Pós-Graduação em Ciência da Computação

Eulogio Gutierrez Huampo

Reconhecimento de Fala em Tempo Real para Robôs Móveis

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2020

Eulogio Gutierrez Huampo

Reconhecimento de Fala em Tempo Real para Robôs Móveis

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestrado em Ciência da Computação.

Área de Concentração: Inteligência Computacional
Orientador: Aluizio Fausto Ribeiro Araújo

Recife
2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

H874r Huampo, Eulogio Gutierrez
Reconhecimento de fala em tempo real para robôs móveis / Eulogio Gutierrez Huampo. – 2020.
82 f.: il., fig., tab.

Orientador: Aluízio Fausto Ribeiro Araújo.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências e apêndices.

1. Inteligência computacional. 2. Reconhecimento de fala. I. Araújo, Aluízio Fausto Ribeiro (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2020 - 111

Eulogio Gutierrez Huampo

"Reconhecimento de Fala em Tempo Real para Robôs Móveis"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 27/03/2020.

BANCA EXAMINADORA

Prof. Dr. Hansenclever de França Bassani
Centro de Informática / UFPE

Prof. Dr. Bruno José Torres Fernandes
Escola Politécnica de Pernambuco / UPE

Prof. Dr. Aluizio Fausto Ribeiro Araújo
Centro de Informática / UFPE
(**Orientador**)

Dedico este trabalho primeiramente a Deus Criador, socorro presente na hora da angústia. A toda minha família e aos meus amigos com quem convivi nesses espaços ao longo desses anos.

AGRADECIMENTOS

Agradeço à Organização de Estados Americanos (OEA) e ao Grupo Coimbra de Universidades Brasileiras (GCUB) pela bolsa de estudo. Ao Prof. Aluizio F. R. Araujo pela orientação. À equipe de Laboratório de Pesquisa pela grande ajuda. À Universidade Federal de Pernambuco (UFPE) pela oportunidade de realizar o curso de Mestrado em Ciência da Computação e pelo curso de Português para Estrangeiros no Núcleo de Línguas e Culturas (NLC) e no Núcleo de Línguas – Idiomas sem Fronteiras (Nucli-IsF). À equipe de Suporte PAL Robotics S.L.

RESUMO

O reconhecimento de fala para robôs é um campo multidisciplinar que envolve processamento de sinais e novos modelos de aprendizagem de máquina. Entre suas aplicações, podemos citar o controle de uma cadeira de rodas por comandos de voz, para ajudar pessoas com restrições de mobilidade. O objetivo deste trabalho é propor e implementar modelos de reconhecimento de comandos de voz em tempo real, fazendo ênfase em aprendizagem on-line e a integração ao robô TIAGo, a fim de executar tarefas de navegação. Modelos independentes do locutor são implementados baseados numa nova base de dados de vocabulário limitado, o extrator de características MFCC e a implementação das redes neurais MLP e oiSGNG, que vem do inglês *online incremental Supervised Growing Neural Gas*, para os quais no treinamento emprega-se a adição de ruído branco e colorido para melhorar seu desempenho. Nos resultados, em termos de taxa de acertos e tempo de processamento, o melhor caso é o MLP, mas requer alterar a estrutura e ajustar os parâmetros para aprender novas categorias. Por outro lado, o classificador oiSGNG não tem esses problemas. Neste caso, apresentamos duas contribuições: (i) nodos inseridos de acordo a uma função exponencial, desse jeito, gera-se uma taxa de acertos mais alta com menos nodos, o que implica menor latência; (ii) aprendizagem on-line de comandos de voz, uma nova implementação, o modelo aprende novos comandos ao mesmo tempo em que os grava. Nos experimentos, a navegação do robô é efetuada via comandos de voz, isto é, movimento da base do robô através dos comandos de velocidade. Para evitar positivos falsos e interferência do ambiente ruidoso, emprega-se uma palavra-chave associada ao comando, no caso do MLP empregamos limiar de ativação. Além disso, os modelos são avaliados para ambientes com ruído acústico, quer dizer, teste com adição de ruído de ventilador e ruído balbuciante. Em geral, a taxa de acertos é mais alta para os comandos gravados com detecção de som em tempo real e, a latência do processamento é menor que 15 ms, portanto, podemos concluir que o reconhecimento de comandos de voz é efetuado em tempo real.

Palavras-chaves: Processamento do sinal de áudio em tempo real. Aprendizagem on-line incremental. MLP com opção de rejeição. Ruído aditivo. Redes neurais em ROS.

ABSTRACT

Speech recognition in robotics is a multidisciplinary field that involves signal processing and new models of machine learning. Among its applications, we can mention the control of a wheelchair by voice commands, to help people with mobility restrictions. The objective of this work is to propose and implement voice recognition models in real time, emphasizing online learning and integration with the TIAGo robot, in order to perform navigation tasks. Speaker-independent models are implemented based on a new limited vocabulary database, the Mel-Frequency Cepstral Coefficients (MFCC) and neural networks. In this case, the Multi-Layer Perceptron (MLP) and the online incremental Supervised Growing Neural Gas (oiSGNG) are implemented, using additive white and color noise to improve their performance. In results, in terms of accuracy rate and processing time, the best case is the MLP, but it requires changing the structure and adjusting the parameters to learn new categories. On the other hand, the oiSGNG classifier does not have these problems. So, for this case, we present two contributions: (i) nodes inserted according to an exponential function, that results in a higher accuracy rate with fewer nodes, which implies less latency; (ii) online learning of voice commands, a novel implementation, the model learns new commands at the same time that it records them. In experiments, robot navigation is performed via voice commands, that is, moving the robot base through velocity commands. To avoid false positives and interference from the noisy environment, a keyword associated with the command is used, in the case of MLP we use activation threshold. In addition, the models are evaluated for environments with acoustic noise, that is to say, test with the additive fan and babble noise. In general, the accuracy rate is higher for commands recorded with real-time sound detection, and processing latency is less than 15 ms, so we can conclude that the voice command recognition is performed in real time.

Keywords: Real-time audio signal processing. Incremental online learning. MLP with reject option. Additive noise. Neural networks in ROS.

LISTA DE FIGURAS

Figura 1 – Complemento PAL Jetson TX2 conectado ao TIAGo 18
Figura 2 – Domínio agente-ambiente 20
Figura 3 – Arquitetura ASR *word model* empregando ANN 26
Figura 4 – Sinal no domínio de tempo 27
Figura 5 – Sinal no domínio do tempo após pré-ênfase (α=0.97) 27
Figura 6 – Sobreposição do segmentos 28
Figura 7 – Janela de Hamming 29
Figura 8 – Visualização dos Coeficientes MFCCs 31
Figura 9 – MLP de três camadas 32
Figura 10 – Fluxo de sinal destacando o neurônio-k da saída 32
Figura 11 – Fluxo de sinal relativo à retro-propagação de sinais de erro 34
Figura 12 – Relação sinal-ruído 41
Figura 13 – Nº de neurônios na camada oculta e taxa de acerto 43
Figura 14 – Erro versus Número de épocas 44
Figura 15 – Dependência do OCMR com os parâmetros do oiSGNG 46
Figura 16 – Processamento do sinal de áudio em tempo real 50
Figura 17 – Nodos inseridos on-line 53
Figura 18 – Sensibilidade e Especificidade versus Limiar 54
Figura 19 – Verificação e teste da navegação do robô TIAGo 56
Figura 20 – Banco de 40 filtros na escala de Mel 67
Figura 21 – Espectrograma de sinal no processamento MFCC 68
Figura 22 – Topologias de mapas auto-organizáveis 70
Figura 23 – Duas configurações de grade e níveis de vizinhança 70
Figura 24 – BMU no mapa bidimensional 71
Figura 25 – Acurácia versus Tamanho da grade 75
Figura 26 – Mapa auto-organizado 50x50 75
Figura 27 – Topologia hexagonal SOM 30x30 76
Figura 28 – Localização e navegação em ambientes domésticos 82

LISTA DE TABELAS

Tabela 1 – Tipos de vocabulário 23
Tabela 2 – Notações do algoritmo 36
Tabela 3 – O número total de nodos inseridos 38
Tabela 4 – A base de dados de comandos de voz para navegação assistida 41
Tabela 5 – Taxa de acerto com adição do ruído 44
Tabela 6 – Parâmetros do algoritmo oiSGNG 46
Tabela 7 – Taxa de acerto (%) e nodos inseridos 47
Tabela 8 – Taxa de acerto (%) e nodos inseridos para diferentes frames 47
Tabela 9 – Avaliação das ANNs para ambientes ruidosos 48
Tabela 10 – Taxa de acerto (%) e latência (ms) 51
Tabela 11 – Topologia de vizinhança ($p \in Z : p > 0$) 74
Tabela 12 – Taxa de acerto (%) para 20 e 98 Frames 77
Tabela 13 – Transcrição fonética 79

LISTA DE ABREVIATURAS E SIGLAS

AGV	*Automated Guided Vehicles*
AMCL	*Adaptive Monte Carlo Localization*
ANN	*Artificial Neural Network*
ASR	*Automatic Speech Recognition*
BMU	*Best Matching Unit*
CMU	*Carnegie Mellon University*
CNN	*Convolutional Neural Network*
DCT	*Discrete Cosine Transform*
DFT	*Discrete Fourier Transform*
DTW	*Dynamic Time Warping*
FFT	*Fast Fourier Transform*
GMM	*Gaussian Mixture Model*
GNG	*Growing Neural Gas*
GPU	*Graphics Processing Unit*
HMM	*Hidden Markov Model*
HRI	*Human-Robot Interaction*
IWR	*Isolated Word Recognition*
kNN	*K-Nearest Neighbors*
LHS	*Latin Hypercube Sampling*
LRSOM	*Low Resolution Self-Organized Maps*
lrTSOM	*low resolution Topological Self-Organized Maps*
MFC	*Mel-Frequency Cepstrum*
MFCC	*Mel-Frequency Cepstral Coefficients*
MLP	*Multi-Layer Perceptron*
OCMR	*Online Cumulative Mistake Rate*
oiSGNG	*online incremental Supervised Growing Neural Gas*
RGB	*Red, Green and Blue*
ROC curve	*Receiver Operating Characteristic curve*

ROS	*Robot Operating System*
ROV	*Remotely Operated Vehicle*
RTC	*Real-Time Computing*
SCP	*Secure Copy Protocol*
SLAM	*Simultaneous Localization and Mapping*
SNR	*Signal-to-Noise Ratio*
SOM	*Self-Organized Maps*
STT	*Speech to Text*
SVM	*Support Vector Machines*
TOM	*Temporal Organization Map*
UAV	*Unmanned Aerial Vehicle*
VFH	*Vector Field Histogram*

SUMÁRIO

1 INTRODUÇÃO 14
1.1 MOTIVAÇÃO 14
1.2 APRESENTAÇÃO DO PROBLEMA 15
1.3 OBJETIVOS 17
1.4 ABORDAGENS ATUAIS E LIMITAÇÕES 17
1.5 PROPOSIÇÃO DA DISSERTAÇÃO 18
1.6 APRESENTAÇÃO DE EXPERIMENTOS 19
1.7 ESTRUTURA DO DOCUMENTO 19

2 RECONHECIMENTO DE FALA PARA ROBÔS 20
2.1 AMBIENTES ACÚSTICOS RUIDOSOS 20
2.1.1 Cenários acústicos 21
2.1.2 A Relação Sinal-ruído 21
2.1.3 Cores do Ruído 21
2.2 SISTEMAS ASR 22
2.2.1 Modelos ASR em Tempo Real 23
2.2.1.1 Modelo de Palavras Isoladas 24
2.2.1.2 PocketSphinx 24

3 MODELOS ASR NEURAL ON-LINE 26
3.1 PRÉ-PROCESSAMENTO 26
3.1.1 Pré-ênfase 27
3.1.2 Enquadramento 27
3.1.3 Sobreposição de quadros 28
3.1.4 Janela de Hamming 28
3.2 EXTRATOR MFCC 29
3.2.1 Operadores para Cálculo dos MFCCs 29
3.2.2 Coeficiente Cepstral de Frequência Mel (MFCC) 30
3.3 REDE NEURAL MLP 31
3.3.1 Métodos de Treinamento 33
3.3.2 Algoritmo de Retro-propagação 33
3.3.2.1 Processamento Direto 33
3.3.2.2 Função de Ativação 34
3.3.2.3 Processamento Reverso 34
3.4 CLASSIFICADOR OISGNG 35
3.4.1 Algoritmo oiSGNG Modificado 36

3.4.2 Intervalo de inserção de nodos . . . 37

4 RESULTADOS ASR NEURAL ON-LINE . . . 39
4.1 ADIÇÃO DE RUÍDO . . . 39
4.2 BASES DE DADOS "SPEECH COMMANDS" . . . 41
4.3 RESULTADOS DA REDE NEURAL MLP . . . 42
4.3.1 Parâmetros da Rede Neural MLP . . . 42
4.3.2 Treinamento e Validação . . . 43
4.3.3 Análise dos Resultados do Classificador MLP . . . 44
4.4 RESULTADOS DO ALGORITMO OISGNG+ . . . 45
4.4.1 Parâmetros do oiSGNG+ . . . 45
4.4.2 Algoritmo oiSGNG vs oiSGNG+ . . . 46
4.4.3 Analise da Divisão de Sinal em Frames . . . 47
4.5 TESTE DOS MODELOS ASR NEURAIS EM RUÍDO . . . 48

5 SIMULAÇÕES E EXPERIMENTOS . . . 49
5.1 ROBÔ TIAGO E ROS . . . 49
5.1.1 ROS Kinetic . . . 49
5.1.2 Simulador Gazebo . . . 50
5.2 PROCESSAMENTO DO SINAL DE ÁUDIO EM TEMPO REAL . . . 50
5.3 APRENDIZAGEM ON-LINE DE COMANDOS DE VOZ . . . 51
5.4 INTEGRAÇÃO DOS MODELOS ASR NEURAIS EM ROS . . . 53
5.4.1 Opção de Rejeição no MLP . . . 54
5.4.2 BMU no oiSGNG . . . 55
5.4.3 Tópico de Velocidade . . . 55
5.4.4 Navegação do Robô TIAGo via Comandos de Voz . . . 55

6 CONCLUSÕES E TRABALHOS FUTUROS . . . 57
6.1 CONCLUSÕES . . . 57
6.2 TRABALHOS FUTUROS . . . 58

REFERÊNCIAS . . . 59

APÊNDICE A – CEPSTRUM DE FREQUÊNCIA MEL . . . 66

APÊNDICE B – MAPAS AUTO-ORGANIZÁVEIS . . . 69

APÊNDICE C – POCKETSPHINX EM ROS KINETIC . . . 78

APÊNDICE D – LOCALIZAÇÃO E NAVEGAÇÃO . . . 81

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Reconhecimento de fala ou reconhecimento automático de fala, do inglês *Automatic Speech Recognition* (ASR), é um campo multidisciplinar, essencialmente envolve processamento de sinais e novos modelos de aprendizagem de máquina (YU; DENG, 2016) e (LEE; SOONG; PALIWAL, 2012). Segundo Kothakota (2018, p. 10, tradução nosssa), um sistema ASR "fornece a uma máquina a capacidade de identificar as palavras e frases presentes na linguagem falada e convertê-las em um formato legível por máquina".

Os robôs são agentes físicos que interagem com o ambiente (RUSSELL; NORVIG, 2004), podem ser manipuladores e/ou navegadores. Para executar essas tarefas, eles são equipados com efetuadores e sensores. Atualmente, os robôs usam vários tipos de sensores tais como câmeras, microfones e ultrassons para monitorar o ambiente e; giroscópios e acelerômetros para acompanhar seus próprios movimentos. De acordo com Pérez e Rumipamba (2014) e Polo (2015), o ambiente pode ser hostil aos seres humanos, por exemplo, causar incapacidade de respirar, baixa temperatura ou alta pressão, que podem ser fatais para seres vivos que os ocupem. Por tudo isso, o uso de robôs é adequado, havendo momentos que poder ser a única opção viável para execução de uma dada tarefa. Na robótica há quatro subcampos de estudo importantes: manipulação, navegação, visão e linguagem. Na tarefa de navegação, os robôs móveis se movimentam pelo ambiente usando rodas, pernas ou mecanismos semelhantes, os quais têm sido usados para distribuir refeições em hospitais e, inúmeras tarefas na indústria e para serviços.

Nesse contexto, a motivação principal desta pesquisa é ser pioneiro na integração de modelos ASR para robôs moveis usando uma rede neural artificial, do inglês *Artificial Neural Network* (ANN), implementado através do sistema operacional ROS, do inglês *Robot Operating System*. Na revisão de literatura, não se encontrou muitos casos da implementação das ANNs em plataformas para robôs empregando ROS, especificamente, para o robô TIAGo de Pal Robotics. Este tem disponível um sistema microfones estéreos para processamento de áudio, que pode ser usado para gravar um sinal de áudio e processá-lo, a fim de executar tarefas como reconhecimento de fala em tempo real. Como exemplo mais próximo, uma rede neural convolucional, do inglês *Convolutional Neural Network* (CNN), foi implementada e testada usando o robô TIAGo para realizar o reconhecimento da atividade humana em tempo real (MOCANU et al., 2018). No qual, combina-se informações espaciais e temporais extraídas de imagens adquiridas de câmeras tipo RGB, que vem do inglês *Red, Green and Blue* (RGB), sistema de cores aditivas vermelho, verde e azul. Apesar de lidar com imagens, o modelo comentado realiza reconhecimento de sinais.

1.2 APRESENTAÇÃO DO PROBLEMA

A fala é a forma mais básica, comum e eficiente método de comunicação para que as pessoas interagirem umas com as outras. Portanto, é esperado que os seres humanos desejem interagir com os computadores por meio da fala, em vez de usar interfaces primitivas (por exemplo, o teclado). A implementação de comando de voz pode ser usada para tarefas tais como acender e apagar luzes, ligar e desligar televisões e outros eletrodomésticos, digitar número de telefone apenas pelo som. Isso, segundo Prabhakar e Sahu (2013), pode ser feito por meio de um sistema ASR que converte um sinal de áudio para uma sequência de palavras escritas empregando um algoritmo implementado como um programa de computador. De acordo com Garg e Sharma (2016), o reconhecimento de fala é uma abordagem otimista para converter um sinal de áudio para a sequência de *tokens*, ou seja, sequência de palavras ou fonemas.

Garg e Sharma (2016) argumentam que muitos sistemas ASR atuais perdem desempenho ao lidar com sinais corrompidos ou ruidosos, estas perturbações podem ser causadas por várias fontes de som (por exemplo, ruído de fundo, outras pessoas falando ou ecos). Segundo Ravanelli (2017, p. 45, tradução nosssa), o ruído aditivo pode ser classificado em diferentes tipos de categorias:

> “ *De acordo com sua ocorrência no tempo, o ruído pode ser contínuo (por exemplo, ventiladores), intermitente (por exemplo, telefone tocando) ou impulsivo (por exemplo, uma batida na porta). O ruído também pode ser classificado de acordo com suas características de frequência (por exemplo, branco, rosa, azul) ou de acordo com suas propriedades estatísticas (gaussiano, de Poisson, etc.). Além disso, o ruído pode ser estacionário ou não estacionário, dependendo da evolução temporal de suas características estatísticas.* ”
>
> Mirco Ravanelli, *2017*

A adição do ruído artificialmente tem diversas aplicações práticas. O ruído branco é fundamental em processamento de sinais, utilizado em muitas disciplinas, tais como, física, música, engenharia acústica, telecomunicações, engenharia eletrônica e na ciência da computação. Em muitos casos, para eliminar o ruído de fundo (por exemplo, ruído de avião ou ruído de carro), é implementado o filtro Wiener, que é um filtro adaptativo (SILAGHI et al., 2014). Sob outra perspectiva, os pesquisadores Doyle e Evans (2018) e Vincent et al. (2008) tomam a adição de ruído para o treinamento de redes neurais artificiais com o propósito de melhorar seu desempenho.

Por outro lado, a navegação por robôs móveis é um tópico amplo que abrange diferentes tecnologias e aplicativos. Baseia-se em técnicas muito antigas, bem como em algumas das mais avançadas ciências e engenharia espaciais. DeSouza e Kak (2002) lidam a navegação tanto em ambientes em ambientes confinados quanto em ambientes externos. Na navegação interior, a maneira mais fácil de levar um robô a um local alvo é simplesmente guiá-lo para este local. A guiagem pode ser feita de diferentes maneiras: enterrando um loop indutivo ou ímãs no chão, pintando linhas no chão, colocando *beacons*, marcadores ou códigos de barras, entre outros recursos para determinar posições espaciais do robô. Esses veículos guiados automaticamente, do inglês *Automated Guided Vehicles* (AGV), são tipicamente usados em cenários industriais para tarefas de transporte.

O estudo dos comandos de voz para robôs móveis visa ajudar pessoas com restrições de mobilidade, por exemplo, paraplégicos. Na literatura, AL-Thahab (2011) propõe um sistema de reconhecimento de fala modelo palavras isoladas para controlar uma cadeira de rodas (*wheelchair*) motorizada. Wijoyo e Wijoyo (2011) e Rai e Rai (2014) propõem cinco comandos de voz (para frente, para trás, esquerda, direita e pare) para controlar robôs moveis. Além disso, Zaatri, Azzizi e Rahmani (2015) apresenta o uso da rede neural perceptron multicamada, do inglês *Multi-Layer Perceptron* (MLP), para reconhecimento de voz dedicado à geração dos comandos de robô.

Sendo assim, entre as características estudadas nos modelos ASR para robôs móveis, destacam-se a integração dos modelos em tempo real (TANG; REN; LIU, 2017), fala distante (RAVANELLI, 2017) e ruído acústico gerado pelo ambiente ou pelo movimento do próprio robô (NOVOA et al., 2018). A computação em tempo real (BUTTAZZO, 2011), do inglês *Real-Time Computing* (RTC), descreve os sistemas de hardware e software sujeitos a uma restrição em tempo real. O limite tolerável à latência para processamento de sinal de áudio/vídeo em tempo real são objeto de investigação, mas estima-se que esteja em torno de 20 ms (KUDRLE et al., 2010).

O problema com os modelos ASR treinados é que eles dependem das bases de dados empregadas no treinamento, as quais podem ter vocabulário limitado e que não existam para idiomas diferentes. Por outro lado, os modelos acústicos também são limitados a idiomas específicos. Nessa situação, um modelo ASR com aprendizagem on-line seria aplicável e vantajoso. A propósito de sistemas de aprendizagem, Kuremoto et al. (2012) introduzem dois tipos de sistemas de aprendizado de instruções para robôs *partners*: (1) um sistema de aprendizado de instruções de imagem manual; e (2) três tipos de sistemas de aprendizado de instruções por voz. Mas eles não lidam com aprendizagem on-line de comandos de voz.

1.3 OBJETIVOS

Propor e implementar modelos de reconhecimento de fala para robôs móveis, com ênfase nos modelos em tempo real e aprendizagem on-line usando redes neurais artificias, a fim de executar tarefas de navegação robótica empregando comando de voz.

Com a finalidade de atingir o objetivo geral são definidas as seguintes metas:

- Investigar o estado da arte sobre os modelos de reconhecimento de fala para robótica.
- Realizar o treinamento de modelos ASR para reconhecimento de palavras isoladas, baseadas em redes neurais artificiais e, considerando ruído acústico.
- Propor modelos ASR para aprendizagem on-line de comandos de voz.
- Integrar os modelos ASR no robô TIAGo através do sistema operacional ROS.
- Avaliar os modelos de reconhecimento de fala em tempo real e, sua aplicação à execução de tarefas de navegação assistida.

1.4 ABORDAGENS ATUAIS E LIMITAÇÕES

Em 1952, Devis desenvolveu o primeiro sistema ASR no laboratório Bell, que é um sistema de reconhecimento de dígitos isolados e dependente do locutor. Segundo Garg e Sharma (2016), por mais de 50 anos, o ASR alcançou um enorme progresso, desde o sistema mais simples de reconhecimento de palavras isoladas até o reconhecimento contínuo de fala. Inicialmente, os sistemas ASR foram estabelecidos em dois modelos acústicos: (1) *Word model*, em que o tamanho do vocabulário é pequeno e a palavra é modelada como um todo; e (2) *Phone model*, em que apenas partes das palavras, ou seja, fonemas, são modeladas. Neste tipo de modelagem, é possível o reconhecimento de fala contínua.

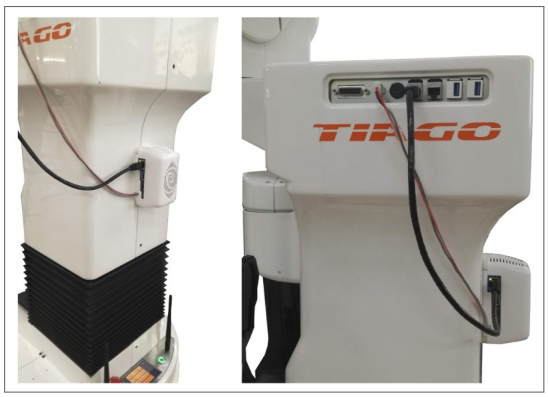
Existem diversos cenários para interação humano-robô, do inglês *Human-Robot Interaction* (HRI). Para reconhecimento de fala em cenários de navegação, propõe-se incorporar a representação e modelagem do ambiente HRI, baseado no modelo de rede neural profunda, as características acústicas limpas com as respostas do canal ruidoso são usadas no treinamento. Para o qual, as sentenças de treinamento foram geradas regravando o banco de dados de testes do Aurora-4 usando o robô móvel PR2, equipado com uma interface de áudio Kinect durante a realização de rotações da cabeça e movimentos da base em direção de uma fonte fixa (NOVOA et al., 2018).

Segundo PAL-Robotics (2019, p. 207, tradução nossa), o pacote de software para reconhecimento de fala incluído no TIAGo, quando ativado, “faz com que o ROS capture o áudio dos microfones do robô, envie-o aos reconhecedores para processamento e retorne

a fala reconhecida" e, possui as seguintes características: (1) reconhecimento contínuo de fala; (2) reconhecimento depois de ouvir uma palavra-chave especial; e (3) capacidade de usar vários reconhecedores de fala. Os requerimentos básicos para a implementação são o pacote de software premium para reconhecimento de fala e um nível apropriado de ruído ambiente. Quanto mais ruidoso for o ambiente, pior serão os resultados do reconhecimento. Os reconhecedores atuais implementados no robô TIAGo, são:

- A API do *Google Cloud Speech* em 80 idiomas, reconhecedor de frases comumente faladas. Requer que o TIAGo esteja conectado à Internet e possa acessar os servidores do Google e ter uma conta válida do *Google Cloud Speech*.
- *DeepSpeech*, uma implementação do *TensorFlow* da arquitetura *DeepSpeech* do Baidu, treinada em inglês americano. Requer NVIDIA Jetson TX2 (Fig. 1), que é um kit de desenvolvimento para máquinas autônomas com inteligência artificial.

Figura 1 – Complemento PAL Jetson TX2 conectado ao TIAGo



Fonte: Adaptado de PAL-Robotics (2019).

No contexto dessas abordagens, e levando em consideração que os reconhecedores de fala para o robô TIAGo mencionados acima devem ser adquiridos e adaptados para executar tarefas de navegação. Por conseguinte, propomos construir novos modelos ASR com ferramentas básicas. Porém, as limitações são: (1) ambientes não muito ruidosos; (2) distancias curtas entre o locutor e o robô; (3) o robô não tem conexão à Internet; e (4) o ASR desenvolvido não contempla usar a unidade de processamento gráfico, do inglês *Graphics Processing Unit* (GPU).

1.5 PROPOSIÇÃO DA DISSERTAÇÃO

A dissertação trata de reconhecimento de comandos de voz usando um robô móvel, para executar tarefas de navegação. Os modelos ASR são treinados para serem independentes do locutor, implementados usando redes neurais artificiais de tal maneira que possua

aprendizagem on-line de comandos de voz. Além disso, para que o modelo seja implementável no robô, deve ter processamento de sinal com detecção de som em tempo real.

1.6 APRESENTAÇÃO DE EXPERIMENTOS

O primeiro experimento realizado é o treinamento com adição de ruído para melhorar o desempenho da rede neural, em seguida, o modelo é avaliado com adição de ruído acústico de um ambiente real. Em seguida, para seleção do modelo neural considera-se o sinal dividida em frames sem e com sobreposição. Além disso, o modelo online incremental é treinado para diferentes números de categorias. Finalmente, compara-se a de taxa de acertos dos modelos empregando comandos de voz gravados com detecção de som em tempo real. Por último, aprendizagem on-line de comandos de voz, experimentos no simulador Gazebo e navegação do robô TIAGo em ambiente real.

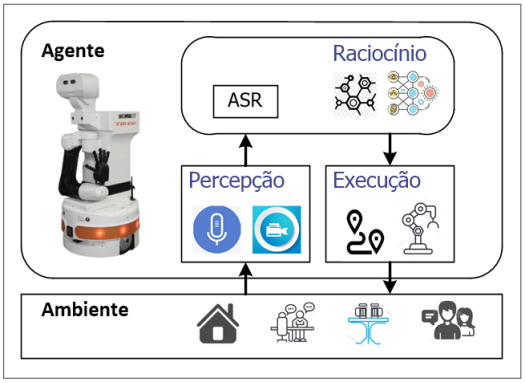
1.7 ESTRUTURA DO DOCUMENTO

Neste trabalho, apresenta-se no Capítulo 2, o estado da arte de reconhecimento de fala para robôs móveis, descrição do sistema robô-ambiente e, ruído acústico. No Capítulo 3, descreve-se processamento de sinais de áudio, ASR modelo palavras isoladas, pré-processamento, extração de características acústicas e, redes neurais artificiais fazendo ênfase em aprendizagem on-line. O Capítulo 4 trata de adição de ruído, tratamento do base de dados, treinamento e validação das redes neurais. O Capítulo 5 trata da integração dos modelos ASR ao robô TIAGo através do ROS, na qual destaca-se a detecção de som em tempo real, aprendizagem on-line de comandos de voz e, simulação e experimentos de navegação do robô via reconhecimento de palavras isoladas. Finalmente. o Capítulo 6 reporta a avaliação dos objetivos atingidos, sugestões e trabalhos futuros.

2 RECONHECIMENTO DE FALA PARA ROBÔS

No caso deste mestrado, o objetivo é a implementação de comandos de voz para robôs moveis, fazendo ênfase nos comandos para executar tarefas de navegação robótica. Portanto, o sistema de navegação assistida pode ser modelado como domínio agente-ambiente (ver Fig. 2), o agente é o robô TIAGo que possui percepção auditiva, raciocínio e, capacidade de navegação com desvio de obstáculos em ambientes internos.

Figura 2 – Domínio agente-ambiente



Fonte: Adaptado de Russell e Norvig (2004).

Neste capitulo tratamos os cenários acústicos e a classificação dos sistemas ASR para robôs. No caso de ambientes acústicos ruidosos, apresentamos a relação sinal-ruído e cores do ruído acústico que são utilizados nos sistemas ASR. Por outro lado, descrevemos os sistemas ASR, fazendo ênfase nos modelos em tempo real.

2.1 AMBIENTES ACÚSTICOS RUIDOSOS

No sistema domínio agente-ambiente (Fig. 2), ambiente representa tudo ao redor do robô que pode ser percebido através de sensores. No caso de sistemas ASR, o ruído acústico e o cenário acústico são objetivos do estudo. Em trabalhos recentes, Ravanelli (2017) e Novoa et al. (2018) mostram que sinais de áudio gravados por matrizes de microfone distantes podem ser processados por um *front-end*, que pode ser composto por aprimoramento de fala e análise de cena acústica. O papel da análise de cena acústica é fornecer informações úteis sobre o cenário acústico, enquanto o aprimoramento da fala tem o objetivo principal de melhorar a qualidade do sinal gravado pelos microfones.

2.1.1 Cenários acústicos

Em modo geral, o sinal de áudio gravado por um microfone distante (RAVANELLI, 2017), é descrito pela seguinte equação:

$$x^{(d)}[n] = s[n] * h[n] + g[n] \tag{2.1}$$

na qual, $s[n]$ é o sinal de fala próximo sem ruído, do inglês *close-talking speech*, corrompido pelo ruído $g[n]$ e pelas reverberações $h[n]$, portanto, o correspondente sinal de fala distante é $x^{(d)}[n]$, do inglês *distant-talking speech*. A resposta impulso $h[n]$, tipicamente é composta por três componentes básicos: o som direto, as primeiras reflexões e a reverberação tardia.

Em particular, o sinal de áudio $x^{(c)}[n]$ gravado por um microfone próximo ao locutor, que corresponde ao sinal de áudio sem ruído $s[n]$, também inclui a contribuição $g[n]$ de ruídos interferentes aditivos (KUTTRUFF, 2016), tais como ruído de carro, avião, música, fãs, balbucio e outros sons.

$$x^{(c)}[n] = s[n] + g[n] \tag{2.2}$$

2.1.2 A Relação Sinal-ruído

O segundo cenário acústico é a abordagem usada ao longo deste trabalho. Então, nesse caso a quantidade de ruído é normalmente medida com a relação sinal-ruído, do inglês *Signal-to-Noise Ratio* (SNR), que também é abreviado como S/R. De acordo com Johnson (2006), o SNR é uma medida usada em ciência e engenharia que compara o nível de um sinal desejado com o nível de ruído de fundo. SNR é definido como a razão entre a potência do sinal (P_S) e a potência do ruído (P_N), geralmente expressa em decibéis (dB).

$$SNR(dB) = 10log_{10}\frac{P_S}{P_N} \ . \tag{2.3}$$

Na literatura, considera-se um SNR de 3 dB como o limiar entre SNRs baixos e altos. A potência de uma variável aleatória é igual ao seu valor médio ao quadrado. Normalmente, o ruído tem média zero, o que torna sua potência igual à sua variância (JOHNSON, 2006).

2.1.3 Cores do Ruído

A cor do ruído diz respeito ao espectro de potência de um sinal de ruído. Os nomes de cores para esses diferentes tipos de sons são derivados de uma analogia entre o espectro da luz visível e o espectro de frequências de ondas sonoras presentes no som (DOYLE; EVANS, 2018).

- **Ruído Branco**. No processamento de sinal, o ruído branco é um sinal aleatório com intensidade igual em diferentes frequências, proporcionando uma densidade

espectral de potência constante. O ruído branco é um sinal, nomeado por analogia à luz branca, espectro de frequência plana quando representado como uma função linear da frequência em Hz (WEID et al., 1997), (DJURIC, 1996) e (KUO, 2018).

- **Ruído Rosa**. O espectro de frequências do ruído rosa é linear em escala logarítmica; tem potência igual em bandas proporcionalmente amplas. A densidade de potência espectral, em comparação com o ruído branco, diminui em 3 dB por oitava (densidade proporcional a $1/f$). Por esse motivo, o ruído rosa costuma ser chamado de "1/f noise"(SZENDRO; VINCZE; SZASZ, 2001).

- **Ruído Vermelho**. A terminologia "ruído vermelho", também chamada de ruído marrom ou ruído browniano, geralmente se refere a uma densidade de potência que diminui 6 dB por oitava com o aumento da frequência (densidade proporcional a $1/f^2$) (GILMAN; FUGLISTER; JR, 1963).

- **Ruído Azul**. O ruído azul também é chamado de “azure noise”. A densidade de potência do ruído azul aumenta 3 dB por oitava com frequência crescente (densidade proporcional a f) em uma faixa de frequência finita (LAU; ULICHNEY; ARCE, 2003).

- **Ruído Violeta**. O ruído violeta também é chamado de “purple noise”. A densidade de potência do ruído violeta aumenta 6 dB por oitava com frequência crescente (densidade proporcional a f^2) em uma faixa de frequência finita. Também é conhecido como ruído branco diferenciado (LAU; ULICHNEY; ARCE, 2003).

2.2 SISTEMAS ASR

Na parte do raciocínio no sistema mostrado na Fig. 2, são estudados os modelos de reconhecimento de fala. Na seleção de um sistema ASR, ou apenas fala para texto, do inglês *Speech to Text* (STT), com o qual pretende-se atingir um objetivo, por exemplo, que o sistema seja independente do locutor, deve-se levar em conta muitos fatores. Tal como no caso dos sistemas treinados, a base de dados utilizado é importante, por exemplo, os áudios que compõem essa base de dados podem ser palavras isoladas ou frases gravadas. Sendo assim, consideramos que “os sistemas de reconhecimento de fala podem ser separados em várias classes diferentes, descrevendo o tipo de expressão da fala, o tipo de modelo do falante, o tipo de canal e o tipo de vocabulário” (PRABHAKAR; SAHU, 2013, p. 1, tradução nosssa).

Em sistemas ASR, um enunciado é a vocalização de uma palavra ou de um conjunto de palavras que possuem um único significado para o computador. Prabhakar e Sahu (2013) classificam os tipos de expressões vocais como: (1) palavras isoladas, (2) palavras conectadas, (3) fala contínua, e (4) fala espontânea.

Além disso, o tamanho do vocabulário de um sistema de reconhecimento de fala afeta a complexidade, os requisitos de processamento e a precisão do sistema. Para os sistemas ASR, segundo Prabhakar e Sahu (2013), os tipos de vocabulários são classificados como na Tabela 1.

Tabela 1 – Tipos de vocabulário

Vocabulário	**Palavras**
Pequeno	Dezenas
Médio	Centenas
Grande	Milhares
Muito grande	Dezenas de milhares
Fora do vocabulário	Palavra desconhecida

Fonte: Adaptado de Prabhakar e Sahu (2013).

Por outro lado, todos os oradores têm suas vozes características que os identificam. Baseados no modelo do locutor, de acordo com Prabhakar e Sahu (2013), os sistemas ASR são classificados em duas categorias principais:

1. **Modelos dependentes do locutor**. Os sistemas dependentes do locutor são projetados para um locutor específico. Esses sistemas geralmente são mais fáceis de desenvolver, mais baratos e mais precisos, mas não tão flexíveis quanto os sistemas adaptativos ou independente de locutores.

2. **Modelos independentes do locutor**. Os sistemas independentes do locutor são projetados para uma variedade de locutores. O modelo reconhece os padrões de fala de um grande grupo de pessoas. Este tipo de modelo é mais difícil de desenvolver, mais caro e oferece menos precisão do que os sistemas dependentes do locutor. No entanto, são mais flexíveis.

Por último, mas não menos importante, sobre os modelos ASR, em vez de enumerar todas as abordagens relacionadas, Zhang et al. (2018) estabelecem uma taxonomia das abordagens mais promissoras, categorizadas por dois princípios: (1) de acordo com o número de canais, que podem ser agrupadas em técnicas de canal único ou multicanal; e (2) de acordo com os tipos de modelos de aprendizagem profunda, podem ser classificadas em técnicas de *front-end*, *back-end* ou conjunta de *front-end* e *back-end*.

2.2.1 Modelos ASR em Tempo Real

No estado de arte, quando modelos em tempo real são objeto de pesquisa, o modelo ASR de palavras isoladas é o alvo principal de estudo. O custo do sistema é baixo e a velocidade de reconhecimento é rápida (ZHI-QIANG et al., 2019).

2.2.1.1 Modelo de Palavras Isoladas

Essencialmente, o modelo consiste em um extrator de características e um classificador de padrões. Para sinais de áudio, o extrator de características mais comum é o processamento de coeficiente cepstral de frequência mel, do inglês *Mel-Frequency Cepstral Coefficients* (MFCC). O processamento do MFCC é baseado na percepção auditiva humana, isto é, a percepção do ouvido humano, do conteúdo de frequência dos sons, não segue uma escala linear (DHINGRA; NIJHAWAN; PANDIT, 2013). Portanto, para cada tom com uma frequência real f, medida em Hz, um tom subjetivo é medido em uma escala chamada escala "mel", cujo valor da frequência é conhecido como frequência mel. Sua escala é constituída por um espaçamento linear de frequência para abaixo de 1000 Hz e um espaçamento logarítmico acima de 1000 Hz.

Na revisão de literatura, Sidiq, Budi e Sa'adah (2015) implementaram um sistema de comando de voz usando dois métodos, chamados métodos de extração de características e métodos de correspondências. Para extração de características é utilizado o método MFCC e como método de correspondência o modelo oculto de Markov, do inglês *Hidden Markov Model* (HMM). Atualmente, o algoritmo DTW, do inglês *Dynamic Time Warping* (DTW), pode ser usado como o algoritmo de reconhecimento de fala eficiente e simples, DTW é um algoritmo usado para medir a similaridade entre duas sequências, que podem variar em tempo ou velocidade. Dhingra, Nijhawan e Pandit (2013) e Mansour, Salh e Mohammed (2015) descrevem a abordagem de reconhecimento de fala usando os métodos MFCC e DTW. Por outro lado, Alex et al. (2018) propõe um sistema de reconhecimento de palavras isoladas, independente do locutor, usando o método de extração de características MFCC e são usadas como classificadores três redes neurais: máquina de vetores de suporte, do inglês *Support Vector Machines* (SVM); a rede neural sigmoide, mais conhecida como perceptron multicamadas (MLP); e a nova rede neural wavelet (WaveNet). O WaveNet utiliza função de ativação de wavelet gaussiana.

Além disso, Curatelli e Mayora-Ibarra (2000) apresentam a comparação de três métodos competitivos de aprendizagem para quantização vetorial de dados de fala de maneira eficiente. Os algoritmos analisados foram dois métodos "batch" (o algoritmo Lloyd generalizado LBG e o método neural gás) e uma técnica "on-line" (algoritmo K-means). Neste caso, os experimentos foram projetados para testes dependentes e independentes do locutor e, a redução do tempo de processamento após o uso dessas técnicas de quantização, levou a aproximar a resposta quase em tempo real.

2.2.1.2 PocketSphinx

O PocketSphinx é um dos vários sistemas de reconhecimento de fala criados na Universidade Carnegie Mellon, do inglês *Carnegie Mellon University* (CMU). De acordo com Dai-

nes (2011), a arquitetura é amplamente herdada do Sphinx-II. Segundo Stenman (2015), o PocketSphinx é o ASR mais rápido que a CMU criou e foi projetado para funcionar em tempo real em plataformas de baixo desempenho, este sistema é dependente do locutor e é construído em modelos de mistura gaussiana, do inglês *Gaussian Mixture Model* (GMM), juntamente com modelos de Markov ocultos (GMM-HMM). Que seja dependente do locutor significa que os resultados de uma tradução podem variar bastante, dependendo do usuário. Por outro lado, o atual sistema de reconhecimento de fala do Google "Google Speech", é independente do locutor e é construído aplicando redes neurais profundas juntamente com modelos Markov ocultos (DNN-HMM). Usando o modelo acústico padrão do inglês americano não treinado e um dicionário não modificado para o PocketSphinx ao avaliar os sistemas, Hjulström (2015) chegou às seguintes conclusões:

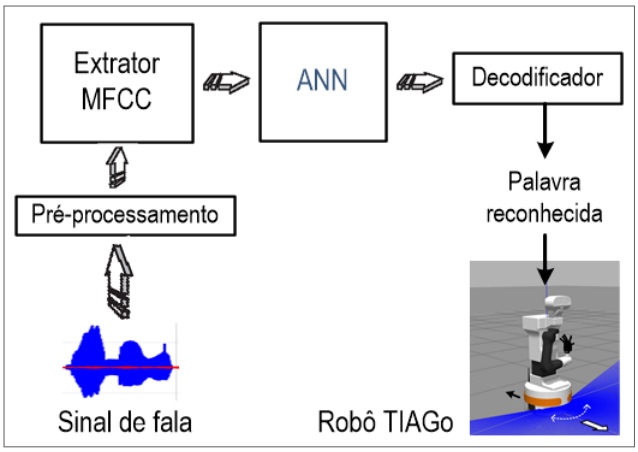
- O Google Speech tem melhor precisão do que o PocketSphinx, quando este não foi treinado.
- O Google Speech lida com o ruído melhor que o PocketSphinx, tanto em relação à precisão quanto ao efeito que o ruído tem no tempo de tradução.
- O PocketSphinx é mais rápido do que o Google Speech.
- O PocketSphinx é mais adequado para controlar um robô, com relação à precisão e ao tempo de tradução.

Voltando à questão dos modelos em tempo real, nos capítulos a seguir, modelos ASR de palavras isoladas são desenvolvidos com ênfase em aprendizagem on-line por meio de redes neurais. Por outro lado, a integração do PocketSphinx ao robô TIAGo é tratada no Apêndice C.

3 MODELOS ASR NEURAL ON-LINE

Neste capítulo é descrito o reconhecimento de fala modelo de palavras isoladas, do inglês *Isolated Word Recognition* (IWR), com ênfase em aprendizagem on-line. O modelo proposto ASR Neural ou ASR *word model* está composta de pré-processamento do sinal áudio, um extrator de características (MFCC), uma rede neural artificial (ANN) e, um decodificador (Fig. 3). A implementação do decodificador é descrita no Capitulo 5.

Figura 3 – Arquitetura ASR *word model* empregando ANN



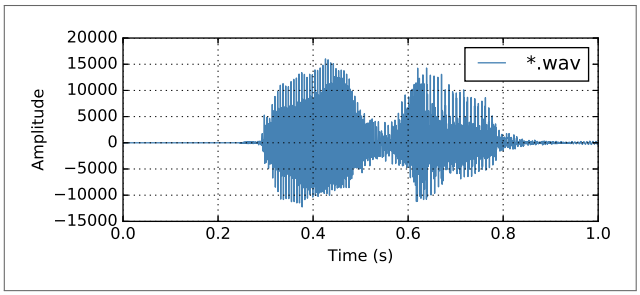
Fonte: Do autor.

No estado de arte, os comandos de voz podem ser reconhecidos através da aplicação de uma rede neural MLP como classificador (ALEX et al., 2018). Por outro lado, propomos um modelo que visa o aprendizado em tempo real, o algoritmo incremental on-line supervisionado GNG de Duque-Belfort, Bassani e Araujo (2017), do inglês *online incremental Supervised Growing Neural Gas* (oiSGNG).

3.1 PRÉ-PROCESSAMENTO

Depois de que o sinal de áudio analógico é convertido em sinal digital, o pré-processamento é o primeiro passo de um sistema de reconhecimento de voz, basicamente consiste de um filtro pré-ênfase, divisão do sinal de áudio em *frames* e a aplicação de funções janelas. Para as ilustrações usaremos as amostras do banco de dados “Speech Commands” de Warden (2018). O sinal no domínio de tempo ilustrada na Fig. 4, é um sinal de áudio bruto (sem processamento), cuja frequência de amostragem é 16 kHz (também conhecida como taxa de amostragem em Hz) e duração 1.0 segundos.

Figura 4 – Sinal no domínio de tempo



Fonte: Adaptado de Fayek (2016).

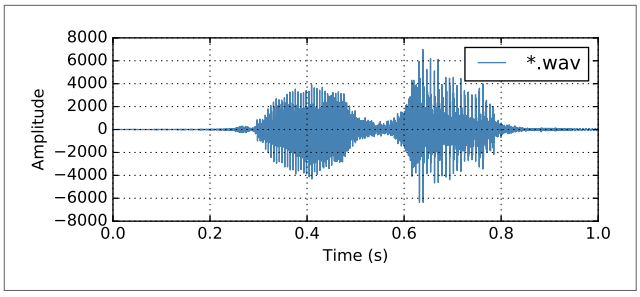
3.1.1 Pré-ênfase

O primeiro passo, de acordo com Fayek (2016), é aplicar um filtro de pré-ênfase no sinal para amplificar as altas frequências. Um filtro de pré-ênfase visa (1) equilibrar o espectro de frequências, pois as altas frequências costumam ter magnitudes menores em comparação às frequências mais baixas; (2) evitar problemas numéricos durante a operação de transformação de Fourier e (3) melhorar o sinal relação ruído. O filtro de pré-ênfase pode ser aplicado a um sinal $x\,[n]$ usando o filtro de primeira ordem.

$$y\,[n] = x\,[n] - \alpha x\,[n-1] \tag{3.1}$$

Os valores típicos do coeficiente de filtro (α) são 0.95 ou 0.97. O sinal no domínio do tempo após a pré-ênfase tem a seguinte forma (Fig. 5):

Figura 5 – Sinal no domínio do tempo após pré-ênfase (α=0.97)



Fonte: Adaptado de Fayek (2016).

3.1.2 Enquadramento

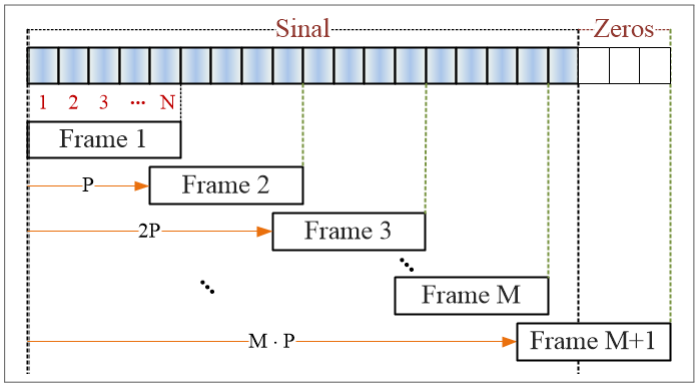
Após a pré-ênfase, aplica-se o enquadramento, do inglês *Framing*, que divide o sinal em segmentos curtos. Tal como explica Fayek (2016), a lógica por trás dessa etapa é que as frequências em um sinal mudam ao longo do tempo; portanto, na maioria dos casos, não faz sentido aplicar a transformada de Fourier para o sinal todo pois se perderiam os contornos no domínio da frequência. Para evitar isso, considera-se que as frequências de um sinal são estacionárias por um curto período de tempo. Portanto, fazendo uma

transformação de Fourier nesse período de tempo pode-se aproximar os contornos de frequência do sinal concatenando os quadros adjacentes.

3.1.3 Sobreposição de quadros

A sobreposição dos quadros (*frames*) visa eliminar a atenuação do início e fim da janela do sinal no cálculo do valor de espectro (BARUA et al., 2014). Na Fig. 6, são mostrados dois casos em que a divisão do sinal de áudio em segmentos e a sobreposição não são do mesmo tamanho que o sinal, isto é: (1) M frames são considerados apenas sem completar o sinal inteiro; e (2) amostras zero são adicionados no final até que o frame M+1 seja concluído.

Figura 6 – Sobreposição do segmentos



Fonte: Do autor.

Segundo Fayek (2016), os tamanhos típicos do *frame* no processamento de fala variam de 20 a 40 ms com sobreposição até 50 % (±10%) entre dois quadros consecutivos. Os parâmetros mais comuns são 25 ms para o tamanho do quadro e sobreposição de 10 ms.

3.1.4 Janela de Hamming

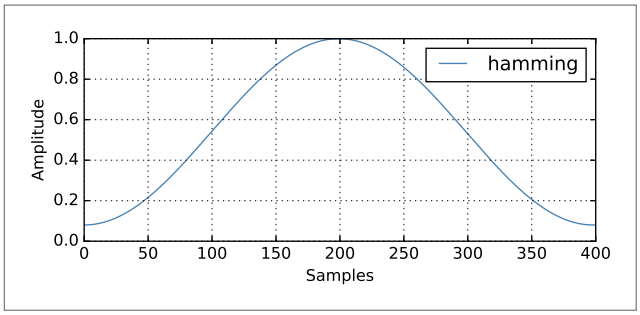
Após dividir o sinal em frames, para minimizar as descontinuidades do sinal, Fayek (2016) aplica uma função de janela a cada frame, especificamente a janela Hamming, afunilando o início e o final de cada quadro para zero. Uma janela é modelada para que seja exatamente zero no início e no final do bloco de dados. Esta função é então multiplicada pelo bloco de dados de tempo, forçando o sinal a ser periódico. A janela de Hamming é usada devido à boa propriedade de resolução de frequência (BARUA et al., 2014). Essa janela de cosseno é definida por:

$$w[n] = 0.54 - 0.46cos(\frac{2\pi n}{N-1}) \tag{3.2}$$

na qual, $0 \leq n \leq N-1$ representa o número das amostras, N é o comprimento da janela, em amostras, da função simétrica de janela discreta no tempo $w[n]$. No caso da frequência de amostragem de 16 kHz para 25 ms de comprimento de frame, as amostras de fala são

divididas em quadros, cada um composto por 400 amostras (Fig. 7). A representação gráfica da equação anterior produz a seguinte plotagem:

Figura 7 – Janela de Hamming



Fonte: Adaptado de Fayek (2016).

3.2 EXTRATOR MFCC

O cepstrum de frequência mel (MFC) é definido como o espectro de potência de intervalo de tempo curto de um som. Os coeficientes cepstrais de frequência mel (MFCCs) são coeficientes que determinam um MFC. Esses coeficientes são calculados a partir de um tipo de representação cepstral de um áudio.

3.2.1 Operadores para Cálculo dos MFCCs

As operações necessárias para determinar os MFCCs são:

1. **Cálculo da Transformada Rápida de Fourier**. O sinal é convertido do domínio do tempo para o domínio da frequência, pela aplicação da Transformada de Fourier (JR; PROAKIS; HANSEN, 1999) e (PROAKIS, 2001). A transformada discreta de Fourier, do inglês *Discrete Fourier Transform* (DFT), de um sinal discreto, composto por N amostras, é definida pela seguinte equação:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi kn}{N}} \qquad k = 0, 1, ..., N-1. \tag{3.3}$$

   Para obter um sinal no domínio da frequência a partir de sinal discreto, o método estudado é a transformada rápida de Fourier (COCHRAN et al., 1967), do inglês *Fast Fourier Transform* (FFT). A FFT é realizada em cada janela do sinal. FFT é uma versão rápida do algoritmo DFT.

2. **Conversão da Frequência para Escala Mel**. Para a audição e percepção humana, o sinal no domínio da frequência é mais apropriada que seja convertido na escala de frequência mel proposta por Stevens, Volkmann e Newman (1937). O ponto de referência entre essa escala e a medição de frequência linear é definido pela

equação que converte um tom de 1000 Hz, 40 dB acima do limite do ouvinte, a um tom de 1000 mels.

A expressão da conversão de frequência f em escala mel é:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{3.4}$$

Para voltar de mels à frequência em Hz.

$$f = 700(10^{m/2595} - 1) \tag{3.5}$$

3. **Cálculo da Transformada Discreta de Cosseno**. O espectro da escala de log mel é convertido no domínio do tempo usando a transformada discreta de cosseno (AHMET; NATARAJAN; RAO, 1974) e (RAO; YIP, 2014), do inglês *Discrete Cosine Transform* (DCT). Para um sinal de N amostras, a DCT é definida pela equação:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \qquad k = 0, 1, ..., N-1. \tag{3.6}$$

Barua et al. (2014) emprega a transformada discreta de cosseno para encontrar o conjunto de coeficientes, os vetores acústicos. Portanto, cada sinal de áudio de entrada é transformado em uma sequência de vetores acústicos.

4. **A Energia de um Frame**. O conteúdo energético de um conjunto de N amostras dentro de um frame, energia do frame (DAS; PAREKH, 2012) e (BALA; KUMAR; BIRLA, 2010), é aproximado pela soma do quadrado das amostras dentro de cada janela:

$$e = \sum_{n=0}^{N-1} x_n^2 \tag{3.7}$$

3.2.2 Coeficiente Cepstral de Frequência Mel (MFCC)

Lyons (2015) e Fayek (2016) apresentaram os passos para implementação do banco de filtros e o extrator de caraterísticas MFCC, cujas etapas estão mais detalhadas no Apêndice A.

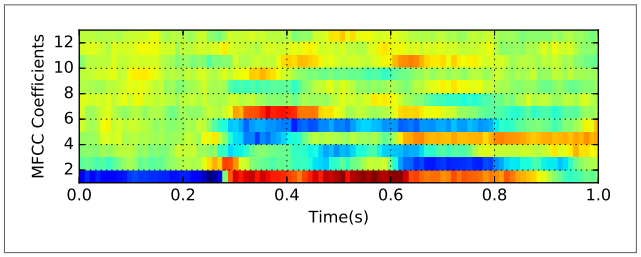
1. Divida o sinal em frames curtos.
2. Para cada frame calcule a estimativa do espectro de potência (*periodogram*).
3. Aplique o banco de filtros mel aos espectros de potência e some a energia em cada filtro.
4. Tome o logaritmo de todas as energias do banco de filtros.

5. Tome o DCT das energias do banco de filtros de log.

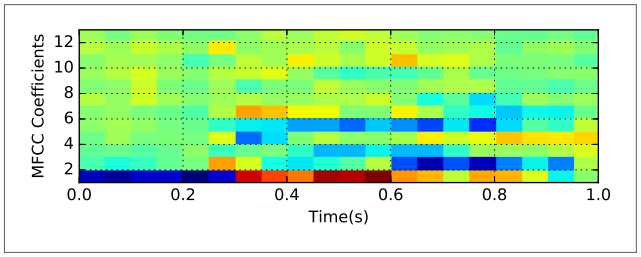
6. Mantenha os coeficientes DCT 2-13, descarte o restante.

Segundo Fayek (2016), os coeficientes dos bancos de filtros são altamente correlacionados, o que pode ser problemático em alguns algoritmos de aprendizagem de máquinas. Portanto, podemos aplicar transformada discreta de cosseno (DCT), para descorrelacionar os coeficientes do banco de filtros e gerar uma representação compactada dos bancos de filtros. Os coeficientes cepstrais resultantes 2-13 são mantidos e o restante é descartado. O descarte se deve às mudanças rápidas que caracterizam os demais coeficientes, característica que não serve como discriminante para o reconhecimento automático de fala. A Figura 8a mostra os coeficientes MFCCs, para o caso dos parâmetros típicos: 25 ms para o tamanho do frame e, sobreposição de 10 ms. Por outro lado, a Figura 8b corresponde ao caso do sinal dividida em 20 frames sem sobreposição. Nos quais, podemos ver que quanto maior o número de frames, mais detalhes são extraídos pelo método extrator MFCC.

Figura 8 – Visualização dos Coeficientes MFCCs



(a) MFCC 98 frames



(b) MFCC 20 frames

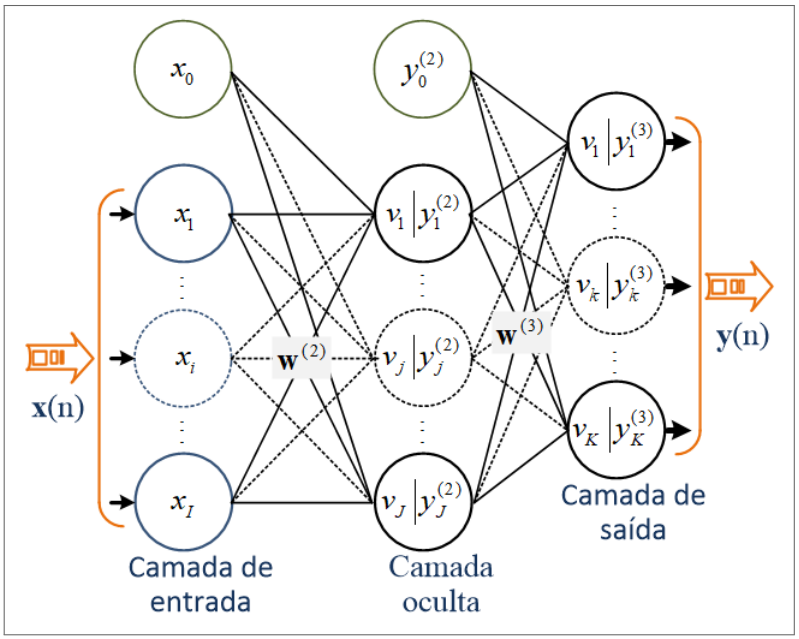
Fonte: Adaptado de Fayek (2016).

No final, além da normalização (SINGH; RANI, 2014), às vezes a energia do quadro é anexada a cada vetor de características acústicas (LYONS, 2015).

3.3 REDE NEURAL MLP

Seja um perceptron de múltiplas camadas, especificamente três camadas; uma camada de entrada, uma camada oculta e uma camada de saída; como ilustrado na Fig. 9.

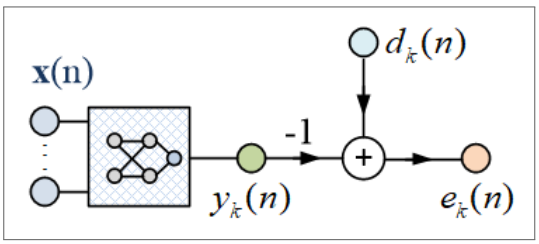
Figura 9 – MLP de três camadas



Fonte: Adaptado de Haykin (2008)

Seguindo a terminologia de Haykin (2008), vamos denotar a amostra de treinamento como $\{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N}$. Na rede MLP, $y_k(n)$ denota o sinal de função produzido na saída do neurônio k na camada de saída pelo estímulo $\mathbf{x}(n)$ aplicado à camada de entrada (ver Fig. 10). O sinal de erro produzido na saída do neurônio k é definido por $e_k(n) = d_k(n) - y_k(n)$, na qual $d_k(n)$ é o k-ésimo elemento do vetor de resposta desejada $\mathbf{d}(n)$.

Figura 10 – Fluxo de sinal destacando o neurônio-k da saída



Fonte: Adaptado de Haykin (2008).

A energia de erro instantânea total de toda a rede é expressa como: $\varepsilon = \frac{1}{2}\sum_{k\in C} e_k^2(n)$. Na qual, o conjunto C inclui todos os neurônios na camada de saída. Desse jeito, a energia de erro média sobre a amostra de treinamento composta por N exemplos é definida por:

$$\varepsilon_{av} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{k\in C} e_k^2(n) \tag{3.8}$$

3.3.1 Métodos de Treinamento

Existem dois métodos clássicos de treinamento supervisionado: *batch learning* e *online learning*. Nesta investigação empregamos o método on-line. No método de aprendizagem on-line, de acordo com Saad (2009) e Haykin (2008), os ajustes nos pesos sinápticos do MLP são realizados exemplo a exemplo, usando o método de gradiente descendente (*gradient descent*). A função de custo a ser minimizada é, portanto, a energia instantânea total do erro (ε).

3.3.2 Algoritmo de Retro-propagação

O algoritmo de retro-propagação de erro, do inglês *back-propagation algorithm*, é um algoritmo utilizado no treinamento de redes neurais multicamadas, e consiste em dois passos de computação: o processamento direto e o processamento reverso. A retro-propagação éo processo que propaga para trás a estimativa de erros por unidade de processamento para ajuste dos parâmetros do modelo (pesos sinápticos e viés). Um ciclo de treinamento através dos N exemplos é chamado de época e as épocas são repetidas até ser alcançado algum critério de parada.

3.3.2.1 Processamento Direto

A propagação direta é empregada para obter a saída e, depois comparar com o valor real para obter o erro. Na rede MLP, os nodos na camada de entrada apenas propagam os dados de entrada. No Algoritmo 1, a ativação é o valor do neurônio após a aplicação de uma função de ativação. Além disso, L_1, L_2 e L_3 incluem todos os neurônios nas camadas entrada, oculta e saída, respectivamente.

Algoritmo 1: Propagação das Entradas

Entrada: $\mathbf{x}(n)$, $\mathbf{w}^{(2)}(n)$, $\mathbf{w}^{(3)}(n)$, $\varphi(\cdot)$
Saída : $\mathbf{y}^{(2)}(n)$, $\mathbf{y}^{(3)}(n)$

for $j \in L_2$ **do**
    Valor do neurônio na camada oculta;
    $v_j(n) \leftarrow \sum_{i \in L_1} w_{j,i}^{(2)}(n) x_i(n)$;
    Valor de ativação na camada oculta;
    $y_j^{(2)}(n) \leftarrow \varphi_j(v_j(n))$;
for $k \in L_3$ **do**
    Valor do neurônio na camada de saída;
    $v_k(n) \leftarrow \sum_{j \in L_2} w_{k,j}^{(3)}(n) y_j^{(2)}(n)$;
    Valor de ativação na camada de saída;
    $y_k^{(3)}(n) \leftarrow \varphi_k(v_k(n))$;

3.3.2.2 Função de Ativação

O algoritmo de retro-propagação do sinal de erro, requer conhecimento da derivada da função de ativação $\varphi(\cdot)$. Em termos básicos, para que essa derivada exista, exigimos que a função $\varphi(\cdot)$ seja contínua. Um exemplo de uma função de ativação não-linear continuamente diferenciável, comumente usada em perceptrons de múltiplas camadas, é a função sigmóide logística. Em sua forma geral, é definida por:

$$y = \varphi(v) = \frac{1}{1 + \exp(-av)} \tag{3.9}$$

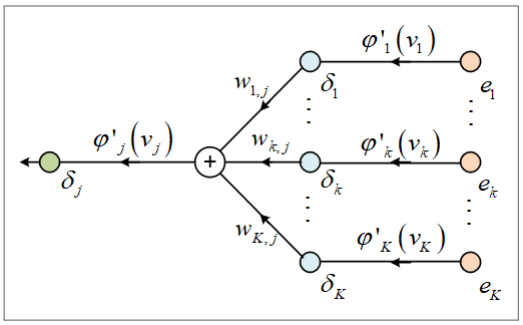
na qual, a é um parâmetro positivo ajustável ($a > 0$). A derivada desta função pode ser expressada como $\varphi'(v) = ay[1 - y]$.

3.3.2.3 Processamento Reverso

A aplicabilidade do aprendizado on-line para o treinamento supervisionado de perceptrons multicamadas, segundo Haykin (2008) foi aumentada pelo desenvolvimento do algoritmo de retro-propagação. O algoritmo de retro-propagação aplica uma correção de peso sinápticos $w_{k,j}$, que é proporcional à derivada parcial, de acordo com a regra da cadeia de cálculo.

Inicialmente, a correção $\Delta w_{k,j}$ aplicada ao nodo de conexão do peso sináptico j (nodo na camada oculta) ao neurônio k (nodo na camada saída) é definido pela gradiente local $\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$ e a regra delta: $\Delta w_{k,j}(n) = \eta\delta_k(n)y_j(n)$. No caso de MLP de três camadas, utilizando a mesma regra, a correção $\Delta w_{j,i}$ aplicada ao nodo de conexão do peso sináptico i (nodo na camada entrada) ao nodo j (nodo na camada oculta) é definido pela gradiente local $\delta_j(n) = \varphi'_j(v_j(n)) \sum_{k \in L_3} \delta_k(n)w_{k,j}(n)$ e a regra delta: $\Delta w_{j,i}(n) = \eta\delta_j(n)x_i(n)$. Assim, o gradiente local $\delta_j(n)$ depende se o neurônio j é um nó de saída ou um nó oculto (ver Fig. 11).

Figura 11 – Fluxo de sinal relativo à retro-propagação de sinais de erro



Fonte: Adaptado de Haykin (2008)

Finalmente, a taxa de aprendizagem η é um hiperparâmetro que controla quanto alterar o modelo em resposta ao erro estimado cada vez que os pesos do modelo são

atualizados. Um valor muito pequeno pode resultar em um longo processo de treinamento, enquanto um valor muito alto pode causar um processo de treinamento instável ou forçar aprendizado de um conjunto subótimo de pesos muito rapidamente.

Algoritmo 2: Retropropagação dos Erros

Entrada: $\mathbf{e}(n)$, $\mathbf{w}^{(2)}(n)$, $\mathbf{w}^{(3)}(n)$, $\mathbf{y}^{(2)}(n)$, $\mathbf{y}^{(3)}(n)$
Saída : $\mathbf{w}^{(2)}(n+1)$, $\mathbf{w}^{(3)}(n+1)$

for $k \in L_3$ **do**
  Produto da derivada $\varphi'_k(v_k(n))$ e o sinal de erro $e_k(n)$;
  $\delta_k(n) \leftarrow a y_k^{(3)}(n)[1 - y_k^{(3)}(n)]e_k(n)$;
  for $j \in L_2$ **do**
    Aplicação da $\Delta w_{k,j}$ ao peso sináptico que liga o neurônio j ao neurônio k;
    $w_{k,j}^{(3)}(n+1) \leftarrow w_{k,j}^{(3)}(n) + \eta\delta_k(n)y_j^{(2)}(n)$;
for $j \in L_2$ **do**
  Produto da derivada associada $\varphi'_j(v_j(n))$ e a soma ponderada de δs calculada para os neurônios na camada de saída;
  $\delta_j(n) \leftarrow a y_j^{(2)}(n)[1 - y_j^{(2)}(n)] \sum_{k \in L_3} \delta_k(n) w_{k,j}^{(3)}(n)$;
  for $i \in L_1$ **do**
    Aplicação da $\Delta w_{j,i}$ ao peso sináptico que conecta o neurônio i ao neurônio j;
    $w_{j,i}^{(2)}(n+1) \leftarrow w_{j,i}^{(2)}(n) + \eta\delta_j(n)x_i(n)$;

3.4 CLASSIFICADOR OISGNG

O método proposto por Fritzke (1995), é diferente dos mapas auto-organizáveis de Kohonen (1982), do inglês *Self-Organized Maps* (SOM), pois o número de unidades é alterado durante o processo de auto-organização. O mecanismo de crescimento das estruturas celulares em crescimento propostas anteriormente (FRITZKE, 1994) e, a geração de topologia da aprendizagem competitiva do Hebbian (MARTINETZ; SCHULTEN et al., 1991), são combinadas em um novo modelo. Começando com poucas unidades, novas unidades são inseridas sucessivamente. Para determinar onde inserir novas unidades, medidas de erro local são coletadas durante o processo de adaptação. Cada nova unidade é inserida próxima à unidade que mais acumulou erros. Ter uma rede com um conjunto crescente de nodos, como o implementado pelo algoritmo GNG, do inglês *Growing Neural Gas* (GNG), foi muito positivo pois o mapa não precisaria ter número máximo de grupos e vizinhança previamente definidas. Partindo da GNG, Prudent e Ennaji (2005) propuseram a Incremental GNG (IGNG) para a qual agrupa-se novas amostras sem degradar a rede já treinada anteriormente. Sun, Liu e Harada (2017) apresentaram um modelo chamado

GNG on-line que aprende de modo não-supervisionado. Enfim, Duque-Belfort, Bassani e Araujo (2017) propõem o algoritmo incremental on-line supervisionado para o GNG (oiSGNG), que visa lidar com grandes quantidades de dados, a rede é inicializada com zero nodos e possui mecanismo de inserção de nodos.

3.4.1 Algoritmo oiSGNG Modificado

Dado um conjunto de instancias e os seus rótulos $\{\mathbf{x}(n), y(n)\}_{n=1}^{N}$, o algoritmo oiSGNG gera um conjunto de nodos inseridos, os quais têm os seguintes atributos: identificador, vetor de peso, lista de conexões, lista de idade de conexões, lista de erros e um único rótulo. O diferencial do algoritmo oiSGNG modificado é o mecanismo de inserção de nodos. A Tabela 2 mostra as notações empregadas neste algoritmo.

Tabela 2 – Notações do algoritmo

$\mathbf{x}$, y	Instância de dado atual e o seu rótulo.
α	Taxa de aprendizagem do vencedor.
β	Taxa de aprendizagem do vizinho.
η	Taxa de decaimento de erros.
g	Idade máxima da conexão.

Fonte: Adaptado de Duque-Belfort, Bassani e Araujo (2017)

Além dos parâmetros apresentados acima, existe um contador de iteração c, que é uma variável interna que controla o número de instâncias de dados. Para cada instância de dados $\mathbf{x}$ e seu rótulo y, a operação normal do oiSGNG pode ser vista no Algoritmo 3 e, o seu funcionamento é descrito nos itens a seguir.

1. **Inicialização**. A rede é inicializada com zero nodos, quando a primeira instância de dado $\mathbf{x}_0$ é apresentada, o primeiro nodo n_0 é inserido, com peso $\mathbf{w}_0$ igual a $\mathbf{x}_0$, e rótulo igual ao rótulo de $\mathbf{x}_0$ (y_0). Depois, quando a segunda instância $\mathbf{x}_1$ é apresentada, outro nodo é inserido de forma semelhante. Os dois nodos se tornam vizinhos e a idade de sua conexão é zerada (DUQUE-BELFORT; BASSANI; ARAUJO, 2017).

2. **Inserção de novos rótulos**. Quando um dado $\mathbf{x}$ que pertence a uma nova categoria é apresentado, um novo nodo n_i é inserido, com $\mathbf{w}_i = \mathbf{x}_i$ e rotulo igual a y_i . Finalmente, n_i é conectado ao nodo com o vetor de peso mais próximo de $\mathbf{x}$, e a idade de sua conexão é zerada (DUQUE-BELFORT; BASSANI; ARAUJO, 2017).

3. **Inserção de nodos**. No algoritmo oiSGNG modificado, a nova característica é a inserção de nodos variante no tempo, definida pelo parâmetro $\phi(t) \in Z$, na qual, t é o número de época ($t = 0, 1, ..., T$). Assim, facilitando o controle dos nodos inseridos

pelo o número da época. Considerando que se pretende ter maior refinamento nas ultimas épocas, propomos nas épocas finais inserir quantidades de nodos muito menores do que no início. Especificamente, para o parâmetro de inserção de nodos sugerimos a parte inteira do valor gerado pela função exponencial, na qual, a variável t está no expoente e cuja base b é uma constante sempre maior que zero e diferente de um, ou seja:

$$\phi(t) = \max\{s \in Z \mid s \leq \phi_0 b^t\} \tag{3.10}$$

na qual, ϕ_0 é o parâmetro de inserção em $t = 0$.

Algoritmo 3: Algoritmo oiSGNG Modificado

Entrada: $\mathbf{x}$, y, α, β, η, g, c, $\phi(t)$
Saída : n_{novo}

1 Encontre o primeiro e o segundo nodo mais próximo de $\mathbf{x}$, s_0 e s_1 e, o nodo mais próximo de $\mathbf{x}$ que pertença a y, s_2 ;
2 Se s_0 e s_1 não estiverem conectados, conecte-os e defina a idade da conexão para zero. Se estiverem conectados, defina a idade da conexão como zero ;
3 Incremente a idade da conexão de todos os vizinhos de s_0 ;
4 Remova todas as conexões com idade maior do que g ;
5 Remova todos os nodos desconectados se eles não forem o último representante do seu rótulo ;
6 Se s_0 não pertence a y, aumente a matriz de erros de s_0 em relação a y com a distância euclidiana entre $\mathbf{w}_{s0}$ e $\mathbf{x}$;
7 Atualização dos pesos dos vizinhos de s_2, $\mathbf{w}_{s2}$ e $\mathbf{w}_n$:
   a: $\mathbf{w}_{s2} \leftarrow \mathbf{w}_{s2} + \alpha(\mathbf{x} - \mathbf{w}_{s2})$
   b: Se o vizinho pertence a y: $\mathbf{w}_n \leftarrow \mathbf{w}_n + \beta(x - \mathbf{w}_n)$
   c: Se o vizinho não pertencer a y: $\mathbf{w}_n \leftarrow \mathbf{w}_n - \beta(\mathbf{x} - \mathbf{w}_n)$;
8 Se c é múltiplo de $\phi(t)$:
   a: Procure o nodo n_{erro} com o maior erro relativo a qualquer rótulo
   b: Insira um novo nodo n_{novo} com vetor de peso igual ao de n_{erro} e defina seu rótulo igual ao que atingiu o valor mais alto de erro em n_{erro}
   c: Conecte n_{erro} a n_{novo}
   d: Reduza todos os vetores de erro da rede multiplicando-os por $0 < \eta < 1$;
9 Incremente c ;

3.4.2 Intervalo de inserção de nodos

Seguindo o método de aprendizagem on-line, o conjunto de dados é apresentada padrão por padrão, porém, para evitar problemas de *underfitting* (ou seja, não aprender tudo o que pudermos a partir dos dados de treinamento), uma época é definida como um ciclo da apresentação de todos os padrões de treinamento, repetida duas vezes aleatoriamente,

quer dizer, que são apresentadas $2N$ padrões por época. Portanto, os nodos inseridos por época podem ser determinados pela seguinte expressão:

$$N_i(t) = \max\{q \in Z \mid q \leq \frac{2N}{\phi(t)}\} \tag{3.11}$$

Para determinar o parâmetro b, usaremos as funções sem arredondar (Tabela 3), ou seja, a quantidade dos nodos inseridos por época pode ser aproximado como: $N_i(t) = \frac{N_0}{b^t}$.

Tabela 3 – O número total de nodos inseridos

t	$\sum_{r=0}^{t} N_i(r)$
0	N_0
1	$N_0 + \frac{N_0}{b}$
2	$N_0 + \frac{N_0}{b} + \frac{N_0}{b^2}$
$\vdots$	$\vdots$
T	$N_0(1 + \frac{1}{b} + \frac{1}{b^2} + \cdots + \frac{1}{b^T})$

Fonte: Do autor.

Denotando $\lambda = \frac{1}{b}$, empregando propriedades dos quocientes notáveis (fórmula da soma dos primeiros $T+1$ termos de uma série geométrica) e considerando um grande número de épocas ($T \to \infty$), a quantidade total de nodos inseridos N_f pode ser expressa como:

$$N_f = N_0 \lim_{T \to +\infty} \frac{1 - \lambda^{T+1}}{1 - \lambda} = \frac{N_0}{1 - \lambda} \tag{3.12}$$

na qual, $|\lambda| < 1$. Dessa forma, o parâmetro b é um constante positivo e maior que um ($b > 1$), e que pode ser obtida pela equação:

$$b = \frac{N_f}{N_f - N_0} \tag{3.13}$$

Nos mapas auto-organizáveis (Apêndice B), de acordo com Vesanto et al. (1999), o número de unidades só depende da quantidade total de padrões do conjunto de treinamento (dada por $5\sqrt{N}$). Portanto, considerando que o modelo oiSGNG pode ser implementado com quantidades menores para ter menor latência, propomos que os nodos inseridos estejam no intervalo $\sqrt{N} < N_i < 2\sqrt{N}$. Isso implica que os parâmetros da função de inserção exponencial sejam: $\phi_0 = 2\sqrt{N}$ e $b = 2$. Na pratica, além do considerar o arredondamento das funções, ao número total de nodos inseridos N_f deve ser adicionado os nodos inseridos na inicialização e diminuir os nodos removidos com idade maior do que g.

4 RESULTADOS ASR NEURAL ON-LINE

Este capítulo trata da avaliação dos modelos ASR Neural. Neste caso, para o treinamento e a seleção dos modelos é usado o banco de dados "Speech Commands" de Warden (2018), no qual, todos os comandos estão em inglês. Portanto, seis comandos são propostos para dirigir o robô, os quais são agrupados em três grupos A, B e C.

$$\underbrace{go, \underbrace{backward, \underbrace{forward, left, right, stop}_{\text{Grupo A}}}_{\text{Grupo B}}}_{\text{Grupo C}}$$

Inicialmente, o Grupo C é usada para treinamento dos modelos ASR Neural, esta base de dados é formada por categorias desbalanceadas, assim o conjunto de dados é equilibrado com repetição aleatória de amostras sem perturbação e com adição de ruído branco e ruído colorido (rosa, vermelho, azul e violeta). Além disso, os modelos são avaliados pela adição do ruído de ventilador e ruído balbuciante.

Por outro lado, neste capitulo apresentamos os resultados do modelo MFCC-oiSGNG, na qual se destaca o tratamento de dados com alta dimensionalidade, ou seja, para os resultados considera-se o sinal dividida em 20 frames sem sobreposição e dividida em 98 frames com sobreposição. Além disso, em termos da taxa de acerto e os nodos inseridos, o algoritmo oiSGNG modificado é comparado ao original. Por outro lado, de acordo aos resultados, podemos notar que não é necessário alterar os parâmetros para a aprendizagem on-line de quatro, cinco ou seis categorias de comandos de voz.

4.1 ADIÇÃO DE RUÍDO

Para realizar os experimentos da adição de ruído, é utilizada a base de dados "Speech Commands", e os ruídos adicionados artificialmente são: ruído branco, ruído colorido (rosa, vermelho, azul e violeta), ruído do ventilador, e ruído balbuciante. Com a ferramenta[1] desenvolvida em Matlab, é possível configurar o nível do ruído SNR (ver Fig. 12). Para o treinamento são utilizadas o ruído branco e o ruído colorido, mas nos testes são empregados o ruído do ventilador e o ruído balbuciante.

Para adição do ruído a uma base de dados de áudio (por exemplo, um conjunto de áudios .wav), são necessários três passos: (l) ler o arquivo (*audioread*), (2) processamento do sinal digital, e (3) gravar o arquivo (*audiowrite*). As ferramentas "audioread" e "audiowrite" precisam conhecer o caminho do arquivo de áudio (*filename*). Então, para o tratamento do base de dados, no caso quando não há lista dos arquivos ".wav", exemplo

[1] https://github.com/egutierrezhu/noiseAdder.git

a base de dados “Speech Commands”, pode-se usar a ferramenta “List= dir(path)” para gerar a lista do conjunto de áudios contidos numa pasta, “path” é o caminho dessa pasta.

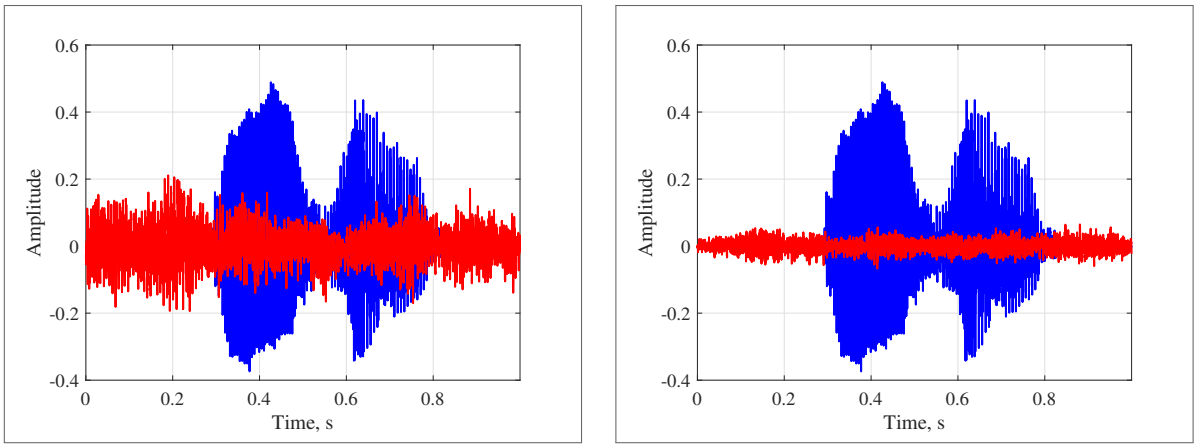
- **Adição do Ruído Branco**. Para o ruído branco, foi empregada a função básica do ruído branco do Matlab, função awgn(X, SNR), essa função adiciona ruído gaussiano branco ao sinal X, no qual a relação sinal-ruído SNR está em dB e a potência de X é assumida como 0 dB. A função correspondente para outros valores de potência é awgn (X, SNR, SIGPOWER), no qual SIGPOWER é o nível da potência do sinal em dB.

- **Adição do Ruído Colorido**. O método para geração de ruído colorido trata da geração de sinais de ruído digital coloridos, com a lei de potência, e a inclinação espectral arbitrária. O método de Zhivomirov (2018) é baseado em geração de um sinal de ruído branco, sua transformação no domínio da frequência, processamento espectral e transformação inversa no domínio do tempo. Para confirmar a consistência do algoritmo, simulações de computador foram realizadas, e a comparação com a função Matlab correspondente. O espectro é proporcional à frequência por um fator de $1/f$, $1/f^2$, f, e f^2; isto é, as amplitudes são proporcionais a $1/\sqrt{f}$, $1/f$, $\sqrt{f}$ e f; para rosa, vermelho, azul e violeta, respectivamente.

- **Adição do ruído de ventilador**. Para o caso de ambientes internos, é muito comum encontrar um ventilador. A fonte de ruídos do ventilador inclui ruído aerodinâmico, além do som operacional do próprio ventilador, que pode ser similar ao ruído gerado pelo ar condicionado. O ruído dinâmico é complexo e difícil de expressar matematicamente (CRIGHTON, 1975). Neste caso, gravou-se o som de um ventilador real, configurar à mesma taxa de amostragem (16 kHz) que a base de dados, e realizar a adição do sinal digital.

- **Adição do ruído balbuciante**. Interpolação é o processo de usar pontos com valores conhecidos para estimar valores em outros pontos desconhecidos. Neste caso, interpolação é necessário usá-lo para prever valores desconhecidos no sinal de ruído balbuciante. Os sinais de áudio digital do banco de dados “Speech Commands” estão na frequência de amostragem de 16 kHz e o áudio gravado do ruído de balbuciante está na frequência de amostragem de 8 kHz (HABETS; COHEN; GANNOT, 2008). Portanto, para que seja possível adicionar os dois sinais digitais, o sinal de ruído balbuciante deve ser interpolado um ponto entre duas amostras adjacentes. Neste caso, utilizou-se um segmento do sinal de ruído de balbucio, o tamanho de esta é maior do que o tamanho do sinal limpa, assim, podemos selecionar aleatoriamente o ponto inicial para interpolar.

Figura 12 – Relação sinal-ruído



(a) SNR 5 dB (b) SNR 15 dB

Fonte: Do autor.

4.2 BASES DE DADOS "SPEECH COMMANDS"

A base de dados para sistemas operados por comandos de voz, é um conjunto de palavras gravadas de vocabulário limitado (WARDEN, 2018), que foi projetada para treinar e avaliar os sistemas de localização através de palavras-chave. O conjunto final de dados é formado por 105.829 áudios gravados de 35 palavras-chave, 2.618 locutores, armazenado como um arquivo no formato ".WAV". Os áudios duram no máximo um segundo, com os dados de amostra codificados canais simples PCM de 16 bits, a uma taxa de 16 kHz. Os arquivos não compactados ocupam aproximadamente 3,8 GB no disco e podem ser armazenados como um arquivo "tar" compactado em "gzip" de 2,7 GB. O Grupo C é o conjunto de comandos de voz que serão usados em todos os casos; portanto, é apresentado o caso de classificação de dados desequilibrada (ver Tabela 4).

Tabela 4 – A base de dados de comandos de voz para navegação assistida

Comando	Conjunto de dados		
	Treinamento	Validação	Teste
Backward	1346	153	165
Forward	1256	146	155
Go	3106	372	402
Left	3037	352	412
Right	3019	363	396
Stop	3111	350	411

Fonte: Do autor.

Nessa situação, o treinamento com conjuntos de dados desequilibrados tende a prejudicar o desempenho dos algoritmos de aprendizado global como o MLP. Funções objetivo

usadas para aprender os classificadores geralmente tendem favorecer as classes maiores. Para tratamento do problema de distribuição desbalanceada, existem vários métodos, o mais simples é replicar dados, porém, podemos ter o problema de *overfitting* (aprender demais, ou seja, se ajusta muito bem ao conjunto de dados anteriormente observado, mas se mostra ineficaz para prever novos resultados). Se for possível, adicionar novos padrões. Mani e Zhang (2003) tratam o problema através da aplicação de uma abordagem de k vizinhos mais próximos, do inglês *K-Nearest Neighbors* (kNN). Cieslak e Chawla (2008) propõem a aplicação de árvores de decisão. E na abordagem proposta por Farquad e Bose (2012), primeiro emprega-se o SVM como pré-processador e os valores-alvo reais dos dados de treinamento são substituídos pelas previsões do SVM treinado.

À vista disso, propomos a réplica dos padrões no conjunto de treinamento adicionando o ruído branco e o ruído colorido (azul, rosa, vermelho e violeta). Para todos os casos, o nível de ruído está no intervalo de 5 dB a 35 dB e, os padrões modificados foram selecionados aleatoriamente. Além do mais, as quantidades de padrões adicionados com ruído foram feitas do jeito que as amostras por categoria são ao total 3600. Primeiro, para as categorias "backward" e "forward" foram replicados todos os dados adicionando o ruído colorido. Depois, para todas as categorias, 392 padrões também são replicados e adicionando o ruído colorido, quer dizer, 98 padrões para cada cor. Finalmente, todas as categorias são completadas com padrões replicados com a adição do ruído branco.

4.3 RESULTADOS DA REDE NEURAL MLP

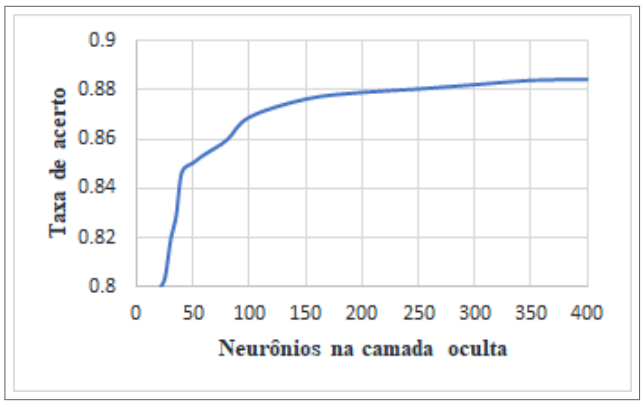
Para treinamento, validação e teste da rede neural MLP, é usada a base de dados dos comandos para navegação assistida Grupo C, cada sinal de áudio é dividido em 20 frames sem sobreposição. Especificamente, o tamanho das janelas e a separação destas janelas, tem o valor da duração do sinal de áudio dividida entre 20, ou seja, 50 ms. Além disso, são empregados 13 coeficientes ceptstrais, o coeficiente cepstral zero é substituído pelo logaritmo da energia total do frame. Assim, no processamento MFCC é gerada 20 frames, cada um com 13 coeficientes ceptstrais, ao todo 260 características, que são as entradas da rede neural MLP.

4.3.1 Parâmetros da Rede Neural MLP

Os pesos sinápticos foram inicializados com valores aleatórios variando de -0.1 a 0.1, os valores máximos e mínimos na saída dos neurônios são: $v_{min} = -16.6$ e $v_{max} = 13.9$. Por outro lado, estas saídas constituem as entradas para a função sigmoide. Para garantir linearidade e maior rapidez de treinamento, o parâmetro da função logística ficou em $a = 1$. Para os resultados que se apresentam nesta seção, a taxa de aprendizagem é $\eta = 0.4$. Na rede neural MLP: a camada da entrada tem 260 unidades, que corresponde à

dimensão do vetor de características do sinal de áudio; a camada saída tem 6 neurônios, que corresponde aos comandos de voz; o número de neurônios da camada escondida é determinado experimentalmente (ver Fig. 13). A melhor taxa de acerto é no caso de 400 neurônios, que ainda poderia melhorar ao aumentar as unidades da camada oculta.

Figura 13 – Nº de neurônios na camada oculta e taxa de acerto



Fonte: Do autor.

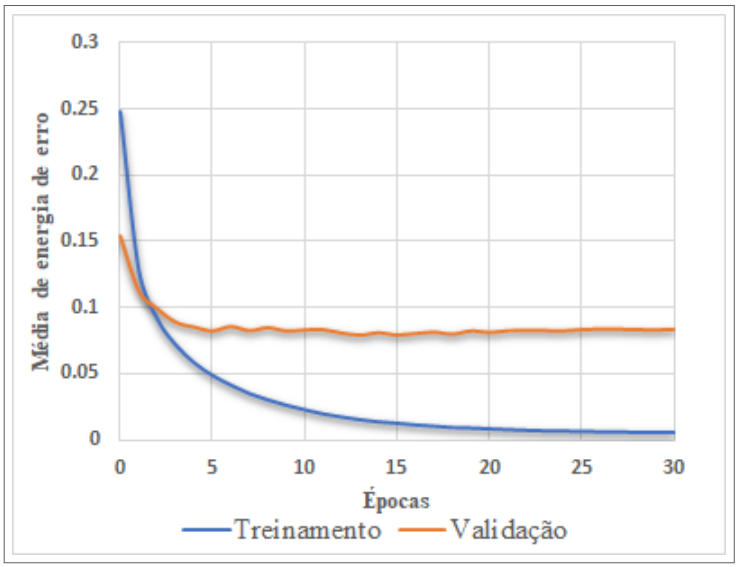
A Fig. 13, evidenciou que os valores de taxa de acerto para os casos de 200 a 400 neurônios são muito próximos, portanto, considerando que o tempo de treinamento é menor quanto o número de neurônios é menor, o modelo selecionado é o modelo com 200 neurônios na camada oculta. Assim, a rede MLP é expressa como (260,200,6).

4.3.2 Treinamento e Validação

O alvo do algoritmo de retro-propagação, segundo Haykin (2008), é realizar um mapeamento de entrada-saída nos pesos e limiares sinápticos de um multiperceptron de várias camadas. A esperança é que a rede se torne bem treinada para aprender o suficiente sobre o passado para generalizar para o futuro. O critério da parada do treinamento pode ser pelo erro mínimo, número de épocas ou validação.

Nesta dissertação, a base de dados de Warden (2018) é dividida em três conjuntos: treinamento, validação e teste, divisão preservada para todos os testes. Assim, o desempenho da generalização do modelo candidato é medido através do conjunto de validação. Para avaliar o desempenho dos modelos candidatos e a análise estadística, num intervalo de 30 épocas ($t = 0, 1, .., 30$), levou-se em conta o critério de parada pelo o máximo número de épocas e o erro mínimo da validação (ver Fig.14). O erro de classificação diminui ao aumentar o número de épocas, mas o erro de validação tem um mínimo na alguma época, e depois aumenta ao passar as épocas.

Figura 14 – Erro versus Número de épocas



Fonte: Do autor.

4.3.3 Análise dos Resultados do Classificador MLP

Nos experimentos com o conjunto de dados Grupo C, apresentamos três casos (Tabela 5): primeiro caso, treinamento da base de dados desequilibrado; em segundo caso, o conjunto de dados foi equilibrado pela inclusão de padrões repetidos sem adicionar ruído; terceiro caso, para equilibrar o conjunto de treinamento, os mesmos padrões do segundo caso foram usados em suas versões ruidosas. Em todos os casos, com os parâmetros mencionados anteriormente e 10 modelos MLP treinados para cada caso, apresentamos a taxa de acerto média e o desvio padrão para cada caso.

Tabela 5 – Taxa de acerto com adição do ruído

Comando	Taxa de acerto		
	Desequilibrado	Equilibrado	Equilibrado+Ruído
Backward	0.735	0.736	0.748
Forward	0.822	0.830	0.844
Go	0.879	0.883	0.894
Left	0.875	0.892	0.896
Right	0.889	0.879	0.886
Stop	0.939	0.929	0.940
Total % (σ)	87.6 (0.3)	87.7 (0.3)	88.6 (0.5)

Fonte: Do autor.

Para o conjunto de dados V2 que a base de dados utilizada nesta investigação, Warden (2018) menciona um ASR implementado com *Tensorflow*, que é uma biblioteca de código aberto para aprendizagem de máquina aplicável a uma ampla variedade de tarefas, cujo

resultado é 88,2%, quando é totalmente treinado. De acordo com o autor, espera-se que isso sirva como uma linha de base a ser excedida por arquiteturas mais sofisticadas.

Os testes de hipótese constituem uma forma de inferência estatística, que é realizado com o objetivo de rejeitar ou não uma hipótese nula H_0. Hipóteses são afirmações sobre parâmetros populacionais e são testadas para ver se são consideradas verdadeiras ou não. Os testes mais conhecidos que lidam com essas questões são o teste t de *student* (comparação de médias) e o teste F (comparação de variâncias). Ambos supõem que os dados tem distribuição Normal. Com respeito aos resultados da Tabela 5, nos casos "Desequilibrado" e "Equilibrado", apenas replica de padrões, estatisticamente, não há diferença significante entre as médias. Sob outra perspectiva, seja a hipóteses nula H_0: a adição do ruído NÃO FAZ EFEITO. Comparação de taxa de acerto nos casos "Desequilibrado" e "Equilibrado+Ruído", replica de padrões com adição de ruído.

- Teste F (0.21) é maior que 0.05. Não rejeita H_0. Variâncias supostamente iguais.
- Teste t de *student* (0.000055) é menor que 0.05. Rejeita H_0. Existe diferença significativa entre as médias. Portanto, o MLP treinado com base em dados com adição de ruído tem uma taxa de acerto mais alta do que o treinado sem adição de ruído.

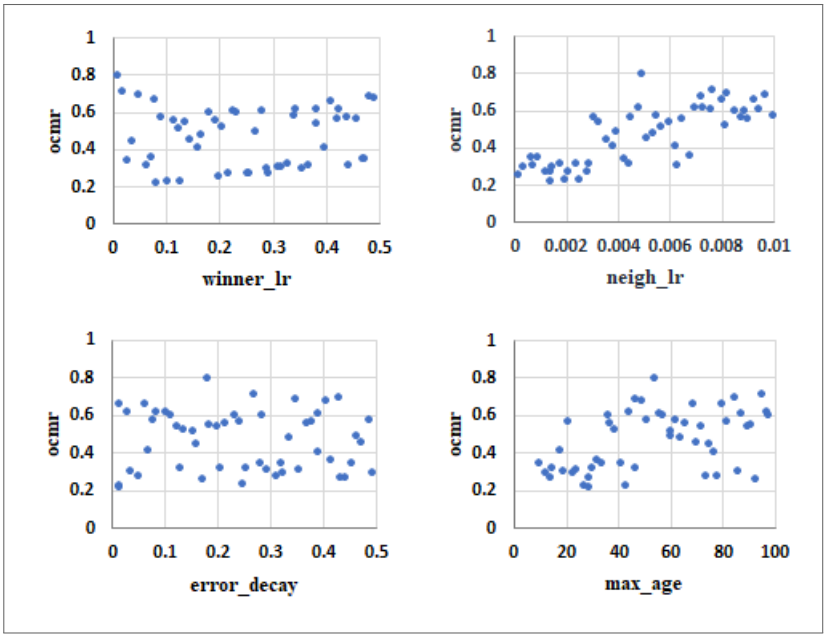
4.4 RESULTADOS DO ALGORITMO OISGNG+

Nos resultados a seguir, o algoritmo oiSGNG modificado é nomeado oiSGNG+. Sendo assim, a base de dados utilizada para determinar os parâmetros do algoritmo oiSGNG+ é o Grupo C, conjunto do treinamento e teste, equilibrada com adição de ruído. Nos algoritmos on-line, a medida de desempenho, segundo Duque-Belfort, Bassani e Araujo (2017) é a taxa de erros cumulativos on-line, do inglês *Online Cumulative Mistake Rate* (OCMR), que é obtida pela equação: $OCMR(q) = \frac{1}{q}\sum_{i=0}^{q} 1|_{\hat{y}_i \neq y_i}$.

4.4.1 Parâmetros do oiSGNG+

Os parâmetros α, β, η e g são ajustados usando o método de amostragem de hipercubo latino, do inglês *Latin Hypercube Sampling* (LHS), que gera amostras quase aleatórias a partir de uma distribuição multidimensional. Especificamente, considerando limites inferiores e superiores, o LHS é implementado com base na ferramenta *lhsdesign(n,p)* de Matlab. Na qual, n é o número de amostras e p é a dimensão ($p = 4$). O conjunto de parâmetros gerados pelo método LHS é avaliado apenas na época $t = 0$, e sua dependência com o OCMR é ilustrada na Figura 15.

Figura 15 – Dependência do OCMR com os parâmetros do oiSGNG



Fonte: Adaptado de Duque-Belfort, Bassani e Araujo (2017)

Dessa maneira, os parâmetros com menor OCMR gerados pelo método LHS são mostrados na Tabela 6. Os quais são utilizados ao longo desta dissertação. Além desses quatro parâmetros, de acordo aos cálculos realizados no Capitulo 3, a função de inserção de nodos ϕ pode ser expressa como $\phi(N,t) = 2\sqrt{N}2^t$, ou seja, depende do número de épocas e do número de amostras no conjunto de treinamento.

Tabela 6 – Parâmetros do algoritmo oiSGNG

Parâmetro	Acrónimo	Resultado do LHS
α	winer_lr	0.0817
β	neigh_lr	0.001384
η	error_decay	0.0141
g	max_age	29

Fonte: Do autor.

4.4.2 Algoritmo oiSGNG vs oiSGNG+

Consideramos o máximo número de épocas $T = 30$ e o sinal de áudio dividida em 20 e 98 frames para todos os casos, na Tabela 7 apresentamos os resultados com intervalos de confiança de 95% para a taxa de acerto e os nodos inseridos, para os comandos de voz dos Grupos A, B e C; de quatro, cinco e seis categorias, respectivamente. Os quais estão em sua versão desequilibra, ou seja, sem replica de dados nem adição de ruído. Neste caso, a

média amostral (μ) é estimada como: $\mu = \bar{x} \pm 1.96 \times \frac{s}{\sqrt{n}}$. Na qual, $\bar{x}$ é a média observada, s é o desvio padrão calculada e n é o tamanho amostral ($n = 6$).

Tabela 7 – Taxa de acerto (%) e nodos inseridos

Comandos	Nodos inseridos		Taxa de acerto	
	oiSGNG	oiSGNG+	oiSGNG	oiSGNG+
Grupo A	263.8±0.3	190.3±3.1	83.3±0.4	90.8±0.5
Grupo B	298.0±1.0	191.7±3.4	80.6±0.9	88.6±0.4
Grupo C	375.5±1.2	213.8±5.6	76.2±1.2	84.0±0.4

Fonte: Do autor.

Desse jeito, podemos observar que os intervalos de taxa de acerto não se sobrepõem, então temos (pelo menos) 95% de confiança de que as verdadeiras médias não são iguais. O caso com o melhor desempenho são os resultados do Grupo C, a taxa de acertos aumenta em 10.2% e os nodos inseridos diminuem em 43.1%, portanto, temos uma taxa de acertos mais alta com menos nodos, o que significa menor latência em processamento do sinal.

A melhoria na precisão se deve ao número de épocas de treinamento ($t = 0, 1, ..., 30$), que, no caso do algoritmo oiSGNG original, apenas uma época é treinada ($t = 0$). Por outro lado, no algoritmo oiSGNG modificado, evidenciou-se que o número de nodos inseridos está no intervalo $\sqrt{N} < N_i < 2\sqrt{N}$, ou seja, não depende do número de categorias. Esse número de nodos depende do parâmetro ϕ, que varia no tempo e também depende do número de padrões treinados. No caso do algoritmo não modificado, o parâmetro de inserção é fixo $\phi_0 = 80$.

4.4.3 Analise da Divisão de Sinal em Frames

Em geral, mapas auto-organizados e redes neurais GNG podem ser treinados com dados de alta dimensão. Nesta seção, o algoritmo oiSGNG+ é treinado para o sinal dividido em 20 e 98 frames (Tabela 8), usando comandos de voz do Grupo C, em sua versão balanceada com adição de ruído.

Tabela 8 – Taxa de acerto (%) e nodos inseridos para diferentes frames

Parâmetros	20 Frames	98 Frames
Nodos inseridos (σ)	258.6 (10.5)	253.4 (7.6)
Taxa de acerto (σ)	83.5 (0.2)	84.7 (0.3)

Fonte: Do autor.

Finalmente, para a análise dos resultados é realizada o teste t de *student* para a hipótese nula H_0: O sinal dividido em 98 frames NÃO FAZ EFEITO. Comparação de

taxa de acertos do algoritmo oiSGNG+ para o sinal de áudio dividido em 20 frames sem sobreposição e 98 frames com sobreposição.

- Teste F (0.59) é maior que 0.05. Não rejeita H_0. Variâncias supostamente iguais.
- Teste t de *student* (0.000034) é menor que 0.05. Rejeita H_0. Existe diferença significativa entre as médias. Portanto, a rede neural oiSGNG (algoritmo modificado) treinado com 98 frames tem maior taxa de acertos que treinado com 20 frames.

4.5 TESTE DOS MODELOS ASR NEURAIS EM RUÍDO

Por último, com respeito ao despenho dos modelos ASR Neurais em ambientes com ruído acústico, na Tabela 9, os modelos ANNs, cujas taxas de acertos são as melhores nos testes sem ruído, são avaliados com adição de ruído no conjunto de teste, para o qual considera-se o ruído de ventilador e ruído balbuciante.

Neste caso, de acordo ao Teorema Central do Limite para uma distribuição Normal, a taxa de acertos com intervalo de confiança de 95% pode ser expressa como uma proporção amostral: $p = \hat{p} \pm 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$. Na qual, p é a verdadeira proporção e $\hat{p}$ é a proporção observada para n amostras do conjunto teste ($n = 1941$).

Tabela 9 – Avaliação das ANNs para ambientes ruidosos

Ruído	SNR	Taxa de acertos	
		MLP	oiSGNG+
Ventilador	0-10dB	67.9±2.1	62.0±2.2
	11-20dB	**84.0±1.6**	79.0±1.8
	21-30dB	**89.3±1.4**	**85.2±1.6**
Balbuciante	0-10dB	49.9±2.2	29.8±2.0
	11-20dB	72.5±2.0	48.1±2.2
	21-30dB	**82.3±1.7**	67.9±2.1
Sem ruído		**89.3±1.4**	**85.2±1.6**

Fonte: Do autor.

De acordo aos resultados da Tabela 9, considerando os modelos com taxa de acertos maior a 80%, no caso do MLP, o modelo ASR Neural é recomendável implementar em ambientes com ruído do ventilador de até 10 dB, ou seja, SNR maior que 10 dB e, com ruído de balbuciante de até 20 dB, ou seja, SNR maior que 20 dB. Por outro lado, no caso do oiSGNG+, o modelo ASR Neural é recomendável implementar apenas em ambientes com ruído de ventilador de até 20 dB, ou seja, SNR maior que 20 dB.

5 SIMULAÇÕES E EXPERIMENTOS

Este capítulo apresenta a integração do reconhecimento de fala em tempo real para robôs moveis, e sua aplicação para tarefas de navegação. Os experimentos, que são basicamente o movimento da base do robô através dos comandos de velocidade, foram realizados primeiro no simulador Gazebo e depois com o robô real, para o qual propomos empregar uma palavra-chave para evitar interferências do ruído acústico gerado pelo ambiente.

Neste capítulo, apresentamos os testes com áudios gravados com detecção de som em tempo real. Primeiro, o sinal de áudio analógico é convertido em um sinal digital, em seguida, o processamento de reconhecimento de fala é efetuado. De acordo ao estado de arte, para os sistemas em tempo real, o processamento de sinal deve ter baixa latência, menos de 20 ms. Por outro, propomos o modelo oiSGNG++, que é o algoritmo oiSGNG modificado para aprendizado on-line de comandos de voz, o modelo aprende novos comandos ao mesmo tempo em que os grava, ou seja, não precisa de nenhum banco de dados pré-gravada.

No sistema operacional ROS, os tópicos são barramentos que permitem a troca de mensagens entre nodos, através da publicação/subscrição semântica. De tal forma que o robô possa responder e agir de acordo com os comandos dados pelo usuário. Portanto, os comandos de voz são diretamente associados aos tópicos do ROS, ou seja, ao tópico de velocidade linear ou angular.

5.1 ROBÔ TIAGO E ROS

O TIAGo é um robô de serviço (manipulador móvel) projetado para trabalhar em ambientes internos; o robô é fabricado pela empresa espanhola PAL Robotics, com destaque mundial no campo da robótica. As características do TIAGo fazem dele a plataforma ideal para pesquisas, especialmente em ambientes domésticos ou indústria leve. Ele combina recursos de mobilidade, percepção, manipulação e interação humano-robô (PAGES, 2019c).

5.1.1 ROS Kinetic

De fato, o robô TIAGo utilizado está com a versão ROS Kinetic como seu *middleware* em uso. A sigla vem do inglês *Robot Operating System* (ROS), que quer dizer sistema operativo robótico e, Kinetic é uma versão de ROS direcionado principalmente para a versão Ubuntu 16.04 (Xenial), embora outros sistemas sejam suportados (FOOTE, 2016).

5.1.2 Simulador Gazebo

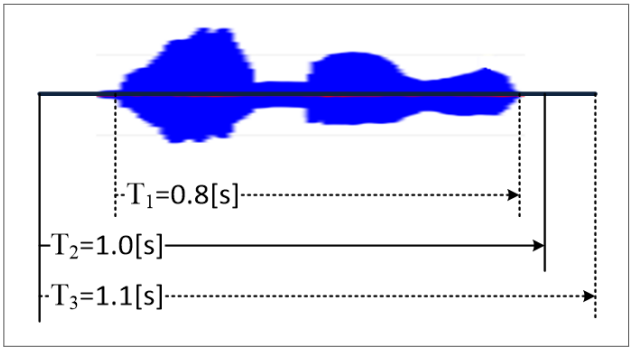
O Gazebo é um simulador de robótica 3D de código aberto, oferece a capacidade de simular com precisão e eficiência de robôs em ambientes internos e externos complexos. O Gazebo foi desenvolvido em 2002 na Universidade do Sul da Califórnia. Segundo Rivera, Simone e Guida (2019) a ideia central foi a criação de um simulador de alta grau de acuracidade que permitisse simular robôs em ambientes externos sob várias condições operacionais e, García e Molina (2020) apresentam uma abordagem para avaliar um sensor LIDAR, e a capacidade de simulação em condições reais de navegação e prevenção de obstáculos em situações simuladas usando uma plataforma de veículo aéreo não tripulado, do inglês *Unmanned Aerial Vehicle* (UAV).

5.2 PROCESSAMENTO DO SINAL DE ÁUDIO EM TEMPO REAL

A maneira mais fácil é gravar um arquivo .wav usando a ferramenta *arecord*, comando de linha em Ubuntu (`arecord -f cd -d 10 -D hw:1,0 test.wav`), em seguida, o arquivo .wav pode ser enviado ao computador via SCP, cópia segura ou simplesmente SCP, do inglês *Secure Copy Protocol* (SCP), SCP é um meio seguro de transferência de arquivos entre um servidor local e um remoto. Mas assim, o problema desta ferramenta que não é adequado para processamento do áudio em tempo real.

Para gravar áudio em tempo real, é preciso realizar gravação por quadros (*frames*), o método e possível implementar com a biblioteca PyAudio de Python. Assim, a captura de sons através do microfone utilizando o PyAudio tem três possibilidades: (1) detecta um som e grava o segmento inteiro até que o silêncio seja detectado; (2) detecta um som, a gravação inicia vários quadros antes e termina exatamente quando o intervalo de tempo de gravação predefinido é concluído; (3) detecta um som, a gravação começa vários quadros antes e termina também vários segmentos após detectar o silêncio (ver Fig. 16). Para o nosso caso, a segunda opção é implementada, pelo motivo de que a base de dados utilizada tem todos os áudios com a duração de 1.0 segundos.

Figura 16 – Processamento do sinal de áudio em tempo real



Fonte: Do autor.

Após a gravação do áudio, o arquivo `.wav` é processado pelo computador do desenvolvimento, através da implementação do código em Python. Desse modo, utilizando um computador com processador Core i5, CPU 2.20 GHz, sistema operacional de 64 bits, a latência do processamento para gerar a palavra reconhecida é mostrada na Tabela 10. Nos resultados, podemos ver que os modelos oiSGNG+ treinados com 98 frames têm maior latência, embora possuam menos nodos, mas essa latência é menor que a restrição de 20 ms. Na Tabela 10, também apresentamos a comparação da taxa de acertos dos sistemas ASR Neurais, para os quais é usado o conjunto de tese da base de dados (BD) formado pelos comandos Grupo C e, a base de dados de comandos de voz gravados com detecção de som em um ambiente real, o novo conjunto de testes possui 332 amostras, 5 locutores. Neste caso, paras o número de unidades da rede neural (RN) considerou-se as camadas oculta e saída no MLP e os nodos inseridos no oiSGNG+.

Tabela 10 – Taxa de acerto (%) e latência (ms)

	MLP	oiSGNG+	
Frames do MFCC	20	20	98
Unidades da RN	206	258.6±9.2	253.4±6.7
Taxa de acerto BD G.C	88.6±0.3	83.5±0.2	84.7±0.3
Taxa de acerto BD Real	90.0±0.9	86.4±3.1	86.7±2.4
Latência	2.72±0.01	9.65±0.25	14.24±0.35

Fonte: Do autor.

Com relação às redes neurais MLP e oiSGNG+, podemos ver que os intervalos de taxa de acertos não se sobrepõem, portanto, podemos concluir que estatisticamente as verdadeiras médias não são iguais. Em geral, nos resultados, a taxa de acertos é mais alta para o novo BD, no qual a principal característica a ser destacada é a gravação com detecção de som em tempo real.

5.3 APRENDIZAGEM ON-LINE DE COMANDOS DE VOZ

Nesta seção, apresentamos o algoritmo oiSGNG++, que é um modelo especificamente para aprendizado on-line de comandos de voz. Assim, as novas características desse algoritmo estão nos itens a seguir:

1. Dado o conjunto de treinamento atual de N_0 amostras, para cada novo exemplo que é gravado com detecção de som em tempo real, o número de amostras é atualizado a $N = N_0 + 1$ e, a rede neural é treinada uma época, isto é, para a aprendizagem é apresentada o conjunto atual de $2N$ amostras aleatoriamente. Portanto, a quantidade de vezes que é treinada a rede neural depende diretamente dos comandos de voz aprendidos de forma on-line.

2. O parâmetro de inserção ϕ é definido como $\phi(N) = 2N$, isto significa, que um nodo é inserido por uma época de treinamento. Na qual, o número de amostras N aumenta com cada novo exemplo, consequentemente, o número de amostras é variante no tempo.

3. A inserção de nodos é efetuada apenas quando o número total de nodos N_i é menor que $2\sqrt{N}$.

Na implementação, levamos em conta que a lista dos nodos inseridos, a lista dos rótulos e o contador c devem ser salvados após a cada treinamento. Além desses três arquivos, o modelo possui outros três arquivos e dois pastas: um arquivo `.corpus`, para armazenar os nomes dos comandos e o seu rótulo; dois arquivos `.csv`, para o conjunto de padrões (vetor de características acústicas), treinamento e teste; e dois pastas para armazenar o conjunto de arquivos `.wav`, treinamento e teste. Na inicialização e funcionamento do aprendizagem on-line de comandos de voz, devemos considerar os seguintes itens para cada novo exemplo.

1. Se existe os arquivos correspondentes aos nodos inseridos, rótulo e contador, então inicializamos o algoritmo com as listas salvadas. No caso contrário, inicializamos em zero.

2. Se existe o arquivo `.corpus`, atualizamos nome e rótulo do novo comando. No caso contrário o arquivo deve ser criado.

3. De acordo ao contador de rótulo, por cada quatro exemplos que é adicionado ao conjunto treinamento, um exemplo é enviado ao conjunto de teste.

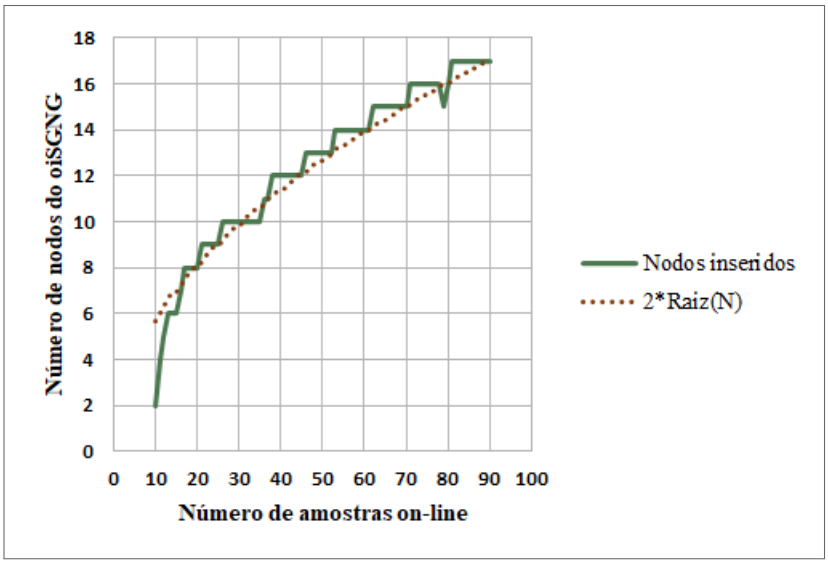
4. Dentro da pasta do conjunto treinamento ou teste, uma subpasta é criada com o nome de cada comando. Os arquivos `.wav` são armazenados dentro de essa sub pasta, com nomes aleatórios.

5. Se existem os arquivos `.csv`, atualizamos o conjunto do vetor de características acústicas. No caso contrário criamos o arquivo correspondente.

6. Finalmente, aplicamos o algoritmo oiSGNG++.

Para iniciar o aprendizado on-line do zero, é recomendável começar com 10 exemplos, duas categorias distribuídas como 4 amostras para treinamento e 1 para teste. Desse jeito, a Figura 17 mostra o número de nodos inseridos ao longo do aprendizado em tempo real, na qual podemos ver que os nodos com idade de conexão maior que a idade máxima são removidos e controlados de acordo com o limite $2\sqrt{N}$.

Figura 17 – Nodos inseridos on-line



Fonte: Do autor.

Nos experimentos, o modelo proposto (algoritmo oiSGNG++ e extrator MFCC de 98 Frames) foi aplicado para aprendizagem on-line do zero. Inicialmente, 5 locutores fizeram parte do grupo de treinamento. Os comandos aprendidos foram agrupados como Grupo E: {*backward*, *forward*, *left*, *right*, *stop*, *tiago*}. Posteriormente, o aprendizado foi estendido a comados em espanhol, formando o Grupo F: {*adelante*, *atrás*, *derecha*, *izquierda*, *alto*}, para o qual fizeram parte 4 locutores. Portanto, para o Grupo E∪F, as amostras on-line ao total foram 326 (264 para treinamento e 62 para teste). Neste caso, podemos determinar o OCMR após apresentação de cada amostra on-line. Desse jeito, na apresentação do último exemplo on-line, o OCMR foi 4.8%. Para o conjunto total de amostras on-line, considerando a inicialização mencionada acima, 33 nodos foram inseridos e, a média da taxa de acertos on-line (1-OCMR) foi 94.1%.

5.4 INTEGRAÇÃO DOS MODELOS ASR NEURAIS EM ROS

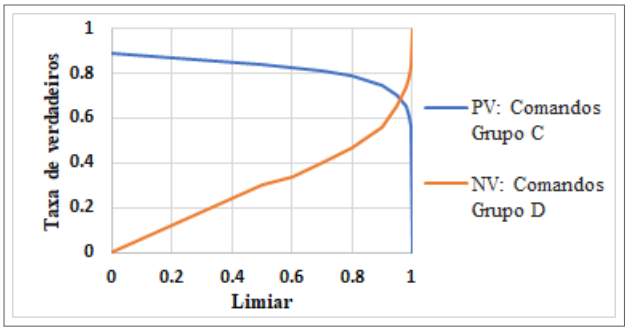
Para realizar a integração dos modelos ANN para reconhecimento de comandos de voz utilizando o robô TIAGo através do ROS, a arquitetura proposta no Capitulo 3 é implementada, na qual, o vetor de características MFCC se torna a entrada das RNs, e a saída final é a palavra reconhecida. Para completar a descrição dessa arquitetura, é necessário detalhar os decodificadores. Por outro lado, a implementação dos modelos ASR Neural no robô TIAGo, é baseada nos códigos Python de Koubaa (2019) e na publicação de um tópico para o movimento da base do robô (PAGES, 2019b).

5.4.1 Opção de Rejeição no MLP

Na arquitetura apresentada na Figura 3, na parte da final, propõe-se um decodificador, cujo objetivo é fornecer a palavra reconhecida a partir da saída do rede neural MLP, essa saída $\mathbf{y}(n)$ é um vetor (*array*) que deve identificar a palavra que corresponde ao sinal do áudio. Mais especificamente, a saída do MLP é o resultado da função de ativação $\varphi(\cdot)$. No caso de palavras desconhecidas, categorias que não fizeram parte da aprendizagem, elas precisam ser rejeitadas, portanto, define-se um limiar de ativação no intervalo 0 a 1, para o qual, acima do limiar a palavra candidata é aceita como verdadeira e abaixo do limiar é considerada falsa. Para os experimentos, vamos empregar outro grupo de comandos, Grupo D: {*Bird*, *eight*, *happy*, *house*, *nine*, *six*}.

A identificação dos comandos de voz pode ser considera como previsão de duas classes, isto é, o Grupo C rotulado como positivo (p) e o Grupo D como negativo (n). Portanto, se o resultado é p e o valor real também é p, então o resultado é um positivo verdadeiro (PV); no entanto, se o valor real é n então é um positivo falso (PF). Por outro lado, um negativo verdadeiro (NV) ocorre quando o resultado da previsão e o valor real são n, e negativo falso (NF) quando o valor real é p. A partir de instâncias totais P positivas e N negativas, a taxa TPV= $\frac{\text{PV}}{\text{P}}$ é conhecida como a sensibilidade e a taxa TNV= $\frac{\text{NV}}{\text{N}}$ é chamada especificidade (FAWCETT, 2006). A curva ROC, do inglês *Receiver Operating Characteristic curve* (ROC curve), é obtida pela representação TPV versus a TPF (1-TNV). Voltando à questão da rede neural MLP, empregamos a representação TPV e TNV versus a limiar de ativação. De acordo aos resultados (Fig. 18), para limiares muito altos, a sensibilidade (taxa dos positivos verdadeiros) diminui e aumenta a especificidade (taxa dos negativos verdadeiros).

Figura 18 – Sensibilidade e Especificidade versus Limiar



Fonte: Do autor.

Na implementação com o robô real, o limiar permite rejeitar positivos falsos causados pelo ruído do ambiente. A taxa dos positivos verdadeiros TPV do MLP é 0.893 quando não tem limiar. Porém, para o limiar 0.4, a sensibilidade é 0.849 e a especificidade é 0.245, em outras palavras, a taxa dos positivos falsos TPF, que era 1 sem o limiar, diminui até 0.755 (1-0.245). Com esse limiar, a TPV do MLP é igual ao do PocketSphinx (Apêndice

C), mas o TPF do último é mais alto (0.774). Por outro lado, pode-se ajustar o limiar de acordo à prioridade do comando. Por exemplo, para evitar colisões, o comando *stop* requer maior sensibilidade, isto é, um limiar baixo.

5.4.2 BMU no oiSGNG

Neste caso, o rotulagem é mais simples do que nos mapas auto-organizáveis (Apêndice B), a unidade vencedora (BMU) já tem o atributo rótulo, quer dizer, $l_{BMU} \in \{1, 2, ..., N_c\}$. Na qual, N_c é o número de categorias. Assim, para $l_{BMU} = k$, a categoria k é identificada como o comando de voz que corresponde ao sinal de áudio da entrada[1].

5.4.3 Tópico de Velocidade

Para o robô avançar e recuar, executam-se os comandos *forward* e *backward*, para esse objetivo, no sistema operacional do robô (ROS), existe o tópico para controlar a base do robô através do comando de velocidade.

```
rostopic pub /mobile_base_controller/cmd_vel geometry_msgs/Twist "linear:
    x: a
    y: 0.0
    z: 0.0
angular:
    x: 0.0
    y: 0.0
    z: b".
```

A instrução básica apresentada por Pages (2019b), publica o tópico que configura a velocidade linear (a=0.5 m/s) ao longo do eixo X, ou seja, o eixo do robô apontando para a frente. Assim, valores positivos para avançar e valores negativos para retroceder.

Além disso, para que o robô TIAGo vire à esquerda e à direita, executam-se os comandos *left* e *right*. Neste caso, para que o robô gire, é necessário publicar o tópico que controla a velocidade angular (b=0.3 rad/s). Na qual, valores positivos são para girar à esquerda e valores negativos para girar à direita[2].

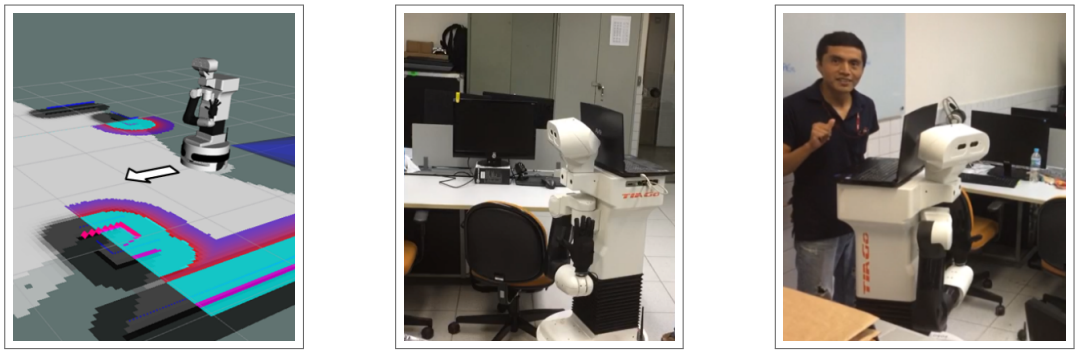
5.4.4 Navegação do Robô TIAGo via Comandos de Voz

Após a seleção dos modelos ASR baseados nas redes neurais (MLP ou oiSGNG) e sua implementação através do ROS, o reconhecimento de comados de voz foi verificado no

[1] https://github.com/egutierrezhu/gng1vn.git
[2] https://github.com/egutierrezhu/ann1vn.git

simulador Gazebo e com o robô TIAGo em ambiente de Laboratório (Figura 19). Essencialmente, verificou-se a navegação do robô TIAGo controlada por comandos de voz[3]. Preliminarmente, os comandos do Grupo C foram empregados na implementação[4], para os quais, deve-se configurar as velocidades linear e angular. As configurações são: velocidade linear para os comandos *backward* e *forward*; a velocidade angular para *left* e *right*; velocidade linear e velocidade angular vão a zero para comando *stop*. O comando *go* é uma palavra-chave para executar as tarefas de ir para frente, para atrás, direita ou esquerda. Nos quatro casos deve ser falado a palavra-chave seguida deste comando, por exemplo, *go forward*. Assim, a palavra-chave permite evitar a execução de falsos comandos e interferências do ambiente ruidoso.

Figura 19 – Verificação e teste da navegação do robô TIAGo



(a) No Gazebo (b) No Laboratório (c) Usuário e Robô

Fonte: Do autor.

Finalmente, para evitar colisões e parar o robô, sugerimos associar outras opções ao comando *stop*, por exemplo, o comando *go* falado duas vezes. Por outro lado, no caso de empregar os comandos do Grupo E∪F, o comando *tiago* é a palavra-chave, similar à implementação do PocketSphinx (ver Apêndice C). Neste caso, os comandos de velocidade podem executados através de comandos em inglês ou espanhol, por exemplo, para parar o robô pode ser falado *stop* ou *alto*. Por último, com relação ao desvio de obstáculos, o ROS possui o tópico da orientação e meta, com o qual podemos criar para o robô pontos alvos no mapa, no Apêndice D apresentamos uma implementação no simulador Gazebo utilizando o robô TIAGo, localização através de comandos de voz e navegação em ambientes domésticos com desvio de obstáculos.

[3] https://youtu.be/5uxSsmu39jM
[4] https://youtu.be/lJYfM4LnAoY

6 CONCLUSÕES E TRABALHOS FUTUROS

6.1 CONCLUSÕES

Para robôs móveis, robô TIAGo de PAL Robotics, propõe-se modelos ASR em tempo real para navegação assistida. Na implementação, para evitar falsos positivos e interferência do ambiente ruidoso, propõe-se o uso de uma palavra-chave.

Essencialmente, no caso de reconhecimento de palavras isoladas, dois métodos são implementados: o extrator de características MFCC e o classificador ANN. Em termos de tempo de processamento e taxa de acertos, a rede neural com melhor desempenho é o MLP. No entanto, o algoritmo oiSGNG modificado é a melhor opção para implementar o aprendizado on-line de comandos de voz, devido ao fato de que é uma rede neural incremental que não precisa de reajuste de parâmetros para lidar com diferentes números de categorias e quantidades de exemplos on-line. Em geral, a taxa de acertos é mais alta para os comandos gravados com detecção de som em tempo real e, a latência do processamento é menor que 15 ms, portanto, podemos concluir que o reconhecimento de comandos de voz foi efetuado em tempo real.

Finalmente, destaca-se as seguintes contribuições para a ciência:

- Com a ferramenta para a adição do ruído a sinais de áudio, código desenvolvido em Matlab, é possível adicionar diversos tipos de ruído, configurar intervalos do valor de relação sinal ruído SNR e, a base de dados a ser adicionado o ruído pode estar contida numa pasta o pode ter uma lista do conjunto áudios. Portanto, para melhorar o desempenho das redes neurais, propõe-se a adição do ruído branco e ruído colorido (rosa, vermelho, azul e violeta) ao conjunto de treinamento.
- No caso do modelo ASR baseado na rede neural MLP, para rejeitar comandos de voz que não fizeram parte da aprendizagem, propõe-se o uso do limiar de ativação para evitar positivos falsos baseada na sensibilidade e especificidade.
- No caso do algoritmo oiSGNG+, propomos um novo mecanismo de inserção de nodos, com intervalo de inserção que varia ao longo do tempo e depende do número de amostras apresentadas, para o qual sugere-se o uso de uma função exponencial com valor arredondado. Assim, facilitando o controle dos nodos inseridos.
- A contribuição mais importante é o algoritmo oiSGNG++ para aprendizagem on-line de comandos de voz, uma nova implementação, o modelo aprende novos comandos ao mesmo tempo em que os grava, assim, pode ser aplicado em diferentes idiomas, nos casos de vocabulário pequeno.

6.2 TRABALHOS FUTUROS

No caso da rede neural MLP, treinar o modelo com mais duma camada escondida. Por outro lado, implementar um modelo incremental para aprendizagem on-line de comandos de voz.

Para ampliar o número de categorias de comandos de voz e considerando cenários acústicos mais complexos, implementar arquiteturas com aprendizagem profunda (BAVU et al., 2019), utilizando o kit de desenvolvimento NVIDIA Jetson TX2, para acelerar o processamento dos dados (PRATAP et al., 2019).

Para reconhecimento de comados de voz com os mapas auto-organizáveis, considerando as topologias de vizinhança, implementar a classificação robusta com opção de rejeição de positivos falsos (SOUSA et al., 2015).

Ao invés de utilizar um canal único, desenvolver algoritmos para reconhecimento de fala, caso multicanais. No caso do robô TIAGo, vem equipado com microfone estéreo. Nesse contexto, usando um array de microfone em um robô móvel é possível realizar a estimativa on-line da posição 3D de uma fonte de som com base na triangulação (SASAKI; TANABE; TAKERNURA, 2018). Desse jeito, pode ser empregado para tarefas de rastreamento e seguimento.

REFERÊNCIAS

AHMET, N.; NATARAJAN, T.; RAO, R. Discrete cosine transform. *IEEE Trans. Computers*, 1974.

AL-THAHAB, A. Controlled of mobile robots by using speech recognition. *Journal of Babylon University, Pure and Applied Sciences*, v. 19, n. 3, p. 1123–1139, 2011.

ALEX, J. S. R.; DAS, A.; KODGULE, S. A.; VENKATESAN, N. A comparative study of isolated word recognizer using svm and wavenet. In: *Computational Signal Processing and Analysis*. [S.l.]: Springer, 2018. p. 139–147.

BALA, A.; KUMAR, A.; BIRLA, N. Voice command recognition system based on mfcc and dtw. *International Journal of Engineering Science and Technology*, v. 2, n. 12, p. 7335–7342, 2010.

BARUA, P.; AHMAD, K.; KHAN, A. A. S.; SANAULLAH, M. Neural network based recognition of speech using mfcc features. In: IEEE. *2014 international conference on informatics, electronics & vision (ICIEV)*. [S.l.], 2014. p. 1–6.

BAVU, E.; RAMAMONJY, A.; PUJOL, H.; GARCIA, A. Timescalenet: A multiresolution approach for raw audio recognition using learnable biquadratic iir filters and residual networks of depthwise-separable one-dimensional atrous convolutions. *IEEE Journal of Selected Topics in Signal Processing*, IEEE, v. 13, n. 2, p. 220–235, 2019.

BORENSTEIN, J.; KOREN, Y. et al. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, v. 7, n. 3, p. 278–288, 1991.

BROWNLEE, J. *Clever algorithms: nature-inspired programming recipes*. [S.l.]: Jason Brownlee, 2011.

BUTTAZZO, G. C. *Hard real-time computing systems: predictable scheduling algorithms and applications*. [S.l.]: Springer Science & Business Media, 2011. v. 24.

CIESLAK, D. A.; CHAWLA, N. V. Learning decision trees for unbalanced data. In: SPRINGER. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. [S.l.], 2008. p. 241–256.

COCHRAN, W. T.; COOLEY, J. W.; FAVIN, D. L.; HELMS, H. D.; KAENEL, R. A.; LANG, W. W.; MALING, G. C.; NELSON, D. E.; RADER, C. M.; WELCH, P. D. What is the fast fourier transform? *Proceedings of the IEEE*, IEEE, v. 55, n. 10, p. 1664–1674, 1967.

CORROCHANO, E. B.; SOBCZYK, G. *Geometric algebra with applications in science and engineering*. [S.l.]: Springer Science & Business Media, 2011.

CRIGHTON, D. Basic principles of aerodynamic noise generation. *Progress in Aerospace Sciences*, Elsevier, v. 16, n. 1, p. 31–96, 1975.

CURATELLI, F.; MAYORA-IBARRA, O. Competitive learning methods for efficient vector quantizations in a speech recognition environment. In: SPRINGER. *Mexican International Conference on Artificial Intelligence*. [S.l.], 2000. p. 108–114.

DAINES, D. H. *An Architecture for Scalable, Universal Speech Recognition*. Tese (Doutorado) — Carnegie Mellon University, 2011.

DAS, B. P.; PAREKH, R. Recognition of isolated words using features based on lpc, mfcc, zcr and ste, with neural network classifiers. *International Journal of Modern Engineering Research*, v. 2, n. 3, p. 854–858, 2012.

DESOUZA, G. N.; KAK, A. C. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 24, n. 2, p. 237–267, 2002.

DHINGRA, S. D.; NIJHAWAN, G.; PANDIT, P. Isolated speech recognition using mfcc and dtw. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, v. 2, n. 8, p. 4085–4092, 2013.

DJURIC, P. M. A model selection rule for sinusoids in white gaussian noise. *IEEE Transactions on Signal Processing*, IEEE, v. 44, n. 7, p. 1744–1751, 1996.

DOYLE, J. A.; EVANS, A. C. What colour is neural noise? *arXiv preprint arXiv:1806.03704*, 2018.

DUQUE-BELFORT, F.; BASSANI, H. F.; ARAUJO, A. F. Online incremental supervised growing neural gas. In: IEEE. *2017 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2017. p. 1034–1040.

DURAND, S.; ALEXANDRE, F. Tom, a new temporal neural net architecture for speech signal processing. In: IEEE. *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. [S.l.], 1996. v. 6, p. 3549–3552.

FARQUAD, M.; BOSE, I. Preprocessing unbalanced data using support vector machine. *Decision Support Systems*, Elsevier, v. 53, n. 1, p. 226–233, 2012.

FAWCETT, T. An introduction to roc analysis. *Pattern recognition letters*, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FAYEK, H. *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between*. [S.l.]: vol, 2016.

FOOTE, T. *ROS Kinetic Kame Released*. [S.l.]: Wiki: kinetic (last edited 2018-01-08 21:55:42 by TullyFoote), 2016.

FRITZKE, B. Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural networks*, Elsevier, v. 7, n. 9, p. 1441–1460, 1994.

FRITZKE, B. A growing neural gas network learns topologies. In: *Advances in neural information processing systems*. [S.l.: s.n.], 1995. p. 625–632.

GARCÍA, J.; MOLINA, J. M. Simulation in real conditions of navigation and obstacle avoidance with px4/gazebo platform. *Personal and Ubiquitous Computing*, Springer, p. 1–21, 2020.

GARG, A.; SHARMA, P. Survey on acoustic modeling and feature extraction for speech recognition. In: IEEE. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. [S.l.], 2016. p. 2291–2295.

GILMAN, D. L.; FUGLISTER, F. J.; JR, J. M. M. On the power spectrum of "red noise". *Journal of the Atmospheric Sciences*, v. 20, n. 2, p. 182–184, 1963.

GOEBEL, R. P. Ros by example, for indigo, vol 1. *Independently Published*, Lulu. com, 2015.

HABETS, E. A.; COHEN, I.; GANNOT, S. Generating nonstationary multisensor signals under a spatial coherence constraint. *The Journal of the Acoustical Society of America*, Acoustical Society of America, v. 124, n. 5, p. 2911–2917, 2008.

HAMILTON, W. R. Lxxviii. on quaternions; or on a new system of imaginaries in algebra: To the editors of the philosophical magazine and journal. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 25, n. 169, p. 489–495, 1844.

HAYKIN, S. Neural networks and learning machines third edition–university hamilton. *Ontario, Canada*, v. 906, 2008.

HJULSTRÖM, R. *Evaluation of a speech recognition system Pocketsphinx*. 2015.

HOLLMÉN, J. *Self-Organizing Map (SOM)*. [S.l.]: EET, 1996. <https://users.ics.aalto.fi/jhollmen/dippa/node9.html>. [Acesso em: 14 janeiro 2020].

JOHNSON, D. H. Signal-to-noise ratio. *Scholarpedia*, v. 1, n. 12, p. 2088, 2006.

JR, J. R. D.; PROAKIS, J. G.; HANSEN, J. H. *Discrete time processing of speech signals*. [S.l.]: Prentice Hall PTR, 1999.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, Springer, v. 43, n. 1, p. 59–69, 1982.

KOHONEN, T. The'neural'phonetic typewriter. *computer*, IEEE, v. 21, n. 3, p. 11–22, 1988.

KOTHAKOTA, S. K. *Reasoning, perception and execution framework for Tiago Robot*. Dissertação (Mestrado) — Universitat Politècnica de Catalunya, 2018.

KOUBAA, A. *Writing a Simple Publisher and Subscriber (Python)*. [S.l.]: Wiki: ROS/Tutorials/WritingPublisherSubscriber(python) (last edited 2019-07-18 19:12:55 by AnisKoubaa), 2019.

KUDRLE, S.; PROULX, M.; CARRIERES, P.; LOPEZ, M. Fingerprinting for solving a/v synchronization issues within broadcast environments. In: SMPTE. *SMPTE Annual Tech Conference & Expo, 2010*. [S.l.], 2010. p. 1–18.

KUO, H.-H. *White noise distribution theory*. [S.l.]: CRC press, 2018.

KUREMOTO, T.; OBAYASHI, M.; KOBAYASHI, K.; FENG, L. Instruction learning systems for partner robots. *Adv. Robot. Model. Control Appl*, v. 8, p. 149–170, 2012.

KURODA, M. C.; VIDAL, A. C.; LEITE, E. P.; DRUMMOND, R. D. Electrofacies characterization using self-organizing maps. *Brazilian Journal of Geophysics*, v. 30, n. 3, 2012.

KUTTRUFF, H. *Room acoustics*. [S.l.]: Crc Press, 2016.

LAMERE, P.; KWOK, P.; GOUVEA, E.; RAJ, B.; SINGH, R.; WALKER, W.; WARMUTH, M.; WOLF, P. The cmu sphinx-4 speech recognition system. In: *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*. [S.l.: s.n.], 2003. v. 1, p. 2–5.

LAU, D. L.; ULICHNEY, R.; ARCE, G. R. Blue and green noise halftoning models. *IEEE Signal Processing Magazine*, IEEE, v. 20, n. 4, p. 28–38, 2003.

LEE, C.-H.; SOONG, F. K.; PALIWAL, K. K. *Automatic speech and speaker recognition: advanced topics*. [S.l.]: Springer Science & Business Media, 2012. v. 355.

LYONS, J. Mel frequency cepstral coefficient (mfcc) tutorial. *Practical Cryptography*, 2015.

MANI, I.; ZHANG, I. knn approach to unbalanced data distributions: a case study involving information extraction. In: *Proceedings of workshop on learning from imbalanced datasets*. [S.l.: s.n.], 2003. v. 126.

MANSOUR, A. H.; SALH, G. Z. A.; MOHAMMED, K. A. Voice recognition using dynamic time warping and mel-frequency cepstral coefficients algorithms. *International Journal of Computer Applications*, Foundation of Computer Science, v. 116, n. 2, 2015.

MARTINETZ, T.; SCHULTEN, K. et al. A"neural-gas"network learns topologies. University of Illinois at Urbana-Champaign Champaign, IL, 1991.

MOCANU, I.; AXINTE, D.; CRAMARIUC, O.; CRAMARIUC, B. Human activity recognition with convolution neural network using tiago robot. In: IEEE. *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*. [S.l.], 2018. p. 1–4.

NOVOA, J.; WUTH, J.; ESCUDERO, J. P.; FREDES, J.; MAHU, R.; YOMA, N. B. Dnn-hmm based automatic speech recognition for hri scenarios. In: ACM. *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. [S.l.], 2018. p. 150–159.

PAGES, J. *Localization and path planning*. [S.l.]: Wiki: Robots/TIAGo/Tutorials/Navigation/Localization (last edited 2019-10-16 08:29:39 by ProcopioStein), 2019.

PAGES, J. *Moving the base through velocity commands*. [S.l.]: Wiki: Robots/TIAGo/Tutorials/motions/cmd$_v el(lastedited 2019-04-21:05:26 by DanCerys)$, 2019.

PAGES, J. *TIAGo*. [S.l.]: Wiki: Robots/TIAGo (last edited 2019-01-07 10:07:42 by Jordi Pages), 2019.

PAL-ROBOTICS. *TIAGo Handbook*. [S.l.]: erbium, 2019. v. 1.19.0.

PANTAZOGLOU, F. K.; PAPADAKIS, N. K.; KLADIS, G. P. Implementation of the generic greek model for cmu sphinx speech recognition toolkit. eRA-12 International Scientific Conference, 2017.

PÉREZ, M.; RUMIPAMBA, J. *Robots Submarinos AUV, ROV y Caminantes*. [S.l.]: YouTube, 2014. <https://www.youtube.com/watch?v=GN1unkazV_g>. [Acesso em: 25 setembro 2017].

PLATON, L.; ZEHRAOUI, F.; TAHI, F. Self-organizing maps with supervised layer. In: IEEE. *2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*. [S.l.], 2017. p. 1–8.

POLO, A. *Robots Submarinos Robots de Servicio*. [S.l.]: YouTube, 2015. <https://www.youtube.com/watch?v=feJG_19bzPY>. [Acesso em: 25 setembro 2017].

PRABHAKAR, O. P.; SAHU, N. K. A survey on: Voice command recognition technique. *International Journal of Advanced Research in Computer Science and Software Engineering*, v. 3, n. 5, 2013.

PRATAP, V.; HANNUN, A.; XU, Q.; CAI, J.; KAHN, J.; SYNNAEVE, G.; LIPTCHINSKY, V.; COLLOBERT, R. Wav2letter++: A fast open-source speech recognition system. In: IEEE. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2019. p. 6460–6464.

PROAKIS, J. G. *Digital signal processing: principles algorithms and applications*. [S.l.]: Pearson Education India, 2001.

PRUDENT, Y.; ENNAJI, A. An incremental growing neural gas learns topologies. In: IEEE. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* [S.l.], 2005. v. 2, p. 1211–1216.

RAI, N.; RAI, B. An ann based mobile robot control through voice command recognition using nepali language. *International Journal of Applied Control, Electrical and Electronics Engineering (IJACEEE)*, v. 2, n. 4, p. 13–22, 2014.

RAO, K. R.; YIP, P. *Discrete cosine transform: algorithms, advantages, applications*. [S.l.]: Academic press, 2014.

RAVANELLI, M. Deep learning for distant speech recognition. *arXiv preprint arXiv:1712.06086*, 2017.

RIVERA, Z. B.; SIMONE, M. C. D.; GUIDA, D. Unmanned ground vehicle modelling in gazebo/ros-based environments. *Machines*, Multidisciplinary Digital Publishing Institute, v. 7, n. 2, p. 42, 2019.

ROSENFELD, B. A. *A history of non-Euclidean geometry: evolution of the concept of a geometric space*. [S.l.]: Springer Science & Business Media, 2012. v. 12.

RUSSELL, S. J.; NORVIG, P. *Inteligencia Artificial: un enfoque moderno*. [S.l.: s.n.], 2004.

SAAD, D. *On-line learning in neural networks*. [S.l.]: Cambridge University Press, 2009. v. 17.

SARASWATI, A.; NGUYEN, V. T.; HAGENBUCHNER, M.; TSOI, A. C. High-resolution self-organizing maps for advanced visualization and dimension reduction. *Neural Networks*, Elsevier, v. 105, p. 166–184, 2018.

SASAKI, Y.; TANABE, R.; TAKERNURA, H. Online spatial sound perception using microphone array on mobile robot. In: IEEE. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2018. p. 2478–2484.

SIDIQ, M.; BUDI, W. T. A.; SA'ADAH, S. Design and implementation of voice command using mfcc and hmms method. In: IEEE. *2015 3rd International Conference on Information and Communication Technology (ICoICT)*. [S.l.], 2015. p. 375–380.

SILAGHI, H.; ROHDE, U.; SPOIALĂ, V.; SILAGHI, A.; GERGELY, E.; NAGY, Z. Voice command of an industrial robot in a noisy environment. In: IEEE. *2014 International Symposium on Fundamentals of Electrical Engineering (ISFEE)*. [S.l.], 2014. p. 1–5.

SIMONIC, M. *A Voice User Interface for Human-Robot Interaction on a Service Robot*. Tese (Doutorado) — B. Sc. Thesis, Wilhelm Schickard Institute for Informatics, University of . . . , 2015.

SINGH, P. P.; RANI, P. An approach to extract feature using mfcc. *IOSR Journal of Engineering*, v. 4, n. 8, p. 21–25, 2014.

SOUSA, R. G.; NETO, A. R. R.; CARDOSO, J. S.; BARRETO, G. A. Robust classification with reject option using the self-organizing map. *Neural Computing and Applications*, Springer, v. 26, n. 7, p. 1603–1619, 2015.

STENMAN, M. *Automatic speech recognition An evaluation of Google Speech*. 2015.

STEVENS, S. S.; VOLKMANN, J.; NEWMAN, E. B. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, ASA, v. 8, n. 3, p. 185–190, 1937.

SUN, Q.; LIU, H.; HARADA, T. Online growing neural gas for anomaly detection in changing surveillance scenes. *Pattern Recognition*, Elsevier, v. 64, p. 187–201, 2017.

SZENDRO, P.; VINCZE, G.; SZASZ, A. Pink-noise behaviour of biosystems. *European Biophysics Journal*, Springer, v. 30, n. 3, p. 227–231, 2001.

TANG, J.; REN, Y.; LIU, S. Real-time robot localization, vision, and speech recognition on nvidia jetson tx1. *arXiv preprint arXiv:1705.10945*, 2017.

TAYMANS, W.; BAKER, S.; WINGO, A.; BULTJE, R. S.; KOST, S. Gstreamer application development manual (1.2. 3). *Publicado en la Web*, 2013.

VESANTO, J.; HIMBERG, J.; ALHONIEMI, E.; PARHANKANGAS, J. et al. Self-organizing map in matlab: the som toolbox. In: *Proceedings of the Matlab DSP conference*. [S.l.: s.n.], 1999. v. 99, p. 16–17.

VIANA, H. O.; ARAUJO, A. F. Adaptive speech model for missing-feature reconstruction. In: IEEE. *2016 IEEE International Conference on Digital Signal Processing (DSP)*. [S.l.], 2016. p. 104–108.

VINCENT, P.; LAROCHELLE, H.; BENGIO, Y.; MANZAGOL, P.-A. Extracting and composing robust features with denoising autoencoders. In: ACM. *Proceedings of the 25th international conference on Machine learning*. [S.l.], 2008. p. 1096–1103.

WARDEN, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

WEID, J. V. D.; PASSY, R.; MUSSI, G.; GISIN, N. On the characterization of optical fiber network components with optical frequency domain reflectometry. *Journal of Lightwave Technology*, IEEE, v. 15, n. 7, p. 1131–1141, 1997.

WEISSTEIN, E. W. Euler angles. Wolfram Research, Inc., 2009.

WIJOYO, S.; WIJOYO, S. Speech recognition using linear predictive coding and artificial neural network for controlling movement of mobile robot. In: *Proceedings of 2011 International Conference on Information and Electronics Engineering (ICIEE 2011)*. [S.l.: s.n.], 2011. p. 28–29.

YU, D.; DENG, L. *AUTOMATIC SPEECH RECOGNITION*. [S.l.]: Springer, 2016.

ZAATRI, A.; AZZIZI, N.; RAHMANI, F. L. Voice recognition technology using neural networks. *Journal of New Technology and Materials*, University of Oum El-Bouaghi, v. 277, n. 2414, p. 1–5, 2015.

ZHANG, Z.; GEIGER, J.; POHJALAINEN, J.; MOUSA, A. E.-D.; JIN, W.; SCHULLER, B. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 9, n. 5, p. 49, 2018.

ZHI-QIANG, H.; JIA-QI, Z.; XIN, W.; ZI-WEI, L.; YONG, L. Improved algorithm of dtw in speech recognition. In: IOP PUBLISHING. *IOP Conference Series: Materials Science and Engineering*. [S.l.], 2019. v. 563, n. 5, p. 052072.

ZHIVOMIROV, H. A method for colored noise generation. *Romanian Journal of Acoustics and Vibration*, v. 15, n. 1, p. 14–19, 2018.

APÊNDICE A – CEPSTRUM DE FREQUÊNCIA MEL

O cepstrum de frequência mel, do inglês *Mel-Frequency Cepstrum* (MFC), é uma representação do espectro de potência de intervalo de tempo curto de um som. No banco de dados "Speech Commands", o sinal de fala é amostrado em 16kHz. Portanto, os espectrogramas e as etapas para a implementação dos MFCCs são:

1. Enquadre o sinal em quadros de 20 a 40 ms. De acordo com Lyons (2015) 25ms é padrão. Isso significa que o comprimento do quadro para um sinal de 16kHz é 0,025 * 16000 = 400 amostras. As próximas etapas são aplicadas a cada quadro, um conjunto de 12 coeficientes MFCC é extraído para cada quadro. Um breve resumo da notação: chamamos $s(n)$ a nosso sinal de domínio do tempo, uma vez enquadrado $s_i(n)$, temos n que varia de 1 a 400, que varia de acordo com o número de quadros. Quando calculamos o complexo DFT, obtemos $S_i(k)$. Assim, $P_i(k)$ é o espectro de potência do quadro.

2. Para realizar a Transformação discreta de Fourier do quadro, faça o seguinte:

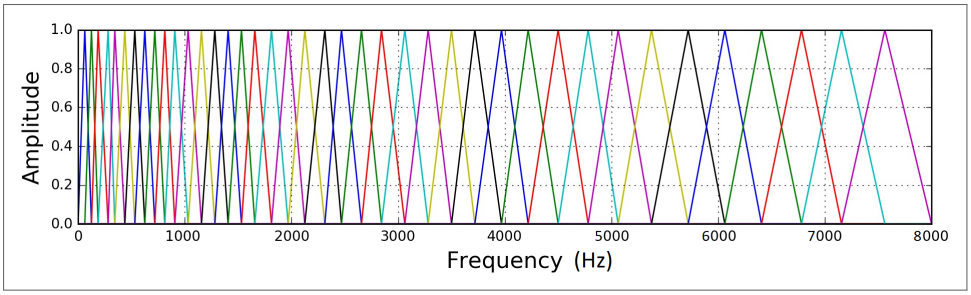
$$S_i(k) = \sum_{n=0}^{N-1} s_i(n)h(n)e^{-j\frac{2\pi k}{N}n} \qquad k = 0, 1, ..., K-1 \tag{A.1}$$

   Na qual, $h(n)$ é uma janela de análise de N amostras (por exemplo, janela hamming) e K é o comprimento da DFT. A estimativa espectral de potência baseada em periodogramas (LYONS, 2015), para o quadro de fala $s_i(n)$ é dada por:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \tag{A.2}$$

3. Calcule o banco de filtros escala Mel. De acordo com Lyons (2015), esse banco de filtros é um conjunto de 20-40 (26 é padrão) filtros triangulares que aplicamos à estimativa espectral de potência do periodograma da etapa 2. Para calcular as energias do banco de filtros, multiplicamos cada banco de filtros pelo espectro de potência e, em seguida, somamos os coeficientes. Uma vez feito isso, ficamos com 26 números que nos dão uma indicação de quanta energia havia em cada banco de filtros (ver Fig.20).

Figura 20 – Banco de 40 filtros na escala de Mel



Fonte: Adaptado de Fayek (2016).

Segundo Lyons (2015), isso pode ser modelado pela seguinte equação:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k < f(m) \\ 1 & k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{A.3}$$

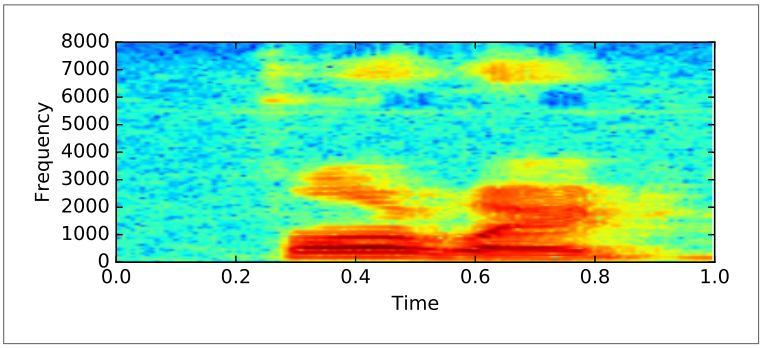
4. Pegue o log de cada uma das 26 energias da etapa 3. Isso nos deixa com 26 energias do banco de filtros de log.

5. Pegue a Transformada discreta de cosseno (DCT) das 26 energias do banco de filtros de log para obter 26 coeficientes cepstrais. Para ASR, segundo Lyons (2015), apenas os 2-13 inferiores dos 26 coeficientes são mantidos. Os valores resultantes (12 números para cada quadro) são chamados de Coeficientes Cepstrais de Frequência Mel.

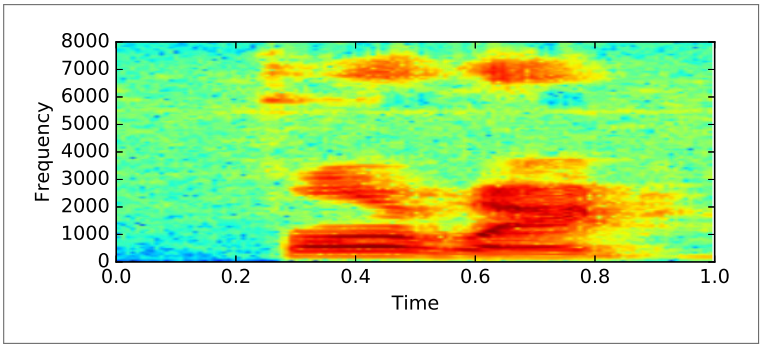
6. (Opcional) O coeficiente zero cepstral é substituído pelo log da energia total do quadro.

Depois de aplicar o banco de filtros ao espectro de potência (*periodogram*) do sinal, obtemos os seguintes espectrogramas (ver Fig. 21).

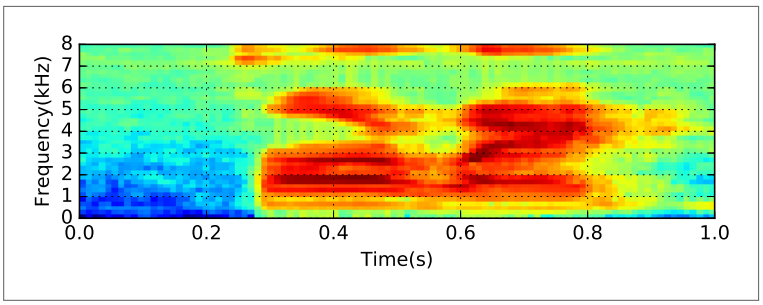
Figura 21 – Espectrograma de sinal no processamento MFCC



(a) Sinal original



(b) Sinal após pré-ênfase



(c) Aplicação do banco de filtros

Fonte: Adaptado de Fayek (2016).

APÊNDICE B – MAPAS AUTO-ORGANIZÁVEIS

Os mapas auto-organizáveis de Kohonen (1982), do inglês *Self-Organizing Maps* (SOM), também podem ser aplicados para reconhecimento de fala. Durand e Alexandre (1996) propõem um modelo de mapa de organização temporal, do inglês *Temporal Organization Map* (TOM), para reconhecimento de dígitos falados. Viana e Araujo (2016) propõem um modelo SOM com campo receptivo adaptativo local, para reconhecimento de fala com reconstrução de características ausentes.

B.1 MAPAS AUTO-ORGANIZÁVEIS LRTSOM

O mapa auto-organizável é um dos modelos de redes neurais mais populares. Pertence à categoria de redes de aprendizagem competitivas. De acordo a Kohonen (1982) e Hollmén (1996), um mapa auto-organizável é baseado em aprendizado não supervisionado no qual cada amostra ou padrão é descrito por um vetor de valores (atributos), mas não precisa do rótulo da categoria associada durante o aprendizado. Neste apêndice, apresentamos os mapas topológicos auto-organizáveis de baixa resolução, no inglês *low resolution Topological Self-Organized Maps* (lrTSOM), pospostos como classificador.

B.1.1 Topologias do Mapa Auto-organizável

Um mapa auto-organizável é um conjunto bidimensional de nodos $\mathbf{M} = \{\mathbf{m}_i\}_{i=1}^{N_u}$. Neste caso, N_u é o número total dos nodos e, um nodo tem um vetor de pesos a ele associado $\mathbf{m}_i$ é chamado vetor protótipo, no inglês *codebook vector*:

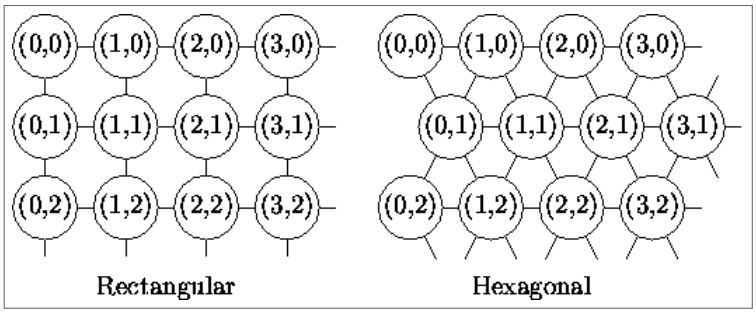
$$\mathbf{m}_i = [m_{i1}, m_{i2}, ..., m_{iq}]^T. \tag{B.1}$$

Para os padrões de entrada $\mathbf{X}$, esse vetor tem a mesma dimensão que os vetores de entrada q-dimensional ($\mathbf{x} \in \mathbf{X}$):

$$\mathbf{x} = [x_1, x_2, ..., x_q]^T. \tag{B.2}$$

Os nodos são conectados a outros adjacentes, os vizinhos, determinando a topologia ou a estrutura do mapa. Geralmente, os neurônios são conectados entre si por topologia retangular ou hexagonal. Na Fig.22, as relações topológicas são mostradas por linhas entre os neurônios.

Figura 22 – Topologias de mapas auto-organizáveis

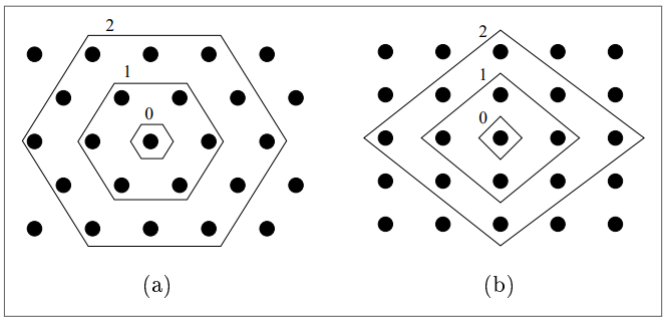


Fonte: Hollmén (1996).

B.1.2 Níveis de Vizinhança

As unidades de mapa geralmente formam uma estrutura bidimensional. A propriedade de preservação da topologia significa que o mapeamento preserva a distribuição de entrada: os pontos próximos do espaço de entrada são mapeados para unidades próximas no SOM. Além disso, o SOM tem a capacidade de generalização. A Fig.23, mostra duas configurações de grade (hexagonal e retangular) e, níveis de vizinhança hexagonal e losango ($p = 0, 1, 2, ...$).

Figura 23 – Duas configurações de grade e níveis de vizinhança: (a) a grade hexagonal e (b) a grade retangular

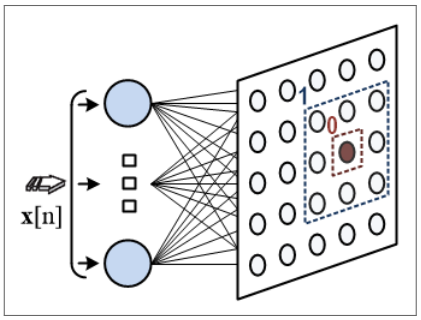


Fonte: Vesanto et al. (1999).

B.1.3 Algoritmo lrTSOM

O algoritmo do treinamento de um SOM proposto por Kohonen (1982), pode ser apresentado envolvendo dois processos computacionais, o competitivo e o cooperativo. O mapa auto-organizado básico é conhecido como SOM de baixa resolução (SARASWATI et al., 2018), do inglês *Low Resolution Self-Organized Maps* (LRSOM). Semelhante à abordagem de Kuroda et al. (2012), leva-se em conta as topologias e os níveis da vizinhança da unidade vencedora, do inglês *Best Matching Unit* (BMU), que é ilustrada na Fig. 24.

Figura 24 – BMU no mapa bidimensional



Fonte: Adaptado de Kohonen (1982).

A vizinhança tipicamente tem uma forma geométrica regular na qual todos os nodos da vizinhança são atualizados usando a mesma taxa de aprendizagem da iteração. Essa taxa de aprendizagem depende da distância de vizinhança usando uma distribuição gaussiana, quer dizer, os vizinhos mais distantes da unidade vencedora, são atualizados de modo mais suave (BROWNLEE, 2011).

B.1.3.1 Processo Competitivo

Um vetor de entrada $\mathbf{x}$ é selecionado aleatoriamente do conjunto de dados de treinamento (SARASWATI et al., 2018). Geralmente, busca-se a distância euclidiana mínima entre $\mathbf{x}$ e o vetor de pesos sinápticos do nodo-i, $\mathbf{m}_i$. O nodo vencedor $\mathbf{m}_c$ tem que satisfazer a seguinte relação:

$$c = \underset{i \in M}{arg\ min} \left| \mathbf{x} - \mathbf{m}_i \right|. \tag{B.3}$$

Em particular, o neurônio-c que satisfaz essa condição é chamado neurônio vencedor ou BMU, para o vetor de entrada $\mathbf{x}$.

B.1.3.2 Processo Cooperativo

Depois que a BMU é determinada pela menor distância, os pesos da unidade vencedora seus vizinhos são atualizados de acordo a grade escolhida e a topologia de vizinhança C, ou seja, elas se aproximam ao vetor de entrada no espaço de saída (KURODA et al., 2012). A atualização do $\mathbf{m}_i$ move a unidade vencedora na direção ao vetor $\mathbf{x}$, de acordo com a função de densidade de probabilidade fornecida por:

$$\Delta \mathbf{m}_i(t) = \begin{cases} \alpha(t) h_{i,c}(t)(\mathbf{x} - \mathbf{m}_i) & Se\ i \in C \\ 0 & caso\ contrário \end{cases} \tag{B.4}$$

A taxa de aprendizagem $\alpha(t)$ é ajustada à medida que o treinamento avança. A função de vizinhança $h_{i,c}(t)$ é responsável por controlar a quantidade pela qual os pesos sináp-

ticos dos neurônios vizinhos são atualizados. De acordo com Haykin (2008), uma escolha popular é a função de vizinhança gaussiana:

$$h_{i,c}(t) = exp(-\frac{d_{i,c}^2}{2\sigma^2(t)}). \tag{B.5}$$

Na cooperação entre os neurônios vizinhos, é necessário que a vizinhança topológica $h_{i,c}$ seja dependente da distância lateral $d_{i,c}$, entre o neurônio vencedor c e o neurônio-i no espaço de saída. No caso de uma rede unidimensional, $d_{i,c}$ é um número inteiro igual a $|i - c|$. Por outro lado, no caso de uma rede bidimensional, é definida por:

$$d_{i,c}^2 = |\mathbf{r}_i - \mathbf{r}_c|^2 \tag{B.6}$$

na qual, o vetor discreto $\mathbf{r}_i$ define a posição do neurônio-i e $\mathbf{r}_c$ define a posição do neurônio vencedor c, os quais são medidos no espaço de saída discreto.

Outra característica exclusiva do algoritmo SOM é que a função da vizinhança topológica $h_{i,c}$ diminui com o tempo, portanto, o tamanho da vizinhança topológica C diminui com o tempo, especificamente, o nível máximo de vizinhança pode ser determina pela parte inteira do $h_{i,c}$. Habitualmente, o decaimento é uma função exponencial, descrito por:

$$\sigma(t) = \sigma_0(\sigma_0)^{-\frac{t}{T}} \tag{B.7}$$

na qual, T é o número máximo das épocas do aprendizado ($t = 0, 1, ..., T$). Ademais, o parâmetro da taxa de aprendizagem $\alpha(t)$ também varia no tempo. Em particular, parte-se de um valor inicial α_0 que diminui gradativamente com o passar do tempo t, até o valor final α_f. Esse requisito pode ser atendido pela seguinte expressão:

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_f}{\alpha_0}\right)^{\frac{t}{T}}. \tag{B.8}$$

O seguinte algoritmo é baseado no pseudocódigo para o SOM de Brownlee (2011), com ênfase na topologia da vizinhança e, considera as funções de taxa de aprendizagem e de vizinhança apresentadas acima, para um mapa $N_x \times N_y$.

B.1.4 SOM Topológico como Classificador

Para lidar com o problema da classificação podemos distinguir duas estratégias: a primeira usa informações do rótulo após a construção do mapa. O segundo usa essas informações durante a etapa de treinamento do SOM.

- **Estratégia 1 (lrTSOM-1)**. A primeira estratégia envolve uma rotulagem de neurônios pós-treinamento. Consiste primeiramente em treinar o SOM da maneira não

Algoritmo 4: Pseudocódigo para o lrTSOM

Entrada: $\mathbf{X}$, α_0, α_f, σ_0, T, N_x, N_y
Saída : $\mathbf{M}$

```
M ← InicializarVetoresProtótipos(N_x, N_y, X)
X ← EmbaralharPadrõesEntrada(X)
for t = 0 To T do
    α(t) ← FunçãoTaxaAprendizagem (α_0, α_f, t, T)
    σ(t) ← FunçãoVizinhança (σ_0, t, T)
    for x ∈ X do
        BMU ← SeleçãoUnidadeVencedora(x, M)
        Vizinhança ← SeleçãoVizinhos(BMU, M, σ(t))
        for m_i ∈ Vizinhança do
            m_i ← m_i + α(t) · σ(t) · (x - m_i)
```

supervisionada. Após o término do treinamento, é necessário apresentar todos os dados do treinamento novamente ao SOM, a fim de encontrar o neurônio vencedor para cada vetor padrão (SOUSA et al., 2015). As unidades são rotuladas por votação majoritária dos dados de entrada correspondentes (PLATON; ZEHRAOUI; TAHI, 2017).

Podem ocorrer duas situações indesejáveis: (1) ambiguidade ou (2) neurônios mortos. A ambiguidade ocorre quando a frequência dos rótulos de classe dos padrões mapeados para um dado neurônio é equivalente. Neurônios mortos são aqueles que nunca são selecionados como vencedores para nenhum dos padrões de entrada (SOUSA et al., 2015). Na fase de classificação, empregamos a votação majoritária das unidades de acordo aos níveis da vizinhança até conseguir o rótulo vencedor, ou seja, existem três possibilidades na identificação da classe: (1) a BMU é um neurônio nulo, portanto, a busca é realizada no seguinte nível de acordo a vizinhança topológica, neste caso, a votação de todos os nós da vizinhança são somados; (2) ambiguidade na BMU, se não houver um máximo único na votação, a busca vai para o próximo nível; e (3) no caso de ter um único máximo, a categoria vencedora é identificada como a classe que corresponde ao padrão de entrada.

- **Estratégia 2 (lrTSOM-2)**. A segunda estratégia é proposta por Kohonen (1988), geralmente chamada de esquema de treinamento SOM auto-supervisionado. Seu princípio é combinar o vetor de rótulo $\mathbf{x}_l$ com os vetores de entrada $\mathbf{x}_v$ para formar um vetor aumentado $\mathbf{x} = [\mathbf{x}_v, \mathbf{x}_l]$. Um SOM treinado em vetores aumentados produz um SOM supervisionado. Neste caso, para cada unidade r, o vetor de peso correspondente $\mathbf{m}_r = \left[\mathbf{m}_r^v, \mathbf{m}_r^l\right]$ é atualizado usando o algoritmo lrTSOM. Os vetores aumentados são usados durante a etapa de competição (localização da BMU) e na etapa de cooperação (atualização dos vetores de peso dos neurônios).

B.2 RESULTADOS DO ALGORITMO LRTSOM

Para os resultados do algoritmo lrTSOM, a base de dados utilizada é o "Grupo 1" (treinamento e teste), comandos de voz para navegação assistida, balanceada com adição de ruído. Neste caso, através do uso das topologias da vizinhança, o tempo de treinamento é menor e melhora a acurácia de classificação com respeito a SOM original. Isso ocorre porque o tempo de processamento está relacionado ao número de unidades no nível da vizinhança. Assim, dado um nível de vizinhança, o número neurônios da topologia losango é menor do que no hexágono e, deste é menor do que na topologia de vizinhança quadrado (ver Tabela 11). De acordo com Hollmén (1996), as relações topológicas e o número de neurônios afetam a precisão e a capacidade de generalização do modelo.

Tabela 11 – Topologia de vizinhança ($p \in Z : p > 0$)

Nível	Unidades		
	Losango	Hexágono	Quadrado
0	1	1	1
1	4	6	8
2	8	12	16
p	4p	6p	8p

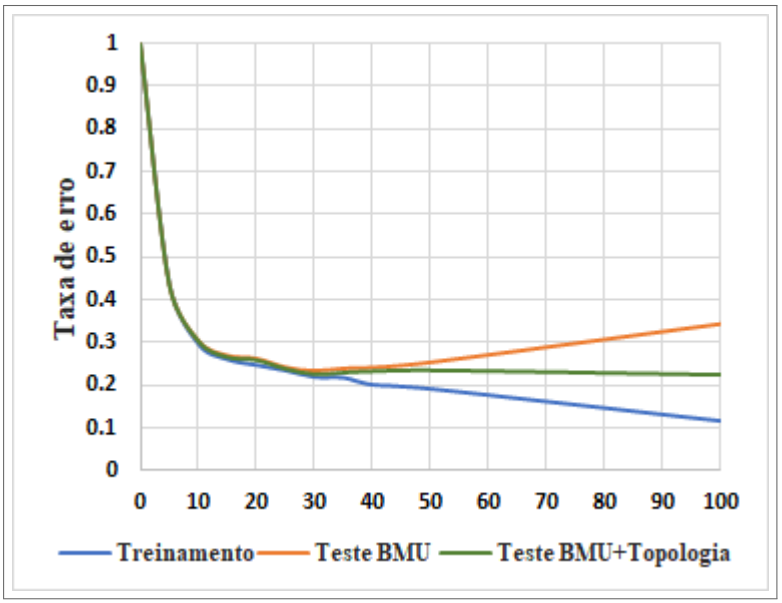
Fonte: Do autor.

A capacidade de generalização significa que a rede pode reconhecer ou caracterizar entradas que nunca encontrou antes. Então, deve-se levar em conta que a generalização (teste) e a precisão (treinamento) são objetivos contraditórios, a melhora de um pode levar à piora do outro. Na Figura 25, apresentamos os resultados do algoritmo lrTSOM, na qual, foram realizados os testes considerando só a unidade vencedora (teste BMU) e por outro lado levando em conta as topologias e níveis de vizinhança (teste BMU+Topologia). O primeiro caso tem o problema de ambiguidade e neurônios nulos, para lidar com esse problema, no segundo caso, consideramos os níveis de vizinhança da BMU.

B.2.1 Parâmetros lrTSOM

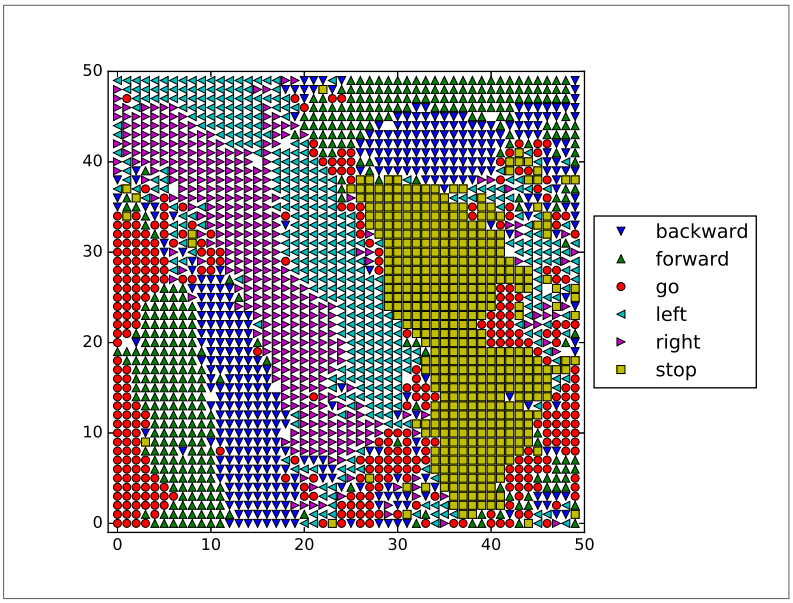
Dado o conjunto $M = \{m_i\}_{i=0}^{N_u-1}$, cada neurônio é localizado como um nodo na grade, para o nosso caso o grade bidimensional, portanto, é necessário ter uma relação entre o número de neurônio e a localização espacial discreta, assim, o neurônio m_i estará na coordenada (x_i,y_i), que é determinada pelo resto da divisão ($y_i = i \ MOD \ N_y$) e quociente da divisão inteira ($x_i = i \ DIV \ N_y$). Na qual, N_y o tamanho da grade no eixo $\mathbf{y}$. Assim, o neurônio-i é determinado pela equação $i = x_i N_y + y_i$. A Figura 26 mostra o resultado dum mapa 50x50 com topologia da vizinhança quadrada ($T = 30$).

Figura 25 – Acurácia versus Tamanho da grade



Fonte: Do autor.

Figura 26 – Mapa auto-organizado 50x50

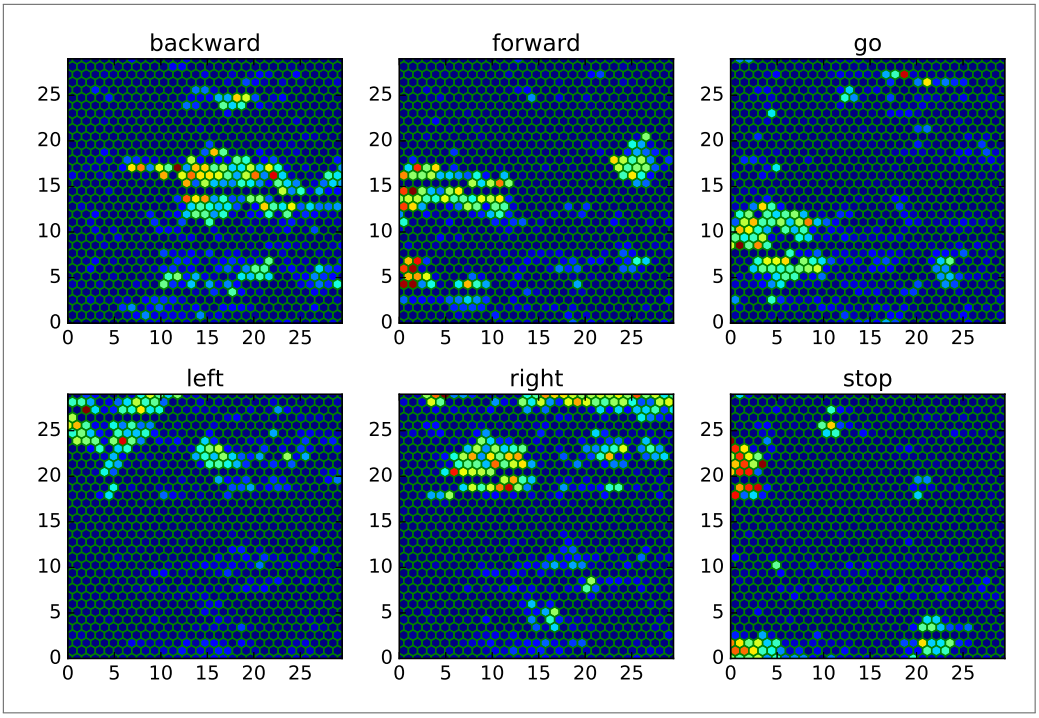


Fonte: Do autor.

Para o treinamento de um conjunto de N amostras, Vesanto et al. (1999) sugerem $5\sqrt{N}$ para o tamanho da unidade de saída (aproximadamente um mapa 27x27 para 21600 amostras). Para o caso SOM bidimensional, dado os valores iniciais de taxa de aprendizagem α_0 e, tamanho inicial de vizinhança σ_0, propõe-se selecionar o tamanho da grade experimentalmente, similar ao método da validação da rede neural MLP, ou seja, ao aumentar o número de nós da grade, existe um mínimo taxa de erro no caso do teste considerando só a BMU (ver Fig. 25). Assim, o mapa topológico com melhor desempenho é 30x30. O resultado obtido é próximo ao mencionado acima. No mapa quadrado ilustrado na Fig. 26

e no mapa hexagonal da Fig. 27, podemos ver que palavras semelhantes foram mapeadas adjacentes, ou seja, *left* ao lado de *right*, *backward* ao lado de *forward* e, *go* ao lado de *forward* e de *stop*.

Figura 27 – Topologia hexagonal SOM 30x30



Fonte: Do autor.

Nesta seção, a taxa de aprendizagem é definido variante ao longo do aprendizagem, de $\alpha_0 = 0.020$ até $\alpha_f = 0.009$, usando a função exponencial $\alpha(t) = \alpha_0 \left(\frac{\alpha_f}{\alpha_0}\right)^{t/T}$. Além disso, considerando que o tamanho inicial da vizinhança deve ser menor ao tamanho da grade, utilizamos $\sigma_0 = max(Nx, Ny)/4$.

B.2.2 Resultados das Estratégias de Classificação

No caso da primeira estratégia, o lrTSOM possui a característica importante de alta dimensionalidade, ou seja, não é necessário alterar o tamanho da grade e os parâmetros no caso de alterações na dimensionalidade dos dados. Assim, para os áudios de duração 1 s, ao dividir o sinal em segmentos de 50 ms, sem superposição é obtida 20 frames, que gera o vetor de características da dimensão 260. No segundo caso, os frames têm o tamanho 25 ms com superposição de 10 ms, portanto, para áudios de 1 s, gera-se 98 frames e, o vetor de características de dimensão 1274 (alta dimensão). Neste caso, os resultados obtidos para 500 épocas são muito próximos (ver Tabela 12), sendo a melhor de taxa de acerto para o caso de 98 frames, 81.1%.

Na segunda estratégia, no caso do lrTSOM supervisionado, o aprendizado é mais rápido do que no caso anterior; portanto, os resultados apresentados na Tabela 12 são

para um máximo de 50 épocas de treinamento. Neste caso, a taxa média de acertos em 8 interações para cada topologia é maior do que na primeira estratégia. Assim, a taxa de acerto mais alta é 84,2 % no caso de 20 frames. Por outro lado, a segunda estratégia não apresenta melhores resultados para dados de alta dimensão.

Tabela 12 – Taxa de acerto (%) para 20 e 98 Frames

Topologia	lrTSOM-1		lrTSOM-2	
	20 Frames	98 Frames	20 Frames	98 Frames
Losango	80.5	80.8	83.5	80.5
Hexágono	79.1	79.2	82.1	80.1
Quadrado	80.0	80.1	83.3	80.5
Total (σ)	79.9 (0.7)	80.0 (0.8)	83.0 (0.7)	80.4 (0.6)

Fonte: Do autor.

APÊNDICE C – POCKETSPHINX EM ROS KINETIC

CMU Sphinx é o termo geral empregado para descrever um grupo de sistemas ASR desenvolvido na CMU. O Sphinx é um sistema ASR de reconhecimento de fala contínua e vocabulário amplo. Na literatura, os modelos implementados em robôs são o Sphinx-4 e o PocketSphinx. Lamere et al. (2003) descreve os principais recursos do decodificador Sphinx-4 e Pantazoglou, Papadakis e Kladis (2017) desenvolvem uma ferramenta baseada em CMU Sphinx para o idioma grego, para tele-operação de dispositivos inteligentes via comandos verbais, a eficácia da abordagem é mostrada através de resultados de simulação para um cenário contendo um veículo operado remotamente, do inglês *Remotely Operated Vehicle* (ROV).

C.1 POCKETSPHINX COM GSTREAMER

O PocketSphinx é uma biblioteca que depende de outra biblioteca chamada SphinxBase, que fornece funcionalidade comum em todos os projetos do CMUSphinx[1]. O PocketSphinx suporta a estrutura de mídia de streaming do GStreamer. O que isso significa é que o decodificador PocketSphinx pode ser tratado como um elemento num *pipeline* de processamento de mídia, especificamente um que filtra o áudio em texto.

C.1.1 GStreamer

O que é o GStreamer? O GStreamer é uma biblioteca para a construção de componentes gráficos de manipulação de mídia. Segundo Taymans et al. (2013), os aplicativos suportados variam de reprodução simples de streaming de áudio ou vídeo a processamento complexo de áudio (mixagem) e vídeo (edição não linear).

Isso é útil porque simplifica muito do trabalho envolvido na criação de aplicativos de fala interativos. Além disso, talvez mais importante, significa que a entrada de fala pode ser integrada a um aplicativo GUI usando o GTK+[2].

C.1.2 Base de Conhecimento do PocketSphinx

Segundo Simonic (2015), a base de conhecimento do PocketSphinx é formada por três elementos: o modelo acústico, o dicionário e, o modelo de linguagem.

[1] https://cmusphinx.github.io/wiki/tutorialpocketsphinx/
[2] https://cmusphinx.github.io/wiki/gstreamer/

O Modelo Acústico contém representações de transformada de Fourier específicas para cada um dos fonemas. Essas representações de referência são comparadas com as características extraídos da entrada real no decodificador (SIMONIC, 2015).

O dicionário é usado para mapear palavras em sequências de elementos do modelo acústico. Normalmente, o dicionário tem a pronúncia *Arpabet*, isto é um código de transcrição fonética, que contém apenas um subconjunto de 39 fonemas reconhecidos pelo IPA[3], encontrados no idioma padrão do inglês. Todo fonema é representado por uma ou duas letras maiúsculas. Além disso, os dígitos são usados como indicadores de estresse e são colocados no final da vogal silábica estressada (SIMONIC, 2015). Assim, por exemplo:

Tabela 13 – Transcrição fonética

Palavra	Transcrição	
	IPA	ARPAbet
backward	ˈbækwərd	B AE K W ER D
forward	ˈfɔrwərd	F AO R W ER D
left	lɛft	L EH F T
right	raɪt	R AY T
stop	stɑp	S T AA P
tiago		T AY AH G OW

Fonte: Do autor.

O modelo de linguagem contém uma representação da probabilidade de ocorrência de palavras. Ele pode ser determinado usando abordagens estatísticas, como N-grams ou abordagens baseadas em gráficos, como gramáticas livres de contexto. Os N-gramas determinam a probabilidade de uma palavra após a outra, dada a presença de N-1 palavras anteriores (SIMONIC, 2015).

O modelo de idioma e o dicionário para o conjunto de palavras necessárias para um sistema de reconhecimento de fala designado pode ser compilado on-line usando o *lmtool*[4] da CMU (SIMONIC, 2015) e (GOEBEL, 2015).

O sistema pode ser treinado para se adaptar a qualquer idioma ou dialeto; no entanto, ele vem com um modelo treinado completo para o inglês americano. O Pocketsphinx usa um modelo de trigrama, que é um modelo N-gram, sendo N = 3. Para os experimentos, o modelo em inglês dos EUA foi escolhido, incluindo um modelo de dicionário e modelo de linguagem.

[3] https://tophonetics.com/
[4] http://www.speech.cs.cmu.edu/tools/lmtool-new.html

C.2 INTEGRAÇÃO DE POCKETSPHINX ATRAVÉS DO ROS

Michael Ferguson[5], implementou reconhecimento de fala usando Pocketsphinx em ROS, distribuição ROS Indigo. Graças a ele foram implementados os comandos de voz para robôs com sistema operacional ROS. Depois disso, para o robô TurtleBot, o Pockecsphinx foi implementado em ROS Kinetic[6]. Mais detalhes sobre a integração do PocketSphinx em ROS podem ser encontrados no livro *ROS By Example* de Goebel (2015).

Neste caso, todos os comandos são implementados com o tópico de velocidade, o qual controla a movimentação da base. Para evitar comandos de voz falsos e interferência de ruído, propõe-se o uso da palavra-chave "tiago". Primeiro, o robô TIAGo aguarda a palavra-chave. Depois, o robô aguarda o comando de voz por tempo definido, neste caso foi usado 5 segundos. Enfim, se o comando de voz estiver correto, a base móvel está executando esse comando (ver Fig. 19).

Para os comandos básico executados pelo robô TIAGo, implementados usando PocketSphinx através do ROS Kinetic[7], é possível associar outros comandos, por exemplo, para o comado *forward* pode ser falado *go forward* ou *move forward*. Dessa forma, para evitar colisões, podemos parar o robô com os comandos de voz *stop* ou *halt*.

[5] https://github.com/mikeferguson/pocketsphinx
[6] https://github.com/sunmaxwll/pocketsphinx
[7] https://github.com/egutierrezhu/pocketsphinx

APÊNDICE D – LOCALIZAÇÃO E NAVEGAÇÃO

O SLAM, siglas em inglês *Simultaneous Localization and Mapping* (SLAM), é uma técnica utilizada por robôs e veículos autônomos para construir um mapa de um ambiente ao mesmo tempo que se localiza. Segundo PAL Robotics, os robôs TIAGo são equipados com um planejador local baseado no método de histograma de campo vetorial, do inglês *Vector Field Histogram* (VFH). O método VFH que foi desenvolvido para prevenção de obstáculos em tempo real para robôs móveis (BORENSTEIN; KOREN et al., 1991), permite a detecção de obstáculos desconhecidos e evita colisões, enquanto simultaneamente guia o robô móvel em direção ao alvo. A proposta mais completa para localização e navegação é o modelo desenvolvido por Pages (2019a), na qual, o robô TIAGo aprende a executar a localização baseada em laser e a navegação autônoma, evitando obstáculos por meio do planejamento de caminho global e local.

D.1 TÓPICO PARA POSIÇÃO E ORIENTAÇÃO DA META

Em matemática, os quaterniões ou quatérnios ($\overline{q} = u + xi + yj + zk$), são uma extensão do conjunto dos números complexos C. Os quais foram descritos pela primeira vez pelo matemático irlandês William Rowan Hamilton em 1843 (HAMILTON, 1844). Suas propriedades, sua álgebra, sua principal aplicação (o operador de rotação), e suas aplicações na robótica são descritas por Corrochano e Sobczyk (2011) e Rosenfeld (2012). Por outro lado, os Ângulos de Euler foram formulados por Leonard Euler para descreverem a orientação de um corpo rígido girante em um espaço euclidiano tridimensional Weisstein (2009).

As posições finais do robô são expressas em coordenadas euclidianas. Porém, para orientação na meta, definimos as orientações como ângulos de Euler, depois os ângulos são convertidos em quaterniões. No entanto, como o primeiro movimento não colocará o robô exatamente na posição e na orientação da meta, é melhor definir as metas em relação ao quadro estático (mapa). Para os experimentos de localização e navegação, Goebel (2015) propõe o cenário quadrado com desvio de obstáculos. Na qual, a sintaxe da mensagem de meta é baseada no tópico *rostopic pub /move_base_simple/goal*.

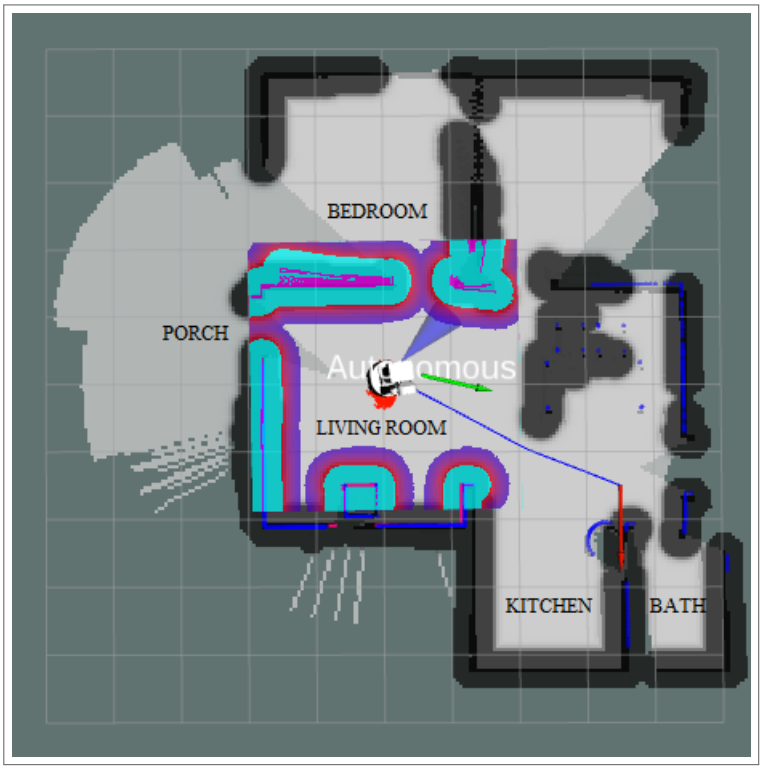
D.2 COMANDOS PARA LOCALIZAÇÃO E NAVEGAÇÃO

Enfim, comandos de voz tipo sentenças foram implementados para localização e navegação[1], comandos tais como *go to the living room* ou *where is the bathroom?*. Para essa finalidade, o ambiente deve ser mapeado, salvar o mapa e, localizar pontos alvos no mapa (ver Fig. 28). No caso ROS Indigo, Goebel (2015) descreve o código em Python para

[1] https://github.com/egutierrezhu/pocketnav

navegação, na qual, usa-se o pacote de localização adaptativa de Monte Carlo, do inglês *Adaptive Monte Carlo Localization* (AMCL). Desse jeito, reescrevendo esse código e integrando com o PocketSphinx, é possível localizar o robô num mapa existente usando os comandos de voz e dados atuais da digitalização de profundidade do robô provenientes do laser.

Figura 28 – Localização e navegação em ambientes domésticos



Fonte: Do autor