UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS
CURSO DE ENGENHARIA ELÉTRICA

PAULO VENÂNCIO RODRIGUES ALVES

FUSÃO SENSORIAL DE UM SONAR E DE UM *LASER SCANNER* UTILIZANDO REDES NEURAIS ARTIFICIAIS

São Luís, MA
2018

PAULO VENÂNCIO RODRIGUES ALVES

FUSÃO SENSORIAL DE UM SONAR E DE UM *LASER SCANNER* UTILIZANDO REDES NEURAIS ARTIFICIAIS

Trabalho de conclusão do curso de Engenharia Elétrica da Universidade Federal do Maranhão para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Professor Dr. Areolino de Almeida Neto

São Luís, MA
2018

Rodrigues Alves, Paulo Venâncio.
FUSÃO SENSORIAL DE UM SONAR E DE UM LASER SCANNER UTILIZANDO REDES NEURAIS ARTIFICIAIS / Paulo Venâncio Rodrigues Alves. - 2018.
65 f.

Orientador(a): Areolino de Almeida Neto.
Monografia (Graduação) - Curso de Engenharia Elétrica, Universidade Federal do Maranhão, São Luís, 2018.

1. Fusão Sensorial. 2. Laser Scanner. 3. Perceptron de Múltiplas Camadas. 4. Redes Neurais. 5. Sonar. I. de Almeida Neto, Areolino. II. Título.

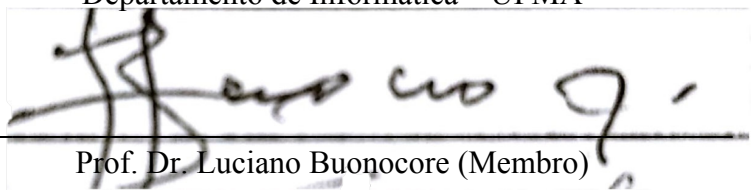
PAULO VENÂNCIO RODRIGUES ALVES

FUSÃO SENSORIAL DE UM SONAR E DE UM *LASER SCANNER* UTILIZANDO REDES NEURAIS ARTIFICIAIS

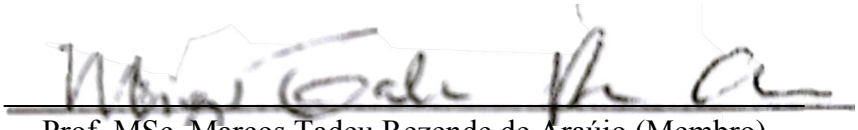
Aprovada em 10/12/2018

BANCA EXAMINADORA

Prof. Dr. Areolino de Almeida Neto (Orientador)
Departamento de Informática – UFMA



Prof. Dr. Luciano Buonocore (Membro)
Departamento de Engenharia Elétrica – UFMA



Prof. MSc. Marcos Tadeu Rezende de Araújo (Membro)
Departamento de Engenharia Elétrica – UFMA

Prof. MSc. Will Ribamar Mendes Almeida (Membro)
Departamento de Engenharia – UNICEUMA

AGRADECIMENTO

Gostaria de agradecer em primeiro lugar a Deus, por sempre cuidar de mim, de minha família e por sempre me proporcionar as melhores situações.

Em seguida, agradeço à minha família por me fornecer as condições necessárias para que eu sempre tivesse uma educação de alto nível, algo que infelizmente não é uma realidade para todos no nosso país. Em especial, agradeço à minha mãe, Vilmária, por me mostrar o real significado e importância da educação desde os meus primeiros anos de vida.

Agradeço todos os professores que participaram da minha formação até aqui, em especial ao Professor Areolino, pela confiança depositada em mim.

Agradeço aos colegas de curso, por estarem comigo durante essa trajetória, compartilhando méritos e também frustrações naturais do processo.

RESUMO

Essa pesquisa tem como objetivo desenvolver um sistema de fusão sensorial baseado em redes neurais artificiais. O trabalho aborda o funcionamento do sensor ultrassônico HC-SR04 e do *laser scanner* RPLIDAR 360, além de algumas técnicas de fusão de dados como seleção ou ponderação, filtro de Kalman e árvore de decisão. No decorrer do trabalho, é definida a configuração da rede neural utilizada no treinamento, sendo esta do tipo *perceptron* de múltiplas camadas de estrutura *feedfoward* com algoritmo de treinamento do tipo *backpropagation*. Além disso, são apresentados os dados utilizados na realização dos treinamentos, com o intuito de a rede aprender a corrigir a média dos valores dos sensores. No final, são apresentados testes de generalização com o objetivo de validar o sistema neural de fusão de dados desenvolvido. Os resultados foram considerados satisfatórios, tendo em vista que, a técnica de fusão sensorial se mostrou mais precisa do que a média das medidas dos sensores em mais de 70% dos casos.

Palavras-chave: Fusão Sensorial, Redes Neurais, Sonar, *Laser Scanner*, *Perceptron* de Múltiplas Camadas

ABSTRACT

This research aims to develop a sensor fusion system based on artificial neural networks. The work addresses the operation of the HC-SR04 ultrasonic sensor and the RPLIDAR 360 laser scanner, as well as some techniques of data fusion, such as selection or weighting, Kalman filter and decision tree. In the course of the work, the configuration of the neural network that is used for training was defined, which is a multilayer perceptron with a feedfoward structure and a backpropagation training algorithm. Also, it is presented the data used for the execution of trainings in order to correct the sensor's mean values. At the end, tests of the generalization are made to validate the data fusion system developed. The results were considered satisfactory, considering that the sensor fusion technique was more accurate than the mean sensor measurements in more than 70% of the cases.

Keywords: Sensor Fusion, Neural Networks, Sonar, Laser Scanner, Multilayer Perceptron.

LISTA DE FIGURAS

Figura 1 – Sonar HC-SR04 16
Figura 2 – Laser Scanner RPLIDAR 360 17
Figura 3 – Exemplo de árvore de decisão 20
Figura 4 – Representação de uma rede neural artificial 21
Figura 5 – Representação de uma rede neural artificial do tipo perceptron 23
Figura 6 – Divisão dos dados de treinamento e generalização 26
Figura 7 – Estrutura desenvolvida para medição 27
Figura 8 – Medição de distância 27
Figura 9 – Interface desenvolvida para o treinamento da rede 28
Figura 10 – Gráficos referentes ao treinamento 1 29
Figura 11 – Gráficos referentes ao treinamento 2 30
Figura 12 – Gráficos referentes ao treinamento 3 31
Figura 13 – Gráficos referentes ao treinamento 4 32
Figura 14 – Erro percentual do teste de generalização 1 33
Figura 15 – Erro percentual do teste de generalização 2 34
Figura 16 – Erro percentual do teste de generalização 3 35
Figura 17 – Erro percentual do teste de generalização 4 36

LISTA DE TABELAS

Tabela 1 – Iterações e erro quadrático médio de cada treinamento 28
Tabela 2 – Dados obtidos no treinamento 1 29
Tabela 3 – Dados obtidos no treinamento 2 30
Tabela 4 – Dados obtidos no treinamento 3 31
Tabela 5 – Dados obtidos no treinamento 4 32
Tabela 6 – Dados obtidos no teste de generalização 1 34
Tabela 7 – Dados obtidos no teste de generalização 2 35
Tabela 8 – Dados obtidos no teste de generalização 3 36
Tabela 9 – Dados obtidos no teste de generalização 4 37

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Objetivos 11
1.1.1 Objetivo Geral 12
1.1.2 Objetivos Específicos 12
1.2 Motivação 12
1.3 Metodologia 12
1.4 Organização do Trabalho 13
2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 Sensores 14
2.1.1 Sonar 15
2.1.2 *Laser Scanner* 16
2.2 Técnicas de Fusão de Sensorial 17
2.2.1 Média Ponderada e Seleção 18
2.2.2 Filtro de Kalman 18
2.2.3 Árvore de Decisão 19
2.3 Redes Neurais Artificiais 20
2.3.1 Processo de Aprendizagem 21
2.3.2 Algoritmos de Treinamento 22
3 EXPERIMENTOS E RESULTADOS 25
3.1 Levantamento das Variáveis de Entrada e Saída 25
3.2 Definição da Rede para Treinamento 26
3.3 Treinamento 28
3.4 Validação do Sistema 28
4 CONCLUSÕES E TRABALHOS FUTUROS 38
BIBLIOGRAFIA 39
APÊNDICE B – CÓDIGO PARA AQUISIÇÃO DE DADOS USANDO O *LASER SCANNER* E ARDUINO 43
APÊNDICE C – CÓDIGO EM MATLAB PARA TREINAMENTO DA REDE NEURAL 44
APÊNDICE D – CÓDIGO EM MATLAB PARA TESTE DE GENERALIZAÇÃO DA REDE NEURAL 56

1 INTRODUÇÃO

O emprego dos sensores mostra-se de fundamental importância na contemporaneidade. Os sensores são utilizados de forma vasta, seja na indústria, para controlar ou otimizar a produção, seja na sociedade, uma vez que a população tem nos meios de transportes, nos serviços de saúde e entre outros, o uso de sensores.

Dentro do campo da robótica, os sensores têm um papel imprescindível, uma vez que um robô necessita de informações relativas ao ambiente de navegação. Através desses dados, é possível explorar o terreno, traçar uma trajetória segura e eficaz, localizar obstáculos, além de realizar mapeamentos (FACELI, 2001).

No entanto, como qualquer dispositivo, os sensores estão sujeitos a intempéries causados por fatores externos (FRADEN, 1993) como temperatura, pressão e umidade. Além disso, determinados tipos de materiais utilizados na sua construção podem limitar a sensibilidade e sua faixa de operação. Podem ocorrer também problemas na calibração e ruído. Tais condições comprometem o resultado de suas medições, tornando sua aplicação inadequada em alguns casos.

Nesse contexto, existe a necessidade da utilização de técnicas que tornem as medições mais confiáveis. Uma forma de tornar mais próximos do real os resultados de uma medição é utilizando o método da fusão sensorial (RAMOS, 2012). A ideia é utilizar dois ou mais sensores de mesmo tipo ou tipos diferentes na mesma aplicação com o objetivo de adquirir uma medida mais precisa, através do processamento dos dados dos sensores.

Para a realização da fusão sensorial, inicialmente se pode utilizar técnicas básicas. Estas baseiam-se em procedimentos matemáticos e consistem basicamente em fazer a ponderação das medidas efetuadas ou uma seleção delas.

No entanto, para obtenção de medidas mais precisas, é conveniente que se utilizem técnicas modernas, ou seja, aquelas baseadas em aprendizado de máquina. E mais recentemente, tem surgido o uso de técnicas de inteligência artificial, assim neste trabalho, é proposto um modelo de fusão sensorial através de redes neurais artificiais.

1.1 Objetivos

Os objetivos estão organizados em geral e específicos como segue.

1.1.1 Objetivo Geral

Desenvolver um método de fusão sensorial de um sonar e de um *laser scanner*, utilizando redes neurais artificiais, a fim de comprovar que tal técnica produz resultados métricos mais próximos do valor real, quando comparada a uma simples média aritmética dos dados de medição provenientes dos sensores.

1.1.2 Objetivos Específicos

- Realizar a fusão dos dados proveniente de um sonar e de um *laser scanner*;
- Validar o método de fusão sensorial a ser desenvolvido;
- Aumentar a precisão de medições realizadas através da fusão de sensores.

1.2 Motivação

A fusão de dados consiste basicamente na determinação de qual deve ser o valor da distância até um obstáculo, uma vez que essa distância é medida por dois ou mais sensores. Quando as medidas dos sensores são iguais, acredita-se que esta deva ser de fato a distância até o obstáculo. Porém, na maioria dos casos, as medidas não coincidem, tornando-se necessário decidir qual o valor correto. E para isto, pode-se fazer uma ponderação das medidas realizadas ou uma seleção delas.

Assim, a fusão de dados consiste inicialmente em definir se a fusão dar-se-á por ponderação ou por seleção. No entanto, ponderar ou selecionar uma medida pode ser um método com baixa precisão, fazendo-se necessário utilizar uma técnica inteligente para a aquisição de dados mais precisos.

Portanto, neste trabalho aplica-se uma técnica de inteligência artificial, mais precisamente redes neurais artificiais, a fim de verificar que tal técnica fornece resultados mais precisos se comparados a uma simples média ou ponderação.

1.3 Metodologia

Para a realização do projeto, foram necessários um exemplar do sensor ultrassônico HC-SR04 e um *laser scanner* RPLIDAR 360. A obtenção das medidas foi realizada utilizando-se o microcontrolador Arduino MEGA 2560 R3. Para a aquisição de dados, foi construída uma estrutura física para comportar os sensores e realizar medições em distâncias pré-determinadas. As etapas de treinamento e teste de generalização foram feitas através do *software* Matlab R2018a.

As atividades iniciaram-se com um estudo dos sensores que seriam utilizados neste projeto. Posteriormente, iniciou-se um estudo sobre alguns tipos de fusão de dados encontrados na literatura. Em seguida, foi feito uma pesquisa sobre redes neurais artificiais com o intuito de definir a estrutura do sistema neural de fusão de dados. Com a estrutura definida, foi possível realizar os treinamentos da rede a fim de aprender uma fusão de dados apropriada. Uma vez o aprendizado sendo adquirido, depois do treinamento da rede, foram realizados testes de generalização para a validação do sistema desenvolvido.

1.4 Organização do Trabalho

O trabalho está organizado em quatro capítulos, incluindo este capítulo introdutório.

O Capítulo 2 é dedicado à fundamentação teórica. São abordados temas como o funcionamento dos sensores utilizados, diferentes tipos de técnicas de fusão de dados presentes na literatura, além da fundamentação sobre redes neurais artificiais, que são a base deste trabalho.

O Capítulo 3 é reservado para a etapa de experimentos e resultados. Nessa parte, é feito o levantamento das variáveis de entrada e saída da rede neural desenvolvida, além de ser abordado o processo de treinamento e de validação do sistema.

Por fim, no Capítulo 4, são apresentadas as conclusões sobre o projeto e eventuais trabalhos futuros. Este trabalho ainda contém apêndices onde são expostos os códigos utilizados para a aquisição de dados, treinamento e validação do sistema desenvolvido.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo fornece todo o embasamento teórico para a realização deste trabalho. Serão abordados os princípios de funcionamento dos sensores utilizados e técnicas de fusão de dados variadas. Além disso, é apresentada a técnica que foi utilizada neste projeto, mais precisamente, redes neurais artificiais.

2.1 Sensores

O advento dos sensores trouxe diversas vantagens e comodidades para a vida contemporânea. Desde a possibilidade de otimizar o funcionamento de um motor ou de uma linha de produção, ou realizar um trabalho cientifico com maior confiabilidade e em menor tempo, facilidade ao estacionar o carro em uma vaga na rua, tais exemplos mostram a utilidade oferecidas pelo uso de sensores (PATSKO, 2016). Os sensores são imprescindíveis no funcionamento das indústrias, pois cada etapa de um processo precisa ser minuciosamente monitorada para garantir a segurança e bem-estar do operador, além qualidade do produto final ou processo (GODOY *et al*., 2016).

O sensor é um dispositivo que responde a estímulos ou variações de grandezas físicas ou químicas, por exemplo, com efeito correspondente e transmite um sinal mensurável (GODOY *et al*., 2016).

Os sensores podem ser de dois tipos: passivos ou ativos (FRADEN, 1993). Os sensores passivos geram um sinal em resposta a um estímulo externo, onde a energia de entrada é convertida em uma energia de saída sem a necessidade de uma fonte externa. Já os sensores ativos precisam de energia externa para seu funcionamento.

Os sensores possuem precisão limitada pois estão sujeitos a ruídos e desgastes ao longo tempo, o que pode gerar falhas. Assim, a utilidade dos sensores depende da sua aplicação, considerando algumas características, como: tempo de operação, estabilidade, alcance, faixa de operação, precisão e resolução (FACELI, 2001).

É muito vasta a diversidade de sensores. Esses dispositivos podem ser classificados de acordo com as suas propriedades (FRADEN, 1993):

- Sinal de estímulo: o que será medido, de grandeza acústica, elétrica, mecânica, entre outros;
- Especificações ou características: podem ser precisão, calibração, seletividade, linearidade, tamanho, velocidade de resposta e outros;
- Fenômeno físico: indica a qual fenômeno físico é sensível;

- Mecanismo de conversão: fenômeno utilizado na conversão, como termoelétrico, eletromagnético, transformação química;
- Material de fabricação: por exemplo, condutor, semicondutor, orgânico;
- Campo de aplicação: por exemplo, comércio, agricultura, militar, segurança, robótica, sensoriamento remoto.

2.1.1 Sonar

O sonar ou sensor ultrassônico é utilizado para medir distância através da duração que um sinal, nesse caso uma onda sonora, leva para ir até um objeto e ser refletido de volta ao emissor (PRESTES, 2012).

Para determinar a distância percorrida pelo sinal, é utilizada a fórmula

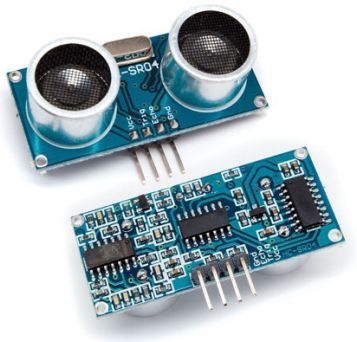
$$\Delta S = V\,\Delta t \qquad (1)$$

onde ΔS é duas vezes a distância entre o emissor e o objeto, V é a velocidade do som e Δt é o tempo que a onda leva para atingir o objeto e retornar ao emissor.

Os sensores ultrassônicos são amplamente utilizados em sistemas robóticos (MURPHY, 2000) seja na modelagem de ambiente, identificação de obstáculos e posicionamento, além de serem bastante disponíveis, pois são de baixo custo e fáceis de usar

O sonar também está sujeito a erros nas medições (BORESTEIN *et al.*, 1996) devido a variações na velocidade de propagação em função de fatores ambientais, como temperatura e umidade. Incertezas na determinação exata do tempo de chegada no pulso refletido, devido à diminuição de energia do pulso emitido, também são fontes de erros. Falta de precisão no circuito de temporização usado para medir o tempo de ida e volta do pulso é outra possibilidade de geração de erro. A energia restante pode dispersar-se em outras direções ou ser absorvida pela superfície do alvo, acarretando em mais incertezas. Na Figura 1 é apresentado o sonar utilizado neste projeto.

Figura 1 – Sonar HC-SR04



Fonte: http://www.hobbyandyou.com/hc-sr04-ultrasonicsonar-distance-measuring-sensor-module-for-arduino

2.1.2 *Laser Scanner*

O *laser scanner* gera coordenadas de pontos sobre uma superfície. Seu princípio de funcionamento é de certa forma simples. Os pulsos de *laser* são gerados e emitidos pelo sistema e, com auxílio de um espelho de varredura, são direcionados e acabam atingindo a superfície dos objetos em vários pontos. Estes objetos refletem o pulso incidente e parte do pulso volta para o sistema. Através desse princípio, a distância entre o sensor e o objeto é determinada através do intervalo de tempo entre a emissão e a reflexão (retorno) do pulso (DALMOLIN & SANTOS, 2004).

O *laser scanner* usa um feixe óptico de alta potência com baixa divergência para definir a distância entre o sensor e os objetos. A faixa do espectro utilizada é condicionada, pois devido à alta potência da energia utilizada, o feixe pode ser nocivo para os olhos humanos. O sistema de varredura *laser* é dividido em três partes principais: a unidade de medição *laser* propriamente dita, encarregada de emitir e receber o sinal, um sistema de varredura e uma unidade de registro de medições (BALTSAVIAS, 1999). A Figura 2 mostra o *laser scanner* utilizado neste projeto.

A varredura *laser* apresenta várias características importantes, como cita TOMMASELLI (2003):

- É um método ativo que não depende da luz visível refletida, embora alguns modelos de *scanners* apresentem comportamento inadequado na ausência total de iluminação;
- Operação remota, o que significa que o objeto não precisa ser tocado;

- O princípio geométrico de cálculo das coordenadas é a triangulação, o intervalo de tempo ou a diferença de fase, dependendo do modelo, mas em todos os casos a varredura pode ser feita com apenas uma estação por visada;
- A resposta está disponível em tempo real, após o término da varredura o operador tem à sua disposição milhões de pontos com coordenadas conhecidas, estando apto a fornecer respostas sobre os objetos, como distâncias entre peças, dimensões, volumes, verticalidade de superfícies, etc;
- Alta densidade de pontos coletados e, conseqüentemente, altíssima redundância na descrição dos objetos;
- É possível realizar o controle de qualidade durante a coleta e refazer a varredura, caso necessário;
- Operação simples e flexível; basta um operador para posicionar e operar o sistema;
- É possível combinar vários modelos numéricos gerados de diferentes posições, o que permite cobrir quase toda superfície visível dos objetos.

Figura 2 – *Laser Scanner* RPLIDAR 360

Fonte: https://wholesaler.alibaba.com/product-detail/RPLIDAR-A1-360-degree-Laser-Scanner_60705323403.html

2.2 Técnicas de Fusão de Sensorial

A utilização de dois ou mais sensores para adquirir dados em uma determinada aplicação se mostra cada vez mais presente. Assim, a necessidade de obter dados seguros e conhecimento baseado nestes dados faz com que técnicas de processamento cresçam em importância. A fusão de sensores é um conceito que resume a integração de dados provenientes de diferentes dispositivos para fornecer aos sistemas dados com maior precisão (DUFFY, 2000; MURPHY, 1994). Alguns pontos positivos da fusão são:

- Melhor tolerância a falhas devido ao uso de vários sensores;

- Alta confiabilidade, pelo fato da leitura de um sensor poder ser confirmada por outras unidades;
- Combinação de dados, para sensores que determinem diferentes aspectos do ambiente e fornece uma informação nova;
- Redução de ambiguidade e incerteza;
- Robustez contra interferências;
- Custo de processamento computacional ou material pode ser menor ao utilizar esse tipo de técnica;

As técnicas de fusão sensorial podem ser utilizadas em diversas aplicações. Em robótica, por exemplo, são utilizados para determinação de trajetórias, reconhecimento de alvos, mapeamento de ambiente, cálculo de distância do robô ao alvo de maneira mais precisa, entre outros (DUFFY, 2000; FACELI, 2001).

2.2.1 Média Ponderada e Seleção

São técnicas que se baseiam em procedimentos matemáticos básicos. A ponderação mostra-se um método simples que consiste no cálculo de pesos que ponderem os dados redundantes provenientes de vários sensores. Nesse caso, os sensores podem ter o mesmo peso ou pesos diferentes.

Já no caso da seleção, as medições oriundas dos sensores são analisadas, pois dependendo da situação, um sensor pode não conseguir realizar uma medição ou adquirir um dado muito impreciso, nesses casos, tal medida seria descartada. Portanto, tal método consiste em uma escolha de qual medição será utilizada.

2.2.2 Filtro de Kalman

O filtro de Kalman foi desenvolvido na década de 1960, sendo inicialmente aplicado na área de engenharia elétrica, relacionada à teoria de controle de sistemas. Posteriormente, ele passou a ser aplicados em outras áreas, como por exemplo, inteligência artificial e robótica (MURPHY, 2000).

Este filtro consiste em um conjunto de equações matemáticas que constitui um algoritmo recursivo e eficiente para estimação, uma vez que o erro quadrático é minimizado. O filtro de Kalman é um estimador imediato de determinado estado perturbado por ruído, em um sistema dinâmico linear. Mostra-se eficiente com relação a qualquer função quadrática de erros de estimativa (GREWAL & ANDREWS, 2014), na qual suas equações formam um processo

recursivo capaz de diminuir a soma dos quadrados das diferenças entre os valores medidos e os estimados. Este método é considerado um propagador de distribuição probabilística, tendo em vista que fornece uma caracterização global do atual estado de um dado sistema, incluindo referências passadas, sem necessitar de valores anteriores.

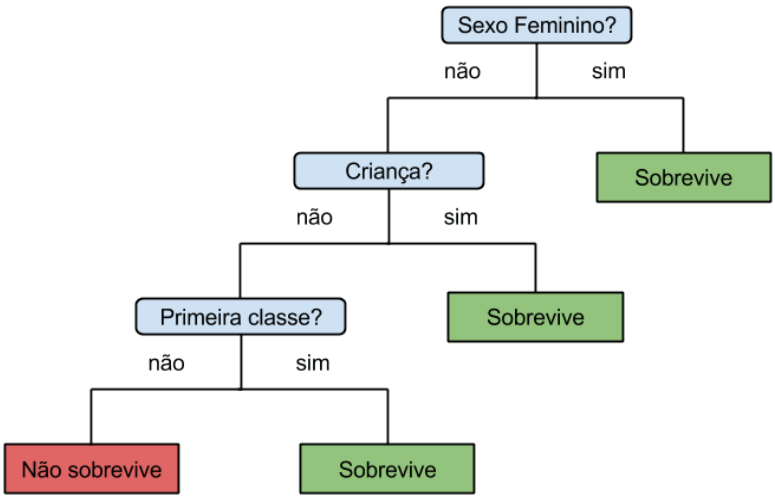
O método é constituído das fases de predição e atualização. Essas duas etapas funcionam em conjunto, uma alimentando a outra. A predição, que também é conhecida como estimativa *a priori,* estima o estado atual apenas com dados estimados até o passo anterior, não incluindo dados observados no tempo atual. Em seguida, o estado atual é atualizado, a estimativa *a priori* é corrigida com a observação do tempo atual, resultando em uma estimativa combinada, conhecida como estimativa *a posteriori* (AIUBE, 2005).

2.2.3 Árvore de Decisão

Uma árvore de decisão é uma técnica que pode ser utilizada para, por meio de uma simples regras de decisão, dividir sucessivamente uma grande coleção de registros em conjuntos menores. Ao longo de cada divisão realizada, os dados são divididos de acordo com características em comum até chegar a pontos indivisíveis, que representam as classes. Neste caso, um problema complexo é decomposto em subproblemas (TAN *et al.,* 2009)

Cada nó da árvore representa um teste que será realizado e as arestas definem um caminho para cada resultado desses testes. O nó raiz é aquele que não possui nenhuma aresta de entrada, havendo zero ou mais arestas na saída. Os nós intermediários possuem apenas uma aresta de entrada e duas ou mais arestas de saída. Os nós folhas da árvore são os pontos indivisíveis, os quais representam as classes (HOSOKAWA, 2011). A Figura 3 mostra um exemplo de árvore de decisão tendo como temática o filme *Titanic* (1997), onde durante o naufrágio, foram selecionadas as pessoas que se salvariam do desastre.

Figura 3 – Exemplo de árvore de decisão



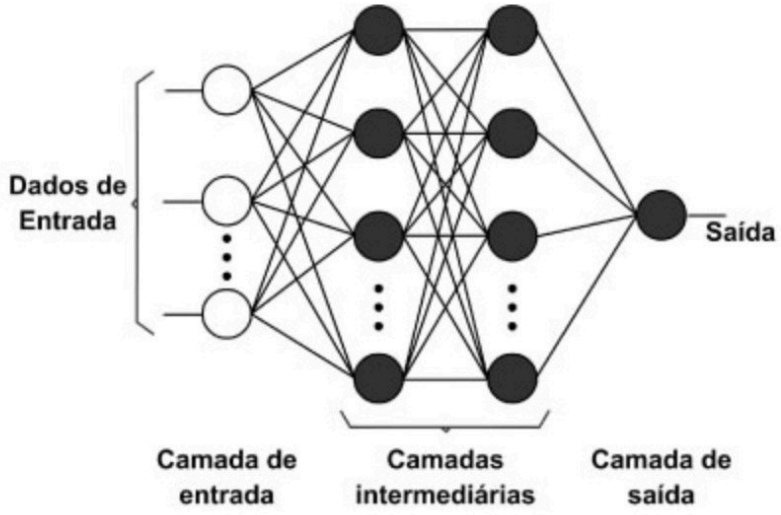
Fonte: http://blog.caelum.com.br/r-titanic-e-data-science/

2.3 Redes Neurais Artificiais

As redes neurais artificiais constituem um tipo de metodologia voltada para a solução de problemas por meio de um aprendizado, a partir de um sistema que baseia-se no cérebro humano, isto é, seu comportamento, ou seja, aprendendo, errando e fazendo descobertas (AZEVEDO *et al.,* 2000). Em outras palavras, uma rede neural artificial é uma técnica computacional que representa uma estrutura neural própria de organismos inteligentes, onde o conhecimento é adquirido através da experiência (GUIMARÃES *et al.,* 2007). A facilidade das redes neurais artificiais em abordar problemas complicados tornou-as uma importante opção na solução de problemas ligados a engenharia e ciências (SILVA et al., 2001).

Uma rede neural artificial pode ser representada através de um grafo direcional pois é uma estrutura de processamento de dados distribuída de forma paralela, onde os nós correspondem aos neurônios e as conexões, ou arestas, funcionam como as sinapses, sendo atribuída a elas os pesos (GUIMARÃES *et al.,* 2007). Também é possível representar a arquitetura da rede por camada de entrada, camada(s) escondida(s) ou oculta(s) e camada de saída, conforme é exemplificado na Figura 4.

Figura 4 – Representação de uma rede neural artificial



Fonte:http://www.scielo.br/scielo.php?script=sci_arttext&pid=S180611172011000100009

2.3.1 Processo de Aprendizagem

Aprendizado pode ser definido como a habilidade natural ou capacidade de se adaptar, modificar e melhorar o comportamento e as suas reações, sendo, portanto, um dos aspectos mais importantes dos seres inteligentes, sejam eles humanos, ou não (OSÓRIO & BITTENCOURT, 2000). As redes neurais artificiais buscam uma reprodução computacional deste comportamento. Em se tratando desse aprendizado, pode-se definir como sendo o procedimento em que os parâmetros livres de uma rede neural são adaptados por meio de um processo de estimulação pelo ambiente onde a rede está inserida, sendo que o tipo de aprendizagem é determinado conforme o modo como a modificação dos parâmetros ocorre (HAYKIN, 2001).

O aprendizado é um procedimento gradual e iterativo, onde os pesos são alterados várias vezes, obedecendo-se uma regra de aprendizado que estabelece a maneira como estes pesos são atualizados. O aprendizado realiza-se utilizando um conjunto de dados de disponível. O treinamento da rede ocorre em etapas, que são conhecidas como épocas ou ciclos, os quais correspondem ao número de iterações do processo, onde são aplicados os dados de entrada do treinamento, para diminuir o erro médio (GUIMARÃES *et al.,* 2007).

Existem alguns tipos de aprendizado na literatura, dois deles são o supervisionado e não supervisionado. O aprendizado supervisionado é realizado sob uma supervisão externa que tem a função de acompanhar a resposta da rede, para cada entrada, conhecendo preliminarmente a

saída desejada. O aprendizado não supervisionado é baseado na auto-organização, não existindo o conhecimento das saídas esperadas para as dadas entradas (HAYKIN, 1994).

Exemplos típicos de problemas que utilizam o aprendizado supervisionado são a aproximação de funções, modelagem de sistemas e classificação de dados. O aprendizado não supervisionado aplica-se geralmente a problemas de categorização de dados (GUIMARÃES *et al.,* 2007).

As redes necessitam de uma grande quantidade de dados históricos, para conseguirem extrair de forma satisfatória as características fundamentais existentes no conjunto de dados. Tendo o treinamento correto, a rede é capaz, não somente de aproximar qualquer função, mas também de generalizar, proporcionando saídas corretas para entradas não conhecidas anteriormente. Um sistema que tem boa generalização é aquele que responde corretamente aos exemplos contidos na fase de treinamento e também a outros exemplos diferentes dos usados durante o processo de aprendizagem, estando contidos em um conjunto de teste (ROHN & MINE*,* 2003).

Deste modo, a capacidade de generalizar é a principal vantagem de uma rede, porém, pode acontecer de uma rede especializar-se de forma demasiada em relação aos exemplos contidos na base de aprendizado. Este tipo de comportamento gera uma adversidade, conhecida como super-aprendizado ou super-ajuste (*over-training / over-f*itting). Outra situação que pode ocorrer é a rede não conseguir generalizar, gerando o problema chamado de sub-ajuste (*under-fitting*).

2.3.2 Algoritmos de Treinamento

O treinamento de uma rede neural é o ponto onde se alcança o êxito ou a reprovação da rede, pois, neste procedimento, submete-se a rede ao aprendizado, onde alguns fatores são relevantes, dentre eles: o algoritmo de treinamento e o número de épocas (ciclos ou iterações) (GUIMARÃES *et al.,* 2007).

Umas das primeiras redes neurais artificiais, cuja arquitetura foi inspirada em um neurônio biológico, foi o *perceptron*. O objetivo desta rede é classificar as entradas x_i em duas classes através de um hiperplano (BRAGA, 2007). Para o caso simples de um espaço em duas dimensões, o hiperplano fica reduzido a uma reta, cuja equação é representada por:

$$y_i = \varphi\left(\sum x_i\, w_i + w_0\right) \tag{2}$$

Onde,

y_i = Saída

x_i = Entradas da rede

w_i = Pesos sinápticos

w_0 = Bias

A ativação do neurônio artificial é realizada através da função de ativação φ, a qual desempenha função similar à sinapse no neurônio biológico, transmitindo ou bloqueando os impulsos nervosos (HAYKIN, 2001).

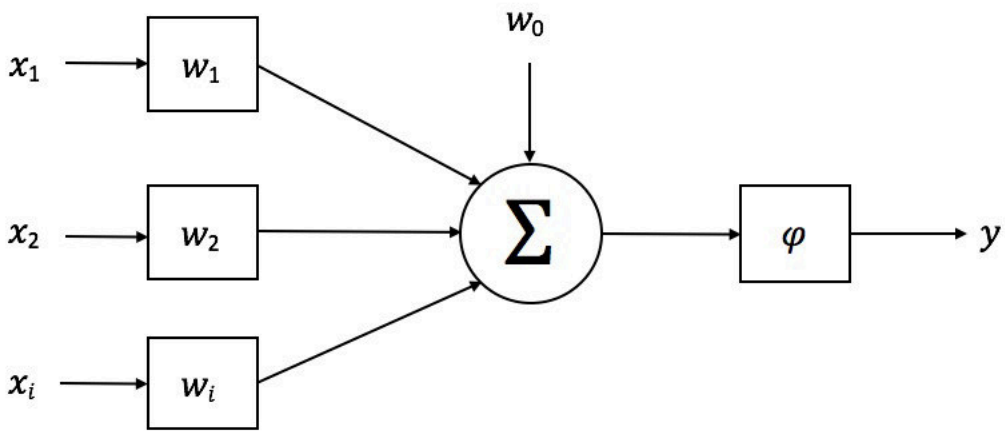
De uma forma geral, a aprendizagem das redes *perceptron* ocorre através da atualização dos pesos sinápticos. O valor do peso sináptico w_i^{t+1}, no passo $t+1$, será determinado em função do seu valor na iteração anterior w^t pela equação

$$w_i^{t+1} = w_i^t + \Delta w_i^t \quad (3)$$

A atualização dos pesos depende do algoritmo, mas geralmente baseia-se na minimização do erro, entre os valores previstos pela rede e as saídas y_i desejadas.

Desta forma, o aprendizado (ou treinamento) em uma rede neural artificial é definido como o ajuste iterativo dos pesos sinápticos, de forma a minimizar o valor do erro. A Figura 5 mostra o esquema de uma rede do tipo *perceptron.*

Figura 5 – Representação de uma rede neural artificial do tipo *perceptron*



Fonte: Autor.

As redes neurais *perceptron* de múltiplas camadas surgem no intuito de solucionar problemas não lineares. Ela permite aproximar funções e solucionar problemas não linearmente separáveis transformando em um problema linearmente separável (HAYKIN, 2001).

A estrutura de uma rede *perceptron* de múltiplas camadas é de uma rede *feedforward,* ou seja, os valores de saída de uma camada são utilizados como entrada para a camada posterior (BRAGA, 2007). Sendo de aprendizado supervisionado, o conjunto de valores de saída da rede

é comparado com os valores desejados. Assim, a rede aprende através do ajuste dos parâmetros baseado no erro.

$$e_i = d_i - y_i \tag{4}$$

Onde,

e_i = Erro da saída

d_i = Valor desejado da saída

y_i = Saída

O aprendizado da rede segue o algoritmo *backpropagation*, também conhecido como retropropagação de erros. O princípio de aprendizagem deste algoritmo é estimar o erro das camadas intermediárias através de uma estimativa do efeito que estas causam no erro da camada de saída (HAYKIN, 2001). Esse erro é usado para ajustar os pesos das conexões de cada camada utilizando o gradiente descendente segundo a equação

$$\Delta w_{ij} = \eta\ e_j\ x_i \tag{5}$$

em que η é a taxa de aprendizado, e_j é o erro de saída do neurônio j e x_i é a entrada deste neurônio i. O produto destes valores, ∇w_{ij}, é a variação dos pesos entre os neurônios j e i. O aprendizado da rede acontece a cada iteração onde os valores são acrescidos desta variação.

A minimização do erro indica que a rede otimizou os valores dos pesos e a rede alcançou o aprendizado. A rede neural treinada pode ser utilizada para aproximar valores de uma função para valores de entrada diferentes dos valores apresentados durante o treino da rede. Esta atividade é denominada generalização da rede.

3 EXPERIMENTOS E RESULTADOS

3.1 Levantamento das Variáveis de Entrada e Saída

Tendo em vista que o treinamento da rede é supervisionado, ela aprende a partir de dados de entrada e saída. Os dados de entrada correspondem aos dados adquiridos com as medições feitas pelos sensores. A saída esperada da rede representa uma correção da média, ou seja, a saída é a diferença entre o valor real (distância medida com a trena) e a média das medidas dos sensores (sonar e *laser scanner*), que é dado por:

$$\text{Saída} = \text{Valor}_{\text{Real}} - \frac{Sensor\ 1 + Sensor\ 2}{2} \tag{6}$$

Foram realizadas coletas de dados entre 20 a 297 centímetros de distância. As posições de medição variavam de forma aleatória, ou seja, foram feitas medições com 20, 24, 27, 30, 35... 297 centímetros de distância. A cada posição eram feitas 70 medições com cada sensor. Após a aquisição, os dados passavam por um filtro, para que as medidas que se distanciassem muito do valor real fossem descartadas.

Ao todo, foram coletadas 4200 medidas, de 60 posições diferentes. Após o filtro, restaram 3988 medidas adequadas para o uso, fazendo-se necessário descartar 212 medidas. No entanto, a fim de ter um número igual de medidas por posição, optou-se por utilizar 3600 medidas, sendo 60 medidas por posição.

O sonar manteve-se fixo na estrutura, objetivando coletar dados em uma direção apenas. Já o *laser scanner* coleta dados em 360 graus, contudo foram utilizados os dados provenientes de um feixe de 10 graus, formando assim um "cone", similar à onda acústica do sonar, objetivando também adquirir dados em uma direção apenas.

Para fins de treinamento e generalização, as 3600 medidas foram divididas em quatro grupos: A, B, C e D. Cada grupo possuindo 15 medidas de cada posição. Sendo assim, foram realizados quatro treinamentos e quatro testes de generalização. De forma que, os dados de treinamento possuíam, cada um, 2700 medidas provenientes de três grupos e os dados de generalização possuíam, cada um, 900 medidas provenientes de um único grupo. A Figura 6 ilustra a forma na qual a divisão dos dados de treinamento e generalização foi realizada.

Figura 6 – Divisão dos dados de treinamento e generalização

Fonte: Autor

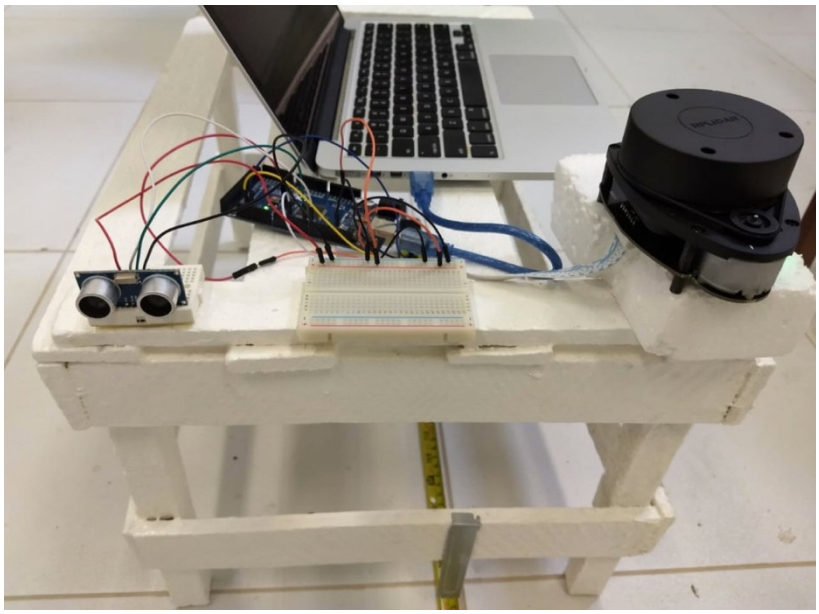
3.2 Definição da Rede para Treinamento

Nesse trabalho, desenvolveu-se uma rede neural artificial simples do tipo *perceptron* de múltiplas camadas de estrutura *feedfoward* com algoritmo de treinamento do tipo *backpropagation* através do *software* Matlab. O treinamento foi realizado de forma supervisionada.

A estrutura da rede implementada conta com dois neurônios na camada de entrada e um na camada de saída. A camada intermediária detém uma quantidade arbitrária de neurônios. No caso deste trabalho, foram testadas diversas configurações de quantidades nessa camada, variando desde três, até 700 neurônios. A configuração que obteve os melhores resultados continha 80 neurônios, havendo ainda *bias* nessa camada e na camada de saída. Também de forma empírica, determinou-se a taxa de aprendizagem. Foram testados valores entre 1×10^{-8} e 1×10^{-4}. O valor que obteve os melhores resultados foi $\eta = 1 \times 10^{-7}$. Os neurônios intermediários utilizaram a função sigmoidal como função de ativação, enquanto que na camada de saída foi utilizada a função linear.

Para a realização do experimento, foi necessária a construção de uma estrutura que comportasse o sonar e o *laser scanner*. A Figura 7 mostra a estrutura desenvolvida.

Figura 7 – Estrutura desenvolvida para medição



Fonte: Autor

Através de tal configuração, foi possível realizar medidas da distância entre uma parede do laboratório e a estrutura montada, como mostra a Figura 8. Entre a parede e a estrutura não haviam obstáculos. Os valores adquiridos eram armazenados em um banco de dados juntamente com o seu valor real, que por sua vez, era medido com o auxílio de uma fita métrica (trena).

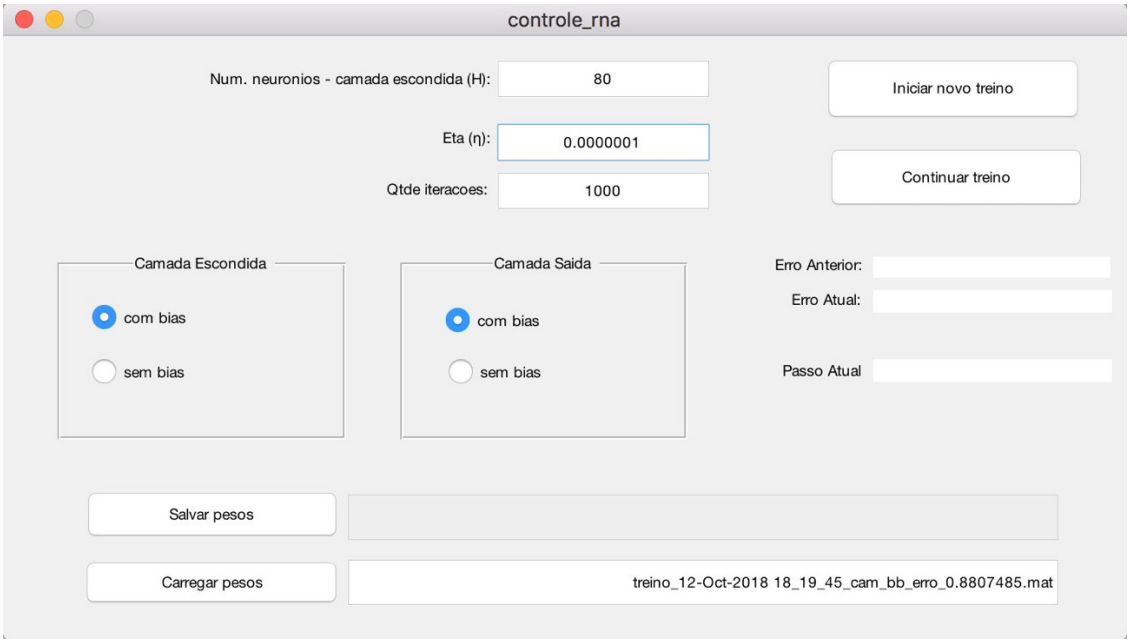
Figura 8 – Medição de distância

Fonte: Autor

3.3 Treinamento

Uma interface foi desenvolvida e é apresentada na Figura 9. Esta interface e a rede neural foram implementadas utilizando a ferramenta Matlab. Através desta interface é possível salvar o treinamento atual (através do salvamento dos pesos) ou carregar o treinamento anterior, acionando no botão "Carregar Pesos". Também é possível acompanhar a redução do erro ao longo das iterações através dos campos "Erro Atual" e "Erro Anterior".

Figura 9 – Interface desenvolvida para o treinamento da rede



Fonte: Autor

A Tabela 1 traz algumas informações sobre os treinamentos: quantidade de iterações e erro quadrático médio final de cada treinamento.

Tabela 1 – Iterações e erro quadrático médio de cada treinamento

	Erro Quadrático Médio	**Quantidade de Iterações**
Treinamento 1	0,8807485	5812000
Treinamento 2	0,8559455	22911000
Treinamento 3	0,7626144	8052000
Treinamento 4	0,8362051	3311000

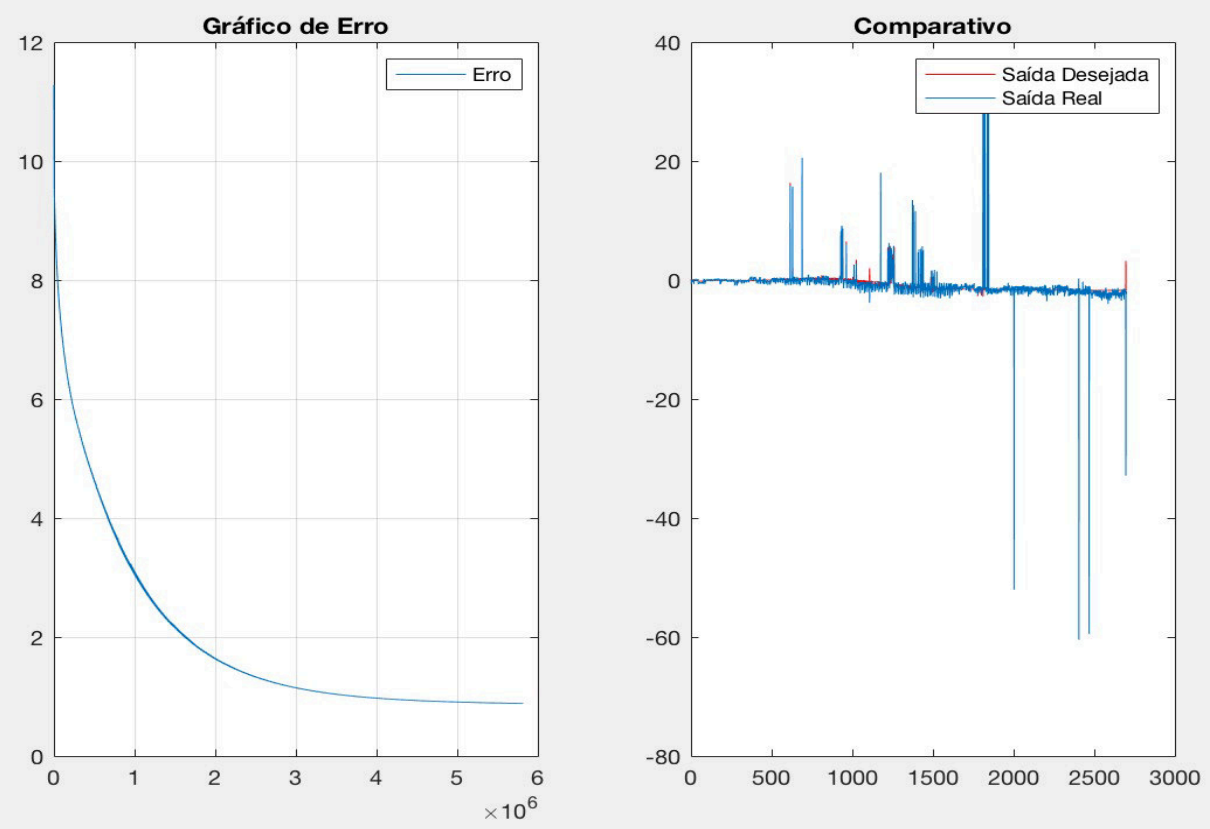
Fonte: Autor

3.4 Validação do Sistema

Após os treinamentos, foram gerados os gráficos do decaimento do erro ao longo das iterações e o gráfico da saída comparada à saída desejada da rede. As Figuras 10, 11, 12 e 13 mostram os gráficos dos treinamentos 1, 2, 3 e 4, respectivamente.

As Tabelas 2, 3, 4 e 5 apresentam alguns dados quantitativos sobre os treinamentos citados. Nessas tabelas, são apresentadas as medições que obtiveram: melhor média entre as medidas obtidas dos sensores, pior média destes valores, melhor e pior resultados utilizando a rede neural. Em cada treinamento, foram utilizadas 2700 medidas extraídas de 60 posições.

Figura 10 – Gráficos referentes ao treinamento 1



Fonte: Autor

Tabela 2 – Dados obtidos no treinamento 1

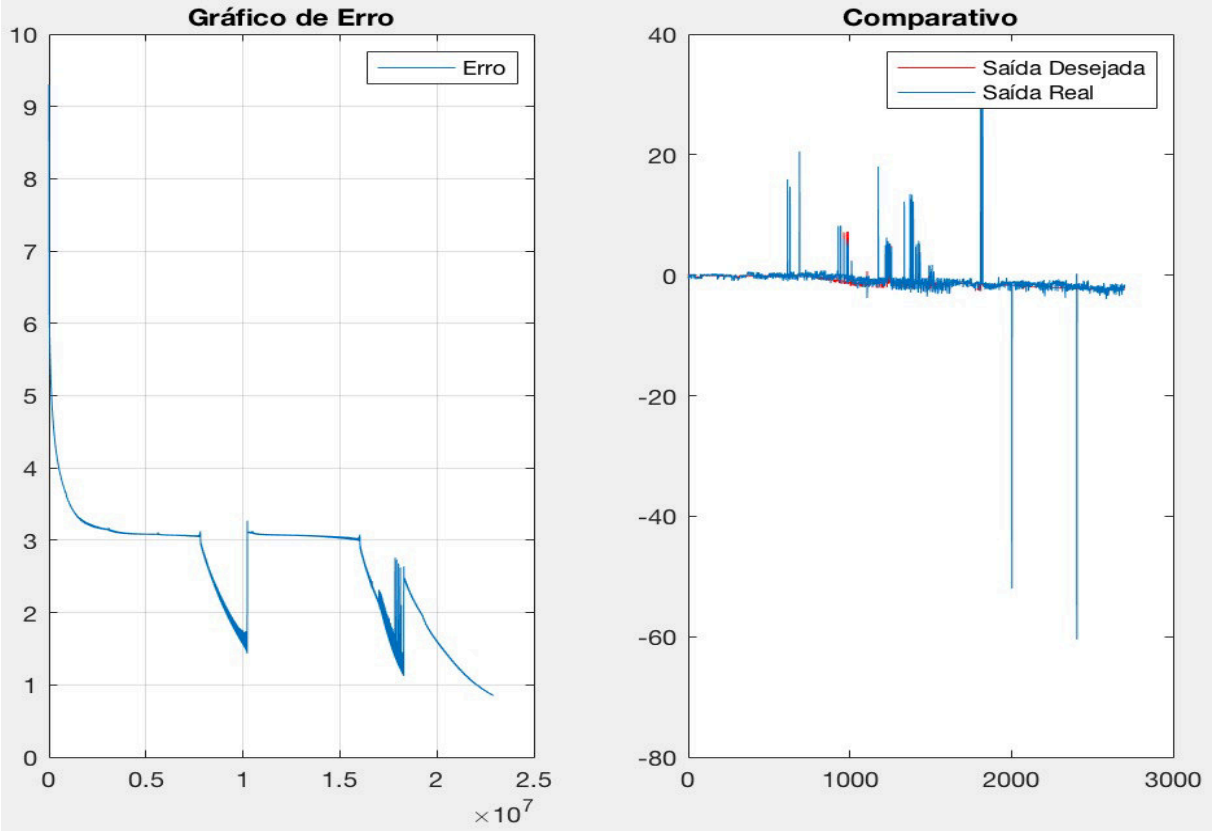
Caso	Real	Média	Média + Rede	Erro Percentual Média	Erro Percentual Média + Rede
Melhor Média	74	74	74,0621901	0,000000%	0,084041%
Pior Média	272	332,4625	278,70194	22,228860%	2,463949%
Melhor Média + Rede	27	27,025	27,0000510	0,092593%	0,000189%
Pior Média + Rede	297	329,8625	333,0881978	11,064815%	12,150908%

Fonte: Autor

Dentre as 2700 medidas do treinamento 1, a correção da média utilizando a rede obteve melhores resultados em 2035 casos, equivalente a 75% do total. Para o pior caso da média, o

erro percentual foi 22,228860%, enquanto que para a pior correção foi 12,150908%, o que sugere uma vantagem da correção em relação a média.

Figura 11 – Gráficos referentes ao treinamento 2



Fonte: Autor

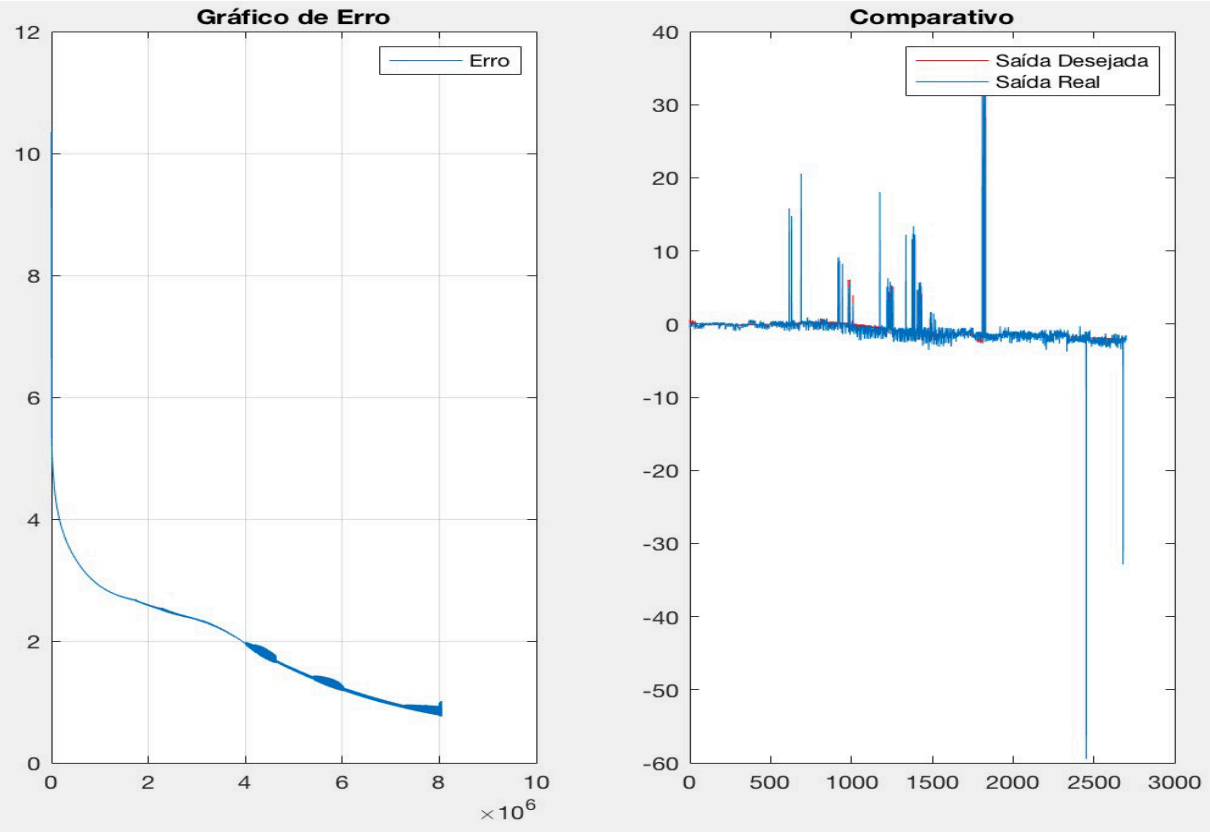
Tabela 3 – Dados obtidos no treinamento 2

Caso	Real	Média	Média + Rede	Erro Percentual Média	Erro Percentual Média + Rede
Melhor Média	83	83	82,9335156	0,000000%	0,080102%
Pior Média	272	332,4625	300,5038167	22,228860%	10,479344%
Melhor Média + Rede	292	294,4375	291,999865	0,834760%	0,000046%
Pior Média + Rede	272	332,4625	300,5038167	22,228860%	10,479344%

Fonte: Autor

Dentre as 2700 medidas do treinamento 2, a correção da média utilizando a rede obteve melhores resultados em 1948 casos, equivalente a 72% do total. Para o pior caso da média, o erro percentual foi 22,228860%, enquanto que para a pior correção foi 10,479344%, o que sugere uma vantagem da correção em relação a média.

Figura 12 – Gráficos referentes ao treinamento 3



Fonte: Autor

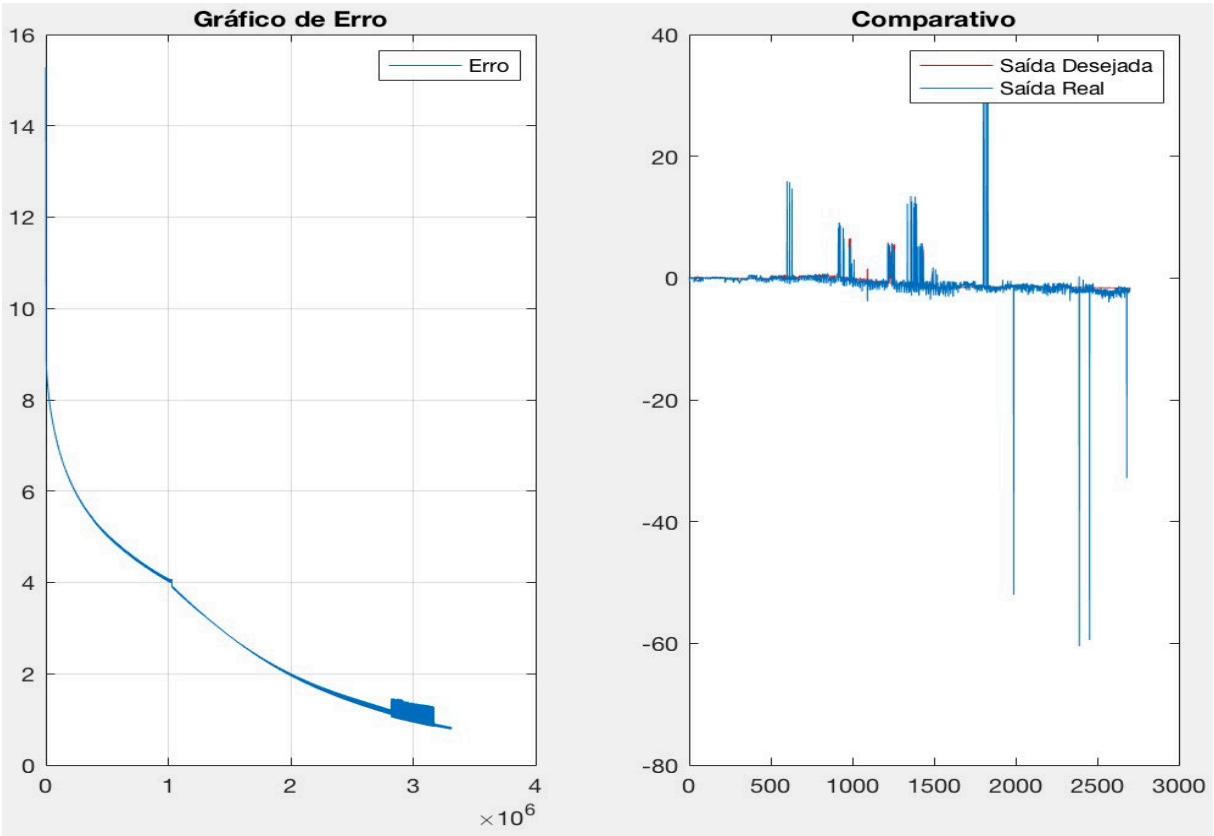
Tabela 4 – Dados obtidos no treinamento 3

Caso	Real	Média	Média + Rede	Erro Percentual Média	Erro Percentual Média + Rede
Melhor Média	56	56	55,8950915	0,000000%	0,187337%
Pior Média	275	334,4625	310,4458291	21,622727%	12,889392%
Melhor Média + Rede	239	240,575	238,9996779	0,658996%	0,000135%
Pior Média + Rede	275	334,4625	310,4458291	21,622727%	12,889392%

Fonte: Autor

Dentre as 2700 medidas do treinamento 3, a correção da média utilizando a rede obteve melhores resultados em 2109 casos, equivalente a 78% do total. Para o pior caso da média, o erro percentual foi 21,622727%, enquanto que para a pior correção foi 12,889392%, o que sugere uma vantagem da correção em relação a média.

Figura 13 – Gráficos referentes ao treinamento 4



Fonte: Autor

Tabela 5 – Dados obtidos no treinamento 4

Caso	Real	Média	Média + Rede	Erro Percentual Média	Erro Percentual Média + Rede
Melhor Média	105	105	105,2361399	0,000000%	0,224895%
Pior Média	272	332,4625	290,5147494	22,228860%	6,806893%
Melhor Média + Rede	297	298,65	297,0005771	0,555556%	0,000194%
Pior Média + Rede	297	329,8625	316,0404513	11,064815%	6,410926%

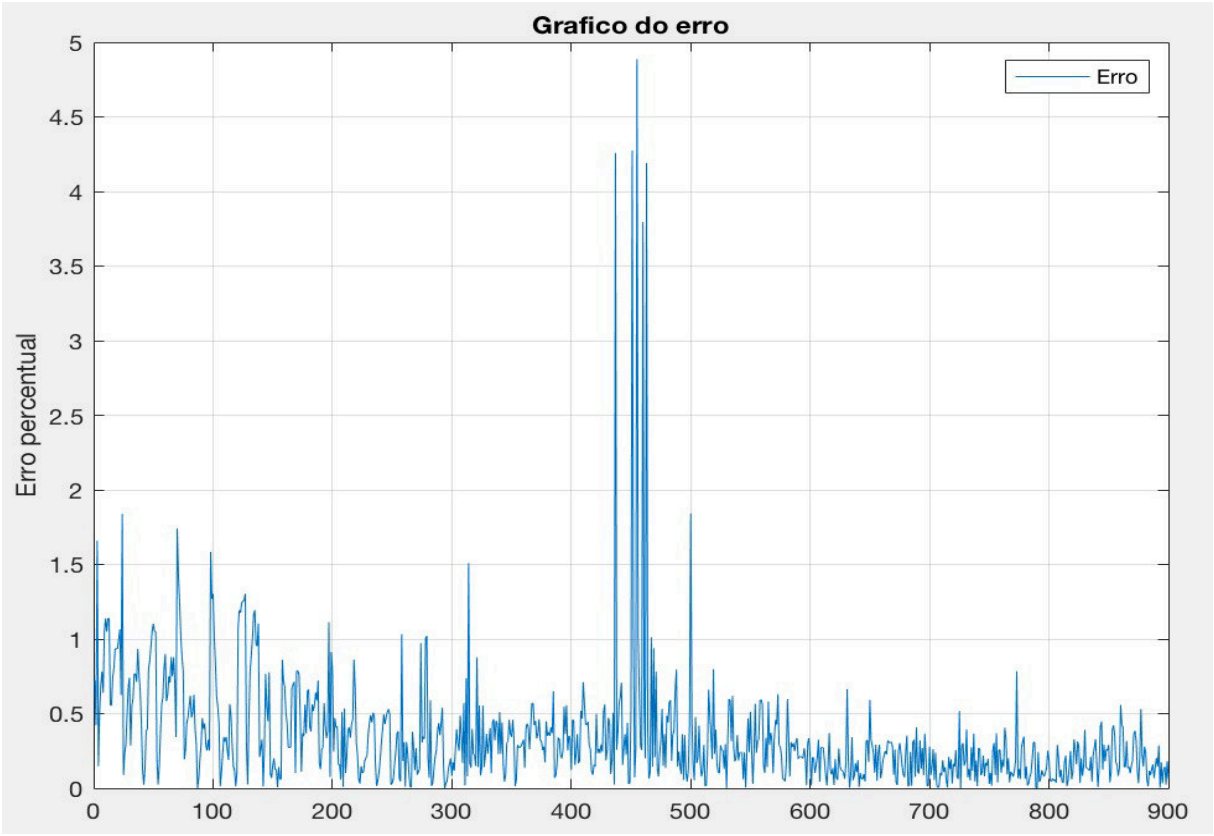
Fonte: Autor

Dentre as 2700 medidas do treinamento 4, a correção da média utilizando a rede obteve melhores resultados em 2044 casos, equivalente a 76% do total. Para o pior caso da média, o erro percentual foi 22,228860%, enquanto que para a pior correção foi 6,410926%, o que sugere uma vantagem da correção em relação a média.

Na etapa dos testes de generalização foram gerados os gráficos do erro percentual em relação ao índice da medida. As Figuras 14, 15, 16 e 17 mostram os gráficos dos testes de generalização 1, 2, 3 e 4, respectivamente. Para cada teste, utilizou-se um conjunto de 900 medidas, referentes às 60 posições distintas, como foi ilustrado no item 3.1.

As Tabela 6, 7, 8 e 9 apresentam alguns dados quantitativos referentes aos testes de generalização. Nessas tabelas, são apresentadas as medições que obtiveram: melhor média entre as medidas obtidas dos sensores, pior média destes valores, melhor e pior resultados utilizando a rede neural.

Figura 14 – Erro percentual do teste de generalização 1



Fonte: Autor

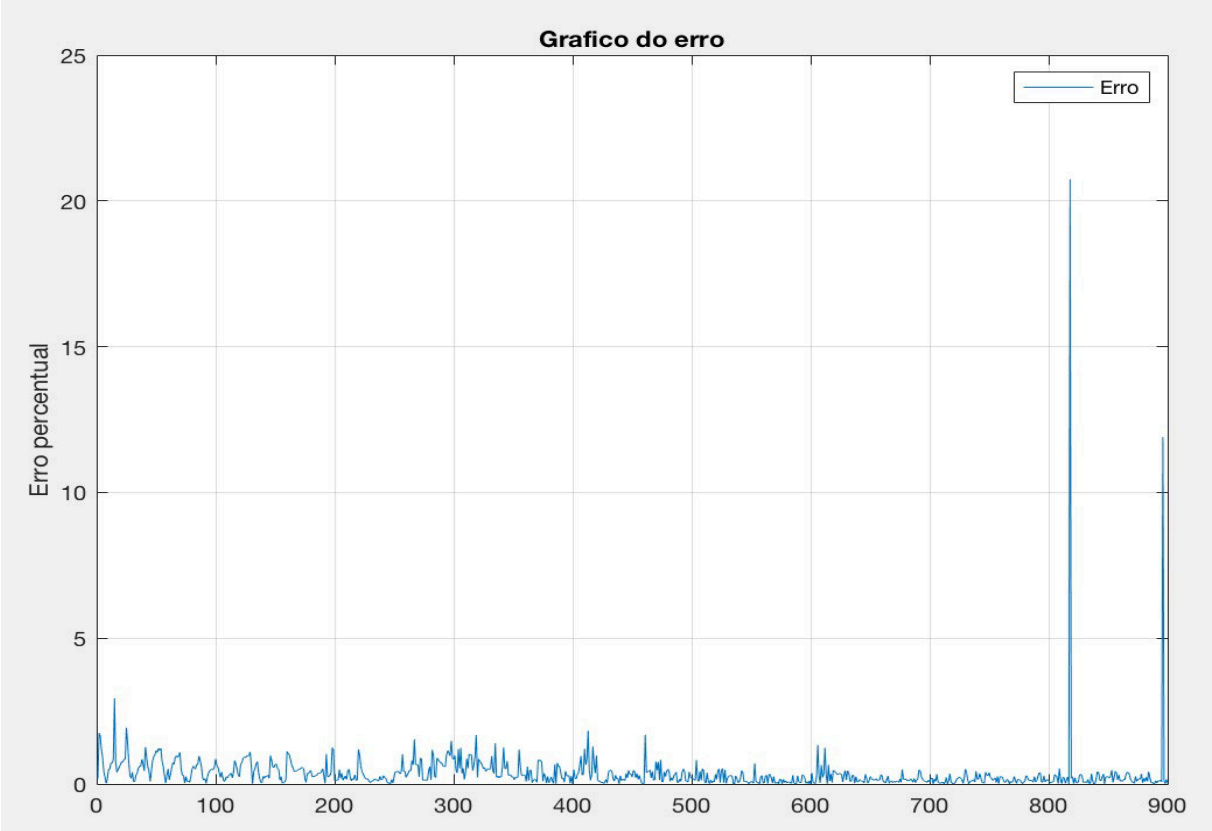
Tabela 6 – Dados obtidos no teste de generalização 1

Caso	Real	Média	Média + Rede	Erro Percentual Média	Erro Percentual Média + Rede
Melhor Média	24	24	23,6625	0,000000%	1,406250%
Pior Média	74	59,3125	58,8	19,847973%	20,540541%
Melhor Média + Rede	244	245,3625	244	0,558402%	0,000000%
Pior Média + Rede	205	206,1625	242,7375	0,567073%	18,408537%

Fonte: Autor

Dentre as 900 medidas do teste de generalização 1, a correção da média utilizando a rede obteve melhores resultados em 583 casos, equivalente a 65% do total. Para o pior caso da média, o erro percentual foi 19,847973%, enquanto que para a pior correção foi 18,408537%, o que sugere uma leve vantagem da correção em relação a média.

Figura 15 – Erro percentual do teste de generalização 2



Fonte: Autor

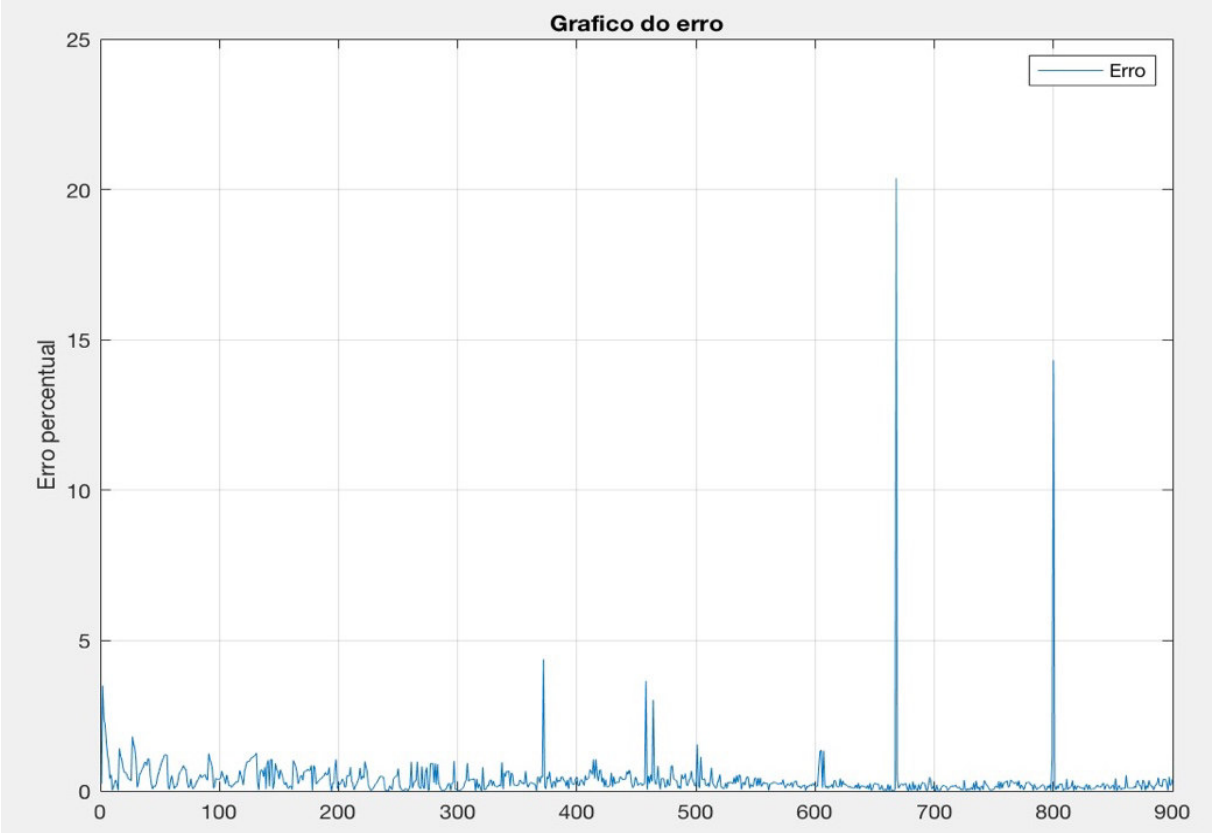
Tabela 7 – Dados obtidos no teste de generalização 2

Caso	Real	Média	Média + Rede	Erro Percentual Média	Erro Percentual Média + Rede
Melhor Média	56	56	56,5	0,000000%	0,892857%
Pior Média	275	334,4625	332,9875	21,622727%	21,086364%
Melhor Média + Rede	192	193,075	192	0,559896%	0,000000%
Pior Média + Rede	272	273,975	213,5125	0,726103%	21,502757%

Fonte: Autor

Dentre as 900 medidas do teste de generalização 2, a correção da média utilizando a rede obteve melhores resultados em 674 casos, equivalente a 75% do total. Para o pior caso da média, o erro percentual foi 21,622727%, enquanto que para a pior correção foi 21,502757%, o que sugere uma leve vantagem da correção em relação a média.

Figura 16 – Erro percentual do teste de generalização 3



Fonte: Autor

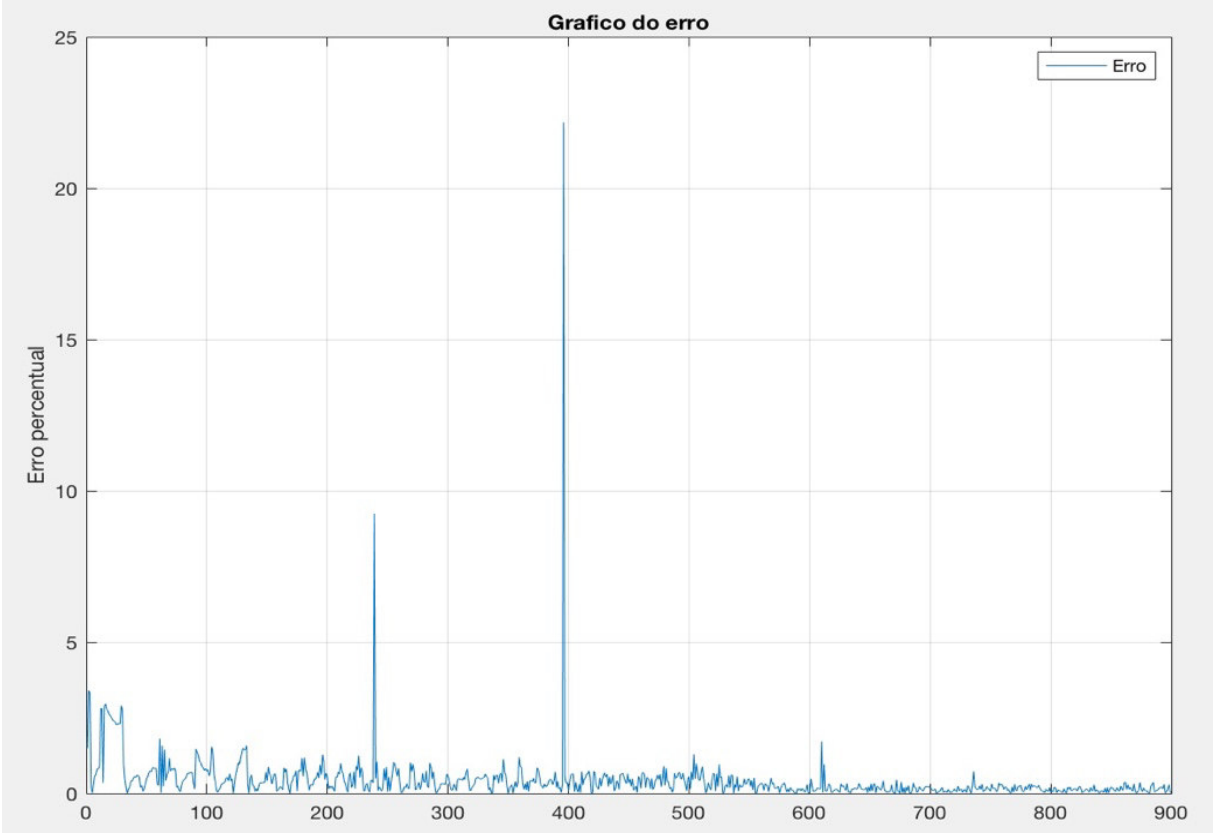
Tabela 8 – Dados obtidos no teste de generalização 3

Caso	Real	Média	Média + Rede	Erro Percentual Média	Erro Percentual Média + Rede
Melhor Média	30	30	30,0625	0,000000%	0,208333%
Pior Média	272	332,4625	330,4875	22,228860%	21,502757%
Melhor Média + Rede	130	131,375	130	1,057692%	0,000000%
Pior Média + Rede	272	332,4625	330,4875	22,228860%	21,502757%

Fonte: Autor

Dentre as 900 medidas do teste de generalização 3, a correção da média utilizando a rede obteve melhores resultados em 699 casos, equivalente a 78% do total. Para o pior caso da média, o erro percentual foi 22,228860%, enquanto que para a pior correção foi 21,502757%, o que sugere uma leve vantagem da correção em relação a média.

Figura 17 – Erro percentual do teste de generalização 4



Fonte: Autor

Tabela 9 – Dados obtidos no teste de generalização 4

Caso	**Real**	**Média**	**Média + Rede**	**Erro Percentual Média**	**Erro Percentual Média + Rede**
Melhor Média	27	27	27,125	0,000000%	0,462963%
Pior Média	205	168,425	167,2625	17,841463%	18,408537%
Melhor Média + Rede	292	293,4625	292	0,500856%	0,000000%
Pior Média + Rede	205	168,425	167,2625	17,841463%	18,408537%

Fonte: Autor

Dentre as 900 medidas do teste de generalização 4, a correção da média utilizando a rede obteve melhores resultados em 606 casos, equivalente a 67% do total. Para o pior caso da média, o erro percentual foi 17,841463%, enquanto que para a pior correção 18,408537%, o que sugere uma leve vantagem da média em relação a correção.

4 CONCLUSÕES E TRABALHOS FUTUROS

Os resultados obtidos mostram que a saída da rede tem aproximado-se bastante da resposta desejada. Na grande maioria dos casos, a fusão sensorial via redes neurais mostrou-se a melhor opção. Além disso, a correção da média dos valores obtidos pelos sensores utilizando a rede neural tem permitido obter maior precisão para o valor de distância.

Já era esperado que os resultados obtidos no treinamento fossem melhores que os obtidos nos testes de generalização. Em dois dos quatro testes de generalização, a porcentagem de medidas onde a correção da média obteve melhor resultado em relação a média, ficou abaixo de 70%, algo que não aconteceu em nenhum dos treinamentos realizados.

A quantidade de iterações para reduzir este erro da rede, em alguns casos, foi grande, fazendo com que os treinamentos levassem algumas semanas para serem concluídos. Nesse trabalho, assumiu-se como razoável um erro quadrático médio abaixo de um. Três dos treinamentos precisaram de cerca de dois a oito milhões de iterações para atingir valores de erro satisfatórios, enquanto que um dos treinamentos levou cerca de 23 milhões de interações para atingir um valor de erro aceitável.

Os resultados poderiam ter sido ainda melhores caso houvesse mais tempo para o treinamento da rede, pois em todos os treinamentos, o erro quadrático médio mostrava uma tendência de queda com o aumento das iterações.

As possibilidades de trabalhos futuros baseados no estágio atual deste trabalho são diversas, dentre elas podemos citar:

- Continuar os treinamentos, a fim de obter-se resultados ainda mais precisos;
- Adicionar outros tipos de sensores, quanto maior a diversidade e quantidade dos dados adquiridos, mais precisos serão resultados da fusão sensorial;
- Viabilizar o arranjo de sonares para fundir alguns pontos da varredura do laser, visando futuro uso em mapeamento e localização de robôs móveis.

BIBLIOGRAFIA

AZEVEDO, F. M.; BRASIL, L. M.; OLIVEIRA, R. C. L. **Redes neurais com aplicações em controle e em sistemas especialistas**. Florianópolis: Visual Books Editora, 2000, 401p.

BALTSAVIAS, E. P. ***Airbone Laser Scanning: Basic Relations and Formulas***. ISPRS *Journal of Photogrammetry and Remote Sensing*, Volume 54 Número 2-3, pg 199-214, 1999.

BORENSTEIN, J. EVERETT, H.R, FENG, L. ***Where am I? Sensors and Methods for Mobile Robot Positioning, 1996.*** Disponível em: <http://wwwpersonal.umich.edu/~johannb/position.htm> Acesso em: 19 out. 2018.

BRAGA, A. P. **Redes neurais artificiais: teoria e aplicações.** 2ed. Rio de Janeiro, RJ: LTC, 2007.

DALMOLIN, Q.; SANTOS, D. R. **Sistema Laserscanner: Conceitos e Princípios de Funcionamento**. 3a Edição – UFPR – Curitiba/PR – Brasil. 97 pg – 2004.

DUFFY, B. R**. *Sensor Fusion for Social Robotic***s. In: *31st International Symposium on Robotics. 2000. Palais des congrs*. p. 155-170.

FACELI, K. **Combinação de métodos de inteligência artificial para fusão de sensores**. 2001, dissertação –ICMC, USP

FRADEN, J. AIP ***Handbook of modern sensors: physics, designs and applications***. *New York, USA: American Institute of Physics, 1993.*

GODOY, R. S.; ASSIS, W. O.; GONÇALVES, H. S. B**. Desenvolvimento de um Sistema para Monitoramento de Sinais de Sensores Utilizados em Aplicações Automotivas. Escola** de Engenharia Mauá, São Caetano do Sul, 2016. Disponível em: < https://maua.br/files/122016/desenvolvimento-um-sistema-para-monitoramento-sinais-sensores-utilizados-aplicacoes-automotivas-270958.pdf >. Acesso em: 19 out. 2018.

GREWAL, M. S.; ANDREWS, A. P. ***Kalman filtering: Theory and practice with matlab. John Wiley & Sons****. New York, USA, 2014.*

GUIMARÃES, A. M.; MATHIAS, I. M.; DIAS A. H.; FERRARI J.W.; CAZELATTO JUNIOR, C. R**. Módulo De Validação Cruzada Para Treinamento De Redes Neurais Artificiais Com Algoritmos *Backpropagation E Resilient Propagatio*n**.** 2007

HAYKIN, S. ***Neural networks: a comprehensive foundation****. New York: Macmillan College Publishing, 1994*

HAYKIN, S. **Redes neurais: princípios e prática**. Trad. Paulo Martins Engel. 2 ed. Porto Alegre, SC: Bookman, 2001.

HOSOKAWA, E. **Técnica de Árvore de Decisão em Mineração de Dados. 2011**. Monografia - Faculdade de Tecnologia de São Paulo.

MURPHY, R. R. ***Sensor Fusion. In Handbook of Brain Theory and Neural Networks****. Colorado School of Mines, 1994.*

MURPHY, R. ***Introduction to AI Robotics***, MIT Press, 2000.

OSÓRIO, F. S.; BITTENCOURT, J. R. **Sistemas Inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens**. Workshop de Inteligência Artificial, 2000. Apostila-seminário, Santa Cruz do Sul, UNISC – Universidade de Santa Cruz do Sul – Departamento de Informática, 2000.

PATSKO, L. F. **Tutorial, Aplicações, Funcionamento e Utilização de Sensores**. 2016. Disponível em: <http://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_aplicacoes_e_funcionamento_de_sensores.pdf>. Acesso em: 19 out. 2018.

PRESTES, E. **Introdução a Robótica Móvel.** 2012. Disponível em: <http://www.inf.ufrgs.br/~prestes/Courses/Robotics/Slides/RAula8-9-10.pdf>. Acesso em: 19 out. 2018.

ROHN, M. da C.; MINE, M. R. M. **Uma aplicação de redes neurais artificiais à previsão de chuvas de curtíssimo prazo**. Simpósio Brasileiro De Recursos Hídricos, 2003. Curitiba, Associação Brasileira de Recursos Hídricos, 2003, CD-ROM.

SILVA, I. N. **Projeto e análise de uma rede neural para resolver problemas de programação dinâmica**. Sba Controle & Automação , v.12, n.1, 2001, p.1-10.

TAN, P.; STEINBACH, M.; KUMAR, V. **Introdução ao *Data Mining***. Ciência Moderna, 2009.

***Titanic*,** 1997. Filme. Dirigido por James CAMERON. EUA: Twentieth Century-Fox Film Corporation

TOMMASELLI, A. M. G**. Um Estudo Sobre as Técnicas de Varredura a Laser e Fotogrametria para Levantamentos 3D a Curta Distância**. GEODÉSIA Online - Revista da Comissão Brasileira de Geodésia. 4 / 2003 [ISSN 1415-1111].

APÊNDICE A – CÓDIGO PARA AQUISIÇÃO DE DADOS USANDO O SONAR E ARDUINO

```
#include <Ultrasonic.h>
//e onde esta ligado o trig(8) e o echo(7) respectivamente

#include <Ultrasonic.h>

 int i = 1;

 Ultrasonic ultrassom(8,7); // define o nome do sensor(ultrassom)
 int led = 13;
 void setup() {
 Serial.begin(9600); //Habilita Comunicação Serial a uma taxa de 9600 bauds.
 long distancia;
 pinMode(led, OUTPUT);

 }

// Função que se repete infinitamente quando a placa é ligada
void loop()
{
  long distancia;

  for(i ; i<=70; i++)
  {

    distancia = ultrassom.Ranging(CM);// ultrassom.Ranging(CM) retorna a distancia em
                          // centímetros(CM) ou polegadas(INC)
    Serial.println(distancia); //imprime o valor da variável distancia
    //Serial.println(" cm");
```

```
    delay(10);

  }
```

APÊNDICE B – CÓDIGO PARA AQUISIÇÃO DE DADOS USANDO O *LASER SCANNER* E ARDUINO

```
#include <RPLidar.h>

RPLidar lidar;

#define RPLIDAR_MOTOR 3 // The PWM pin for control the speed of RPLIDAR's motor.
                        // This pin should connected with the RPLIDAR's MOTOCTRL signal

 bool transmit = false;
 byte incomingByte;
 int vezes = 1;
void setup() {
 // bind the RPLIDAR driver to the arduino hardware serial
 lidar.begin(Serial3);
 Serial.begin(115200);
 // set pin modes
 pinMode(RPLIDAR_MOTOR, OUTPUT);
}

void loop() {
 if (IS_OK(lidar.waitPoint())) {
   float distance = lidar.getCurrentPoint().distance; //distance value in mm unit
   float angle    = lidar.getCurrentPoint().angle; //anglue value in degree
   bool  startBit = lidar.getCurrentPoint().startBit; //whether this point is belong to a new scan
   byte  quality  = lidar.getCurrentPoint().quality; //quality of the current measurement

   //perform data processing here...

   if ((angle <= 185) && (angle >= 175))
```

```
    {
      delay(100);
      Serial.println(distance);
      vezes++;
      if(vezes >= 71)
      {
         delay(30000);
      }
    }

  } else {
    analogWrite(RPLIDAR_MOTOR, 0); //stop the rplidar motor

    // try to detect RPLIDAR...
    rplidar_response_device_info_t info;
    if (IS_OK(lidar.getDeviceInfo(info, 100))) {
      // detected...
      lidar.startScan();

      // start motor rotating at max allowed speed
      analogWrite(RPLIDAR_MOTOR, 255);
      delay(1000);
    }
  }
}
```

APÊNDICE C – CÓDIGO EM MATLAB PARA TREINAMENTO DA REDE NEURAL

C.1 - Função variáveis_globais

```
clear all
close all
clc
global H eta iter Hbias Obias
global pesos erro Y
```

controle_rna

C.2 - Função controle_rna

```
function varargout = controle_rna(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @controle_rna_OpeningFcn, ...
                   'gui_OutputFcn',  @controle_rna_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before controle_rna is made visible.
function controle_rna_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for controle_rna
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes controle_rna wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = controle_rna_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
function ed_H_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ed_H_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

qtdeH = 7;
set(hObject, 'String', num2str(qtdeH));

function ed_eta_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ed_eta_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
valor_eta = 0.0001;
set(hObject, 'String', num2str(valor_eta));

function ed_iter_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
```

```
function ed_iter_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
qtdeIter = 1000;
set(hObject, 'String', num2str(qtdeIter));

% --- Executes on button press in pb_iniciartreino.
function pb_iniciartreino_Callback(hObject, eventdata, handles)
set(handles.txt_ok, 'String', '')
global H pesos eta iter Hbias Obias erro Y

H = 0;
pesos = [];
Y = [];
eta = 0;
iter = 0;
Hbias = 0;
Obias = 0;
erro = [];

H = str2num( get(handles.ed_H, 'String') );
eta = str2num( get(handles.ed_eta, 'String') );
iter = str2num( get(handles.ed_iter, 'String') );

%%% Bloco que verifica se deve ser inserido bias ou nao nas camadas
%%% escondida (Hbias) e de saida (Obias)
bias = get(handles.bg_Hbias, 'SelectedObject');
if isequal(get(bias, 'Tag'), 'rb_Hsbias')
   Hbias = 0;
elseif isequal(get(bias, 'Tag'), 'rb_Hcbias')
```

```
   Hbias = 1;
end

bias = get(handles.bg_Obias, 'SelectedObject');
if isequal(get(bias, 'Tag'), 'rb_Osbias')
   Obias = 0;
elseif isequal(get(bias, 'Tag'), 'rb_Ocbias')
   Obias = 1;
end
%%% ------------------------------------------------------------
continuar = 0;

[pesos, erro, Y] = rnatreino(pesos, erro, H, Hbias, Obias, eta, iter, continuar);

sp = sprintf('%0.7f', erro(numel(erro)) );
set(handles.txt_merroAtual, 'String', sp );
set(handles.txt_merroAnterior, 'String', '');
set(handles.ed_passo, 'String', num2str(numel(erro)) );

set(handles.pb_continuar, 'Enable', 'on')

% --- Executes on button press in pb_continuar.
function pb_continuar_Callback(hObject, eventdata, handles)
% hObject    handle to pb_continuar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.txt_ok, 'String', '')
global pesos H Hbias Obias eta iter erro Y

eta = str2num( get(handles.ed_eta, 'String') );
iter = str2num( get(handles.ed_iter, 'String') );

continuar = 1;
```

```
[pesos, erro, Y] = rnatreino(pesos, erro, H, Hbias, Obias, eta, iter, continuar);

set(handles.txt_merroAnterior, 'String', get(handles.txt_merroAtual, 'String') );
sp = sprintf('%0.7f', erro(numel(erro)) );
set(handles.txt_merroAtual, 'String', sp );
set(handles.ed_passo, 'String', num2str(numel(erro)) );

% --- Executes on button press in pb_salvar.
function pb_salvar_Callback(hObject, eventdata, handles)
global pesos erro Hbias Obias H Y

ult_erro = sprintf('%0.7f', erro( numel(erro) ) );
data = datestr(clock);
data = strrep(data, ':','_');
camada = num2str(H);
bias = '';
if Hbias == 0,
   bias = 's';
else
   bias = 'b';
end
camada = strcat( camada, bias );
if Obias == 0
   bias = 's';
else
   bias = 'b';
end
camada = strcat( camada, bias );

%%% se o diretorio ja tiver sido criado ele apresenta uma mensagem
%%% de Warning: Directory already exists.
mkdir('pesos')
filename = strcat( 'pesos/treino_', data, '_cam_', camada, '_erro_', ult_erro, '.mat'  );
```

```
save( filename , 'pesos', 'erro', 'Y');

set(handles.ed_salva, 'String', filename)
set(handles.txt_ok, 'String', 'OK')

function ed_carregar_Callback(hObject, eventdata, handles)
function ed_carregar_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pb_carregar.
function pb_carregar_Callback(hObject, eventdata, handles)
set(handles.txt_ok, 'String', '')

[filename, pathname] = uigetfile('*.mat');

arquivo = strcat(pathname, filename);

load(arquivo, 'pesos', 'erro', 'Y');
pesos1 = pesos;
erro1 = erro;
Y1 = Y;
clear pesos erro Y

load('dadosfiltrados.mat', 'desejada');
D = desejada;

% apresenta o grafico do erro do treinamento carregado
figure(1)
subplot(1, 2, 1)
```

```
grid on
plot(erro1)
title('Gráfico de Erro')
legend('Erro')
grid on

% figure(2)
subplot(1, 2, 2);
plot(1:numel(Y1), Y1, 'r', 1:numel(D), D)
title('Comparativo')
legend('Saída Desejada', 'Saída Real')
% -------
global pesos erro Hbias Obias
pesos = pesos1;
erro = erro1;


%=====================================================

d = strsplit(filename, '_erro');  % divide string em dois
e = d{1};
tam = numel(e);
bias = e((tam-1):tam);
disp(bias)

if isequal(bias(1), 's')
   hand1 = findobj('Tag', 'rb_Hsbias');
   set(handles.bg_Hbias, 'SelectedObject', hand1)
   Hbias = 0;
elseif isequal(bias(1), 'b')
   hand1 = findobj('Tag', 'rb_Hcbias');
   set(handles.bg_Hbias, 'SelectedObject', hand1)
   Hbias = 1;
```

```
end

if isequal(bias(2), 's')
   hand2 = findobj('Tag', 'rb_Osbias');
   set(handles.bg_Obias, 'SelectedObject', hand2)
   Obias = 0;
elseif isequal(bias(2), 'b')
   hand2 = findobj('Tag', 'rb_Ocbias');
   set(handles.bg_Obias, 'SelectedObject', hand2)
   Obias = 1;
end
%================================================
qtH = size(pesos.Whi, 1);
set(handles.ed_H, 'String', num2str(qtH) )

% mostra o arquivo que foi carregado
set(handles.ed_carregar, 'String', filename)

% habilita o botao continuar
set(handles.pb_continuar, 'Enable', 'on')

function ed_salva_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ed_salva_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

C.3 - Função rnatreino

```
function [pesos, erro, Y] = rnatreino(pesos, erro, H, Hbias, Obias, eta, iter, continua)
clc
```

```
disp('Treinamento em execucao...')
load dados_treinamento entrada desejada real

% numero de neuronios na camada de entrada
I = size(entrada,2);

% -------- BLOCO DE CONFIGURACAO DE PARAMETROS ------ %
epocas = iter;
% ------------------------------------------------------%


O = size(desejada,2);
X = entrada';       % valores de entrada
D = desejada';      % valores desejados

%---------VARIAVEIS PARA TREINO -----------------%
if continua

   if ~isempty(pesos)
      Whi = pesos.Whi;
      bias_hi = pesos.bias_hi;
      Woh = pesos.Woh;
      bias_oh = pesos.bias_oh;
   else
      errordlg('Pesos nao inicializados.');
      return
   end

else
   % inicializa dos pesos e do bias
   Whi = rand(H,I) - 0.5;
   bias_hi = (rand(H,1) - 0.5)*Hbias;
   Woh = rand(O,H) - 0.5;
```

```
   bias_oh = (rand(O,1) - 0.5)*Obias;

   erro_treino_total = [];
end
%-----------------------------------------------%

k = 1;          % funcao de ativacao da camada de saida (linear)
erro_p = zeros(1, epocas);
%%------------ aprendizado da rede ---------------%%
 for iter=1:epocas

   net_h = Whi*X + bias_hi*ones(1, size(X,2))*Hbias;
   Yh = logsig(net_h);   % funcao de ativacao da camada intermediaria (nao-linear)

   net_o = Woh*Yh + bias_oh*ones(1, size(Yh,2))*Obias;
   Y = k*net_o;

   % erro do aprendizado e retro-propagado na camada de saida
   E = D - Y;

   df = k*ones(size(net_o));
   delta_bias_oh = eta*sum((E.*df),2);
   delta_Woh = eta*(E.*df)*Yh';

   % erro do aprendizado e retro-propagado na camada escondida
   Eh = Woh'*(E.*df);

   df = logsig(net_h) - (logsig(net_h).^2);
   delta_bias_hi = eta*sum((Eh.*df),2);
   delta_Whi = eta*(Eh.*df)*X';

   % Ajuste dos pesos e bias
   Whi = Whi + delta_Whi;
```

```
   bias_hi = bias_hi + delta_bias_hi*Hbias;
   Woh = Woh + delta_Woh;
   bias_oh = bias_oh + delta_bias_oh*Obias;

   % Erro quadratico medio
   erro_p(iter) = meansqr(E);        % vetor com o erro quadratico medio
 end

 erro_treino_total = [erro erro_p];
 erro = erro_treino_total;

%----------------------PLOTA GRAFICOS-------------------------%
% Plotagem do grafico de erro
figure(1)
subplot(1, 2, 1);
plot(erro)
title ('Gráfico do erro')
xlabel('Número de iterações')
ylabel('Erro quadrático médio')
legend('Erro')
grid on

dim = numel(Y);
% figure(2)
subplot(1, 2, 2);
plot(1:numel(D), D, 'b', 1:numel(Y), Y, 'r') % saida desejada e % saida da rede
title('Comparativo')
legend('Saída Desejada - RNA', 'Saída Real - RNA')
grid on
xlabel('Índice da medida')
ylabel('Valor em centímetros')

media = sum(entrada,2)./size(entrada,2);
```

```
postreino = media' + Y;
%%%%
% figure(3)
figure(2)
subplot(1, 2, 1);
plot(1:dim, real, 'b', 1:dim, postreino, 'r' )
title('Compara: Valor Real x RNA + Média')
legend('Valor Real', 'RNA + Média')
xlabel('Índice da medida')
ylabel('Valor em centímetros')
grid on

subplot(1, 2, 2);
dify = real' - postreino;
plot(1:dim, dify, 'g' )
title('Diferença: Real x Saída')
legend('Erro de Média')
xlabel('Índice da medida')
ylabel('Valor em centímetros')
grid on
% %-------------------------------------------------------------%

pesos.Whi = Whi;
pesos.bias_hi = bias_hi;
pesos.Woh = Woh;
pesos.bias_oh = bias_oh;

disp('Treinamento finalizado!!')

end
```

APÊNDICE D – CÓDIGO EM MATLAB PARA TESTE DE GENERALIZAÇÃO DA REDE NEURAL

D.1 – Função var_generaliza

```
clear all
clc

% declaracao de variaveis globais
global entrada
global saida real
global erro erroAbs
global pesos

load('dadosfiltradost2.mat','entrada','desejada','real');

entrada = entrada;
real = real;
saida = desejada;
% --- calcula a saida para comparar com a rede --- %
media = sum(entrada,2)./size(entrada,2);
saida = real - media;
% ------------------------------------------------ %
controle_generaliza
```

D.2 – Função controle_generaliza

```
function varargout = controle_generaliza(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @controle_generaliza_OpeningFcn, ...
                   'gui_OutputFcn',  @controle_generaliza_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before controle_generaliza is made visible.
function controle_generaliza_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = controle_generaliza_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes on button press in pb_testeGeneraliza.
function pb_testeGeneraliza_Callback(hObject, eventdata, handles)
global entrada saida pesos erro erroAbs real
erro = [];
erroAbs = [];

filename = get(handles.ed_pathPesos, 'String');

% ------------- VERIFICA O USO DE BIAS --- %%
d = strsplit(filename, '_erro');  % divide string em dois
e = d{1};
tam = numel(e);
bias = e((tam-1):tam);
```

```
Hbias = 0;
if isequal(bias(1), 's')
   Hbias = 0;
elseif isequal(bias(1), 'b')
   Hbias = 1;
end

Obias = 0;
if isequal(bias(2), 's')
   Obias = 0;
elseif isequal(bias(2), 'b')
   Obias = 1;
end

% -------- VERIFICA QUAL QTDADE DE DADO DE ENTRADA ---
sele = get(handles.pan_entrada, 'SelectedObject');
switch get(sele, 'Tag')
   case 'rb_todos'
      [erroAbs, erro] = rnageneraliza(entrada, saida, pesos, Hbias, Obias, real);
   case 'rb_linha'
      lin = get(handles.ed_linha, 'String');
      linha = str2num( lin );
      if (linha < 1) || (linha > size(entrada,1))
         errordlg('Linha fora dos limites permitidos!');
      else
         set(handles.pan_linResultados, 'Visible', 'on')
         [erroAbs, erro] = rnageneraliza(entrada(linha,:), saida(linha), pesos, Hbias, Obias, real);

         set(handles.ed_s1, 'String', num2str( entrada(linha,1) ))
         set(handles.ed_s2, 'String', num2str( entrada(linha,2) ))
         set(handles.ed_s3, 'String', num2str( entrada(linha,3) ))
         set(handles.ed_media, 'String', num2str( mean(entrada(linha,:)) ))
```

```
            saidarna = saida(linha) - erroAbs;
            set(handles.ed_rna, 'String', num2str( saidarna ))
            set(handles.ed_rnaMedia, 'String', num2str( mean(entrada(linha,:))+saidarna ))
        end
end

if ~isempty(erro)
    % ------ erros percentuais --------- %
    set(handles.ed_erroMax, 'String', num2str( max(erro) ) )
    set(handles.ed_erroMin, 'String', num2str( min(erro) ) )
    set(handles.ed_erroMedio, 'String', num2str( mean(erro) ) )

    % ------ erros absolutos --------- %
    set(handles.ed_absMax, 'String', num2str( max(erroAbs) ) )
    set(handles.ed_absMin, 'String', num2str( min(erroAbs) ) )
    set(handles.ed_absMedio, 'String', num2str( mean(erroAbs) ) )

end


% --- Executes on button press in pb_selecionaPesos.
function pb_selecionaPesos_Callback(hObject, eventdata, handles)
global pesos

[filename, pathname] = uigetfile('*.mat');
arquivo = strcat(pathname, filename);
set(handles.ed_pathPesos, 'String', filename);

load(arquivo, 'pesos');
pesos = pesos;

function ed_pathPesos_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
```

```
function ed_pathPesos_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_erroMax_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function ed_erroMax_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_erroMin_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function ed_erroMin_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_erroMedio_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function ed_erroMedio_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_linha_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
```

```
function ed_linha_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_s1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ed_s1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function ed_s2_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ed_s2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_s3_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function ed_s3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_media_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
```

```
function ed_media_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_rna_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function ed_rna_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_rnaMedia_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ed_rnaMedia_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_absMax_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ed_absMax_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function ed_absMin_Callback(hObject, eventdata, handles)
function ed_absMin_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function ed_absMedio_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function ed_absMedio_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes when selected object is changed in pan_entrada.
function pan_entrada_SelectionChangeFcn(hObject, eventdata, handles)

switch get(eventdata.NewValue, 'Tag')
   case 'rb_todos'
      set(handles.ed_linha, 'Visible', 'off')
      set(handles.pan_linResultados, 'Visible', 'off')
   case 'rb_linha'
      set(handles.ed_linha, 'Visible', 'on', 'String', '')
      % set(handles.pan_linResultados, 'Visible', 'on')
end
```

D.3 – Função rnageneraliza

```
function [erroAbs, erro] = rnageneraliza(entrada, saida, pesos, Hbias, Obias, real)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
```

```
clc
disp('Iniciando teste...')

% numero de neuronios na camada de entrada
I = size(entrada,2);

% quantidade de neuronios da camada de saida
O = size(saida,2);

X = entrada';       % valores de entrada
D = saida';      % valores desejados
R = real';

% pesos
Whi = pesos.Whi;
bias_hi = pesos.bias_hi;
Woh = pesos.Woh;
bias_oh = pesos.bias_oh;

k = 1;          % funcao de ativacao da camada de saida (linear)
net_h = Whi*X + bias_hi*ones(1, size(X,2))*Hbias;
Yh = logsig(net_h);   % funcao de ativacao da camada intermediaria (nao-linear)

net_o = Woh*Yh + bias_oh*ones(1, size(Yh,2))*Obias;
Y = k*net_o;

% erro do aprendizado e retro-propagado na camada de saida
erroAbs = D-Y;
E = 100*abs(erroAbs)./abs(R);

erro = E;

%----------------------PLOTA GRAFICOS-------------------------%
```

```
% Plotagem do grafico de erro
figure(1)
plot(erro)
title ('Grafico do erro')
% xlabel('')
ylabel('Erro percentual')
legend('Erro')
grid on


disp('Teste finalizado!!')

end
```