 2005 (Thrun *et al.*, 2006). 3
1.3 Veículos de testes pertencentes ao projeto temático CaRINA. (a) Veículo elétrico conhecido como CaRINA 1 e (b) Palio Adventure denominado de CaRINA 2. . 6

2.1 Projeto ALVINN (Pomerlau, 1995). 10
2.2 A visão do robô simulado (Kunzle, 2010). 10
2.3 Posição lateral do robô atual e futura (Darter e Gordon, 2005). 11
2.4 (a) Robô. (b) Estação de controle. (c) Pista usada nos testes (Markelic *et al.*, 2009). 12
2.5 O robô autônomo (Stein e Santos, 2010). 12
2.6 (a) Câmera estéreo. (b) Computador. (c) Monitor no carro (Markelic *et al.*, 2011). 13
2.7 *Cart* utilizado para a coleta de dados (Weiss *et al.*, 2006). 14
2.8 Tipos de terrenos utilizados. 1: piso interno 2: asfalto 3: cascalho 4: grama 5: pedra de calçada e 6: areia (Weiss *et al.*, 2006). 14
2.9 Plataforma de teste utilizada (Weiss *et al.*, 2008). 15

3.1 Neurônio artificial (Braga *et al.*, 2007). 18
3.2 RNA MLP usando o algoritmo de treinamento *Backpropagation* (Haykin, 1999). 19
3.3 Curvas de erro em aprendizado e validação (Pessin, 2013). 23
3.4 Exemplo da predição de GP com três funções de covariância (Guizilini, 2013). 28
3.5 Exemplo de funcionamento do Processo Gaussiano (Guizilini, 2013). 30

3.6 (a) Dados gerados a partir de um GP com hiperparâmetros (l, σ_f, σ_n) = (1, 1, 0.1), como mostrado pelos símbolos +. Usando a predição de GP com estes hiperparâmetros, obtendo uma região de confiança de 95% para a função f (mostrado em cinza). Painéis (b) e (c) mostram novamente a região de confiança de 95%, mas desta vez para os hiperparâmetros (0.3, 1.08, 0.00005) e (3.0, 1.16, 0.89), respectivamente (Rasmussen e Williams, 2006). 31
3.7 Um exemplo do uso de otimização Bayesiana para um problema de 1-D. As figuras ilustram uma aproximação do GP da função objetivo sobre quatro iterações de valores amostrados da mesma. Também ilustra a função de aquisição nas curvas verdes sombreadas. A aquisição é alta, onde o GP prevê um objetivo alto (explotação) e onde a incerteza da predição é alta (exploração), áreas com ambos os atributos são primeiro amostradas. A área à esquerda permanece não-amostrada, ao mesmo tempo que se tem uma incerteza (variância) alta (Brochu *et al.*, 2010). 32

4.1 Plataforma de teste CaRINA 1 da Universidade de São Paulo em São Carlos. . 36
4.2 Esquema geral do sistema de navegação autônoma usando aprendizado. 37
4.3 (a) Classificador, (b) Imagem Real e (c) Mapa de Navegabilidade. 37
4.4 Imagem com os blocos de dimensão (10 x 10) *pixels* (Shinzato, 2010). 38
4.5 Geometrias definem as direções do veículo. (a) Curva rígida à esquerda. (b) Curva suave à esquerda. (c) Frente. (d) Curva suave à direita. (e) Curva rígida à direita. 40
4.6 Os formatos dos *templates* são: (a) Esquerda. (b) Centro. (c) Direita. 40
4.7 Estrutura da RNA utilizada para produzir os comandos de esterçamento e velocidade do CaRINA 1. Azimute representa a diferença entre a orientação para o destino (coordenadas de GPS) e a orientação do veículo (obtida a partir de uma bússola). A sigla AO significa a área de ocupação de cada modelo (*template*). 41
4.8 Trajeto realizado pelo CaRINA 1 no primeiro cenário externo (aproximadamente 200 m). P0 até P6 representam os pontos das coordenadas de GPS utilizados nesse ambiente. A linha amarela ilustra o caminho aproximado realizado pelo veículo. 43
4.9 Histograma dos erros (valor esperado - valor obtido da RNA) considerando as melhores RNAs usando os dados de teste. Eixo x representa o erro. O eixo y representa a densidade baseada em esterçamento ou velocidade usadas no cenário 1. 46
4.10 A condução humana (esperado) e os valores da RNA usando os dados de teste. 47
4.11 Trajeto realizado pelo CaRINA 1 (aproximadamente 1,08 km). P0 até P6 representam os pontos de GPS usados nesse segundo cenário. A linha amarela ilustra o caminho aproximado realizado pelo CaRINA 1. As linhas vermelhas ilustram a orientação calculada pelo veículo para se deslocar de um ponto a outro. . . . 48

4.12 Histograma dos erros (valor esperado - valor obtido da RNA) considerando as melhores RNAs das topologias 4 (execução 4 usando SL) e 5 (execução 3 usando TH) usando os dados de teste. O eixo x representa o erro usado para análise. O eixo y representa a densidade baseada nos dados de esterçamento no cenário 2. 49
4.13 Ângulo de esterçamento usando os dados de teste. 50

5.1 Plataforma de teste Husky - *Clearpath Robotics* (Husky, 2013). 54
5.2 Diagrama de bloco do método proposto. 54
5.3 Aplicação de dois filtros em dados brutos da IMU. 56
5.4 Resultados dos filtros de pré-processamento e função de covariância. RMSE obtidos a partir de 30% dos dados de teste, e tempo de treinamento para maximizar LML com 700 observações. Os índices das funções de covariância são mostrados na Tabela 5.1. 61
5.5 (a) O cenário utilizado para o primeiro experimento delimitado pelo mapa produzido pelo robô. (b) Média da vibração preditiva. (c) A variância da predição. (d) Média preditiva após a navegação autônoma. A distância está em metros ($[m]$) e a predição está em $[m/s^2]$. 62
5.6 Comparação entre as estratégias BO e Entropia. (a) e (f) mostram as áreas do experimento com um retângulo e os obstáculos que produzem vibração são marcados com círculos. (b), (c), (g) e (h) mostram uma superfície para a predição da aceleração vertical em m/s^2 e o eixo em metros. (d), (e), (i) e (j) mostram um mapa de calor do tempo gasto do robô em cada posição durante os experimentos. 64

Lista de Tabelas

4.1 Atributos de entrada das RNAs utilizadas para identificar áreas navegáveis e não navegáveis (Shinzato e Wolf, 2011). Componentes R, G, B são representados pelo vermelho (*red*), verde (*green*) e azul (*blue*). Componentes H, S, V são representados pelo matiz (*hue*), saturação (*saturation*) e brilho (*value*). Componentes Y, U, V são representados pelo brilho, tonalidade azul e tonalidade vermelha. Md = Média, Norm = Normalizado, Ent = Entropia, En = Energia e Var = Variância. 39

4.2 Resultados das RNAs para cada topologia. Diversas avaliações foram realizadas usando diferentes topologias e duas funções de ativação: Tangente Hiperbólica (TH) e Sigmóide Logística (SL). O erro da RNA é obtido através do EMQ (10^{-3}). Além disso, apresenta-se o resultado do Ponto Ótimo de Generalização (POG) de cada execução das RNAs. Em negrito, apresenta-se os melhores (menores) valores. 44

4.3 Resultados do teste Shapiro-Wilk. 45

4.4 Resultados do teste Man-Whitney. 45

4.5 Resultados do erro (valor esperado - valor obtido da RNA) usando como entrada os dados de teste para as RNAs de Topologias 4 e 5 (Cenário 2). 49

5.1 Resultados dos filtros de pré-processamento e função de covariância. RMSE obtidos a partir de 30% dos dados de teste, e tempo de treinamento para maximizar LML com 700 observações, onde os valores do RMSE estão dispostos em $(10^{-2}[m/s^2])$ e o Tempo está descrito em (s). 61

5.2 Aceleração média em $[m/s^2]$ para os planejadores BO e Entropia. 64

Lista de Siglas

ALVINN	*Autonomous Land Vehicle In a Neural Network*
BO	*Bayesian Optimisation*
BP	*Backpropagation*
CARINA	Carro Robótico Inteligente para Navegação Autônoma
CPU	*Central Processing Unit*
DARPA	*Defense Advanced Research Projects Agency*
DUCB	*Distance-based Upper Confidence Bound*
EMQ	Erro Médio Quadrático
FAPESP	Fundação de Amparo à Pesquisa do Estado de São Paulo
GP	*Gaussian Process*
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
ICMC	Instituto de Ciências Matemáticas e de Computação
IMU	*Inertial Measurement Unit*
INCT-SEC	Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos
LML	*Log Marginal Likelihood*
LRM	Laboratório de Robótica Móvel
MLP	*Multi-Layer Perceptron*
POG	Ponto Ótimo de Generalização
RBF	*Radial Basis Function*
RMSE	*Root Mean Squared Error*
RNA	Rede Neural Artificial
ROS	*Robot Operating System*
RPROP	*Resilient Backpropagation*
SLAM	*Simultaneous Localization and Mapping*
SNNS	*Stuttgart Neural Network Simulator*
SVM	*Support Vector Machine*

CAPÍTULO

1

Introdução

1.1 Motivação

A robótica é a ciência que percebe e atua no mundo físico (e.g. ambiente externo) por meio de dispositivos computacionais (Thrun *et al.*, 2005). Para compreender o ambiente é necessário o uso de sensores, sendo os principais encontrados na literatura: câmeras de vídeo, radares, GPS, laser (*laser range finders*) e unidades de medida inercial (Dahlkamp *et al.*, 2006; Petrovskaya e Thrun, 2009; Thrun *et al.*, 2006). Os atuadores usados em um robô são: pernas, garras e rodas. A Figura 1.1 apresenta um exemplo de interação do robô, em que ele percebe o ambiente, decide e age sobre o mesmo, e cujas ações afetarão o ambiente, e por consequência, suas percepções no instante de tempo seguinte (Wolf *et al.*, 2009).

Figura 1.1: Robô móvel autônomo: percepção-decisão-ação (Wolf *et al.*, 2009).

A robótica móvel lida com três níveis de autonomia, os robôs tele-operados, os semi-autônomos e os robôs autônomos (Dudek e Jenkin, 2010). Cada vez mais têm crescido o interesse e a demanda por robôs móveis autônomos (Siegwart *et al.*, 2011; Thrun *et al.*, 2005), os quais são capazes de operar no ambiente (interno ou externo), apreendendo e tomando decisões corretas de modo que suas tarefas sejam realizadas com êxito (Carvalho e Kowaltowski, 2009) sem a necessidade da intervenção humana, tanto em atividades de pesquisa como industriais.

Pesquisas em robótica móvel requerem conhecimentos em diversas áreas, tais como Computação, Matemática, Física, Engenharia Elétrica, Engenharia Mecânica, entre outras. Nos últimos 5 anos, os robôs móveis autônomos são aplicados com sucesso em diversas aplicações externas (em ambientes não estruturados), como por exemplo, em operações militares (Lin *et al.*, 2008), explorações espaciais (Tunstel *et al.*, 2009), minas (Murphy *et al.*, 2009), portos (Bandyophadyay *et al.*, 2010), agricultura (Chen *et al.*, 2012), entre outras aplicações.

Parte da robótica móvel se especializa em navegação autônoma, que é uma tarefa fundamental em robôs móveis (Thrun *et al.*, 2006). Segundo Stentz *et al.* (2007), o conceito de navegação autônoma é abordado como sendo a tarefa de conduzir um robô autonomamente de forma adequada, onde o robô é equipado com sensores para a leitura das propriedades do ambiente.

Para a realização da navegação autônoma são necessários computadores, capazes de interpretar os dados e atuadores que permitem a direção, aceleração e frenagem do robô, a partir de uma posição inicial até uma posição de destino sobre um determinado terreno, de modo que o robô funcione com segurança e com um desempenho correto para a realização da operação, com pouca (sistemas semiautônomos) ou nenhuma intervenção humana (sistemas autônomos).

Existem diversos sistemas de navegação robóticos, como por exemplo, os robôs de exploração espaciais (Bajracharya *et al.*, 2008), robôs domésticos com a capacidade de limpar a casa (iRobot, 2014), os sistemas robóticos utilizados para o desarme de bombas (Cobham, 2014), sistemas de alerta ao motorista usando câmeras térmicas (Flir, 2014), cadeira de rodas inteligente (MIT, 2014), entre outros.

Um outro exemplo muito interessante de navegação robótica são os veículos autônomos inteligentes (Cheng, 2011; Jung *et al.*, 2005; Ozguner *et al.*, 2011), os quais pretendem conseguir uma diminuição do número de acidentes em ruas e rodovias, redução nos gastos com acidentes, aumento da eficiência no trânsito das grandes capitais e diminuição no gasto de combustível. Além disso, permitem uma mobilidade para os deficientes físicos e idosos.

A Figura 1.2 apresenta o Stanley, veículo terrestre não tripulado realizando a navegação de maneira autônoma no deserto próximo ao estado da Califórnia nos Estados Unidos da América. Stanley da Universidade de Stanford foi o vencedor do DARPA *Grand Challenge* em 2005.

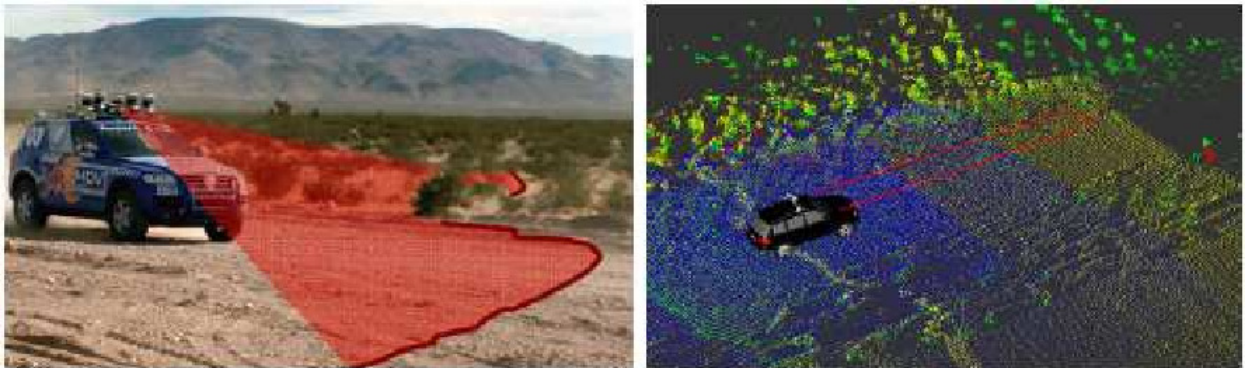


Figura 1.2: Stanley, vencedor do *Grand Challenge* em 2005 (Thrun *et al.*, 2006).

A navegação autônoma em ambientes externos constitui-se como uma tarefa extremamente complexa e desafiadora. Esses ambientes são caracterizados pela sua estrutura, movimento de pessoas, objetos dinâmicos (e.g. bicicletas, carros, caminhões) e fatores climáticos (e.g. chuvas, neve, neblina, vento, entre outros). Entretanto, grande parte das dificuldades de programar um robô móvel para que este realize a navegação de maneira autônoma, deve-se aos seguintes desafios: imprecisão das informações obtidas através de sensores e atuadores, pois os robôs navegam em ambientes não estruturados; a resposta precisa ser dada em tempo real, exigindo uma alta carga de processamento nos computadores embarcados nos robôs; os ambientes em que os robôs operam são dinâmicos e imprevisíveis; e por fim, os sistemas de navegação autônoma devem ser tolerantes a falhas, para que os robôs móveis não causem danos nos seres humanos e prejuízos para a população.

Em contraste com esses desafios, os seres humanos, devido à sua capacidade de perceber o ambiente a sua volta, sabem conduzir de forma eficaz os robôs móveis. Assim, através de exemplos fornecidos por um ser humano ou dados obtidos de sensores e utilizando técnicas de Aprendizado de Máquina, mais especificamente o aprendizado supervisionado, os sistemas de aprendizado inteligentes que serão desenvolvidos nesta tese terão a capacidade de conduzir o robô de maneira autônoma, de forma segura e eficiente.

O Aprendizado de Máquina dedica-se ao desenvolvimento de algoritmos, os quais permitem que os computadores aprendam a partir de exemplos fornecidos pelo usuário. Aprendizado de Máquina é uma área da Inteligência Artificial em que são investigadas técnicas computacionais de aprendizado e aquisição de conhecimento (Rezende, 2003).

As abordagens atuais de Aprendizado de Máquina apresentam os aprendizados supervisionado, não-supervisionado e semi-supervisionado, os quais se diferenciam pela presença de rótulos nos dados (Bruce, 2001; Jain *et al.*, 1999; Mitchell, 1997). O aprendizado supervisionado baseia-se em um conjunto de dados, mais especificamente um conjunto de exemplos

de treinamento, em que as respostas ou saídas corretas (desejadas) são fornecidas e, com base neste conjunto de treinamento, os algoritmos em geral generalizam para responder corretamente a todas as entradas possíveis. Esse tipo de aprendizado é também conhecido na literatura como aprendizado a partir de exemplos (Marsland, 2009).

A proposta apresentada nesta tese consiste de dois métodos de navegação autônoma baseados no aprendizado supervisionado. Mais especificamente, são abordados dois problemas no âmbito da robótica móvel: navegação autônoma baseada no condutor humano e navegação autônoma baseada na irregularidade do terreno.

O primeiro problema é a navegação autônoma baseada no condutor humano em um ambiente urbano. Este apresenta uma série de desafios: o veículo elétrico (CaRINA 1) deve seguir os pontos de GPS pelo trajeto, respeitar as áreas não navegáveis (e.g. calçadas, vegetação) e manter-se em uma via urbana permitindo a navegação de maneira autônoma, segura e eficiente. Para lidar com esses desafios, foi desenvolvido um sistema de navegação autônoma baseado em visão computacional e localização por GPS, utilizando Redes Neurais Artificiais para aprender os dados fornecidos por um motorista humano, enquanto o CaRINA 1 percorre o trajeto.

O segundo problema discorre sobre a navegação autônoma baseada na irregularidade do terreno em um ambiente externo. Este apresenta um desafio interessante, onde um robô real (Husky) deve reduzir a quantidade de vibração experimentada durante a navegação autônoma em diferentes terrenos; como consequência, apresentará uma redução do consumo de energia. Para conseguir isso, foi desenvolvido um sistema de navegação autônoma baseado em laser e uma Unidade de Medida Inercial utilizando Processos Gaussianos e Otimização Bayesiana para aprender uma representação da irregularidade do terreno, favorecendo o movimento do robô real para áreas navegáveis que produzam níveis de vibração mais baixos.

1.2 Justificativa

Atualmente, os robôs móveis autônomos apresentam uma série de aplicações desenvolvidas com sucesso nos ambientes externos. O primeiro problema abordado nesta tese relaciona-se com a condução autônoma ou assistida de robôs móveis (e.g. veículo) em ambientes urbanos, para permitir uma diminuição do número de acidentes, gastos com atropelamentos de pedestres e mobilidade para deficientes físicos. O segundo problema é destinado ao monitoramento ambiental de robôs móveis em ambientes não estruturados, para que os robôs percorram e monitorem áreas específicas do ambiente de maneira autônoma ou assistida por um guarda ambiental.

O aprendizado supervisionado aplicado ao problema de navegação autônoma utilizando robôs móveis apresenta uma importância significativa para a robótica móvel, pois ele permite

um robô navegar de maneira autônoma e eficiente sobre um ambiente externo. Nesses ambientes é muito difícil programar um robô que percorra áreas seguras de forma autônoma, pois os ambientes são imprevisíveis e não estruturados. Dessa forma, a existência de um sistema de navegação inteligente que aprenda exemplos fornecidos por um motorista humano ou dados provenientes de sensores posicionados sobre os robôs móveis, permitirá ao robô realizar a navegação de maneira autônoma e eficiente sobre um ambiente externo.

Um ser humano conduz o robô de forma eficaz, pois ele pensa, percebe quem está à sua volta e direciona o robô em ambientes navegáveis. Além disso, o sistema inteligente evitará obstáculos, mantendo o robô em áreas seguras, tendo, por consequência, preservando a vida útil do robô. Finalmente, o robô não causará danos nas pessoas e o sistema permitirá uma boa qualidade ao acesso às informações dos sensores, pois o robô não permanecerá em áreas com uma agitação excessiva.

Além disso, os trabalhos desenvolvidos nesta tese fazem parte de um projeto temático, que tem por propósito principal a criação de veículos autônomos inteligentes para a navegação em ambientes urbanos. O projeto conhecido como CaRINA (Carro Robótico Inteligente para Navegação Autônoma) é coordenado pelo Laboratório de Robótica Móvel (LRM) do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo em São Carlos, tendo como parceira o INCT-SEC (Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos).

O projeto CaRINA apresenta diversos trabalhos em andamento, os quais são os módulos de desenvolvimento do veículo, como por exemplo, arquitetura do sistema, localização, mapeamento, controle do carro, planejamento e aprendizado. Esta tese está diretamente relacionada com o módulo de aprendizado, a fim de permitir uma contribuição significativa para o mesmo. Atualmente, o projeto CaRINA possui dois veículos reais para a realização de testes (experimentos), um veículo elétrico (Carro de Golfe) destacado na Figura 1.3 (a) e um veículo de passeio (Palio Adventure), como pode ser visto pela Figura 1.3 (b).

(a) (b)

Figura 1.3: Veículos de testes pertencentes ao projeto temático CaRINA. (a) Veículo elétrico conhecido como CaRINA 1 e (b) Palio Adventure denominado de CaRINA 2.

1.3 Objetivos

1.3.1 Objetivo Geral

O desenvolvimento de sistemas de navegação autônoma para robôs móveis usando algoritmos de aprendizado supervisionado.

1.3.2 Objetivos Específicos

- Desenvolver sistemas de percepção capazes de oferecer informações para os algoritmos de aprendizado supervisionado;
- Desenvolver sistemas de aprendizado supervisionado capazes de relacionar informações do sistema de percepção com os comandos de controle desejados para cada situação;
- Desenvolver sistemas de navegação capazes de conduzir o robô de maneira autônoma e segura, baseados nas informações obtidas pelo sistema de aprendizado supervisionado.

1.4 Contribuições

As principais contribuições desta tese para o problema de navegação autônoma usando aprendizado supervisionado são:

- O desenvolvimento de um sistema de percepção usando visão computacional, capaz de identificar a geometria da via navegável (Souza *et al.*, 2011b);
- O desenvolvimento de um sistema que utiliza as informações de percepção, câmera, GPS e bússola como entrada de uma RNA para estimar o esterçamento e a velocidade de um veículo autônomo (Souza *et al.*, 2013a);
- O modelo de estimação da irregularidade do terreno a partir de dados de aceleração construídos utilizando regressão de Processos Gaussianos (Souza *et al.*, 2014);
- O algoritmo de planejamento baseado em Otimização Bayesiana que refina o modelo do ambiente ao mesmo tempo que evita as áreas de maior vibração (Souza *et al.*, 2014).

1.5 Organização da Tese

O documento está organizado em 6 Capítulos, cujos conteúdos se encontram estruturados abaixo:

- Capítulo 2 descreve os trabalhos relacionados aos problemas de navegação autônoma baseada no condutor humano e a navegação autônoma baseada na irregularidade do terreno;
- Capítulo 3 aborda as técnicas de aprendizado supervisionado utilizadas neste trabalho, apresentando suas características e limitações;
- Capítulo 4 apresenta a navegação autônoma baseada no condutor humano, descrevendo suas limitações e as perspectivas futuras;
- Capítulo 5 descreve a navegação autônoma baseada na irregularidade do terreno, apresentando suas limitações e os trabalhos futuros;
- Capítulo 6 apresenta a conclusão dos trabalhos realizados nesta tese.

CAPÍTULO

2

Trabalhos Relacionados

Este capítulo descreve os trabalhos relacionados à navegação autônoma utilizando aprendizado supervisionado baseada na condução de um humano, que podem ser visualizados na Seção 2.1, enquanto que a Seção 2.2 apresenta os trabalhos envolvidos com a navegação autônoma utilizando o aprendizado supervisionado baseada na irregularidade do terreno.

2.1 Navegação utilizando aprendizado supervisionado baseada na condução humana

Autonomous Land Vehicle in a Neural Network (ALVINN) (Pomerlau, 1995), um dos primeiros robôs inteligentes baseados em aprendizado, usava Redes Neurais Artificiais (RNAs) e processamento de imagem para conduzir um robô de maneira autônoma em diversos tipos de ambiente.

Inicialmente, as imagens obtidas de uma câmera montada na parte superior do robô (ver Figura 2.1), tinham sua resolução diminuída para reduzir a quantidade de dados e os *pixels* dessa imagem alimentavam uma RNA (Haykin, 1999). Essa rede treinada fornecia como resultado o ângulo que deveria ser aplicado à direção do robô para mantê-lo dentro dos limites da estrada. A arquitetura do sistema possuía 960 unidades de entrada totalmente conectadas com uma camada escondida com 4 unidades, também conectada com as 30 unidades da camada de saída. A desvantagem desta abordagem é o alto tempo requerido para gerar as 1200 cenas de estrada

sintética (simuladas) na etapa de treinamento. Além disso, as diferenças entre as imagens de estrada sintética em que a RNA foi treinada e as imagens reais em que a RNA foi testada, muitas vezes resultaram em um desempenho ruim em situações de condução real, devido aos exemplos fornecidos nas imagens de estrada sintética, que não cobriram o suficiente para conduzir o robô em uma linha única da estrada de teste.

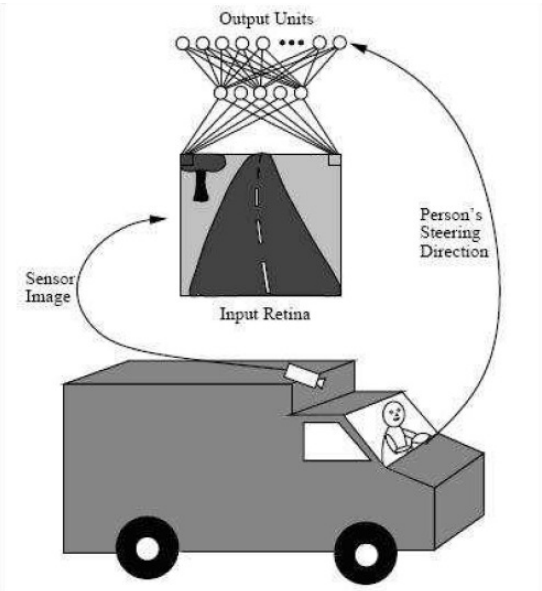


Figura 2.1: Projeto ALVINN (Pomerlau, 1995).

Em Kunzle (2010), um robô simulado, como apresentado na Figura 2.2, utilizou RNAs para calcular a aceleração, com o intuito de desviar dos obstáculos à sua frente. As entradas da RNA foram representadas pelas distâncias dos obstáculos até o robô, a fim de imitar o comportamento de um motorista humano. As saídas do sistema são a aceleração e a direção do robô. Os testes são simulados, com a distância do robô para os obstáculos sempre conhecida (Kunzle, 2010).

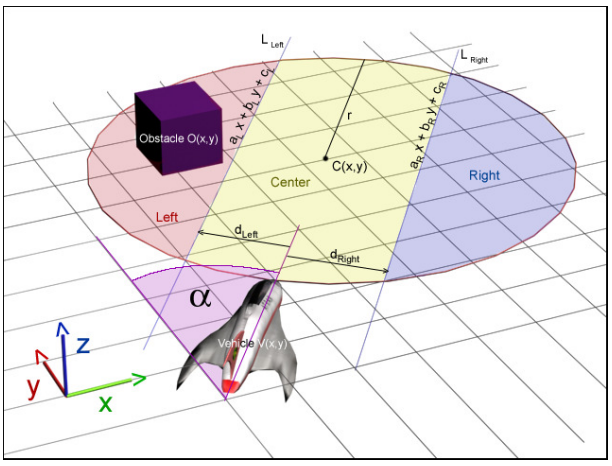


Figura 2.2: A visão do robô simulado (Kunzle, 2010).

Darter e Gordon (2005) utilizou grupos de RNAs para melhorar a segurança e a eficiência do controle de direção de um robô de remoção de neve, nas quais as informações obtidas pelo robô foram apresentadas ao motorista através de um computador. Também obtiveram o cálculo da posição lateral futura do robô (ver Figura 2.3) com base em leituras de vários sensores. No entanto, nenhuma implementação real foi desenvolvida, e os dados utilizados para o treinamento e validação das RNAs foram gerados e não adquiridos do mundo real (Darter e Gordon, 2005).

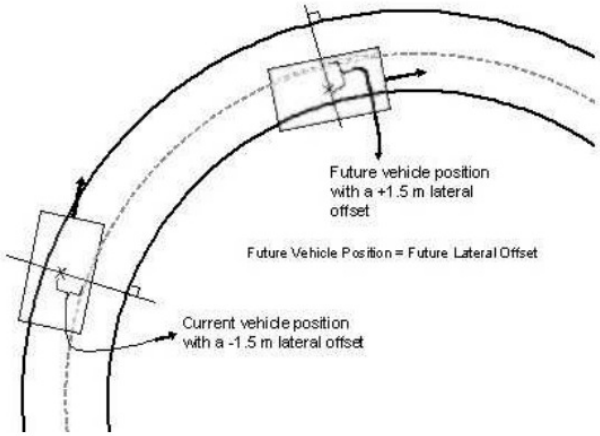


Figura 2.3: Posição lateral do robô atual e futura (Darter e Gordon, 2005).

Chan *et al.* (2007) apresenta um controle de direção e velocidade que permite ao robô antecipar e realizar as curvas. O sistema usa *Generic Self-Organizing Fuzzy Neural Network* (GenSoFNN-Yager), que inclui a inferência Yager (Oentaryo e Pasquier, 2006). GenSoFNN-Yager tem a habilidade de induzir a partir de um baixo nível de informação *fuzzy if-then*. Os resultados mostram que a robustez do sistema em aprender através de um humano é significativa, pois o sistema trabalha com novas estradas não vistas. Os valores de esterçamento gerados pelo motorista autônomo mostram que a antecipação não é sempre suficiente para imitar o comportamento humano. O sistema foi desenvolvido e testado em um ambiente de simulação e grandes variações nas regras foram observadas, o que implica uma alta complexidade do sistema.

Markelic *et al.* (2009) usaram um robô (ver Figura 2.4) que aprende a navegar através de exemplos fornecidos por um condutor humano e dados visuais obtidos de uma câmera posicionada sobre um robô móvel. O robô associa a informação visual com as ações humanas. Essa informação é obtida dos limites da pista, que é detectada em cada imagem. Dois módulos foram usados, um controlador reativo e um planejador. O primeiro produz mapas de informação para o controle da direção e o segundo planeja a ação, isto é, sequências para o controle da direção e velocidade do robô. Os resultados foram satisfatórios, nos casos em que o robô realizou o percurso com êxito em um ambiente específico, entretanto, o sistema não é capaz de prever manobras futuras.

Figura 2.4: (a) Robô. (b) Estação de controle. (c) Pista usada nos testes (Markelic *et al.*, 2009).

No trabalho de Stein e Santos (2010), foi desenvolvido um método que calcula a direção de um robô autônomo em um ambiente real (ver Figura 2.5). RNAs foram usadas para aprender comportamentos baseados em exemplos dos motoristas humanos (Stein e Santos, 2010).

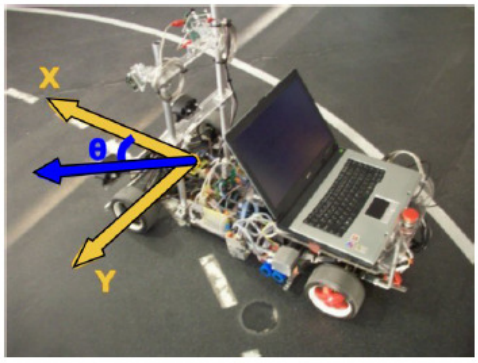


Figura 2.5: O robô autônomo (Stein e Santos, 2010).

A extração dos *pixels* dos limites da estrada são as entradas da RNA e a saída é o ângulo de esterçamento do robô. Para validar as RNAs, foram realizados testes reais e o robô realizou com êxito várias voltas no circuito de teste, circuito inverso e em diferentes caminhos, mostrando boa capacidade de generalização com os dados de treino. Um dos problemas dessa abordagem é a impossibilidade de validar uma RNA na etapa de treinamento sem testar com o robô real.

Um sistema de condução baseado em visão a partir de um supervisor humano foi proposto por Markelic *et al.* (2011). O sistema possui uma arquitetura *multi-thread*, CPU/GPU paralela em um carro real (ver Figura 2.6) e utilizou dados reais para gerar o controle de direção e aceleração para seguir ruas (Markelic *et al.*, 2011). A abordagem utilizada nesse sistema usa um algoritmo de movimento de objetos independentes, a fim de detectar obstáculos utilizando uma câmera estéreo. Uma sequência de ação prévia é comparada com as ações do motorista e um alerta é emitido se houver uma discrepância. A capacidade do sistema para imitar o comportamento humano é analisada em ruas conhecidas e desconhecidas, e os resultados obtidos sugerem seu uso para auxílio à direção, mas a aceleração é definida para a realização de curvas.

(a) (b) (c)

Figura 2.6: (a) Câmera estéreo. (b) Computador. (c) Monitor no carro (Markelic *et al.*, 2011).

As principais diferenças das abordagens apresentadas até o momento com o sistema de navegação autônomo baseada na experiência do condutor humano desenvolvido no Capítulo 4, é que este permite um veículo elétrico CaRINA 1 percorrer rotas (pontos de GPS) de maneira segura e eficiente enquanto que ao mesmo tempo ele mantém-se na rua em um ambiente urbano.

2.2 Navegação utilizando aprendizado supervisionado baseada na irregularidade do terreno

Os ambientes não estruturados podem ser prejudiciais para o robô quando este percorre as regiões com uma elevada vibração. O segundo problema desta tese permite que um robô reduza a quantidade de vibração experimentada durante a navegação em diferentes terrenos, como consequência, apresentando uma redução do consumo de energia. O robô pode navegar em áreas com menor vibração, aumentando sua vida útil e tornando a operação mais segura.

Weiss *et al.* (2006) desenvolveram um método para classificação de terreno baseado na vibração de um robô móvel (ver Figura 2.7). O objetivo deste método é estimar qual tipo de terreno o robô está atualmente atravessando. A estimação é baseada em dados de aceleração linear do eixo vertical do robô, os quais são obtidos através de um acelerômetro.

Figura 2.7: *Cart* utilizado para a coleta de dados (Weiss *et al.*, 2006).

Durante a fase de treinamento, o robô pilotado por um humano coleta dados de aceleração sobre cada terreno (ver Figura 2.8). Os dados de aceleração foram divididos dentro de segmentos, nos quais cada segmento representa um segundo (1s) de viagem do robô. Foram aplicados filtros sobre o segmento de dados de aceleração, a fim de extrair mais informações sobre os dados.

Figura 2.8: Tipos de terrenos utilizados. 1: piso interno 2: asfalto 3: cascalho 4: grama 5: pedra de calçada e 6: areia (Weiss *et al.*, 2006).

Após a extração de característica, o algoritmo SVM usando a função de *kernel* RBF treina *offline* os dados (sinais de vibração com 1s de viagem do robô). Após o treinamento do SVM, novos sinais de vibração podem ser classificados durante a operação do robô. O classificador atribui dados de terreno coletados para uma das classes. Para avaliação, foram divididos aleatoriamente os dados em treino e teste. A classificação dos tipos de terreno apresenta bons resultados, com uma taxa de erro de classificação total em cerca de 5% utilizando os dados de teste.

Weiss *et al.* (2008) propuseram uma abordagem de classificação de terreno que combina predições de terreno baseadas em imagens com predições baseadas em vibração (Weiss *et al.*, 2006). Utilizando as imagens, o robô (ver Figura 2.9), que está equipado com uma câmera estéreo (os autores utilizaram apenas a câmera da esquerda para obter as imagens coloridas) classifica o terreno em sua frente. Quando o robô atravessa áreas que já foram classificadas pelo método baseado em imagens, ele utiliza dados de vibração para verificar sua predição.

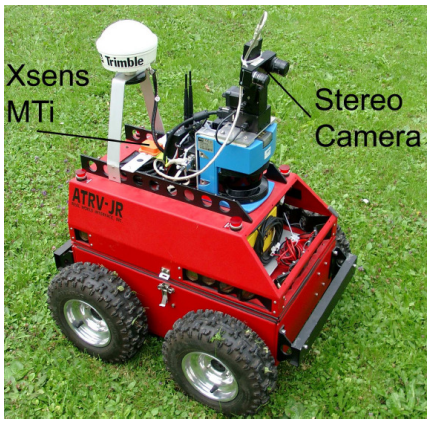


Figura 2.9: Plataforma de teste utilizada (Weiss *et al.*, 2008).

O robô classifica a área à sua frente com base nas Texturas das imagens usando Integrais Invariantes (TII) (Schael, 2002). Se o robô mais tarde percorre uma área que já tenha sido classificada baseada em imagens, ele também classifica a área usando dados de vibração. Nesse caso, combinam-se as predições de ambas as abordagens para compor o método proposto.

O método de classificação baseado em visão tem a fase de treino *offline* e a classificação *online*. Na fase de treino, um humano pilota o robô e usa uma câmera para coletar imagens. A imagem é dividida dentro de sub-imagens, a fim de utilizá-las individualmente. Para cada sub-imagem, calcula-se um histograma de característica baseado em TII. Rotula-se cada sub-imagem com o tipo de terreno verdadeiro e treina-se um SVM com os vetores de características. Na fase de classificação, o robô move-se e adquire novas imagens em sua frente. Da mesma forma que as imagens de treino, divide-se cada imagem de teste em sub-imagens e calcula-se seus histogramas. Por fim, utiliza-se SVM para predizer o tipo de terreno de cada sub-imagem.

Os dados são representados por imagens e vibrações das 14 classes de terrenos. Combinando as predições de ambos os métodos baseado em visão e vibração, a taxa de classificação correta apresentada foi de 87,33%, representando uma melhoria de 12,26% em relação ao método usando apenas vibração, e 16,04% em relação ao método utilizando apenas visão.

A desvantagem principal dessa abordagem (classificação de terreno que combina predições de terreno baseado em visão e vibração) é que ela apenas classifica diferentes tipos de terrenos,

enquanto que o trabalho desenvolvido no Capítulo 5 é capaz de predizer a quantidade de vibração esperada em um domínio contínuo, com um nível de incerteza associado.

2.3 Considerações Finais

Neste Capítulo foram descritos os trabalhos relacionados com a tese, métodos de navegação autônoma baseados no aprendizado supervisionado. Mais especificamente, foram apresentados dois problemas no âmbito da robótica móvel: a navegação autônoma baseada no condutor humano e navegação autônoma baseada na irregularidade do terreno.

O ALVINN (Pomerlau, 1995) navegou de forma autônoma em uma estrada, mas apresentou limitações. Uma das limitações foi o alto tempo requerido para gerar as 1200 cenas de estrada sintética na etapa de treinamento. As diferenças entre as imagens de estrada sintética em que a RNA foi treinada e as imagens reais em que a RNA foi testada, muitas vezes resultaram em um desempenho ruim em situações de condução real, pois os exemplos fornecidos nas imagens sintéticas não foram suficientes para conduzir o robô em uma linha única da estrada de teste.

Markelic *et al.* (2009) desenvolveram um sistema que aprende através do ser humano e dados visuais utilizando um robô seguidor de linhas. Nesse sistema foram utilizados dois módulos, o primeiro é um controlador reativo e o segundo é um planejador. Os resultados foram satisfatórios, pois o robô percorreu a trajetória com êxito, porém o sistema não é capaz de prever manobras futuras.

Um método que calcula a direção de um robô em um ambiente real foi desenvolvido por Stein e Santos (2010), onde RNAs foram utilizadas para aprender dados visuais e comportamentos baseados em motoristas humanos. Uma das limitações encontradas nessa abordagem é a impossibilidade de validar uma RNA na etapa de treinamento sem testar com o robô real.

Um sistema de condução baseado em visão a partir de um supervisor humano é proposto em Markelic *et al.* (2011). O sistema alerta o motorista, caso haja uma discrepância nos valores comparados com uma sequência de ação prévia com as ações do motorista. Avaliando o desempenho do sistema em pistas conhecidas e desconhecidas, mostrou-se que o sistema funciona de forma segura para o esterçamento da direção, mas a aceleração é definida para negociar curvas.

Weiss *et al.* (2006, 2008) apresentaram bons resultados de classificação, porém uma desvantagem dessas abordagens é que apenas classifica terrenos, enquanto que o trabalho desenvolvido no Capítulo 5 prediz uma quantidade de vibração esperada em um domínio contínuo com um nível de incerteza associado usando Processos Gaussianos. Este nunca tinha sido utilizado para a estimação da irregularidade do terreno a partir de dados de aceleração sobre um robô móvel.

CAPÍTULO

3

Aprendizado Supervisionado

Aprendizado de Máquina é uma área da Inteligência Artificial na qual são investigadas técnicas computacionais de aprendizado e aquisição de conhecimento (Rezende, 2003). As abordagens atuais apresentam os aprendizados supervisionado, não-supervisionado e semi-supervisionado, que se diferenciam pela presença de rótulos nos dados (Jain *et al.*, 1999; Mitchell, 1997).

A classificação e a regressão são problemas comumente encontrados nesses tipos de aprendizado, os quais apresentam diferenças. Para o primeiro problema, os valores das classes pertencem a um conjunto de valores discretos (Hastie *et al.*, 2009). O segundo problema, as classes ou rótulos assumem valores contínuos (Cogger, 2010).

O aprendizado supervisionado baseia-se em um conjunto de dados, mais especificamente um conjunto de exemplos de treinamento, em que as respostas ou saídas corretas (desejadas) são fornecidas. Com base neste conjunto de treinamento, os algoritmos em geral generalizam para responder corretamente a todas as entradas possíveis. Este tipo de aprendizado é também denominado de aprendizado a partir de exemplos (Marsland, 2009).

Existem diversas técnicas de aprendizado supervisionado aplicadas na Robótica Móvel, por exemplo, Árvores de Decisão, Máquinas de Suporte Vetorial, Redes Bayesianas, entre outras Hamzei e Hossein (1998); Shen e Hu (2007). Para o primeiro problema abordado nesta tese, a navegação autônoma baseada no condutor humano em um ambiente urbano, foram usadas Redes Neurais Artificiais, para permitir o desenvolvimento de um sistema de navegação autônoma.

O segundo problema discorre sobre a navegação autônoma baseada na irregularidade do terreno em um ambiente externo usando GP e BO, para permitir um sistema de navegação autônoma.

3.1 Redes Neurais Artificiais

As Redes Neurais Artificiais são modelos matemáticos inspirados no cérebro humano, de forma mais específica, em como o sistema nervoso humano processa suas informações. Essas técnicas têm sido desenvolvidas há décadas e existem diversos modelos de RNAs propostos e consolidados na literatura (Alpaydin, 2010; Bishop, 1996; Haykin, 1999).

As RNAs podem também ser chamadas de sistemas paralelos distribuídos (Mitchell, 1997), compostos por unidades de processamento simples (neurônios artificiais) que calculam determinadas funções matemáticas. Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos, essas conexões estão associadas a pesos, os quais armazenam o conhecimento adquirido pelo modelo e servem para ponderar a entrada recebida para cada neurônio da rede (Braga *et al.*, 2007).

As características mais relevantes das RNAs são: capacidade de se adaptar ou aprender por meio de exemplos (e.g. padrões); generalizar o conhecimento adquirido; atuar como mapeadores de entrada e saída (aprendizado supervisionado) e a tolerância a falhas (Haykin, 1999).

McCulloch e Pitts (1943) propuseram o primeiro modelo artificial de um neurônio biológico (Braga *et al.*, 2007). O modelo é composto de n entradas $(x_1, x_2, ..., x_n)$, onde cada entrada tem um peso (w_i) associado. A saída linear u produzida por este neurônio é obtida através da soma ponderada das entradas (x_i) com os respectivos pesos (w_i). Após a obtenção deste resultado, uma função $f(.)$ é aplicada sobre a saída linear u para produzir a saída y do neurônio de McCulloch e Pitts (1943). A função $f(.)$ é conhecida na literatura como função de ativação e esta apresenta diversas formas, por exemplo, limiar, linear, sigmóide logística, tangente hiperbólica, entre outras. A Figura 3.1 apresenta o neurônio artificial proposto por McCulloch e Pitts (1943).

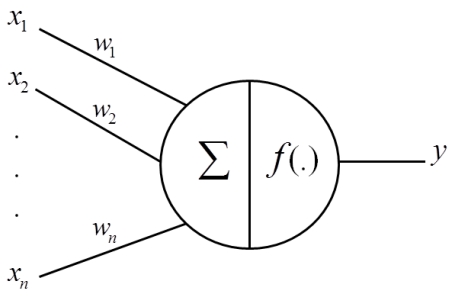


Figura 3.1: Neurônio artificial (Braga *et al.*, 2007).

A seguir serão ilustradas as Equações das funções de ativação sigmóide logística (Equação 3.1) e a tangente hiperbólica (Equação 3.2) utilizadas nesta pesquisa.

$$f(u) = \frac{1}{1 + e^{(-u)}} \tag{3.1}$$

$$f(u) = \frac{e^{(u)} - e^{(-u)}}{e^{(u)} + e^{(-u)}} \tag{3.2}$$

onde u representa a saída linear de um respectivo neurônio (valor de ativação do neurônio), e o e descreve a função exponencial de base e ($e = 2,7182818284$). A amplitude da função de ativação sigmóide logística (Equação 3.1), pode assumir todos os valores entre $[0, 1]$. Entretanto, a função tangente hiperbólica (Equação 3.2), pode assumir todos os valores entre $[-1, 1]$.

Após o desenvolvimento do primeiro neurônio artificial (McCulloch e Pitts, 1943), surgiu a fase do aprendizado em RNAs com o Perceptron (Rosenblatt, 1958). Esse modelo foi bastante usado, porém apenas para solucionar problemas linearmente separáveis. A fim de permitir soluções mais gerais do que a rede descrita anteriormente, (Rumelhart e McClelland, 1986) desenvolveram uma RNA *Multi-Layer Perceptron* (MLP) com o algoritmo *Backpropagation* (BP) (Figura 3.2). A RNA MLP soluciona problemas considerados não linearmente separáveis.

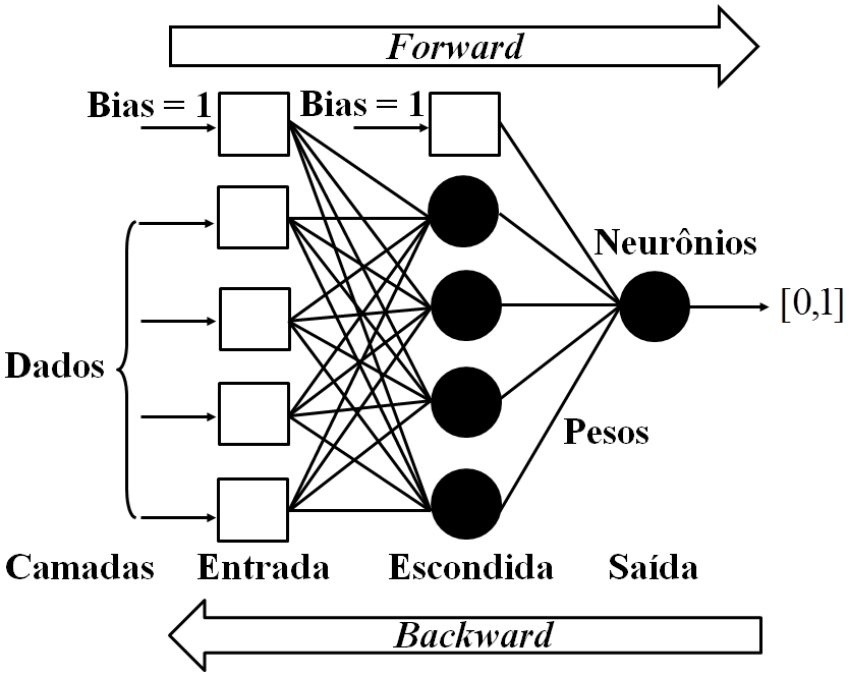


Figura 3.2: RNA MLP usando o algoritmo de treinamento *Backpropagation* (Haykin, 1999).

Normalmente, uma rede neural MLP possui três camadas (Figura 3.2). A camada de entrada, que recebe os dados para serem inseridos na rede, a camada escondida, onde se realiza o cálculo dos pesos com os dados de entrada e a saída, que fornece os resultados (Haykin, 1999).

Cada neurônio de uma camada é conectado com todos os neurônios das camadas anteriores e posteriores, com um determinado peso (isso em uma RNA totalmente conectada). Para que os dados de entrada sejam corretamente relacionados às saídas esperadas, é necessário estimar corretamente os pesos que conectam os neurônios da rede. Essa estimativa é realizada em uma etapa de aprendizado, que consiste em utilizar exemplos pré-classificados e um algoritmo de treinamento de retropropagação do erro ou BP (Rumelhart *et al.*, 1986).

O algoritmo de treinamento das RNAs MLP mais utilizado é o *Backpropagation* que, por ser supervisionado, ajusta os pesos da rede levando em conta a utilização dos pares de entrada e saída. O treinamento ocorre em duas fases: *Forward* e *Backward*. A fase *Forward* é aplicada para definir a saída da rede para um dado padrão de entrada. Já a fase *Backward* utiliza a saída desejada e a saída da RNA para atualizar os pesos de suas conexões (Braga *et al.*, 2007).

O fluxo do algoritmo BP (Braga *et al.*, 2007; Haykin, 1999) é apresentado abaixo. A fase *Forward* é descrita nas etapas seguintes:

1. Os dados de entrada x_i são apresentados à RNA. Cada neurônio pertencente à primeira camada escondida C_1, realizando a soma ponderada v_j (função de soma):

$$v_j(n) = \sum_{i=1}^{m} x_i w_{ji}(n) + b_j \tag{3.3}$$

   onde x_i representa os dados de entrada da RNA e o parâmetro m define o número total de entradas aplicado para o neurônio j. $w_{ji}(n)$ significa o peso sináptico que conecta a saída do neurônio i para a entrada do neurônio j a cada iteração n (apresentação do n-ésimo exemplo de treinamento). O *bias* aplicado para o neurônio j é denotado por b_j e seu efeito é representado por um peso sináptico $w_{j0} = b_j$ conectado a uma entrada fixa igual 1. E por fim, $v_j(n)$ descreve a soma ponderada das entradas (x_i) com os respectivos pesos (w_{ji}). Após realizar a função de soma ($v_j(n)$), a função de ativação (e.g. sigmóide logística - Equação 3.1 ou tangente hiperbólica - Equação 3.2) é aplicada sobre o valor resultante da função de soma, a fim de obter as saídas dos neurônios. A função de ativação que descreve a relação entrada e saída associada com o neurônio j é denotada por $f_j(.)$ e a função normalmente utilizada é a função sigmóide logística. A Equação 3.4 apresenta a saída de cada neurônio j na camada escondida (C_1, C_2, ..., C_h, ..., C_o). Ressalta-se que h representa os índices das camadas da RNA e o representa a camada de saída.

$$y_j(n) = f_j(v_j(n)) \tag{3.4}$$

2. As saídas da camada escondida C_1 serão as entradas da camada escondida C_2, dessa forma, calculando as saídas da camada escondida C_2 do mesmo modo que foi calculada para a camada escondida C_1 (item 1). A única diferença está nos dados de entrada, pois os dados de entrada da camada escondida C_2 são representados pelas saídas da camada escondida C_1. O processo é repetido até alcançar a camada de saída C_o. As Equações 3.5 e 3.6 representam de forma geral a função de soma ($v_j^h(n)$) e a saída do neurônio j ($y_j^h(n)$) pertencente à camada escondida h.

$$v_j^h(n) = \sum_{i=1}^{N} p_i w_{ji}^h(n) + b_j^h \tag{3.5}$$

$$y_j^h(n) = f_j^h(v_j^h(n)) \tag{3.6}$$

onde N representa a quantidade de conexões que chegam ao neurônio j e p_i representa os valores (sinais) que chegam (no sentido da fase *forward*) ao neurônio j. Os demais parâmetros foram descritos no item 1.

3. As saídas produzidas pelos neurônios da camada de saída C_o, que são representadas por $y_j^o(n)$ (saída obtida da RNA) são comparadas com as saídas desejadas $y_j^d(n)$ para o conjunto de dados de entrada (x_i). O erro $e_j(n)$ representa o erro do neurônio j da camada de saída C_o na iteração n, o qual é definido pela Equação 3.7. A Equação 3.8 apresenta o erro dos neurônios da camada de saída na iteração n.

$$e_j(n) = (y_j^d(n) - y_j^o(n)) \tag{3.7}$$

$$E(n) = \frac{1}{2} \sum_{j=1}^{M} (y_j^d(n) - y_j^o(n))^2 \tag{3.8}$$

onde M representa a quantidade de neurônios da camada de saída C_o e E significa a soma dos quadrados do erro dos neurônios de saída.

A fase *Backward* também apresenta algumas etapas, as quais são descritas abaixo:

1. O erro produzido pela camada de saída C_o é utilizado para ajustar os pesos da RNA considerando o gradiente descendente do erro. A fórmula utilizada para o ajuste dos pesos de uma RNA é definida pela Equação 3.9.

$$w_{ji}^{h}(n+1) = w_{ji}^{h}(n) + \eta\delta_{j}^{h}(n)p_i(n) \tag{3.9}$$

onde η representa a taxa de aprendizado, $p_i(n)$ representa os valores (sinais) que chegam (no sentido da fase *forward*) ao neurônio j a cada iteração n.

2. Calcula-se o gradiente local ($\delta_j(n)$) para o neurônio j da camada de saída C_o (Equação 3.10), que é igual ao produto do sinal do erro $e_j(n)$ com a derivada da função de ativação $f'(v_j(n))$ associada. Depois disso, o gradiente local ($\delta_j(n)$) é utilizado para o ajuste dos pesos (Equação 3.9) que estão entre a camada escondida e a camada de saída.

$$\delta_j(n) = e_j(n)f_j'(v_j(n)) \tag{3.10}$$

3. A Equação 3.11 apresenta o cálculo do gradiente local ($\delta_j^h(n)$) para o neurônio j da camada escondida h. Em seguida, o gradiente local ($\delta_j^h(n)$) é utilizado para o ajuste dos pesos (Equação 3.9) que estão entre as camadas escondidas. Além disso, a Equação 3.11 permite o ajuste dos pesos que estão entre a camada de entrada e a camada escondida.

$$\delta_j^h(n) = {f'}_j^h(v_j^h(n)) \sum_{k=1}^{L} \delta_k(n)w_{kj}^h(n) \tag{3.11}$$

onde L é a quantidade de conexões que partem (no sentido da fase *forward*) do neurônio j, δ_k representa o gradiente local do neurônio k; ressaltando que k representa os neurônios da camada seguinte à h (no sentido da fase *forward*). Além disso, $w_{kj}(n)$ significa os pesos que conectam a saída do neurônio k para a entrada do neurônio j a cada iteração n.

4. O item 3 é repetido até que o pesos da camada C_1 sejam ajustados, realizando o ajuste de todos os pesos da RNA para os dados de entrada x_i e sua respectiva saída desejada $y_j^d(n)$.

Cybenko (1989) realizou um estudo para identificar qual o número de camadas intermediárias (ocultas) necessárias em uma RNA. Após o estudo, Cybenko (1989) concluiu que uma camada intermediária é necessária para aproximar qualquer função contínua, e duas camadas são necessárias para aproximar qualquer função (Cybenko, 1988).

O número de neurônios nas camadas intermediárias são definidos de forma empírica. Não existe uma regra geral para a escolha do número de neurônios na camada escondida, mas

Braga *et al.* (2000) sugeriram alguns fatores que devem ser considerados, como por exemplo: o número de exemplos de treinamento, a quantidade de ruído nos exemplos, a complexidade da função a ser aprendida e a distribuição estatística dos dados de treinamento.

Não é recomendado utilizar um número muito grande de neurônios na camada intermediária, a RNA pode memorizar os padrões de treino, ao invés de extrair as características necessárias para a RNA generalizar, ou seja, identificar padrões que não foram apresentados na fase de treino (essa memorização é conhecida como *overfitting*). No caso extremo, com poucos neurônios na camada escondida, a RNA pode gastar um tempo excessivo para buscar a representação ótima (este problema é chamado de *underfitting*) (Braga *et al.*, 2007).

Para evitar o problema de *overfitting*, Reed (1993) sugere uma estimação do erro de generalização durante a fase de treinamento. Dessa forma, apresenta uma divisão dos dados em conjunto de treinamento e conjunto de validação (dados de teste), onde os dados de treinamento são utilizados para o ajuste dos pesos e os dados de teste são empregados para estimar a capacidade de generalização da RNA (Braga *et al.*, 2007) (ver Figura 3.3).

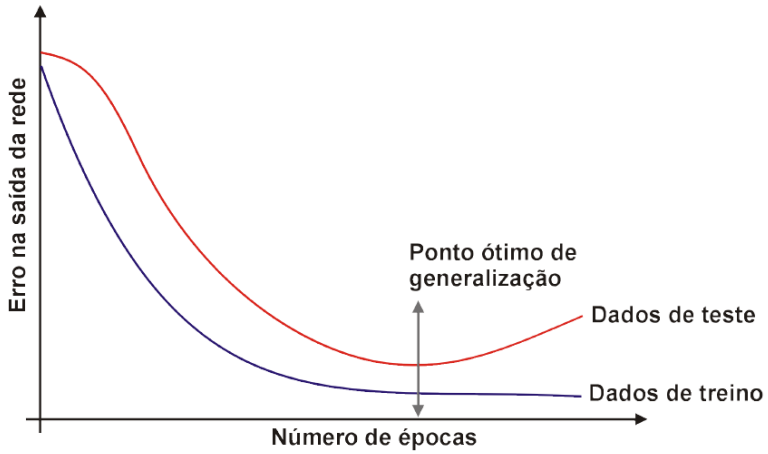


Figura 3.3: Curvas de erro em aprendizado e validação (Pessin, 2013).

A Figura 3.3 ilustra as curvas de erro em aprendizado (dados de treino) e validação (dados de teste). A escolha da RNA nos resultados obtidos do Capítulo 4 é baseada no ponto ótimo de generalização, que apresenta o erro mínimo de validação e máxima capacidade de generalização. Na seção de resultados (Seção 4.2), diversas comparações de topologias de RNAs são realizadas com duas funções de ativação, buscando uma melhor configuração de RNA baseada nas medidas de desempenho Média, Desvio Padrão e o Erro Médio Quadrático (EMQ).

Dentre os diversos algoritmos de treinamento desenvolvidos para RNAs do tipo MLP, o algoritmo BP destaca-se por ser o mais utilizado e difundido na literatura. Esse algoritmo é baseado na regra de Widrow e Hoff (1960) (delta generalizada) e os pesos são ajustados considerando o gradiente descendente do erro (Braga *et al.*, 2007).

A regra delta generalizada requer que as funções de ativação utilizadas pelos neurônios da camada intermediária sejam contínuas (Braga *et al.*, 2000). Para superfícies simples, o uso do gradiente descendente do erro pode encontrar a solução com o erro mínimo, entretanto, para superfícies complexas não existe certeza, acarretando em mínimos locais. Dessa forma, não permite a identificação do mínimo global, que é uma região que apresenta o erro menor comparado com o mínimo local. O mínimo global pode ser encontrado através de diversas inicializações dos pesos da RNA de forma aleatória.

Conforme descrito por Braga *et al.* (2007), o algoritmo BP é lento para várias aplicações, piorando o seu desempenho para problemas maiores e complexos. Além disso, considerando problemas simples, o algoritmo requer que todos os padrões de treino sejam apresentados centenas ou milhares de vezes. A partir disso, existem outros algoritmos de treinamento para as RNAs utilizando MLP que podem ser considerados, como BP com *momentum* (Rumelhart e McClelland, 1986), Quickprop (Fahlman, 1988), RPROP (Riedmiller, 1994), entre outros.

O RPROP é um algoritmo de adaptação global, que realiza o treinamento supervisionado *batch* (aprendizado por época) em RNAs do tipo MLP (Riedmiller, 1994). O aprendizado por época significa que a atualização do peso é realizada após a apresentação de todo o conjunto de treinamento a ser mostrado à RNA. Riedmiller e Braun (1992) afirmam que o algoritmo de treinamento RPROP apresenta benefício com relação ao tempo de convergência e robustez (capacidade da RNA funcionar mesmo em condições anormais).

A escolha da taxa de aprendizado influencia na convergência da RNA (Riedmiller e Braun, 1992). Em virtude disso, o RPROP busca eliminar a influência negativa do valor da derivada parcial do erro para o ajuste dos pesos da RNA para minimizar o erro e este convergir para o mínimo global. Ao contrário do BP, o RPROP utiliza o sinal da derivada (Equação 3.13) e o valor de atualização (Equação 3.14) para o ajuste dos pesos da RNA (Braga *et al.*, 2007).

A Equação 3.12 apresenta o ajuste dos pesos da RNA utilizando o algoritmo de treinamento RPROP, onde $\Delta w_{ij}(t)$ é descrito pela Equação 3.13.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{3.12}$$

Segundo Braga *et al.* (2007), o sinal da derivada indica a direção do ajuste dos pesos da RNA (e.g. aumentar ou diminuir o peso anterior) e o ajuste do peso é representado por um valor de atualização $\Delta_{ij}(t)$, que é descrito pela Equação 3.14, onde $\frac{\partial E}{\partial w_{ij}}$(t) indica a direção do gradiente sobre todos os padrões do conjunto de treinamento.

$$\Delta w_{ij}(t) = \left\{ \begin{array}{rll} -\Delta_{ij}(t), & \text{se} & \frac{\partial E}{\partial w_{ij}}(\text{t}) > 0 \\ +\Delta_{ij}(t), & \text{se} & \frac{\partial E}{\partial w_{ij}}(\text{t}) < 0 \\ 0, & \text{se} & \frac{\partial E}{\partial w_{ij}}(\text{t}) = 0 \end{array} \right\} \tag{3.13}$$

Após se ter conhecimento do sinal da derivada através da Equação 3.13, faz-se necessário saber quanto de ajuste será realizado sobre os pesos da RNA (Equação 3.14). Em virtude disso, o valor de atualização $\Delta_{ij}(t)$ é representado por um processo de adaptação que depende do sinal da derivada do erro com relação ao peso para o mesmo ser ajustado (Braga *et al.*, 2007), onde $0 < \eta^- < 1 < \eta^+$. Segundo Riedmiller e Braun (1993) e usando o SNNS (Zell *et al.*, 1995), o ambiente de desenvolvimento e treinamento das RNAs aplicado em nossos experimentos, os parâmetros são definidos como: $\eta^- = 0.5$, $\eta^+ = 1.2$, $\Delta_0 = 0.1$ (valores iniciais para todos os Δ_{ij}) e $\Delta_{max} = 50$ (o limite superior para a atualização dos valores de Δ_{ij}).

$$\Delta_{ij}(t) = \left\{ \begin{array}{rll} \eta^+\Delta_{ij}(t-1), & \text{se} & \frac{\partial E(t-1)}{\partial w_{ij}} \frac{\partial E(t)}{\partial w_{ij}}(\text{t}) > 0 \\ \eta^-\Delta_{ij}(t-1), & \text{se} & \frac{\partial E(t-1)}{\partial w_{ij}} \frac{\partial E(t)}{\partial w_{ij}}(\text{t}) < 0 \\ \Delta_{ij}(t-1), & \text{se} & \frac{\partial E(t)}{\partial w_{ij}}(\text{t}) = 0 \end{array} \right\} \tag{3.14}$$

A regra de adaptação do RPROP é descrita da seguinte maneira. Se a derivada parcial do erro em relação a um peso w_{ij} mantém o seu sinal, há a indicação de que o último ajuste realizado diminuiu o erro e então o valor de atualização $\Delta_{ji}(t)$ é aumentado pelo fator η^+, acelerando a convergência do treinamento. Caso contrário, se a derivada muda de sinal, isso indica que o último ajuste foi grande demais e, dessa forma, o valor da atualização $\Delta_{ji}(t)$ é reduzido pelo fator η^-, alterando a direção do ajuste (Braga *et al.*, 2007).

3.2 Processos Gaussianos

O Processo Gaussiano (do Inglês, *Gaussian Process* - GP) é uma abordagem probabilística que relaciona os dados observados a fim de estimar os valores desconhecidos. O seu funcionamento baseia-se em uma distribuição gaussiana multivariável (Rasmussen e Williams, 2006).

Usando um conjunto de pontos de treinamento, um GP pode prover uma predição de uma função desconhecida com uma incerteza associada sobre o domínio contínuo. Formalmente, um GP é definido por uma função média $m(x)$ e uma função de covariância $k(x, x')$. Ele aprende a representação de uma função desconhecida f em um aprendizado supervisionado usando um conjunto S de observações conhecidas. $S = \{x_i, y_i\}_{i=1}^N$, onde $x_i \in \mathcal{R}^D$ sendo x as N entradas em um espaço de dimensão D e $y_i \in \mathcal{R}$ são as saídas ruidosas correspondentes.

Dado um ponto de teste x^*, o GP retorna a predição do valor de $f(x^*)$ com a incerteza correspondente. Observações da função $f(x)$ são modeladas como uma Gaussiana, cujo $y = f(x) + \mathcal{N}(0, \sigma_n^2)$, ou seja, média 0 e variância σ_n^2. Dado uma função de covariância $k(x, x')$, a matriz de covariância K pode ser calculada avaliando-se a função de covariância para todas as observações. Onde K é uma matriz de $N \times N$ dados de treinamento, em que os elementos (i, j) são iguais a $k(x_i, x_j)$. $K = k(x, x)$, $K_*^T = k(x^*, x)$, $K_* = k(x, x^*)$ e $K_{**} = k(x^*, x^*)$. Os dados de treinamento y (saída desejada) pertencem ao conjunto S de observações conhecidas, que são modelados através de uma distribuição normal multivariada:

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \quad \begin{bmatrix} K + \sigma_n^2 I & K_* \\ K_*^T & K_{**} \end{bmatrix}\right) \tag{3.15}$$

Esta equação pode ser redefinida em termos de probabilidade condicional $p(y_*|y)$, considerando uma distribuição gaussiana. A distribuição preditiva para os dados de teste x^* é:

$$y^*|y \sim \mathcal{N}(\mu^*, \Sigma^*) \tag{3.16}$$

Onde,

$$\mu^* = K_*^T[K + \sigma_n^2 I]^{-1} y \tag{3.17}$$

$$\Sigma^* = K_{**} - K_*^T[K + \sigma_n^2 I]^{-1} K_* \tag{3.18}$$

Assim sendo, μ^* estima a média de y^* e Σ^* resulta na variância do valor de y^* para entrada x^*. I representa a matriz identidade. As matrizes de covariância K, K_* e K_{**} representam a correlação entre as variáveis, as quais estão definidas abaixo:

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1), & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{3.19}$$

$$K_* = \begin{bmatrix} k(x^*, x_1) & k(x^*, x_2) & \cdots & k(x^*, x_n) \end{bmatrix} \tag{3.20}$$

$$K_{**} = k(x^*, x^*) \tag{3.21}$$

A função de covariância é fundamental para o Processo Gaussiano, pois ela codifica as suposições sobre a função em que o modelo GP deseja aprender. A função de covariância quantifica a similaridade entre dois pontos, ou seja, quão similar cada ponto está um do outro no espaço de entrada x. Isto é importante em aprendizado supervisionado, onde a inferência dos novos dados baseia-se nos exemplos anteriores da função $f(x)$. Por fim, duas entradas próximas uma da outra apresentarão saídas semelhantes (Rasmussen e Williams, 2006).

As funções de covariância normalmente utilizadas são Linear, Exponencial, Exponencial Quadrática, Matérn 3 e Matérn 5 (Rasmussen e Williams, 2006), onde suas equações são dadas por:

$$k_{\text{LI}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \mathbf{x}\mathbf{x}', \tag{3.22}$$

$$k_{\text{EXP}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sqrt{r}\right), \tag{3.23}$$

$$k_{\text{SQEXP}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{r}{2}\right), \tag{3.24}$$

$$k_{\text{MAT3}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{3r}\right) \exp\left(-\sqrt{3r}\right), \tag{3.25}$$

$$k_{\text{MAT5}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \sqrt{5r} + \frac{5r}{3}\right) \exp\left(-\sqrt{5r}\right), \tag{3.26}$$

onde $r = (\mathbf{x} - \mathbf{x}')\mathrm{L}(\mathbf{x} - \mathbf{x}')^T$, L é uma matriz diagonal de tamanho D, cujos elementos são $\mathrm{L}_{ii} = 1/l_i^2$, um parâmetro de comprimento de escala associado a cada dimensão do espaço de entrada. A matriz L e o fator σ_f, são os hiper-parâmetros dessas funções de covariância. O parâmetro l representa o comprimento de escala (fator de escala), que indica quão distante dois pontos têm de estar um do outro para a saída do GP alterar significativamente. Já o parâmetro σ_f representa a variância do sinal, responsável por escalar (dimensionar) a função de covariância.

A Figura 3.4 ilustra um exemplo da predição de GP usando diferentes funções de covariância, mostrando que a suavidade da predição depende da função de covariância escolhida. As funções de covariância apresentadas são: Exponencial Quadrática, Matérn 3 e Matérn 5, as quais são representadas pelas Equações 3.24, 3.25 e 3.26, respectivamente.

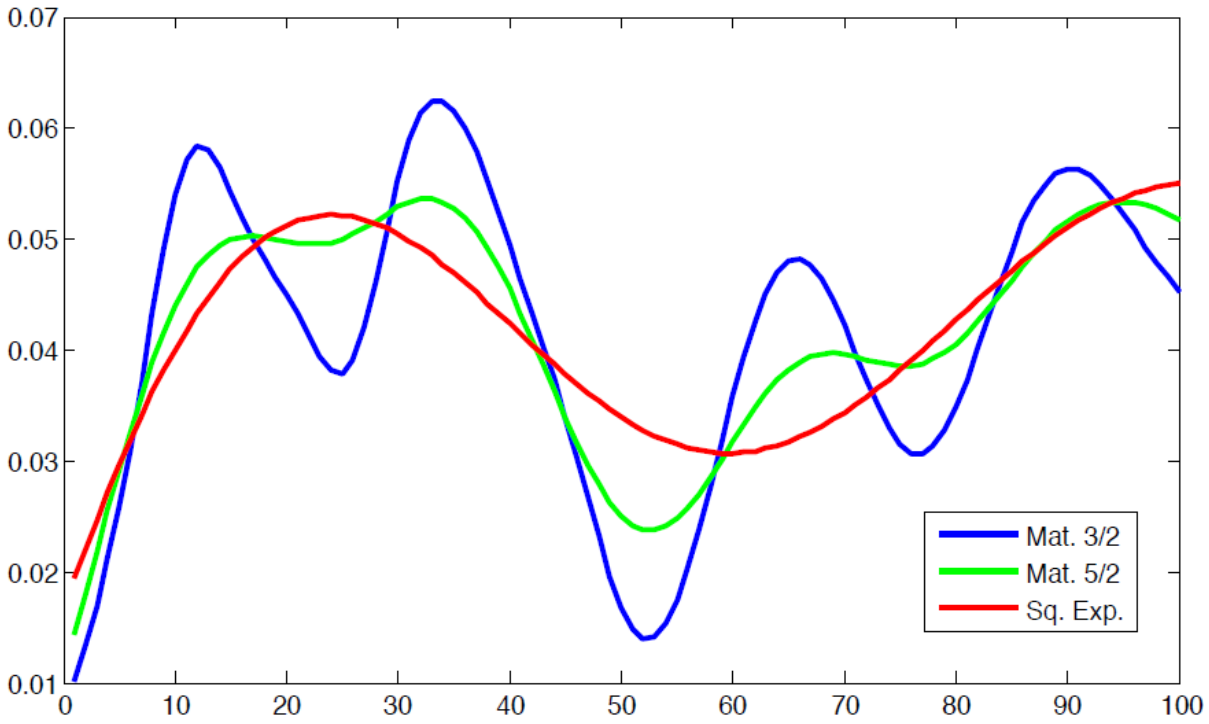


Figura 3.4: Exemplo da predição de GP com três funções de covariância (Guizilini, 2013).

A função de covariância $k(\mathbf{x}, \mathbf{x}')$ deve ser escolhida, ou seja, avaliada dependendo da aplicação a ser observada. A função de covariância frequentemente utilizada na literatura é a função exponencial quadrática (Ebden, 2008), definida pela Equação 3.24.

Assim sendo, σ_f (variância do sinal) restringe o valor máximo da covariância e l (comprimento de escala) preocupa-se em ajustar a proximidade dos valores entre $\mathbf{x}$ e $\mathbf{x}'$. Dessa forma, quanto maior $k(\mathbf{x}, \mathbf{x}')$, maior será a correlação entre $\mathbf{x}$ e $\mathbf{x}'$.

O aprendizado do Processo Gaussiano é o processo de selecionar automaticamente hiper-parâmetros θ da função de covariância escolhida. Esta fase de aprendizado é conhecida como fase de treinamento discutida previamente, onde o GP está condicionado a um conjunto de observações y. O conjunto de hiper-parâmetros que determina completamente um GP é dado por: $\theta = \{l, \sigma_f, \sigma_n\}$. Ajustando esses hiper-parâmetros, o comportamento do GP é adaptado para conseguir uma boa representação da função desconhecida. O conjunto ótimo dos hiper-parâmetros θ^* pode ser encontrado maximizando a verossimilhança marginal do log conhecido por Rasmussen e Williams (2006) como *Log Marginal Likelihood* (LML),

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ \mathrm{LML}(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \ , \tag{3.27}$$

com

$$\mathrm{LML}(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi \ . \tag{3.28}$$

onde $K = K + \sigma_n^2 I$ é a matriz de covariância para as saídas ruidosas y. A Equação 3.28 apresenta a verossimilhança marginal do log, a fim de otimizar os hiper-parâmetros θ, os quais são os parâmetros da função de covariância.

Existem três termos da expressão LML, definida pela Equação 3.28. O primeiro termo é o único termo que envolve os alvos observados y (ajuste dos dados), penalizando os modelos que não são capazes de explicar y. O segundo termo reflete a complexidade do modelo, penalizando os modelos que são definidos para y (depende apenas da função de covariância e os valores de entrada das observações). O terceiro termo é uma constante de normalização. Como a função de verossimilhança marginal do log é negativa e inclui ambos ajuste dos dados e a complexidade do modelo, ele é reconhecido como sendo resistente ao *overfitting* e generaliza bem, mesmo durante o treinamento em dados ruidosos (Rasmussen e Williams, 2006).

Para encontrar o conjunto ótimo de hiper-parâmetros θ^*, deve-se maximizar a verossimilhança marginal do log definida pela Equação 3.28. Isso pode ser conseguido com algoritmos de otimização padrão, tais como gradiente descendente.

A Figura 3.5 ilustra o funcionamento do GP para o problema de Regressão. Os dados de treinamento são representados pelas cruzes vermelhas, que determinam a saída $f(x)$ dos valores x desconhecidos. A sequência da inserção das observações sobre o modelo GP dá-se na parte superior da esquerda para à direita (Figura 3.5). O primeiro painel mostra que não existe nenhuma observação, então o modelo preditivo tem média zero e alta variância (área cinza) ao longo de todo o espaço de entrada x. À medida que os dados são coletados, o modelo GP torna-se mais preciso em torno das observações e diminui a incerteza (variância). A regressão de cada um dos valores resulta em sua média e variância, que são representadas pelas curvas vermelhas e regiões cinzas, respectivamente.

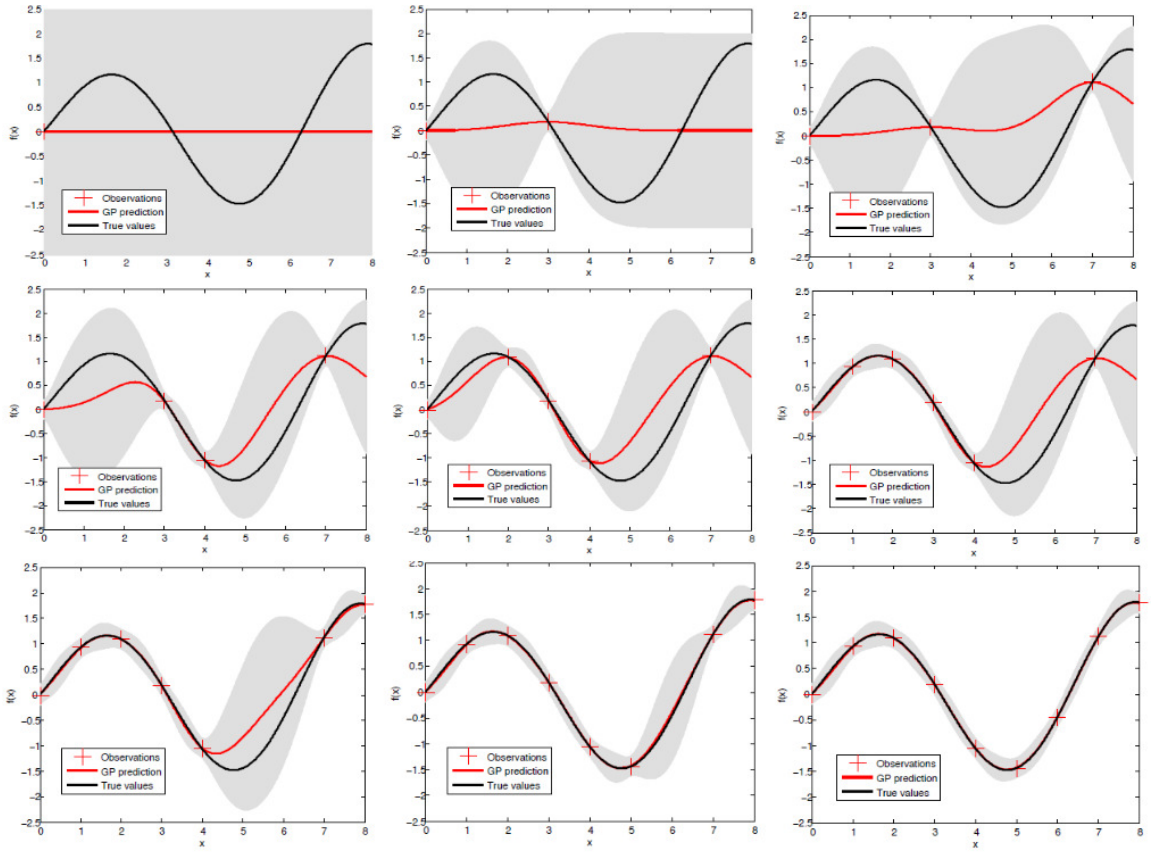


Figura 3.5: Exemplo de funcionamento do Processo Gaussiano (Guizilini, 2013).

A Figura 3.6 apresenta um exemplo de GP para o problema de Regressão variando o parâmetro comprimento de escala (l). Os dados de treinamento são representados pelas cruzes, determinando a saída dos valores desconhecidos. A regressão de cada um dos valores resulta em sua média e variância, representadas respectivamente pelas curvas azuis e áreas cinzas.

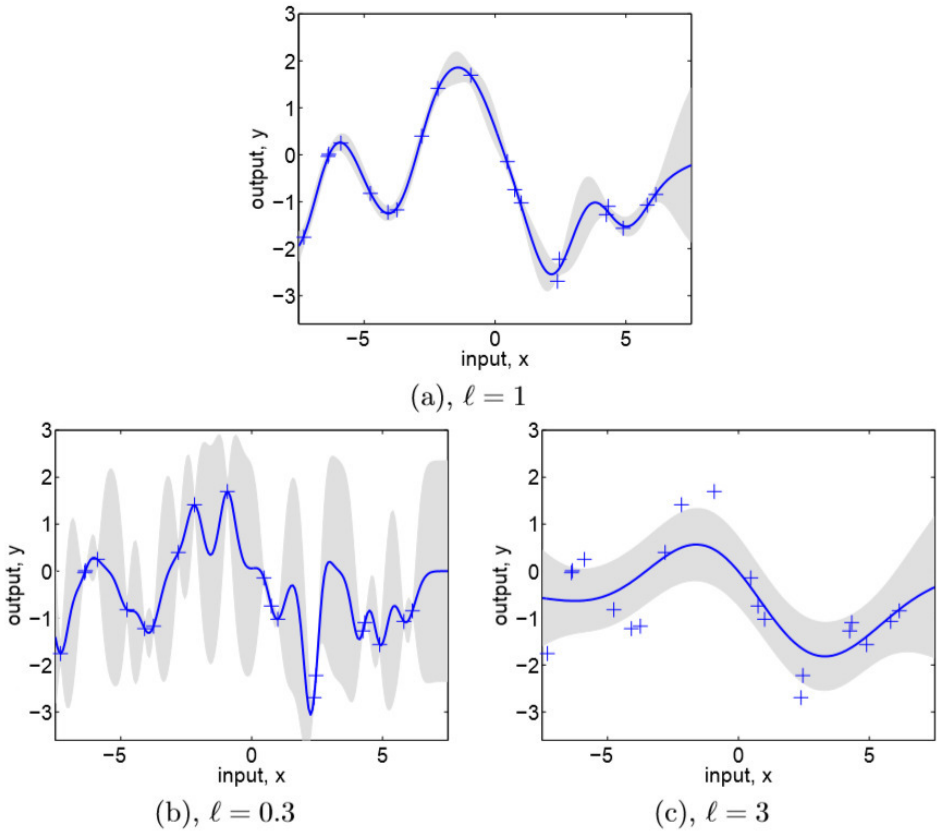


Figura 3.6: (a) Dados gerados a partir de um GP com hiperparâmetros (l, σ_f, σ_n) = (1, 1, 0.1), como mostrado pelos símbolos +. Usando a predição de GP com estes hiperparâmetros, obtendo uma região de confiança de 95% para a função f (mostrado em cinza). Painéis (b) e (c) mostram novamente a região de confiança de 95%, mas desta vez para os hiperparâmetros (0.3, 1.08, 0.00005) e (3.0, 1.16, 0.89), respectivamente (Rasmussen e Williams, 2006).

3.3 Otimização Bayesiana

A otimização Bayesiana (Brochu *et al.*, 2010) é uma estratégia de otimização que trata com funções desconhecidas, onde as avaliações são custosas e ruidosas. Essa estratégia é aplicada em situações em que não se tem uma expressão definida da função, mas se pode obter observações (possivelmente ruidosas) dessa função em valores amostrados. A otimização Bayesiana é favorável quando essas avaliações são custosas, quando o problema a ser analisado não é convexo, ou seja, quando o objetivo é a busca por uma otimização global. Vale salientar que as técnicas de otimização Bayesiana são abordagens eficientes em termos de avaliações da função que se pretende analisar (Mockus, 1994; Sasena, 2002; Streltsov e Vakili, 1999).

A Figura 3.7 apresenta uma execução do algoritmo de otimização Bayesiana para um problema de uma dimensão. O algoritmo de otimização inicializa com dois pontos (cor preta). A cada iteração, a função de aquisição (definida pela cor verde) é maximizada a fim de determinar a próxima amostra (ponto) da função objetivo, ou seja, a função de aquisição leva em conta a média e a variância das predições sobre o espaço para modelar a utilidade da amostragem. O objetivo do algoritmo de otimização Bayesiana é o argumento máximo da função de aquisição, onde o Processo Gaussiano é atualizado e o processo é repetido (Brochu *et al.*, 2010).

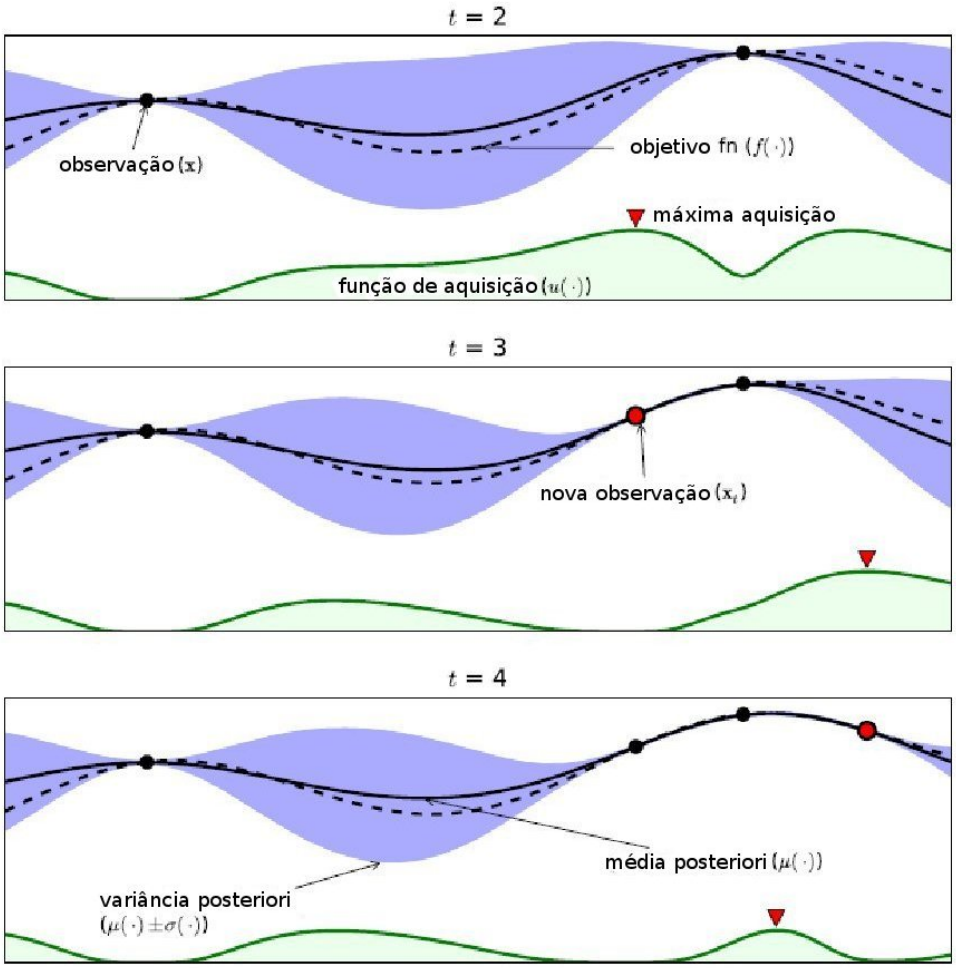


Figura 3.7: Um exemplo do uso de otimização Bayesiana para um problema de 1-D. As figuras ilustram uma aproximação do GP da função objetivo sobre quatro iterações de valores amostrados da mesma. Também ilustra a função de aquisição nas curvas verdes sombreadas. A aquisição é alta, onde o GP prevê um objetivo alto (explotação) e onde a incerteza da predição é alta (exploração), áreas com ambos os atributos são primeiro amostradas. A área à esquerda permanece não-amostrada, ao mesmo tempo que se tem uma incerteza (variância) alta (Brochu *et al.*, 2010).

A otimização é um campo fundamental da matemática (Brochu *et al.*, 2010). Para aproveitá-la para os fins do algoritmo de Otimização Bayesiana é preciso restringi-la, onde a restrição é definida pela maximização. A maximização de uma função de um valor real pode ser dada pela Equação 3.29 e a maximização da função transformada $[-f(x)]$ é dada pela Equação 3.30.

$$\mathbf{x}^\star = \underset{\mathbf{x}}{\operatorname{argmax}}\, f(\mathbf{x}) \tag{3.29}$$

ou mínimo

$$\mathbf{x}^\star = \underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmax}}\, (-f(\mathbf{x})) \ . \tag{3.30}$$

O algoritmo de otimização Bayesiana utiliza a priori (GP) e evidências (função de aquisição) para definir uma distribuição posteriori ao longo do espaço de funções. O modelo Bayesiano permite uma forma sofisticada para que as informações a priori (parâmetros iniciais da função de covariância de um GP e a mesma) possam descrever os atributos da função objetivo, tal como a suavidade ou as posições mais prováveis de serem escolhidas (o máximo da função), mesmo quando a própria função objetivo não é conhecida (Brochu *et al.*, 2010).

O processo de decidir qual será a próxima amostra (e.g. posição sob um espaço 2-D), requer a escolha de uma função de utilidade e uma forma de otimizá-la. A fim de esclarecer qual função estamos discutindo, mencionaremos a função utilidade como sendo a função de aquisição.

A função de aquisição é útil para orientar a busca para o ótimo. As funções de aquisição são definidas de forma que alta aquisição corresponde potencialmente a valores altos da função objetivo, pois a predição é alta, a incerteza é ótima. A maximização da função de aquisição é utilizada para selecionar o próximo ponto onde se avalia a função (Brochu *et al.*, 2010).

O Algoritmo 1 (Brochu *et al.*, 2010) foi modelado para encontrar a função f desconhecida, ruidosa e custosa, ou seja, para encontrar o máximo como definido pela Equação 3.29.

Algorithm 1 Otimização Bayesiana

1: $\mathbf{x}_i$: ponto da amostra escolhido na iteração i.
2: s: função de aquisição.
3: f: função desconhecida. (irregularidade do terreno)
4: **for** $i = 1, 2, 3, \ldots$ **do**
5: Encontre $\mathbf{x}_i = \operatorname{argmax}_{\mathbf{x}} s(\mathbf{x})$
6: Adquira uma amostra de f na posição $\mathbf{x}_i$.
7: Atualize o modelo GP de f com a nova amostra.
8: **end for**

A Otimização Bayesiana (BO, do Inglês, *Bayesian Optimisation*) é definida por dois elementos, um GP e uma função de aquisição que codifica a utilidade de selecionar uma amostra

apropriada na posição $\mathbf{x}$. O primeiro elemento (GP, Seção 3.2) usa amostras adquiridas $\mathbf{x}_i$ para construir um modelo da função desconhecida e realizar a regressão sobre o espaço não-amostrado.

O segundo elemento é a função de aquisição, que estabelece a próxima posição $\mathbf{x}$ para amostra. A função de aquisição definida neste trabalho é a função $DUCB$, do Inglês, *Distance Upper Confidence Bound*, dada pela Equação 5.10. Existem outras funções de aquisição, como *Lower Confidence Bound*, *Upper Confidence Bound*, entre outras (Brochu *et al.*, 2010). A combinação desses dois elementos proporciona ao sistema proposto lidar de forma inteligente com a exploração e explotação (refinamento) sobre o ambiente interno ou externo.

Em cada iteração (Algoritmo 1), uma nova posição da amostra é selecionada pelo conhecimento das informações previamente adquiridas. Ao maximizar a função de aquisição s, a melhor posição é selecionada para testar a função desconhecida da irregularidade do terreno.

Existem diversos trabalhos de otimização Bayesiana, os quais podem ser encontrados no trabalho descrito por Brochu *et al.* (2010). Além disso, um trabalho de Otimização Bayesiana utilizando Processos Gaussianos tem sido aplicado com êxito para otimização denominada de otimização global eficiente (Santner *et al.*, 2003).

Também existem diversos testes de consistência para o algoritmo de Otimização Bayesiana, por exemplo, para o problema de uma dimensão definido por Locatelli (1997) e para uma simplificação do algoritmo utilizando particionamento quando trata-se com dimensões elevadas (Zilinskas e Zilinskas, 2002). O trabalho descrito de Vasquez e Bect (2008) trata da convergência do algoritmo de Otimização Bayesiana usando Processos Gaussianos.

Otimização Bayesiana tem sido aplicada em diversas áreas de aprendizado de máquina, tais como aquelas para aprender um conjunto de parâmetros que maximiza a velocidade de um robô AIBO (Lizotte, 2008). Osborne (2010) utiliza otimização Bayesiana para selecionar as posições de um conjunto de sensores em um sistema. Além dessas aplicações, Brochu *et al.* (2010) apresentam outras utilizando otimização Bayesiana.

CAPÍTULO

4

Navegação Autônoma Baseada no Condutor Humano

Este capítulo apresenta a metodologia utilizada e os resultados obtidos para o primeiro problema abordado nesta tese: a navegação autônoma baseada no condutor humano em um ambiente urbano. O problema consiste em desenvolver um sistema capaz de conduzir um veículo em um ambiente urbano seguindo uma trajetória pré-definida e evitando obstáculos e regiões não navegáveis. O sistema deve ser capaz de utilizar informações fornecidas por um condutor humano. Para tanto, foram utilizadas as técnicas *Template Matching* e Redes Neurais Artificiais.

4.1 Metodologia

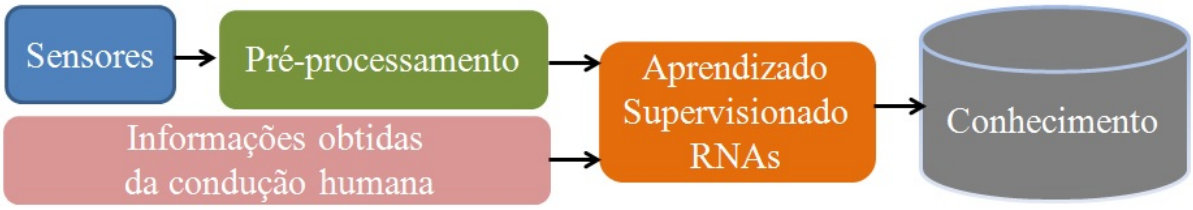
A metodologia utilizada (Souza *et al.*, 2013a) permitiu que um veículo elétrico conhecido como CaRINA 1 (ver Figura 4.1) adquirisse conhecimento para realizar uma rota em um ambiente urbano baseado em pontos esparsos de GPS (sete coordenadas de GPS foram utilizadas em um percurso de aproximadamente 1,08 km) de forma autônoma. As Figuras 4.8 e 4.11 ilustram os ambientes urbanos utilizados para avaliação da metodologia desenvolvida.

Figura 4.1: Plataforma de teste CaRINA 1 da Universidade de São Paulo em São Carlos.

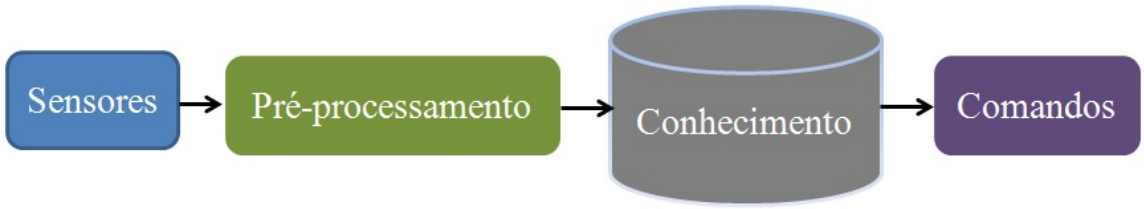
A metodologia do sistema de navegação autônoma utilizando aprendizado (ver Figura 4.2) possui diversos sensores, tais como: uma câmera de vídeo, uma bússola e um GPS. Essa metodologia é dividida em duas etapas: a etapa de aprendizado e a etapa de aplicação.

A etapa de aprendizado é composta de 3 fases. A primeira e a segunda realizam o pré-processamento, no qual uma imagem colorida é obtida por uma câmera posicionada no veículo e a rua é identificada usando RNAs. O resultado desta fase é um mapa de navegabilidade que é usado na próxima fase. Na segunda fase, o algoritmo *template matching* é aplicado para identificar a geometria da rua à frente do veículo (linha reta, curvas suaves e rígidas para esquerda ou direita), além disso, é calculado um percentual de ocupação para cada um dos *templates*. Aplicando-se um limiar sobre os percentuais de ocupação, o sistema distingue áreas livres de ocupadas baseado no mapa de navegabilidade e a geometria da rua. Por fim, as RNAs são aplicadas para definir a ação que o veículo deve realizar para se manter na rua com êxito, produzindo, assim, um conhecimento para ser usado posteriormente na etapa de aplicação.

A etapa de aplicação é composta de 2 fases. Na primeira fase é realizado o pré-processamento da mesma forma que na etapa de aprendizado e a segunda fase é baseada no conhecimento adquirido pela etapa de aprendizado, o qual produz os comandos de esterçamento e velocidade para serem inseridos no CaRINA 1. Todas essas etapas serão descritas nas próximas subseções.



(a) Etapa de aprendizado.



(b) Etapa de aplicação.

Figura 4.2: Esquema geral do sistema de navegação autônoma usando aprendizado.

4.1.1 Identificação da rua

Adotou-se o método proposto de Shinzato e Wolf (2011), que utiliza RNAs para serem aplicadas dentro de uma tarefa de identificação da rua. Baseado nos resultados, um sistema formado por seis RNAs MLP foi proposto para identificar as regiões navegáveis em ambientes urbanos (Figura 4.3 (a)). Uma imagem real desse ambiente pode ser vista na Figura 4.3 (b). O resultado dessa combinação de saída das RNAs é um mapa de navegabilidade (Figura 4.3 (c)). Esta etapa divide uma imagem dentro de blocos de *pixels*, onde os blocos mais claros são as áreas mais prováveis de serem consideradas navegáveis. A vantagem desta abordagem é que as RNAs podem ser treinadas para identificar diferentes tipos de regiões navegáveis e não navegáveis (e.g. ruas pavimentadas, asfalto e não pavimentadas, calçadas, vegetação).

(a) (b) (c)

Figura 4.3: (a) Classificador, (b) Imagem Real e (c) Mapa de Navegabilidade.

Inicialmente, a etapa de identificação da rua divide a imagem dentro de blocos de $(K_b \times K_b)$ *pixels*, como mostrado na Figura 4.4. Diversas características são calculadas para cada bloco. Uma característica é representada pelo cálculo de uma medida estatística em um bloco de *pixels* considerando um canal de cor específico. As medidas estatísticas utilizadas estão especificadas em Shinzato (2010): média, entropia, variância e energia. Essas medidas são relacionadas com os canais de cor RGB, HSV, YUV e RGB normalizado. Os componentes R, G, B são representados pelo vermelho (*red*), verde (*green*) e azul (*blue*). Os componentes H, S, V são representados pelo matiz (*hue*), saturação (*saturation*) e brilho (*value*). Os componentes Y, U, V são representados pelo brilho, tonalidade azul e tonalidade vermelha. O RGB normalizado é composto por $(R/(R+G+B))$, $(G/(R+G+B))$, $(B/(R+G+B))$. Nesta etapa, um frame com resolução de $(M \times N)$ *pixels* foi deslocado nos grupos com $(K_b \times K_b)$ *pixels*. Numa imagem representada por uma matriz I de tamanho $(M \times N)$, o elemento $I(m, n)$ corresponde ao *pixel* na linha m e coluna n da imagem, onde $(0 <= m < M)$ e $(0 <= n < N)$. Portanto, o grupo $G(i, j)$ contém todos os *pixels* de $I(m, n)$, de tal forma que $((i*K_b) <= m < ((i*K_b)+K_b))$ e $((j*K_b) <= n < ((j*K_b)+K_b))$. Esta estratégia tem sido utilizada para reduzir a quantidade de dados, permitindo um processamento mais rápido.

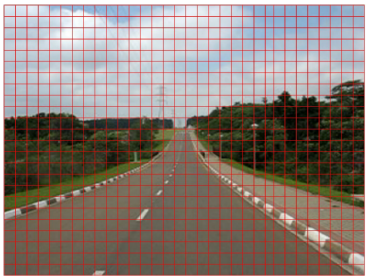


Figura 4.4: Imagem com os blocos de dimensão (10 x 10) *pixels* (Shinzato, 2010).

Uma vez que um bloco é processado, seus atributos são usados como entradas nas RNAs. As RNAs são usadas para classificar os blocos considerando os seus atributos. A RNA usada nesta etapa de identificação da rua consiste de uma MLP usando o algoritmo RPROP. A topologia das RNAs é representada por três camadas, nas quais cada RNA contém uma camada de entrada com os neurônios de acordo com as características de entrada da imagem (Tabela 4.1), uma camada escondida com cinco neurônios e uma camada de saída com apenas um neurônio. Esse processo é repetido para cada bloco da imagem. Após a etapa de treinamento, a RNA retorna valores reais entre 0 e 1 como saídas. Esse valor real pode ser interpretado como o grau de certeza da classificação de um bloco específico. As diferenças principais entre as seis RNAs (Figura 4.3 (a)) são os conjuntos de atributos da imagem usados como entrada. Todos esses conjuntos de atributos (Tabela 4.1) são calculados durante o bloco de segmentação da imagem.

A escolha dos atributos foi baseada nos resultados de Shinzato e Wolf (2011), que apresentam avaliações usando diferentes atributos de imagem, a fim de obter os mais adequados para serem a entrada das seis RNAs (Tabela 4.1). O método de avaliação utilizado relaciona informações de erro médio e precisão com o grau de certeza que uma RNA possui sobre um determinado padrão. Baseado nos resultados obtidos, foram definidos três grupos (RNA1, RNA2 e RNA3) de atributos de imagem para compor o sistema de identificação de rua. Com base nesses três grupos de atributos escolhidos, foram compostas seis RNAs contendo duas instâncias para cada conjunto, como mostrado na Tabela 4.1. A combinação das seis RNAs tem apresentado bons resultados para a tarefa de identificação de áreas navegáveis em ambientes urbanos.

Tabela 4.1: Atributos de entrada das RNAs utilizadas para identificar áreas navegáveis e não navegáveis (Shinzato e Wolf, 2011). Componentes R, G, B são representados pelo vermelho (*red*), verde (*green*) e azul (*blue*). Componentes H, S, V são representados pelo matiz (*hue*), saturação (*saturation*) e brilho (*value*). Componentes Y, U, V são representados pelo brilho, tonalidade azul e tonalidade vermelha. Md = Média, Norm = Normalizado, Ent = Entropia, En = Energia e Var = Variância.

RNAs	Atributos de entrada das RNAs
RNA1	Md U, Md V, Md B Norm, Ent H, En G Norm e Md H
RNA2	Md V, Ent H, En G Norm, Md G, Md U, Md R, Md H, Md B Norm, Md G Norm e Ent Y
RNA3	Md U, Md B Norm, Md V, Var B, Md S, Md H, Md G Norm e Ent G Norm
RNA4	Md U, Md V, Md B Norm, Ent H, En G Norm e Md H
RNA5	Md V, Ent H, En G Norm, Md G, Md U, Md R, Md H, Md B Norm, Md G Norm e Ent Y
RNA6	Md U, Md B Norm, Md V, Var B, Md S, Md H, Md G Norm e Ent G Norm

Após a obtenção das seis saídas das RNAs referentes para cada bloco, o classificador calcula a média desses valores para compor um único valor de saída. Esses valores representam cada um dos blocos obtidos a partir da imagem original, formando a matriz do mapa de navegabilidade. A Figura 4.3 (c) apresenta o mapa de navegabilidade em escala de cinza, onde a cor preta representa as áreas não navegáveis, a branca representa as áreas navegáveis e a cinza representa os valores intermediários. É importante mencionar que a RNA é previamente treinada usando exemplos supervisionados de regiões navegáveis e não navegáveis, selecionadas pelo usuário uma única vez em uma imagem inicial. Depois, a RNA treinada é integrada no sistema de controle do veículo e usada como informação para o sistema de controle de navegação autônoma. O mapa de navegabilidade é usado na próxima etapa para a identificação de geometrias da rua.

4.1.2 Identificação de geometrias

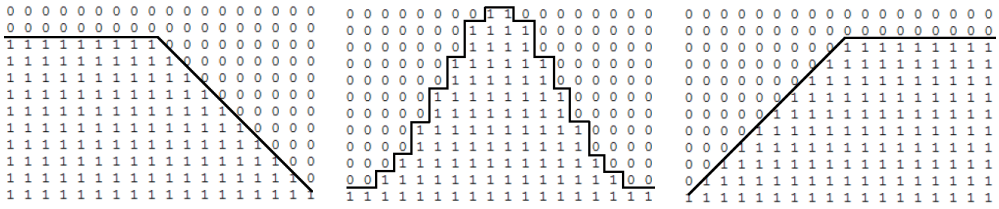
Após a obtenção do mapa de navegabilidade, cinco possíveis modelos diferentes de ruas são posicionados sobre a imagem, com o intuito de identificar a geometria da rua. Alguns exemplos

desses modelos são apresentados na Figura 4.5. Um deles identifica uma rua em linha reta, dois identificam curvas suaves esquerda e direita e dois curvas rígidas esquerda e direita.

(a) (b) (c) (d) (e)

Figura 4.5: Geometrias definem as direções do veículo. (a) Curva rígida à esquerda. (b) Curva suave à esquerda. (c) Frente. (d) Curva suave à direita. (e) Curva rígida à direita.

Cada modelo (*Template*) é composto por uma máscara de uns e zeros (Figura 4.6), como proposto em Souza *et al.* (2011a). O valor de cada máscara é multiplicado pelo valor correspondente da matriz do mapa de navegabilidade (Figura 4.3 (c)). A pontuação total ou nota (*score*) para cada modelo é a soma dos produtos dividido pela quantidade de uns existentes do modelo, resultando em um valor real entre 0 e 1.



(a) (b) (c)

Figura 4.6: Os formatos dos *templates* são: (a) Esquerda. (b) Centro. (c) Direita.

Um bom desempenho foi obtido em uma navegação autônoma por um ambiente urbano, onde o veículo CaRINA 1 desviou dos obstáculos (e.g. pessoas, cones e caixas) em sua frente e permaneceu na rua com êxito baseado nos modelos ou *templates* (Souza *et al.*, 2011b). As notas de todos os modelos são obtidas nesta etapa de identificação de geometrias e o percentual de ocupação é calculado para cada um deles. O valor do percentual de ocupação é obtido dividindo-se a nota do modelo pelo máximo valor da nota do modelo em questão.

Houve uma primeira tentativa de treinar a RNA com os valores do percentual de ocupação (valores reais entre 0 e 1) para cada um dos modelos, porém não apresentou uma capacidade de aprendizado satisfatória. Dessa forma, foi adotada uma abordagem baseada em um limiar (valor lógico). Após a obtenção do valor do percentual de ocupação para cada um dos modelos, o sistema verifica se o valor do percentual de ocupação a partir de cada modelo é menor que um limiar (*threshold*). Se o valor do percentual de ocupação do modelo em questão é menor que um limiar, a área de ocupação para esse determinado modelo está ocupada (área não navegável,

obtendo valor 0), e se é maior que um limiar, então a área de ocupação para esse determinado modelo está livre (área navegável, obtendo valor 1). Os valores livres ou ocupados são parte de entrada do sistema para a próxima etapa de navegação baseada em aprendizado.

4.1.3 Navegação por aprendizado

O controle de um sistema de navegação autônomo baseado em visão e GPS é realizado por uma RNA, que pode ser visualizada na Figura 4.7. Essa ilustração apresenta em detalhes uma estrutura básica de uma RNA MLP utilizando o algoritmo de treinamento RPROP (Seção 3.1).

Primeiramente, um humano pilotou o CaRINA 1 sobre um determinado cenário urbano, a fim obter as informações necessárias do veículo e posteriormente essas informações serem inseridas em uma RNA para que o veículo navegasse de forma autônoma.

As entradas da RNA são representadas pelo azimute que é definido pela diferença entre a orientação para o destino (coordenadas de GPS) e a orientação do veículo (obtida a partir de uma bússola), além disso, os valores das áreas de ocupação são obtidos de cada modelo (*template*), ou seja, áreas de ocupação (AO) livres ou ocupadas descritas na Seção 4.1.2. As saídas da RNA são representadas pelo ângulo de esterçamento e velocidade obtidos a partir de um motorista humano. A Figura 4.7 apresenta a estrutura da RNA utilizada, que foi aplicada para aprender os dados de entrada e saída e, posteriormente, controlar o veículo de forma autônoma.

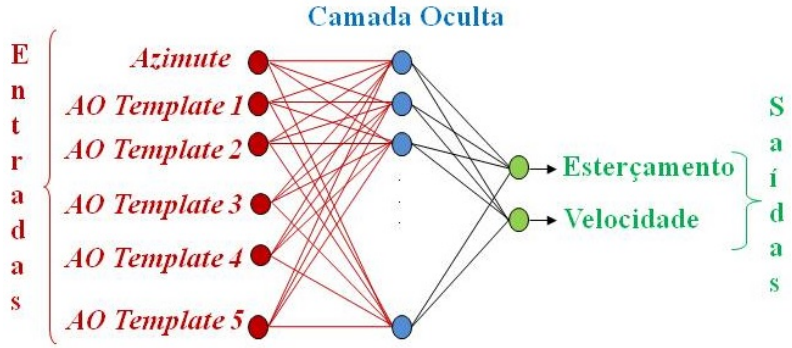


Figura 4.7: Estrutura da RNA utilizada para produzir os comandos de esterçamento e velocidade do CaRINA 1. Azimute representa a diferença entre a orientação para o destino (coordenadas de GPS) e a orientação do veículo (obtida a partir de uma bússola). A sigla AO significa a área de ocupação de cada modelo (*template*).

Foram avaliadas diferentes quantidades de neurônios e duas funções de ativação na camada oculta de uma RNA MLP (Figura 4.7). As funções de ativação utilizadas são a função sigmóide logística e a função tangente hiperbólica, pois são bastante difundidas e comumente utilizadas na comunidade científica de aprendizado de máquina. Os resultados obtidos dessa metodologia proposta serão descritos em detalhes na próxima seção.

4.2 Resultados

Os experimentos usaram o CaRINA 1 (Figura 4.1), veículo capaz de navegar autonomamente em uma via urbana. Ele é equipado com uma câmera monocular *VIDERE DSG* montada sobre o teto do carro, um controlador de motor *ROBOTEQ AX2580* para o esterçamento do volante, uma bússola *TNT Revolution GS* para orientação do veículo, um GPS *GARMIN 18X-5Hz* para localização do carro e um circuito baseado em Arduino para aceleração do veículo (Fernandes *et al.*, 2012). A resolução da aquisição da imagem foi configurada para 320×240 *pixels*. As RNAs foram aplicadas usando a biblioteca FANN (2014). Utilizou-se o OpenCV (Bradski e Kaehler, 2008) para o desenvolvimento da aquisição da imagem e do algoritmo *Template Matching*.

O sistema de identificação da rua converte uma imagem de 320×240 *pixels* em uma matriz de 32×24 blocos, onde K_b é representado pelo valor 10 (Seção 4.1.1). Cada bloco é individualmente classificado. Durante a etapa de aprendizado, a qual é realizada como uma etapa inicial do sistema proposto para navegação autônoma, o usuário seleciona manualmente partes de uma ou mais cenas de uma imagem como áreas navegáveis ou não navegáveis para fornecer os dados de treinamento para a RNA. Na etapa de navegação por aprendizado, o sistema utiliza a imagem classificada como entrada para a identificação de geometria, resultando em áreas de ocupação para cada modelo (*template*). Essas áreas de ocupação são as entradas da RNA e os controles desejados (esterçamento e velocidade) são as saídas da RNA (Seção 4.1.3).

As próximas subseções descrevem os experimentos para validação da metodologia. Os experimentos foram realizados em um ambiente urbano em dois cenários com características distintas. O cenário 1 apresenta uma área fechada com diversos objetos, tais como edifícios, árvores, paredes, placas de trânsito, bicicletas, carros e ônibus. O cenário 2 apresenta um ambiente mais aberto, com alguns objetos na cena, tais como árvores, grama, vegetação e rotatórias.

4.2.1 Cenário 1

Sete coordenadas de GPS foram usadas para definir a trajetória desejada, cujo trajeto apresenta aproximadamente 200 m. Para navegação autônoma, o veículo CaRINA 1 utilizou uma câmera monocular para se desviar dos obstáculos. O experimento foi realizado com sucesso, pois o CaRINA 1 seguiu os pontos de GPS e desviou dos obstáculos em sua frente. Em alguns trechos em linha reta, o veículo teve que evitar o meio-fio (calçada), ao mesmo tempo foi atraído pelas coordenadas de GPS, resultando em algumas oscilações na trajetória final sobre a via urbana. A Figura 4.8 apresenta a trajetória do veículo obtida pelas coordenadas de GPS, porém não apresenta o caminho exato das coordenadas dos experimentos, apenas o caminho aproximado.

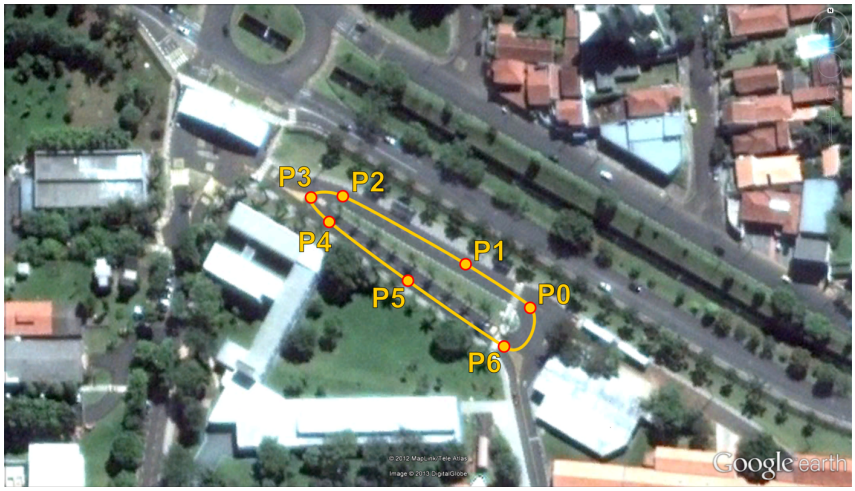


Figura 4.8: Trajeto realizado pelo CaRINA 1 no primeiro cenário externo (aproximadamente 200 m). P0 até P6 representam os pontos das coordenadas de GPS utilizados nesse ambiente. A linha amarela ilustra o caminho aproximado realizado pelo veículo.

A Tabela 4.2 apresenta os resultados da metodologia (Seção 4.1) durante a navegação do veículo CaRINA 1. Cinco diferentes topologias de RNAs foram analisadas usando duas funções de ativação (sigmóide logística e tangente hiperbólica) na camada oculta da RNA. Essas topologias representam a arquitetura da segunda RNA (Seção 4.1.3) usada em nosso sistema proposto.

As topologias de RNAs foram testadas com o intuito de obter o erro mínimo de validação e máxima capacidade de generalização, ou seja, Ponto Ótimo de Generalização (POG), além da obtenção da RNA considerando o menor Erro Médio Quadrático (EMQ) sobre os dados de teste. A topologia da RNA também foi selecionada considerando-se o número de neurônios na camada oculta ou escondida usando o algoritmo de treinamento RPROP em RNAs MLP e considerando os valores obtidos a partir do EMQ e o POG.

Foram avaliadas as RNAs alterando a utilização das funções de ativação na camada oculta da rede. A diferença entre essas funções é que a sigmóide logística produz valores positivos entre 0 e 1, e a função tangente hiperbólica, valores dentro de um intervalo entre -1 e 1.

Tabela 4.2: Resultados das RNAs para cada topologia. Diversas avaliações foram realizadas usando diferentes topologias e duas funções de ativação: Tangente Hiperbólica (TH) e Sigmóide Logística (SL). O erro da RNA é obtido através do EMQ (10^{-3}). Além disso, apresenta-se o resultado do Ponto Ótimo de Generalização (POG) de cada execução das RNAs. Em negrito, apresenta-se os melhores (menores) valores.

			Exec 1	Exec 2	Exec 3	Exec 4	Exec 5	Média	Desv. Pad.
	TH	EMQ	4.119	3.418	3.737	3.393	3.310	3.595	0.335
Topologia 1	TH	POG	1400	2300	1000	2000	2600	-	-
$6 \times 3 \times 2$	SL	EMQ	5.266	3.922	3.747	5.407	4.856	4.640	0.765
	SL	POG	600	3000	1400	700	1000	-	-
	TH	EMQ	3.271	3.413	2.148	5.404	3.217	3.491	1.182
Topologia 2	TH	POG	2200	700	1900	9700	26.400	-	-
$6 \times 6 \times 2$	SL	EMQ	2.490	3.257	3.250	3.670	3.414	3.216	0.440
	SL	POG	7600	1200	8300	1700	5600	-	-
	TH	EMQ	3.162	3.318	3.650	2.352	4.292	3.355	0.709
Topologia 3	TH	POG	1100	1100	800	800	400	-	-
$6 \times 9 \times 2$	SL	EMQ	3.774	5.013	4.584	5.050	3.190	4.322	0.815
	SL	POG	800	600	500	900	2300	-	-
	TH	EMQ	4.040	3.899	2.959	3.595	3.895	3.678	0.433
Topologia 4	TH	POG	400	1400	800	300	7400	-	-
$6 \times 12 \times 2$	SL	EMQ	2.836	3.797	3.746	2.048	5.645	3.614	1.345
	SL	POG	91.800	15.200	1000	23.600	1400	-	-
	TH	EMQ	3.232	1.999	**1.881**	3.510	2.989	**2.722**	0.739
Topologia 5	TH	POG	2600	76.100	10.600	2500	1000	-	-
$6 \times 15 \times 2$	SL	EMQ	4.226	4.597	4.000	5.319	3.188	4.266	0.783
	SL	POG	400	600	1300	400	700	-	-

As Equações 4.1, 4.2 e 4.3 apresentam respectivamente as medidas de desempenho (média, desvio padrão e EMQ) que foram usadas para avaliação de cada topologia da RNA (Tabela 4.2).

$$M = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{4.1}$$

onde M representa a média aritmética, e é calculada através da adição de um grupo de números (x_i, amostras, os valores do EMQ), depois disso, a divisão pela quantidade desses números (n, quantidade de elementos) avaliados.

$$D = \sqrt{\frac{\sum_{i=1}^{n} (x_i - M)^2}{n - 1}} \tag{4.2}$$

onde D representa o desvio padrão, considera que seus números são uma amostra da população. M é a média da amostra (os valores do EMQ), x_i representa as amostras (valores obtidos do EMQ) e n representa o tamanho da amostra (quantidade de elementos) avaliados.

$$EMQ = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.3}$$

onde EMQ representa o Erro Médio Quadrático, y_i é um vetor dos valores reais (verdadeiros, saídas esperadas), $\hat{y}_i$ é um vetor de predições (saídas obtidas através da RNA) e n representa a quantidade de elementos (padrões) avaliados.

A inferência estatística dos resultados foi avaliada utilizando-se o método (Shapiro e Wilk, 1965), que se baseia nas melhores topologias de RNAs da Tabela 4.2. Observa-se que a hipótese nula (adequação normal), não é satisfeita (ver Tabela 4.3) utilizando a melhor topologia de RNA (6x15x2) para os dados de teste.

Tabela 4.3: Resultados do teste Shapiro-Wilk.

	Teste de normalidade Shapiro-Wilk	
	p-valor	
	Esterçamento	Velocidade
Topologia 1	**0.2468**	3.84e-09
Topologia 2	5.43e-14	1.97e-15
Topologia 3	6.81e-14	2.68e-15
Topologia 4	6.69e-13	8.60e-15
Topologia 5	2.20e-16	1.59e-15

Valor em negrito - Aceito como normal
Outros valores - Não aceito como normal

O método não-paramétrico de Man-Whitney (Fay e Proschan, 2010) foi utilizado para verificar as diferenças entre as topologias (ver Tabela 4.4). Para a velocidade, o método não rejeita a hipótese nula (igualdade) (os p-valores são maiores que 0.05) e para o esterçamento, a hipótese nula é rejeitada apenas entre as topologias (T) 1 e 5 (p-valor é menor que 0.05). De acordo com esse método, e utilizando um nível de confiança de 95%, não existem evidências de diferenças estatísticas entre os melhores resultados, exceto para T1 e T5 em relação ao esterçamento.

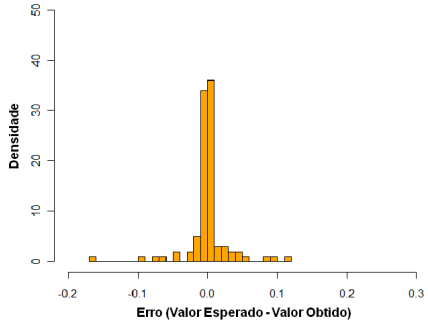
Tabela 4.4: Resultados do teste Man-Whitney.

Teste Man-Whitney					
		p-valor			
		T1	T2	T3	T4
Esterçamento	T5	0.031	0.115	0.397	0.114
Velocidade	T5	0.464	0.403	0.715	0.585

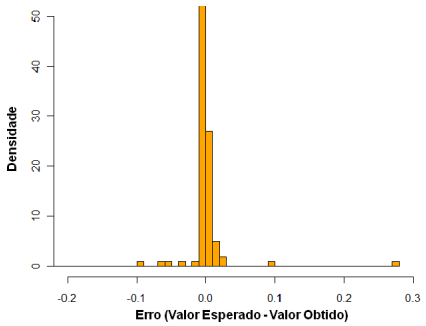
Utilizando o primeiro cenário (Figura 4.8), diversas avaliações foram realizadas, considerando-se cinco diferentes topologias de RNAs, sendo que cada topologia também foi analisada

alterando a função de ativação na camada oculta da RNA. Cinco diferentes execuções foram realizadas alterando a semente de inicialização de pesos da RNAs usando o algoritmo de treinamento RPROP. Os resultados dessa avaliação podem ser visualizados na Tabela 4.2. Após a análise, a topologia 5 da RNA ($6 \times 15 \times 2$), utilizando a função de ativação tangente hiperbólica, apresentou o resultado com o menor Erro Médio Quadrático (média de cinco execuções).

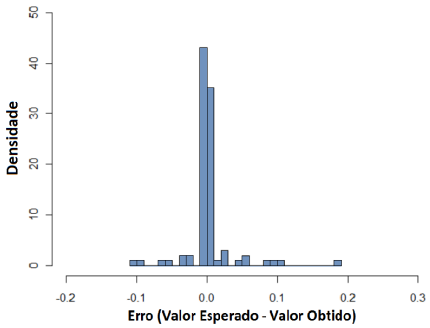
A Figura 4.9 apresenta os histogramas dos erros (valor esperado - valor obtido da RNA), considerando as duas melhores RNAs (Topologia 4 com execução 4 usando SL e Topologia 5 com execução 3 usando TH) obtidas da Tabela 4.2. A Figura 4.9 (b) apresenta o erro concentrado em zero e com uma menor dispersão comparado com a Figura 4.9 (a). A Figura 4.9 (d) também apresenta uma menor dispersão comparado com a Figura 4.9 (c). Assim, a Topologia 5 foi selecionada como a mais adequada atendendo aos requisitos do sistema de navegação.



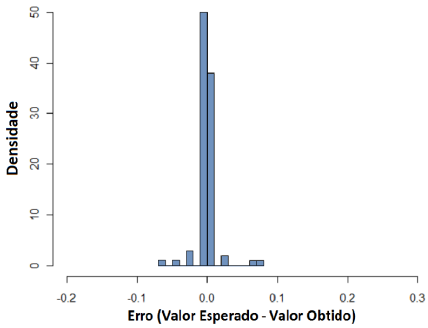
(a) Esterçamento (Top. 4, Exec. 4, SL)



(b) Esterçamento (Top. 5, Exec. 3, TH)



(c) Velocidade (Top. 4, Exec. 4, SL)



(d) Velocidade (Top. 5, Exec. 3, TH)

Figura 4.9: Histograma dos erros (valor esperado - valor obtido da RNA) considerando as melhores RNAs usando os dados de teste. Eixo x representa o erro. O eixo y representa a densidade baseada em esterçamento ou velocidade usadas no cenário 1.

A Figura 4.10 ilustra o ângulo de esterçamento do CaRINA 1 usando a topologia 6x15x2 para o cenário 1, apresentando os valores baseados na condução humana e os resultados obtidos pelo aprendizado da RNA. Oscilações pequenas estão presentes nos dados de aprendizado, pois

o humano manteve o esterçamento constante, resultando na linearidade dos dados (o problema para a RNA foi aprender uma curva rígida, porém esse aspecto não interferiu nos resultados).

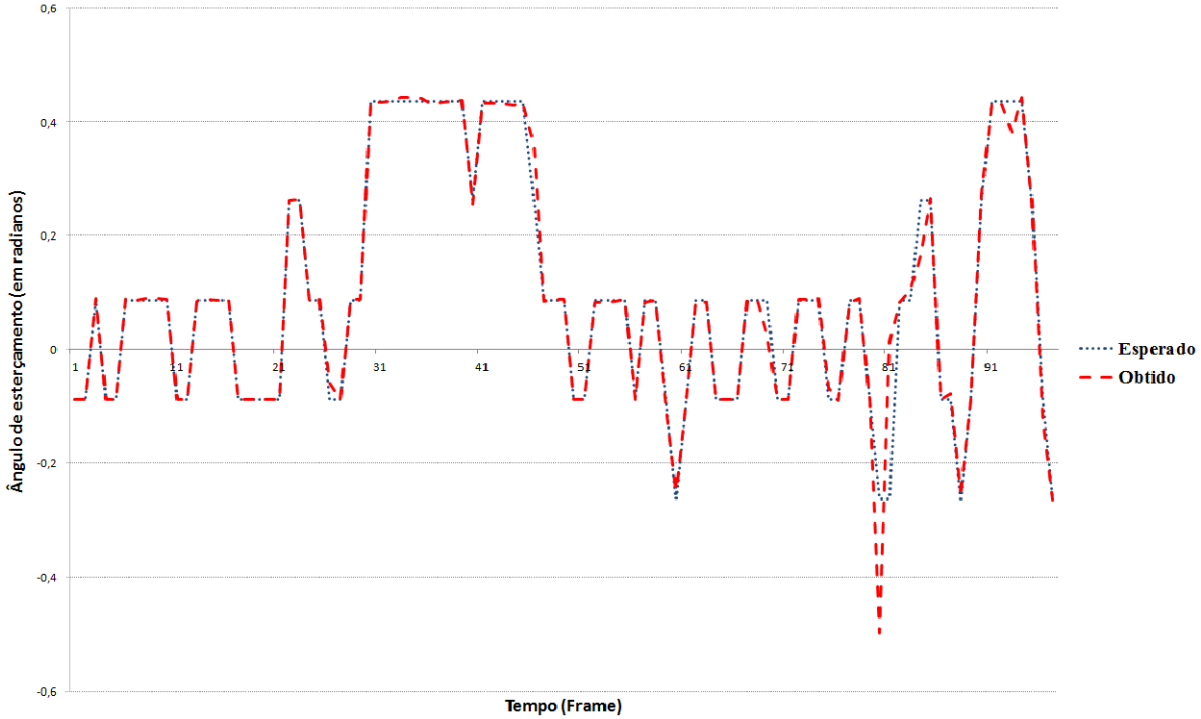


Figura 4.10: A condução humana (esperado) e os valores da RNA usando os dados de teste.

4.2.2 Cenário 2

Sete coordenadas de GPS foram utilizadas para definir a trajetória desejada, cujo trajeto apresenta aproximadamente 1,08 km (Figura 4.11). As coordenadas de GPS estavam mais de 100 metros de distância uma da outra e elas estavam separadas por calçadas (meio-fio) e áreas não navegáveis (e.g. campos de grama, árvores e vegetação).

A Figura 4.11 apresenta a trajetória do veículo obtida pelas coordenadas de GPS, porém não apresenta o caminho exato das coordenadas dos experimentos (aproximado). P0 até P6 representam os pontos das coordenadas de GPS usados nesse segundo cenário. As linhas vermelhas pontilhadas ilustram a orientação calculada pelo veículo para se deslocar de um ponto a outro, porém não é possível seguir exatamente essas linhas, visto que de um ponto para o outro existem calçadas, árvores e vegetação. A linha amarela ilustra o caminho aproximado realizado pelo veículo CaRINA 1, que permaneceu na rua usando imagens da câmera em todo o seu trajeto. Na Figura 4.11, apresentamos apenas as linhas de orientação entre os pontos P0 e P1 (linha vermelha) e entre P1 e P2 para manter a imagem original das coordenadas de GPS.

Para navegação autônoma, o veículo CaRINA 1 utilizou uma câmera monocular para se desviar dos obstáculos e permanecer na via urbana. O experimento foi realizado com êxito,

pois o CaRINA 1 seguiu os pontos de GPS e realizou as curvas e rotatórias propostas em sua frente (1,08 km autonomamente). Em alguns trechos em linha reta, o veículo teve que evitar a calçada e áreas com vegetação. Ao mesmo tempo, foi atraído pelas coordenadas de GPS, resultando em pequenas oscilações na trajetória final do cenário 2 sobre a via urbana.

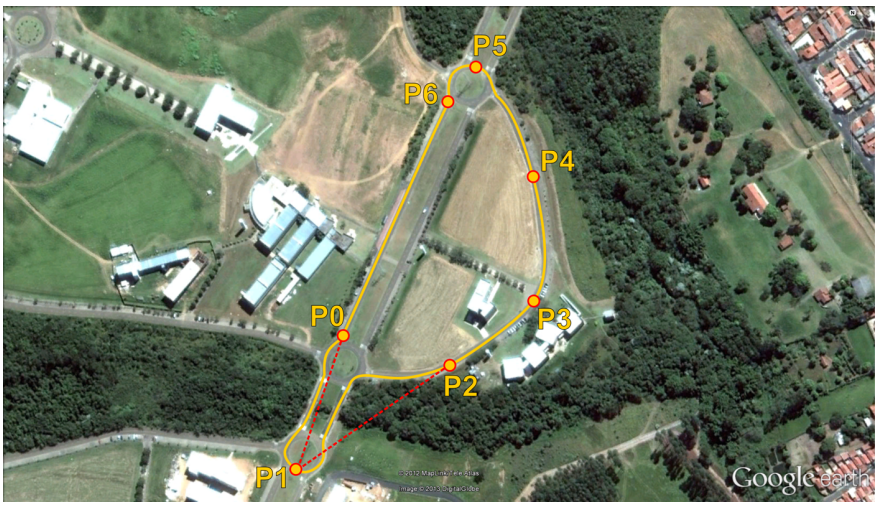
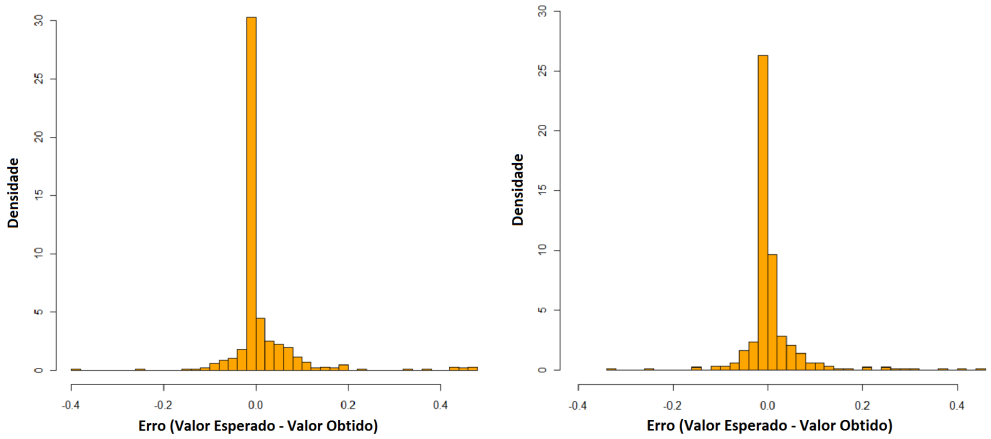


Figura 4.11: Trajeto realizado pelo CaRINA 1 (aproximadamente 1,08 km). P0 até P6 representam os pontos de GPS usados nesse segundo cenário. A linha amarela ilustra o caminho aproximado realizado pelo CaRINA 1. As linhas vermelhas ilustram a orientação calculada pelo veículo para se deslocar de um ponto a outro.

A Figura 4.12 apresenta o histograma dos erros (valor esperado - valor obtido da RNA) para o ângulo de esterçamento, considerando as duas melhores RNAs com topologias (Top. 4 com execução 4 usando SL e Top. 5 com execução 3 usando TH) obtidas a partir da Tabela 4.2. Ambas as imagens (Figuras 4.12 (a) e 4.12 (b)), apresentam alguns valores atípicos mas, em sua grande maioria, os erros estão concentrados próximos de zero com menor dispersão.



(a) Topologia 4 ($6 \times 12 \times 2$ - Exec. 4, SL) (b) Topologia 5 ($6 \times 15 \times 2$ - Exec. 3, TH)

Figura 4.12: Histograma dos erros (valor esperado - valor obtido da RNA) considerando as melhores RNAs das topologias 4 (execução 4 usando SL) e 5 (execução 3 usando TH) usando os dados de teste. O eixo x representa o erro usado para análise. O eixo y representa a densidade baseada nos dados de esterçamento no cenário 2.

A Tabela 4.5 apresenta uma análise sobre os resultados do erro (valor esperado - valor obtido da RNA) de esterçamento e velocidade usando como entrada os dados de teste para as melhores RNAs (Topologias 4 e 5). A Topologia 5 apresentou um menor erro considerando a média e o desvio padrão (medidas de desempenho) comparado com a Topologia 4. Dessa forma, a Topologia 5 será usada em nosso sistema proposto para a navegação autônoma do CaRINA 1.

Tabela 4.5: Resultados do erro (valor esperado - valor obtido da RNA) usando como entrada os dados de teste para as RNAs de Topologias 4 e 5 (Cenário 2).

Topologias	Esterçamento		Velocidade	
	Média	Desvio Padrão	Média	Desvio Padrão
Topologia 4 (exec. 4, SL)	0.014762	0.073536	0.002138	0.038567
Topologia 5 (exec. 3, TH)	**0.007629**	**0.055944**	**0.000528**	**0.035246**

A Figura 4.13 ilustra o ângulo de esterçamento do CaRINA 1, apresentando os valores esperados e os valores obtidos pelo aprendizado das RNAs 15 (topologia 5) e 12 (topologia 4). Os resultados são similares usando ambas as topologias para o sistema de navegação autônoma.

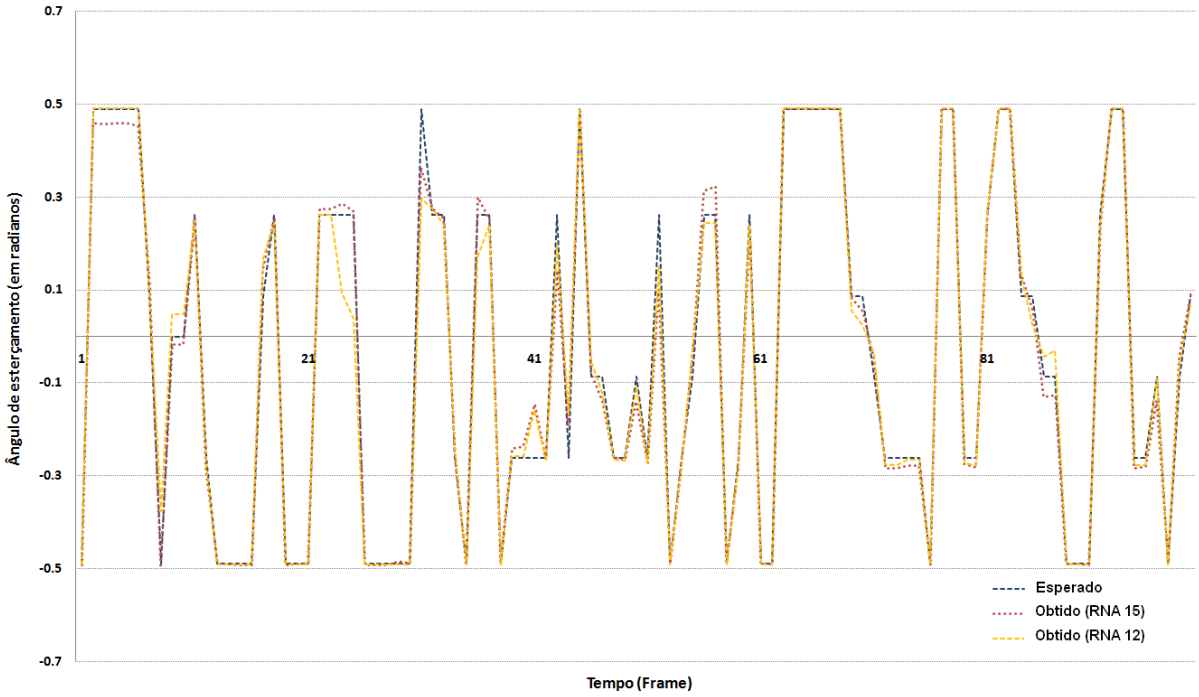


Figura 4.13: Ângulo de esterçamento usando os dados de teste.

Usando a RNA com a Topologia 5 em nosso sistema proposto, o CaRINA 1 foi capaz de navegar de forma autônoma em uma via urbana. O veículo se manteve em áreas navegáveis evitando obstáculos. Ele foi capaz de realizar a navegação de maneira segura e eficiente. Cabe ressaltar que a mesma topologia obteve os melhores resultados em ambos os cenários avaliados. Isso demonstra a robustez do sistema de navegação desenvolvido em diferentes situações.

4.3 Considerações Finais

O Capítulo 4 apresenta o desenvolvimento e os resultados da navegação baseada no condutor humano em vias urbanas. A navegação autônoma é uma tarefa importante em robôs móveis, visto que a comunidade científica tem apresentado projetos avançados e vem discutindo esse assunto em conferências internacionais de alto nível para difundir essa área do conhecimento.

A RNA permitiu que o CaRINA 1 navegasse de forma autônoma em vias urbanas desviando de obstáculos e evitando calçadas. Além disso, o veículo foi capaz de seguir as coordenadas de GPS que foram fornecidas pelo sistema. Contudo, o sistema apresenta restrições. A identificação da rua (Shinzato e Wolf, 2011) combinado com a identificação de geometrias (Souza *et al.*, 2011b) permite o veículo mudar de faixa sem que ele esteja disposto a fazê-lo. A identificação de geometrias direciona o veículo para qualquer área livre navegável, com isso em uma pista dupla, o CaRINA 1 pode oscilar. Esse problema é um dos trabalhos futuros da pesquisa.

A metodologia desenvolvida apresenta a capacidade de localização (posição atual e destino do veículo baseada em coordenadas de GPS), orientação (obtida a partir de uma bússola) e percepção (identificação de modelos em áreas navegáveis e não navegáveis) baseado na integração de uma câmera monocular e GPS. A estrutura da RNA provê o desvio de obstáculos de forma reativa (CaRINA 1 observa usando a câmera e age) sobre as ruas pavimentadas. Além disso, apresenta uma deliberação (CaRINA 1 observa, planeja para onde deve se deslocar e age), pois a estrutura da RNA é direcionada por pontos de GPS. Esta estrutura pode ser usada em outros mapas se estes fornecerem os pontos de GPS. O erro grande de GPS (3 m), mudança de tempo e a posição da câmera são aspectos considerados como investigação em trabalhos futuros.

CAPÍTULO

5

Navegação Autônoma Baseada na Irregularidade do Terreno

Este capítulo apresenta a metodologia, os resultados alcançados e as considerações finais do segundo problema abordado nesta tese: a navegação autônoma baseada na irregularidade do terreno em um ambiente externo. Os algoritmos que foram usados são: *Wavelet*, Filtro Passa-Baixa, Mapeamento usando SLAM, Localização Monte Carlo, Processos Gaussianos e a Otimização Bayesiana. Para avaliação dos algoritmos, foram utilizadas duas medidas de desempenho: a Raiz do Erro Médio Quadrático e o tempo computacional em segundos.

5.1 Metodologia

O objetivo é reduzir a quantidade de vibração experimentada pelo robô Husky (Figura 5.1) durante a navegação em diferentes terrenos. Para conseguir isso, foi proposto um método (Souza *et al.*, 2014) para aprender uma representação da irregularidade do terreno, favorecendo, assim, o movimento através de áreas navegáveis que produzam níveis de vibração mais baixos.

Figura 5.1: Plataforma de teste Husky - *Clearpath Robotics* (Husky, 2013).

A metodologia desenvolvida, apresentada na Figura 5.2 consiste de quatro etapas:

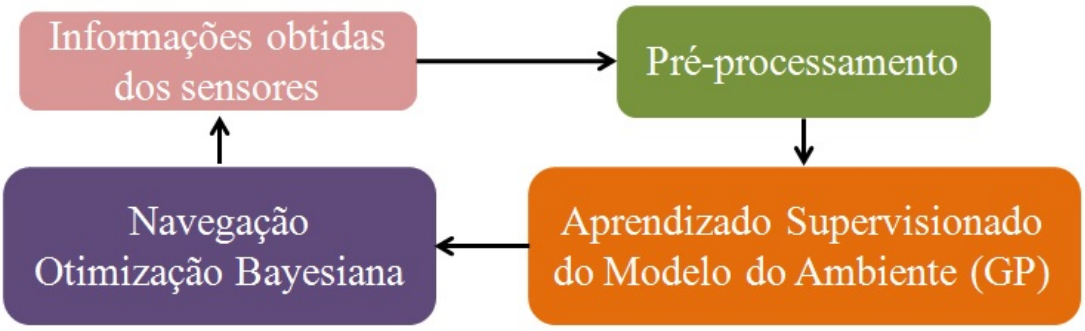


Figura 5.2: Diagrama de bloco do método proposto.

- *Pré-processamento*: Os filtros são aplicados às medidas ruidosas providas por uma Unidade de Medida Inercial para suavizar os dados como descrito na Seção 5.1.1.
- *Modelagem de Vibração*: A regressão de Processo Gaussiano é usada para criar um modelo estatístico de vibração quando o robô atravessa um terreno e para predizer os níveis de vibração através de áreas não-amostradas. Os detalhes são apresentados na Seção 5.1.2.
- *Planejamento*: A Otimização Bayesiana (ver Seção 5.1.3) é aplicada para selecionar locais (posições) a serem visitados, dependendo do nível esperado de vibração e a incerteza na predição. Isso está descrito na Seção 5.1.4.

- *Localização e Mapeamento*: A localização é usada pela etapa de Modelagem para atribuir medições de vibração para diferentes partes do terreno. Esta etapa localiza o robô em um ambiente desconhecido e cria um mapa da área usando módulos disponíveis no ROS [1].

5.1.1 Pré-processamento

Para prevenir uma quantidade excessiva de ruído sendo transferido para a etapa de Modelagem de Vibração, foi proposto um pré-processamento etapa de filtragem. Dado o sinal original $w(t)$, foi avaliado o desempenho de duas estratégias diferentes para a filtragem de ruídos indesejados, produzindo um sinal filtrado $w_f(t)$.

a *Filtro Wavelet*: A transformada de *wavelet* discreta (Burrus *et al.*, 1998) fornece uma versão de multi-resolução contínua do sinal original. Manteve-se apenas o quarto nível dos coeficientes da transformada discreta de *Daubechies*, na qual foi obtida uma aproximação filtrada do sinal original em que o ruído de alta frequência não está mais presente.

b *Filtro Passa-Baixa*: O filtro passa-baixa simples não-recursivo (Siqueira e Diniz, 2005) foi desenvolvido para suavizar o sinal original. A saída do sinal é dado pela seguinte expressão:

$$w_f(t) = \sum_{j=0}^{N} \frac{w(t-j)}{N} \ . \tag{5.1}$$

Dependendo da largura de banda (diretamente relacionada com N) do filtro passa-baixa, um atraso é observado no sinal de saída. Definindo $N = 100$, foi fixado uma largura de banda de filtragem que remove o ruído e mantém um limite superior sobre o atraso.

A Figura 5.3 apresenta ambos os filtros aplicados para os dados brutos (reais) adquiridos com um acelerômetro. A partir desses gráficos, é difícil decidir o que produz bons resultados. Dessa forma, essa discussão será retomada na seção de resultados (Seção 5.2). Além disso, foi avaliado o desempenho dos filtros no contexto de todo o sistema proposto para selecionar a estratégia de filtragem que proporciona os melhores resultados.

[1] Robot Operating System http://www.ros.org

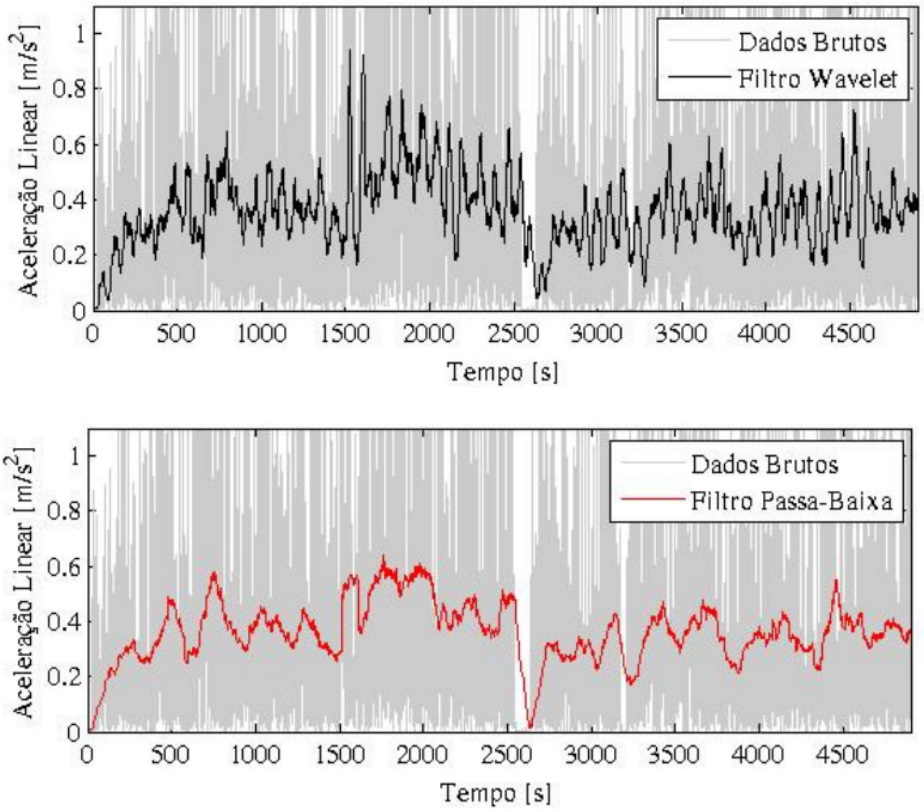


Figura 5.3: Aplicação de dois filtros em dados brutos da IMU.

5.1.2 Processo Gaussiano para Modelagem de Vibração

Um Processo Gaussiano é definido por uma função média $m(\mathbf{x})$ e uma função de covariância $k(\mathbf{x}, \mathbf{x}')$. No Processo Gaussiano treina-se uma função desconhecida f em um aprendizado supervisionado configurado usando um conjunto S de observações conhecidas, $S = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, onde $\mathbf{x}_i \in \mathbb{R}^D$ sendo x as N entradas em um espaço de dimensão D e $y_i \in \mathbb{R}$ são as saídas ruidosas correspondentes. Dado um ponto de teste $\mathbf{x}^*$ (posição não-amostrado), o Processo Gaussiano retorna uma predição do valor de $f(\mathbf{x}^*)$ com a incerteza correspondente. As observações ruidosas da função subjacente $f(\mathbf{x})$ são modeladas como uma distribuição Gaussiana, sendo que $y = f(\mathbf{x}) + \epsilon$, onde $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Dada uma função de covariância $k(\mathbf{x}, \mathbf{x}')$, a matriz de covariância K pode ser calculada avaliando-se a função de covariância para todas as observações. A distribuição preditiva para as posições de teste x^* é dada por

$$y^*|y \sim \mathcal{N}(\mu^*, \Sigma^*) \tag{5.2}$$

Onde,

$$\mu^* = K_*^T[K + \sigma_n^2 I]^{-1} y \tag{5.3}$$

$$\Sigma^* = K_{**} - K_*^T[K + \sigma_n^2 I]^{-1} K_* \tag{5.4}$$

Selecionando funções de covariância (Capítulo 3), o GP para Modelagem de Vibração pode adaptar-se facilmente para modelos de vibração complexos. O fator σ_f, ou variância do sinal, l_D (comprimento de escala para cada dimensão), σ_n (ruído) são os hiper-parâmetros das funções de covariância. Portanto, o conjunto de hiper-parâmetros que determina o GP é dado por:

$$\boldsymbol{\theta} = \{\sigma_f, l_1, l_2, \ldots, l_D, \sigma_n\} \tag{5.5}$$

O comportamento da regressão do Processo Gaussiano é adaptado para se conseguir uma boa representação da função desconhecida. O conjunto ótimo de hiper-parâmetros $\boldsymbol{\theta}^*$ pode ser encontrado maximizando o *Log Marginal Likelihood* (LML),

$$\boldsymbol{\theta^*} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ \ \mathrm{LML}(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \,, \tag{5.6}$$

com,

$$\mathrm{LML}(\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_X^{-1}\mathbf{y} - \frac{1}{2}\log|K_X| - \frac{n}{2}\log 2\pi \,. \tag{5.7}$$

Foram combinadas as funções de covariância para se obterem novas funções. A soma e o produto de duas funções de covariância são funções de covariância válidas (Brahim-Belhouari e Bermak, 2004; Schöolkopf e Smola, 2002):

$$k_{\mathrm{SUM}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{M} k_i(\mathbf{x}, \mathbf{x}') \tag{5.8}$$

$$k_{\mathrm{PROD}}(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{M} k_i(\mathbf{x}, \mathbf{x}') \tag{5.9}$$

Dependendo do número de funções de covariância M, a dimensionalidade do vetor de hiper-parâmetro pode crescer rapidamente, tornando a otimização neste espaço dimensional mais custosa computacionalmente.

5.1.3 Otimização Bayesiana

O GP foi usado priori como modelo estatístico da função desconhecida para o desenvolvimento do algoritmo BO. Esse modelo usa amostras adquiridas $\mathbf{x}_i$ para construir um modelo da função

desconhecida e realiza a regressão sobre o espaço não-amostrado. As Equações 5.3 e 5.4 apresentam a média preditiva μ e a variância Σ do algoritmo BO (Brochu *et al.*, 2010) (Seção 3.3).

O segundo elemento fundamental no algoritmo BO para a otimização é a função de aquisição, a qual estabelece a próxima posição $\mathbf{x}$ para a amostra. A combinação desses dois elementos (um GP priori e a função de aquisição) proporciona ao sistema desenvolvido lidar de forma inteligente com a exploração e explotação (refinamento) sobre o ambiente usando um robô móvel.

Em cada iteração, uma nova posição da amostra é selecionada pelo conhecimento das informações previamente adquiridas. Ao maximizar a função de aquisição s, foi selecionada a melhor posição para testar a função desconhecida da irregularidade do terreno.

5.1.4 Otimização Bayesiana para Planejamento Informativo

A escolha de um modelo adequado (e.g. GP) para representar o fenômeno estudado é necessária, mas não suficiente para alcançar bons resultados. As observações desempenham um papel fundamental na qualidade do modelo resultante, e a sua precisão da predição. A abordagem comumente utilizada para escolher posições baseada no sensoriamento ocorre através da maximização da informação mútua (considera apenas a variância) entre o domínio amostrado e não-amostrado (Singh *et al.*, 2007). No entanto, Marchant e Ramos (2012) têm demonstrado que a Otimização Bayesiana pode ser usada como uma estratégia adequada, pois não só considera a incerteza no modelo, mas também faz uso do valor predito para escolher posições da amostra, automaticamente lidando com a exploração e a explotação. Usa-se essa estratégia para aprender a irregularidade do terreno, enquanto, ao mesmo tempo, considera-se essa irregularidade para selecionar pontos de interesse (*waypoints*) para a navegação em áreas com menos vibração.

Foi usado um procedimento de amostra ativa baseado na Otimização bayesiana (ver Seção 3.3). Ao maximizar a função de aquisição s, foi selecionada a melhor posição para testar a função desconhecida. Para a estimação da irregularidade do terreno, foi usada a distância do limite superior de confiança ($DUCB$, do Inglês, *Distance Upper Confidence Bound*), dado por

$$DUCB(\mathbf{x}|\mathbf{x}^{-}) = \mu(\mathbf{x}) + \kappa \cdot \sigma(\mathbf{x}) + \gamma \cdot d(\mathbf{x}, \mathbf{x}^{-}), \tag{5.10}$$

onde $d(\mathbf{x}, \mathbf{x}^{-})$ é a distância Euclidiana entre a última posição amostrada $\mathbf{x}^{-}$ e a posição candidata $\mathbf{x}$ (obtida ao maximizar a função de aquisição s), e $\gamma < 0$ assegura que as posições alvo são próximas umas das outras, com o objetivo de reduzir o custo de atravessar o ambiente.

Os valores preditos $\mu(\mathbf{x})$ dado pelo modelo GP são grandes para os níveis de vibração maiores (elevados), e ao minimizar f (Equação 3.30), reduziu-se a vibração total experimentada

pelo robô ao atravessar o ambiente. Ao maximizar a Equação 5.10, notou-se que o primeiro termo ($\mu(\mathbf{x})$) favorece áreas de menor vibração e o segundo termo ($\sigma(\mathbf{x})$) favorece áreas com maior incerteza (variância). Em resumo, o robô tenderá a favorecer rotas com menor vibração enquanto visita áreas que reduzem a incerteza em suas predições.

Inicialmente, a incerteza será alta sobre todo o domínio, ou seja, o segundo termo da Equação 5.10 ($\sigma(\mathbf{x})$) prevalecerá, favorecendo a exploração do terreno desconhecido. Como o robô percebe (detecta) as posições desconhecidas, a incerteza diminui. Depois de algum tempo, os valores preditos do fenômeno (primeiro termo da Equação 5.10, ($\mu(\mathbf{x})$) tornam-se mais importantes para planejar as posições de sensoriamento, ou seja, favorecendo a explotação. A suavidade da transição entre a exploração e a explotação depende da suavidade da função de covariância e o parâmetro κ utilizado.

5.2 Resultados

Para avaliar a capacidade e o desempenho do método proposto, foi desenvolvido o sistema descrito na Seção 5.1 e testado seu desempenho em um robô com rodas, conhecido como Husky da *Clearpath Robotics* (ver Figura 5.1). O robô móvel está equipado com um sensor laser SICK LMS 200, um sensor YEI 3-Space IMU e a odometria. Ele foi aplicado ao ar livre em uma área com diferentes características no terreno. O sensor laser SICK LMS 200 em conjunto com a odometria foi utilizado para realizar o mapeamento do ambiente e também para localizar o robô sobre o ambiente. Além disso, o sensor YEI 3-Space IMU foi utilizado para obter dados de aceleração linear do eixo vertical do robô, a fim de que ele se mantenha em lugares com menor vibração possível, enquanto ao mesmo tempo atravessa o ambiente.

Assume-se que a vibração é independente da orientação do veículo durante a navegação, e a velocidade foi mantida constante ao longo de todo o percurso. Esses fatores simplificam significativamente o modelo, assim sendo, pode-se aprender uma função de mapeamento a partir de uma posição para vibração diretamente, a um custo pequeno de precisão na área de teste - o GP aumenta automaticamente o nível de ruído σ_n^2 para acomodar essas simplificações. O aprendizado de modelos mais complicados de posição, velocidade e orientação à vibração é possível, mas requer trajetórias de exploração mais longas, pois o problema torna-se multidimensional. Esta questão será abordada em trabalhos futuros para terrenos mais irregulares.

O método proposto (ver Seção 5.1) é executado em tempo real sobre um laptop padrão e é implementado na linguagem de programação C++. Foi utilizado ROS para realizar a comunicação entre o robô e o método, bem como para fornecer a localização do robô.

5.2.1 Modelo

No primeiro experimento, o foco foi encontrar um modelo adequado que pudesse representar o ambiente corretamente. Para esse fim, precisa-se escolher uma estratégia de filtragem para o pré-processamento dos dados e uma função de covariância que produzisse bons resultados de predição. Foi comparado o desempenho preditivo do modelo, combinando um filtro passa-baixa ou um filtro *wavelet* com as funções de covariância detalhadas na Seção 5.1.2.

O robô foi conduzido de forma aleatória sobre o ambiente externo mostrado na Figura 5.5(a), apenas com o propósito de coleta de dados. Os dados de aceleração linear ao longo do eixo vertical do robô Husky foram coletados a uma frequência de 10Hz usando-se o sensor IMU e cada medição foi espacialmente referenciada utilizando-se a localização do robô, a qual se baseia no sensor laser e no mapa do ambiente. Esse conjunto de dados que consiste em 1000 medições, foi dividida em 70% para treinamento e 30% para fins de teste. Para cada combinação de filtros e funções de covariância, foram executados os experimentos 10 vezes, comparando-se o seu desempenho por meio de dois indicadores com a respectiva média e desvio padrão: o tempo que leva para aprender a representação em segundos e RMSE sobre o conjunto de teste definido em m/s^2. Esses resultados são apresentados na Tabela 5.1 e também pela Figura 5.4.

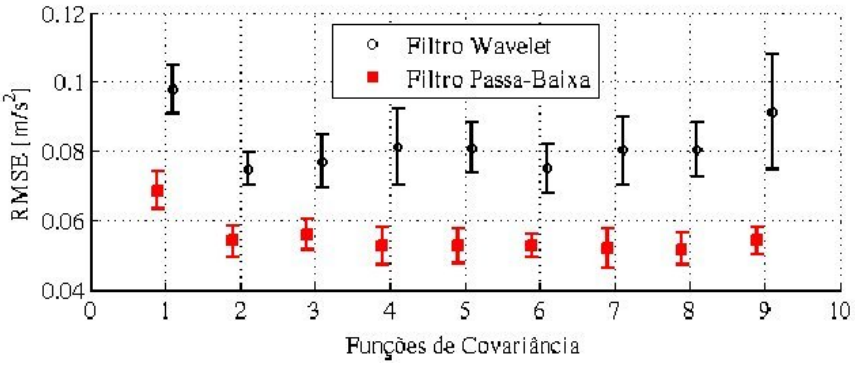
A Equação 5.11 apresenta uma das medidas de desempenho (RMSE) que foi considerada na avaliação de cada filtro de pré-processamento para cada função de covariância (Tabela 5.1).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{5.11}$$

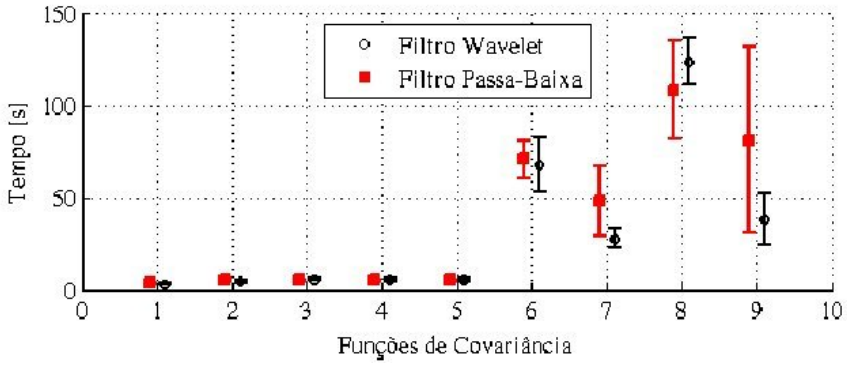
$RMSE$ representa a raiz quadrada do erro médio quadrático, y_i é um vetor dos valores reais (saídas esperadas), $\hat{y_i}$ é um vetor de predições (saídas obtidas através do Processo Gaussiano) e n representa a quantidade de elementos (padrões) observados.

Tabela 5.1: Resultados dos filtros de pré-processamento e função de covariância. RMSE obtidos a partir de 30% dos dados de teste, e tempo de treinamento para maximizar LML com 700 observações, onde os valores do RMSE estão dispostos em $(10^{-2}[m/s^2])$ e o Tempo está descrito em (s).

		Filtro Passa-Baixa		Filtro Wavelet	
Índice	**Função de Covariância**	**RMSE**	**Tempo**	**RMSE**	**Tempo**
1	Linear (LI)	6.8836 ± 0.5393	3.2836 ± 0.3272	9.7869 ± 0.7096	3.1382 ± 0.4614
2	Exponencial (EP)	5.4013 ± 0.4583	5.1260 ± 0.7097	7.4870 ± 0.4479	4.3600 ± 0.7276
3	EP Quadrático (SE)	5.5994 ± 0.4510	5.1191 ± 0.5014	7.7127 ± 0.7563	5.6706 ± 0.5639
4	Matérn 3 (M3)	5.2796 ± 0.5485	5.0507 ± 0.5947	8.1409 ± 1.0812	5.3240 ± 0.6323
5	**Matérn 5 (M5)**	$\mathbf{5.2741 \pm 0.4984}$	$\mathbf{4.9580 \pm 0.6434}$	8.0989 ± 0.7176	5.5355 ± 0.6924
6	Som (SE, M3, M5)	5.2848 ± 0.3319	70.806 ± 10.283	7.5104 ± 0.7193	67.500 ± 14.940
7	Prod (SE, M3, M5)	5.2042 ± 0.5799	47.945 ± 19.168	8.0247 ± 0.9952	27.813 ± 5.0636
8	Som (LI, EP, SE, M3, M5)	5.1685 ± 0.4569	107.84 ± 26.586	8.0398 ± 0.7777	123.64 ± 12.516
9	Prod (LI, EP, SE, M3, M5)	5.4207 ± 0.3824	80.988 ± 50.401	9.1381 ± 1.6532	37.976 ± 14.008



(a) RMSE



(b) Tempo

Figura 5.4: Resultados dos filtros de pré-processamento e função de covariância. RMSE obtidos a partir de 30% dos dados de teste, e tempo de treinamento para maximizar LML com 700 observações. Os índices das funções de covariância são mostrados na Tabela 5.1.

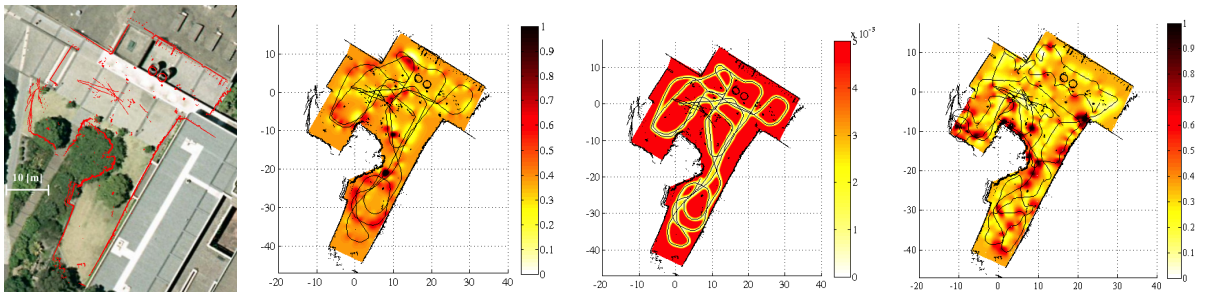
O filtro passa-baixa apresenta o erro menor comparado com o filtro *wavelet* para cada função de covariância. Há evidências claras de que somas ou produtos de funções de covariância apre-

sentam um tempo longo para treinar do que funções individuais, o que é um resultado esperado dado o aumento da dimensionalidade do espaço de busca do hiper-parâmetro. A função Linear apresenta o tempo de treinamento mais curto, no entanto, o erro é maior do que as demais funções de covariância para ser aceitável. Considerando as funções de covariância relativamente curtas, Matérn 5 é a que produz o menor erro, sem comprometer o tempo de treinamento.

O melhor conjunto de hiper-parâmetros para a função de covariância Matérn 5, encontrado através da maximização da Equação 5.7, foi:

$$\{\sigma_f, l_{posx}, l_{posy}, \sigma_n\} = \{0.104, 1.016, 0.990, 0.028\} \tag{5.12}$$

A Figura 5.5(b) apresenta a média e a Figura 5.5(c) a variância do modelo GP aprendido utilizando-se um filtro passa-baixa e a função de covariância Matérn 5 com os hiper-parâmetros ótimos representados pela Equação 5.12. A média da estimativa de vibração apresenta uma distinção clara entre os dois terrenos explorados: grama e asfalto.



(a) Mapa do cenário (b) Média (c) Variância (d) Média preditiva após a navegação autônoma

Figura 5.5: (a) O cenário utilizado para o primeiro experimento delimitado pelo mapa produzido pelo robô. (b) Média da vibração preditiva. (c) A variância da predição. (d) Média preditiva após a navegação autônoma. A distância está em metros ($[m]$) e a predição está em $[m/s^2]$.

5.2.2 Planejamento de Trajetória

O segundo experimento consiste em, de forma autônoma, selecionar posições da amostra utilizando o algoritmo BO descrito na Seção 5.1.4. A função de covariância selecionada (Matérn 5) na Seção 5.2.1 e os hiper-parâmetros ótimos foram determinados a partir da Equação 5.12. Foi utilizado um filtro passa-baixa para remover o ruído de alta frequência a partir da IMU, devido à sua simplicidade e aos bons resultados obtidos na seção anterior. Os parâmetros da função de aquisição DUCB, $\{\kappa, \gamma\}$, são definidos por $\{1.25, 0.0125\}$. Estes foram manualmente ajustados, para resultar na quantidade certa de exploração (κ) e escolher posições de sensoriamento

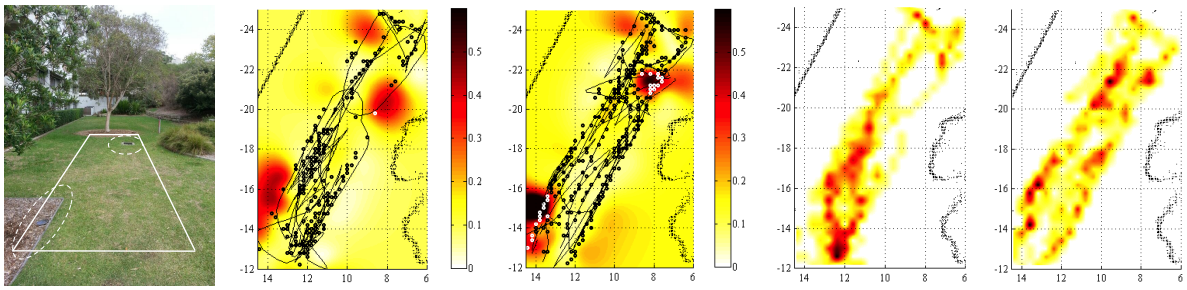
consecutivos suficientemente próximos entre si (γ). Entretanto, uma estratégia como a utilizada por Snoek *et al.* (2012) poderia ser usada para otimizar esses parâmetros automaticamente.

Em primeiro lugar, avaliou-se o comportamento exploratório do método proposto. O robô foi programado para selecionar de forma autônoma as posições da amostra usando o algoritmo BO sobre um ambiente externo apresentado na Figura 5.5(a), com aproximadamente $1000m^2$. A trajetória seguida pelo robô enquanto as posições objetivo são selecionadas, além do valor estimado da vibração do domínio são apresentadas na Figura 5.5(d). O robô cobriu toda a área em cerca de 17 minutos. Ao longo do tempo, o robô explorou áreas de baixa vibração mais vezes do que áreas com alta vibração. Como resultado, áreas de baixa vibração foram modeladas com maior precisão. Isto não é um problema, já que o objetivo não é aprender o melhor modelo de vibração ao longo de toda a região, mas minimizar a vibração durante a navegação. Os valores preditos para vibração estão de acordo com os resultados esperados, visto que as posições dos obstáculos são identificadas como picos (alta vibração) na Figura 5.5(d) e a área de asfalto apresenta um nível de vibração mais baixo em média.

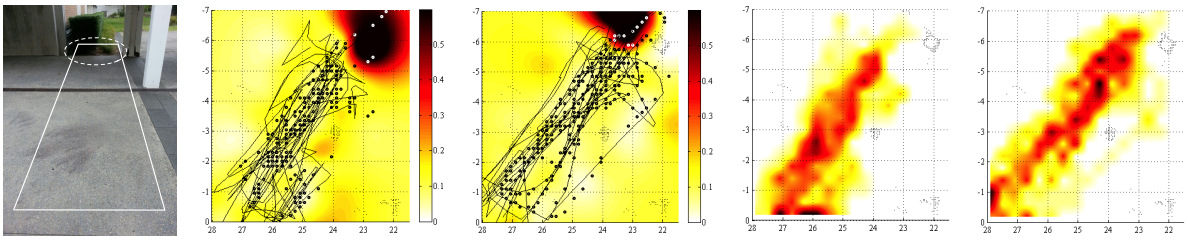
Finalmente, compara-se o método de amostragem com um planejador baseado em entropia (Singh *et al.* (2010)), no qual as posições da amostra são selecionadas com base somente na redução da entropia (considera somente a variância e não a média preditiva, ver Equação 5.10). A comparação é realizada em dois cenários pequenos, nos quais o robô é forçado a permanecer em uma área restrita, como mostrado nas Figuras 5.6(a) e 5.6(f). Esses cenários contêm obstáculos que produzem mudanças acentuadas (bruscas) nos níveis de vibração. A Figura 5.6 apresenta o caminho realizado pelo robô, as posições da amostra selecionadas e a predição da vibração sobre o domínio para cada cenário.

O modelo produzido para cada uma das estratégias de planejamento apresenta corretamente as áreas de alta vibração. No entanto, o resultado mais interessante e esperado é que o algoritmo BO evitou áreas de alta vibração após identificá-las durante o comportamento exploratório, diferente do planejador baseado em entropia, que escolhe posições de amostra apenas com base na incerteza (variância), sem considerar a quantidade de vibração esperada. Mapas de calor baseados em posição foram produzidos para o caminho realizado pelo robô. O robô usando o algoritmo BO cobriu toda a área, porém concentrou-se em posições com baixa vibração (Figuras 5.6(d) e 5.6(i)), enquanto a abordagem baseada em entropia (Figuras 5.6(e) e 5.6(j)) foi espalhada sobre todo o domínio de amostragem de forma homogênea.

A Tabela 5.2 apresenta uma comparação entre a aceleração total experimentada pelo robô em cada cenário. O planejador usando o algoritmo BO resulta em menos vibração encontrada pelo robô, em termos de média e desvio padrão, e consegue um modelo preciso do ambiente.



(a) Cena Grama (b) Mapa de vibração e caminho usando BO (c) Mapa de vibração e caminho com Entropia (d) Mapa de calor da posição com BO (e) Mapa de calor da posição Entropia



(f) Cena Asfalto (g) Mapa de vibração e caminho usando BO (h) Mapa de vibração e caminho com Entropia (i) Mapa de calor da posição usando BO (j) Mapa de calor da posição Entropia

Figura 5.6: Comparação entre as estratégias BO e Entropia. (a) e (f) mostram as áreas do experimento com um retângulo e os obstáculos que produzem vibração são marcados com círculos. (b), (c), (g) e (h) mostram uma superfície para a predição da aceleração vertical em m/s^2 e o eixo em metros. (d), (e), (i) e (j) mostram um mapa de calor do tempo gasto do robô em cada posição durante os experimentos.

Tabela 5.2: Aceleração média em $[m/s^2]$ para os planejadores BO e Entropia.

Planejadores	Grama		Asfalto	
	Média	Desvio Padrão	Média	Desvio Padrão
Entropia	0.154	0.148	0.127	0.256
BO	**0.096**	**0.068**	**0.103**	**0.122**

5.3 Considerações Finais

O Capítulo 5 aborda o desenvolvimento e os resultados da navegação baseada na irregularidade do terreno em um ambiente externo. A navegação autônoma é um problema relevante na robótica móvel e a comunidade científica tem consolidado esta área do conhecimento.

Foi proposto um método de aprendizado simples, mas eficaz para aprender as características do terreno em ambientes ao ar livre. Particularmente, foi modelada a irregularidade do terreno usando um GP treinado com medições de aceleração sobre o eixo vertical do robô, a fim de manter o robô o mais seguro possível. Dessa forma, foi usada uma abordagem BO para a escolha das posições da amostra, de forma automática, lidando com a exploração e a explotação

quando necessário. Uma representação da irregularidade do terreno foi aprendida escolhendo-se uma função de covariância e hiper-parâmetros que melhor predizem o fenômeno de vibração através do espaço. Além disso, foi avaliado o algoritmo BO com uma estratégia de redução da entropia. O robô aprendeu um modelo da irregularidade do terreno, enquanto ao mesmo tempo reduziu exposição desnecessária em áreas que apresentam vibrações excessivas.

As contribuições deste trabalho são uma etapa importante para a autonomia a longo prazo para a robótica ao ar livre. A irregularidade do terreno pode ser rapidamente modelada, reduzindo o risco de danos para o robô de uma maneira eficiente. Os algoritmos que foram desenvolvidos e executados em um robô durante a navegação em um ambiente externo demonstram evidências empíricas de sua eficácia para aplicações em tempo real em sistemas embarcados.

Contudo, o sistema proposto apresenta restrições. Um ser humano conduziu o robô Husky após os algoritmos de planejamento sugerirem os pontos de interesse onde o robô deveria visitar. O mesmo foi usado, porque, na época, o interesse maior era o de validar o sistema proposto e também por não ter sido desenvolvido um planejador local no sistema de controle do robô. Sanar essas lacunas é um dos aspectos considerados como trabalhos futuros da pesquisa.

Foram avaliadas funções de covariância para cada um dos filtros de pré-processamento de dados para obter uma ótima representação da irregularidade do terreno. Foi observado que se poderia ter usado a função Matérn 3, mas a Matérn 5 mostrou RMSE e o tempo computacional menores em termos de média e desvio padrão. Dessa forma, foi realizada uma comparação entre os algoritmos BO e Entropia considerando a Matérn 5 e o melhor conjunto de hiper-parâmetros encontrado através da maximização da Equação 5.7. A Figura 5.6 apresenta a comparação entre as estratégias BO e Entropia, e pode-se ver que nos cenários de grama e asfalto, o BO evita os obstáculos e mantém-se em áreas com baixa vibração. A Tabela 5.2 apresenta os resultados dos algoritmos de planejamento nos diferentes cenários, demonstrando que o BO produziu em termos de média e desvio padrão valores menores comparados com a Entropia.

CAPÍTULO

6

Conclusão

A navegação autônoma é um dos problemas fundamentais na área da robótica móvel. Os algoritmos e soluções desenvolvidos nessa área atuam em ambientes estruturados, como corredores e salas dentro de prédios. No entanto, diversas abordagens têm sido relacionadas à navegação em ambientes externos, não estruturados, como o desenvolvimento de sistemas de navegação autônoma baseada no condutor humano e baseada na irregularidade do terreno, pois são temas de grande interesse dos pesquisadores em robótica móvel.

Este trabalho apresenta o desenvolvimento de dois sistemas de navegação autônoma, um capaz de aprender baseado em comandos fornecidos por um humano e o outro baseado em dados de sensores posicionados sobre os robôs móveis. A primeira aplicação relaciona-se com a condução autônoma ou assistida de robôs móveis (e.g. veículo) em ambientes urbanos, para permitir uma diminuição do número de acidentes, gastos com atropelamentos de pedestres e mobilidade para deficientes físicos e idosos. A segunda, é destinada ao monitoramento ambiental de robôs móveis em ambientes não estruturados, para que os robôs percorram e monitorem áreas específicas do ambiente de maneira autônoma ou assistida por um guarda ambiental.

O primeiro trabalho realizado nesta tese permitiu ao CaRINA 1 navegar de forma autônoma em vias urbanas, desviando de obstáculos, evitando calçadas e não subindo em áreas não navegáveis. Além disso, ele foi capaz de seguir as coordenadas de GPS que foram fornecidas pelo sistema e também percorreu uma distância de 1,1 km de forma autônoma. Contudo, esta aplicação apresenta restrições, como o fato de a identificação da rua combinada com a iden-

tificação de geometrias permitir que o CaRINA 1 mude de faixa sem que ele esteja disposto a fazê-lo. O erro grande de GPS (3 m) e a mudança de luminosidade sobre o ambiente são aspectos considerados como investigação de trabalhos futuros desta aplicação.

No segundo trabalho, o robô foi capaz de navegar em um ambiente externo, minimizando o gasto de energia. Os algoritmos foram desenvolvidos e executados em um robô real durante a navegação em um ambiente não-estruturado, demonstrando evidências empíricas de sua eficácia. Pode ser visto que nos cenários de grama e asfalto, o planejador BO evita os obstáculos (áreas com vibração excessiva) e mantém-se em áreas com baixa vibração.

6.1 Trabalhos Futuros

O primeiro problema abordado nesta tese, a navegação autônoma baseada no condutor humano, apresenta alguns trabalhos futuros que podem ser destacados, tais como:

- Avaliação de outros métodos de classificação de rua (áreas navegáveis e não navegáveis) e algoritmos de decisão para o controle do sistema de navegação autônoma de veículos;
- Integração de câmeras (monocular, estéreo ou térmica) com o sensor *laser* para uma compreensão mais completa do ambiente, a fim de lidar com as depressões, obstáculos, luminosidade, chuvas e neblinas nas ruas;
- Desenvolvimento de um algoritmo de identificação de guias (meio-fio), a utilização de um GPS diferencial e o uso da identificação de geometrias, a fim de permitir que o CaRINA 1 se mantenha em uma faixa de trânsito e não oscile nas vias urbanas;
- Avaliação de técnicas de aprendizado de máquina para que o sistema proposto diferencie o perfil de motoristas humanos (e.g. jovens, adultos e idosos);
- Desenvolvimento e avaliação de algoritmos de planejamento de trajetórias, a fim de permitir uma diminuição do consumo de combustível do CaRINA 2.

O segundo problema destacado nesta tese, a navegação autônoma baseada na irregularidade do terreno, apresenta alguns trabalhos futuros que podem ser destacados, tais como:

- Desenvolvimento de um planejador local para ser inserido no robô, a fim de deslocá-lo para os pontos sugeridos pelo algoritmo de planejamento de trajetórias BO e Entropia;
- Desenvolvimento de um modelo de vibração mais completo, tendo como entradas: velocidade, a direção de viagem e características a partir de um sistema de percepção visual;
- Desenvolvimento de um modelo de estimação do consumo de energia produzido por um robô real (e.g. Husky, Wombot), a fim de permitir que um robô terrestre navegue em áreas não-estruturadas com baixo consumo de energia;
- Desenvolvimento e avaliação de outros filtros de pré-processamento de dados, com o intuito de obter uma representação mais precisa da irregularidade do terreno e do consumo de energia sobre o ambiente em questão;
- Fusão de câmeras (monocular, estéreo ou térmica) com o sensor *laser* e GPS para obter uma compreensão mais completa do ambiente, a fim de lidar com obstáculos dinâmicos, chuva, noite e neblinas nos ambientes externos;
- Desenvolvimento e avaliação de algoritmos de planejamento de trajetórias (baseado em amostragem), a fim de permitir uma suavidade nas trajetórias e uma diminuição do consumo de energia produzido por um robô real.

Referências Bibliográficas

ALPAYDIN, E. *Introduction to machine learning*. second ed. MIT Press, 2010.

BAJRACHARYA, M.; MAIMONE, M. W.; HELMICK, D. Autonomy for mars rovers: Past, present, and future. *IEEE Computer*, v. 41, p. 44–50, 2008.

BANDYOPHADYAY, T.; SARCIONE, L.; HOVER, F. S. A simple reactive obstacle avoidance algorithm and its application in singapore harbor. In: *Field and Service Robotics*, 2010, p. 455–465.

BISHOP, C. M. *Neural networks for pattern recognition*. Oxford, 1996.

BRADSKI, G.; KAEHLER, A. *Learning opencv: Computer vision with the opencv library*. O'Reilly, Cambridge, 2008.

BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B. *Redes neurais artificiais - teoria e aplicações*. LTC, 2000.

BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B. *Redes neurais artificiais - teoria e aplicações*. 2ª ed. LTC, 2007.

BRAHIM-BELHOUARI, S.; BERMAK, A. Gaussian Process for Non-Stationary Time Series Prediction. *Computational Statistics and Data Analysis*, 2004.

BROCHU, E.; CORA, V. M.; DE FREITAS, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. University of British Columbia, Tech. Rep, 2010.

BRUCE, R. A bayesian approach to semi-supervised learning. In: *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium - NLPRS*, 2001, p. 57–64.

BURRUS, C. S.; GOPINATH, R. A.; GUO, H. *Introduction to wavelets and wavelet transforms: A primer*. Prentice Hall, 1998.

CARVALHO, A. C. P. L. F.; KOWALTOWSKI, T. *Atualizações em informática*. PUC-Rio, 2009.

CHAN, M.; PARTOUCHE, D.; PASQUIER, M. An intelligent driving system for automatically anticipating and negotiating road curves. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, p. 117–122.

CHEN, Y.; ZHANG, T.; PENG, X.; LIANG, L.; LIU, Y. Research review on the navigation of agricultural mobile robot. In: *American Society of Agricultural and Biological Engineers (ASABE)*, 2012, p. 1–8.

CHENG, H. *Autonomous intelligent vehicles: Theory, algorithms, and implementation (advances in computer vision and pattern recognition)*. Springer, 2011.

COBHAM Cobham - mission equipment. `http://www.armedforces-int.com/suppliers/telerob-gmbh.html`, acesso em 15 de janeiro, 2014.

COGGER, K. Nonlinear multiple regression methods: A survey and extensions. *Intelligent Systems in Accounting, Finance & Management*, v. 17, p. 19–39, 2010.

CYBENKO, G. *Continuous valued neural networks with two hidden layers are sufficient*. Relatório Técnico, Technical report, Department of Computer Science, Tufts University, 1988.

CYBENKO, G. Approximation by superpositions of a sigmoid function. *Mathematics of Control, Signals and Systems*, v. 2, p. 303–314, 1989.

DAHLKAMP, H.; KAEHLER, A.; STAVENS, D.; THRUN, S.; BRADSKI, G. Self–supervised monocular road detection in desert terrain. In: *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June, 2006.

DARTER, M.; GORDON, V. Vehicle steering control using modular neural networks. In: *Proceedings of International Conference on Information Reuse and Integration*, 2005, p. 374–379.

DUDEK, G.; JENKIN, M. *Computational principles of mobile robotics*. 2nd ed. Cambridge University Press, 2010.

EBDEN, M. Gaussian processes for regression: A quick introduction. 2008.

FAHLMAN, S. E. Faster-learning variations on back-propagation: An empirical study. In: *Proceedings of the Connectionist Models Summer School*, 1988, p. 38–51.

FANN Fast artificial neural network (fann). `http://leenissen.dk/fann/`, acesso em 10 de janeiro, 2014.

FAY, M. P.; PROSCHAN, M. A. Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics Surveys*, v. 4, p. 1–39, 2010.

FERNANDES, L. C.; SOUZA, J. R.; SHINZATO, P. Y.; PESSIN, G.; MENDES, C. C. T.; OSÓRIO, F. S.; WOLF, D. F. Intelligent robotic car for autonomous navigation: Platform and system architecture. In: *Second Brazilian Conference on Critical Embedded Systems*, 2012, p. 12–17.

FLIR Câmeras térmicas para segurança. `http://www.flir.com/BR/`, acesso em 15 de janeiro, 2014.

GUIZILINI, V. C. *Non-parametric learning for monocular visual odometry*. Thesis in computer science, school of information technologies, The University of Sydney, 2013.

HAMZEI, S.; HOSSEIN, G. *Decision tree learning for intelligent mobile robot navigation*. Phd theses (electronic, electrical and systems engineering), Loughborough University, 1998.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: Data mining, inference and prediction*. 2ª ed. Springer-Verlag, 2009.

HAYKIN, S. *Neural networks: A comprehensive foundation*. 2nd ed. Prentice Hall, 1999.

HUSKY Clearpath robotics. `http://www.clearpathrobotics.com/husky/`, acesso em 15 de dezembro, 2013.

IROBOT Cleaning robots (roomba and scooba). `http://www.irobot.com/`, acesso em 15 de janeiro, 2014.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: A review. *ACM Computing Surveys*, v. 31, n. 3, p. 264–323, 1999.

JUNG, C. R.; OSÓRIO, F. S.; KELBER, C. R.; HEINEN, F. J. Computação embarcada: Projeto e implementação de veículos autônomos inteligentes. In: *Jornada de Atualização em Informática – Congresso da SBC*, São Leopoldo – RS, 2005, p. 1358–1406.

KUNZLE, P. Gamedev.net - vehicle control with neural networks. `http://www.gamedev.net/reference/articles/article1988.asp`, acesso em 06 de outubro, 2010.

LIN, P.; BEKEY, G.; ABNEY, K. Autonomous military robotics: Risk, ethics and design. California Polytechnic State University, San Luis Obispo, 2008.

LIZOTTE, D. *Practical bayesian optimization*. Phd thesis, University of Alberta, Edmonton, Alberta, Canada, 2008.

LOCATELLI, M. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 1997.

MARCHANT, R.; RAMOS, F. Bayesian Optimisation for Intelligent Environmental Monitoring. In: *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2012.

MARKELIC, I.; KJAER-NIELSEN, A.; PAUWELS, K.; JENSEN, L. B. W.; CHUMERIN, N.; VIDUGIRIENE, A.; TAMOSIUNAITE, M.; HULLE, M. V.; KRUERGER, N.; ROTTER, A.; WOERGOETTER, F. The driving school system: Learning automated basic driving skills from a teacher in a real car. *International Journal of Vehicle Autonomous Systems*, 2011.

MARKELIC, I.; KULVICIUS, T.; TAMOSIUNAITE, M.; WORGOTTER, F. Anticipatory driving for a robot-car based on supervised learning. In: *Lecture Notes in Computer Science: Anticipatory Behavior in Adaptive Learning Systems*, 2009, p. 267–282.

MARSLAND, S. *Machine learning: An algorithm perspective*. Chapman & Hall/CRC, 2009.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 115–133, 1943.

MIT The mit intelligent wheelchair project developing a voice-commandable robotic wheelchair. `http://rvsn.csail.mit.edu/wheelchair/`, acesso em 15 de janeiro, 2014.

MITCHELL, T. *Machine learning*. McGraw Hill Series in Computer Science, 1997.

MOCKUS, J. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, v. 4, p. 347–365, 1994.

MURPHY, R. R.; KRAVITZ, J.; STOVER, S. L.; SHOURESHI, R. Mobile robots in mine rescue and recovery. *IEEE Robotics & Automation Magazine*, v. 16, p. 91–103, 2009.

OENTARYO, R. J.; PASQUIER, M. Gensofnn-yager: A novel hippocampus-like learning memory system realizing yager inference. In: *International Joint Conference on Neural Networks*, 2006, p. 705–712.

OSBORNE, M. *Bayesian gaussian processes for sequential prediction, optimization and quadrature*. Phd thesis, University of Oxford, 2010.

OZGUNER, U.; ACARMAN, T.; REDMILL, K. *Autonomous ground vehicles*. Artech House, 2011.

PESSIN, G. *Estratégias inteligentes aplicadas em robôs móveis autônomos e em coordenação de grupos de robôs*. Tese em ciência da computação e matemática computacional, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos-SP., 2013.

PETROVSKAYA, A.; THRUN, S. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots Journal*, v. 26, n. 2-3, p. 123–139, 2009.

POMERLAU, D. A. *Neural network vision for robot driving*. 1995.

RASMUSSEN, C. E.; WILLIAMS, C. K. I. *Gaussian processes for machine learning*. The MIT Press, Cambridge, 2006.

REED, R. Pruning algorithms - a survey. In: *IEEE Transactions on Neural Networks*, 1993, p. 740–746.

REZENDE, S. *Sistemas inteligentes: Fundamentos e aplicações*. Manole Ltda, 2003.

RIEDMILLER, M. *Rprop - description and implementation details*. Relatório Técnico, Technical Report, University of Karlsruhe, 1994.

RIEDMILLER, M.; BRAUN, H. Rprop - a fast adaptive learning algorithm. In: *Proceedings of the International Symposium on Computer and Information Science VII*, 1992.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: *Proceedings of the IEEE International Conference on Neural Networks*, 1993, p. 586–591.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol*, v. 65, n. 6, p. 386–408, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge Massachusetts, 1986.

RUMELHART, D. E.; MCCLELLAND, J. L. *Parallel distributed processing*. The MIT Press, 1986.

SANTNER, T. J.; WILLIAMS, B.; NOTZ, W. *The design and analysis of computer experiments*. Springer, 2003.

SASENA, M. J. *Flexibility and efficiency enhancement for constrained global design optimization with kriging approximations*. Phd thesis, University of Michigan, 2002.

SCHAEL, M. Invariant texture classification using group averaging with relational kernel functions. In: *Proceedings of the International Workshop on Texture Analysis and Synthesis*, 2002, p. 129–133.

SCHÖOLKOPF, B.; SMOLA, A. *Learning with Kernels*. The MIT Press, Cambridge, Massachuset, 2002.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, v. 52, p. 591–611, 1965.

SHEN, J.; HU, H. Svm based slam algorithm for autonomous mobile robots. In: *IEEE International Conference on Mechatronics and Automation*, 2007, p. 337–342.

SHINZATO, P. Y. *Sistema de identificação de superfícies navegáveis baseado em visão computacional e redes neurais artificiais*. Dissertação (mestrado em ciência da computação e matemática computacional), Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos-SP., 2010.

SHINZATO, P. Y.; WOLF, D. F. A road following approach using artificial neural networks combinations. *Journal of Intelligent and Robotic Systems*, v. 62, p. 527–546, 2011.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to autonomous mobile robots*. 2nd ed. The MIT Press, Cambridge, Massachusetts, 2011.

SINGH, A.; KRAUSE, A.; GUESTRIN, C.; KAISER, W.; BATALIN, M. Efficient Planning of Informative Paths for Multiple Robots. In: *International Joint Conference on Artificial Inteligence (IJCAI)*, 2007.

SINGH, A.; RAMOS, F.; WHYTE, H.; KAISER, W. Modeling and decision making in spatio-temporal processes for environmental surveillance. In: *IEEE Conference on Robotics and Automation (ICRA)*, 2010.

SIQUEIRA, M. G.; DINIZ, P. S. R. *The Electrical Engineering Handbook, Chapter 2, Digital Filters*. Elsevier Academic Press, 2005.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical Bayesian Optimization of Machine Learning Algorithms. In: *Neural Information Processing Systems (NIPS)*, 2012.

SOUZA, J. R.; MARCHANT, R.; OTT, L.; WOLF, D. F.; RAMOS, F. Bayesian optimisation for active perception and smooth navigation. In: *International Conference on Robotics and Automation (ICRA)*, 2014, p. 1–8.

SOUZA, J. R.; PESSIN, G.; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Vision-based waypoint following using templates and artificial neural networks. *Neurocomputing*, v. 107, p. 77–86, 2013a.
Disponível em `http://dx.doi.org/10.1016/j.neucom.2012.07.040`

SOUZA, J. R.; SALES, D. O.; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Template–based autonomous navigation in urban environments. In: *Proceedings of the 2011 ACM Symposium on Applied Computing*, TaiChung – Taiwan, 2011a, p. 1376–1381.
Disponível em `http://dx.doi.org/10.1145/1982185.1982485`

SOUZA, J. R.; SALES, D. O.; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Template–based autonomous navigation and obstacle avoidance in urban environments. *ACM SIGAPP Applied Computing Review*, v. 11, n. 4, p. 49–59, 2011b.
Disponível em `http://dx.doi.org/10.1145/2107756.2107761`

STEIN, P.; SANTOS, V. Visual guidance of an autonomous robot using machine learning. In: *7th Symposium on Intelligent Autonomous Vehicles*, Lecce Italy, 2010.

STENTZ, A.; BARES, J.; PILARSKI, T.; STAGER, D. The crusher system for autonomous navigation. In: *AUVSIs Unmanned Systems North America*, 2007.

STRELTSOV, S.; VAKILI, P. A non-myopic utility function for statistical global optimization algorithms. *Journal of Global Optimization*, v. 14, p. 283–298, 1999.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics (intelligent robotics and autonomous agents series)*. The MIT Press, Cambridge, Massachusetts, 2005.

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. V.; JENSEN, E.; ALESSANDRINI, P.; DAVIES, G. B. B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P. Stanley, the robot that won the darpa grand challenge. *Journal of Field Robotics*, v. 23, p. 661–692, 2006.

TUNSTEL, E.; DOLAN, J. M.; FONG, T. W.; SCHRECKENGHOST, D. Mobile robotic surveying performance for planetary surface site characterization. In: *8th Workshop on Performance Metrics for Intelligent Systems*, 2009, p. 1–21.

VASQUEZ, E.; BECT, J. *On the convergence of the expected improvement algorithm*. Relatório Técnico, Technical Report, arXiv.org, 2008.

WEISS, C.; FRÖHLICH, H.; ZELL, A. Vibration-based terrain classification using support vector machines. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

WEISS, C.; TAMIMI, H.; ZELL, A. A combination of vision- and vibration-based terrain classification. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008.

WIDROW, B.; HOFF, M. E. *Adaptive switching circuits*. Relatório Técnico, Institute of Radio Engineers, Western Electronic Show and Convention, 1960.

WOLF, D. F.; SIMOES, E. V.; OSÓRIO, F. S.; JR, O. T. *Robótica inteligente: Da simulação às aplicações no mundo real*. XXVIII Jornadas de Atualização em Informática, 1–51 p., 2009.

ZELL, A.; MAMIER, G.; MACHE, M.; HUBNER, R.; HERRMANN, S.; SOYEZ, T.; SCHMALZL, M.; SOMMER, T.; HATZIGEORGIOU, A.; POSSELT, D.; SCHREINER, T.; KETT, B.; CLEMENTE, G.; WIELAND, J. *Stuttgart neural network simulator - user manual - version 4.1*. Relatório Técnico, Technical Report, University of Stuttgart, 1995.

ZILINSKAS, A.; ZILINSKAS, J. Global optimization based on a statistical model and simplical partitioning. *Computers and Mathematics with Applications*, v. 44, p. 957–967, 2002.

APÊNDICE

A

Artigos Publicados

A.1 Publicados em periódicos

1. FERNANDES, L. C.; **SOUZA, J. R.**; PESSIN, G.; SHINZATO, P. Y.; SALES, D. O.; MENDES, C. C.; PRADO, M. G.; KLASER, RAFAEL L.; MAGALHAE, A. C.; HATA, A. Y.; PIGATTO, D.; BRANCO, K. R. L. J. C.; GRASSI JR., V.; OSORIO, F. S.; WOLF, D. F. CaRINA Intelligent Robotic Car: Architectural Design and Applications. Journal of Systems Architecture, 2014. http://dx.doi.org/10.1016/j.sysarc.2013.12.003

2. **SOUZA, J. R.**; PESSIN, G.; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Vision-based waypoint following using templates and artificial neural networks. *Neurocomputing*, vol. 107, p. 77-86, 2013. http://dx.doi.org/10.1016/j.neucom.2012.07.040

3. PESSIN, G.; OSÓRIO, F. S.; **SOUZA, J. R.**; UEYAMA, J.; COSTA, F. G.; WOLF, D. F.; DIMITROVA, D.; BRAUN, T.; VARGAS, P. A. Investigation on the evolution of an indoor robotic localization system based on wireless networks. *Applied Artificial Intelligence*, vol. 27, p. 743-758, 2013. http://dx.doi.org/10.1080/08839514.2013.823328

4. **SOUZA, J. R.**; SALES, D. O.; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Template-based autonomous navigation and obstacle avoidance in urban environments. *Applied Computing Review*, vol. 11, p. 49-59, 2011. http://dx.doi.org/10.1145/2107756.2107761

A.2 Publicados em conferências

1. **SOUZA, J. R.**; MARCHANT, R.; OTT, L.; WOLF, D. F.; RAMOS, F. Bayesian optimisation for active perception and smooth navigation. *IEEE International Conference on Robotics and Automation (ICRA)*, p. 1-7, 2014.

2. **SOUZA, J. R.**; MARCHANT, R.; OTT, L.; WOLF, D. F.; RAMOS, F. Bayesian optimisation for active perception and smooth navigation. *Robotics: Science and Systems - RSS (Workshop on Robotic Exploration, Monitoring, and Information Collection: Nonparametric Modeling, Information-based Control, and Planning under Uncertainty)*, p. 1-8, 2013.

3. **SOUZA, J. R.**; PESSIN, G.; EBOLI, G. B.; MENDES, C. C. T.; OSÓRIO, F. S.; WOLF, D. F. Vision and gps-based autonomous vehicle navigation using templates and artificial neural networks. *27th Annual ACM Symposium on Applied Computing (SAC)*, New York: ACM, p. 280-285, 2012. http://dx.doi.org/10.1145/2245276.2245332

4. **SOUZA, J. R.**; PESSIN, G.; OSÓRIO, F. S.; WOLF, D. F.; VARGAS, P. A. Combining evolution and training in a robotic controller for autonomous vehicle navigation. *13th Towards Autonomous Robotic Systems (TAROS), Advances in Autonomous Robotics*, vol. 7429, p. 426-427, 2012. http://dx.doi.org/10.1007/978-3-642-32527-4_43

5. PESSIN, G.; OSÓRIO, F. S.; **SOUZA, J. R.**; COSTA, F. G.; UEYAMA, J.; WOLF, D. F.; BRAUN, T.; VARGAS, P. A. Evolving an indoor robotic localization system based on wireless networks. *13th International Conference on Engineering Applications of Neural Networks (EANN)*, vol. 311, p. 61-70, 2012. http://dx.doi.org/10.1007/978-3-642-32909-8_7

6. AILINCA, M.; PARANTHAMAN, P. K.; PESSIN, G.; OSÓRIO, F. S.; **SOUZA, J. R.**; WOLF, D. F.; UEYAMA, J.; VARGAS, P. A. Uncovering new neural network topologies in real world robot applications. *12th Annual Workshop on Computational Intelligence (UKCI)*, p. 53-58, 2012.

7. FERNANDES, L. C.; **SOUZA, J. R.**; SHINZATO, P. Y.; PESSIN, G.; MENDES, C. C. T.; OSÓRIO, F. S.; WOLF, D. F. Intelligent robotic car for autonomous navigation: platform and system architecture. *II Brazilian Conference on Critical Embedded Systems (CBSEC)*, Los Alamitos, CA, USA: IEEE Computer Society, p. 12-17, 2012. http://dx.doi.org/10.1109/CBSEC.2012.26

8. **SOUZA, J. R.**; SALES, D. O; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Template-based autonomous navigation in urban environments. *26th Annual ACM Symposium on Applied Computing (SAC)*, New York: ACM, p. 1376-1381, 2011. http://dx.doi.org/10.1145/1982185.1982485

9. **SOUZA, J. R.**; PESSIN, G.; OSÓRIO, F. S.; WOLF, D. F. Vision-based autonomous navigation using supervised learning techniques. *12th Engineering Applications of Neural Networks (EANN)*, IFIP Advances in Information and Communication Technology, Boston: Springer, vol. 363, p. 11-20, 2011. http://dx.doi.org/10.1007/978-3-642-23957-1_2

10. **SOUZA, J. R.**; PESSIN, G.; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Vision-based autonomous navigation using neural networks and templates in urban environments. *I Brazilian Conference on Critical Embedded Systems (CBSEC)*, p. 55-60, 2011.

11. PESSIN, G.; OSÓRIO, F. S.; UEYAMA, J.; **SOUZA, J. R.**; WOLF, D. F.; BRAUN, T; VARGAS, P. A. Evaluating the impact of the number of access points in mobile robots localization using artificial neural networks. *The Fifth International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE)*, New York: ACM, p. 1-9, 2011. http://dx.doi.org/10.1145/2016551.2016561

12. **SOUZA, J. R.**; PESSIN, G.; OSÓRIO, F. S.; WOLF, D. F. Avaliação de árvores de decisão no controle de navegação robótica. *III Workshop on Computational Intelligence (WCI), Joint Conference 2010 Workshop Proceedings (SBIA/SBRN/JRI)*, Porto Alegre/RS: Sociedade Brasileira de Computação (SBC), p. 488-493, 2010.

13. PESSIN, G.; OSÓRIO, F. S.; DIAS, M. A.; **SOUZA, J. R.**; NETO, D. F. Avaliação de técnicas de otimização aplicadas à formação e atuação de grupos robóticos. *III Workshop on Computational Intelligence (WCI), Joint Conference 2010 Workshop Proceedings (SBIA/SBRN/JRI)*, Porto Alegre/RS : Sociedade Brasileira de Computação (SBC), p. 494-499, 2010.