Universidade Católica do Salvador
Bacharelado em Engenharia de Software

Adriano Ricardo Andrade Araújo
Luiz Henrique Brito Rios

ROBÔ COBRA: Sistema para Identificação de Organismos Vivos em Estruturas Colapsadas

Salvador
2020

Adriano Ricardo Andrade Araújo

Luiz Henrique Brito Rios

ROBÔ COBRA: Sistema para Identificação de Organismos Vivos em Estruturas Colapsadas

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software.
Orientador: Prof. Me. Marcelo Indio dos Reis
Coorientador: Prof. Me. André Brasil Vieira Wyzykowski

Universidade Católica do Salvador

Salvador
2020

Adriano Ricardo Andrade Araújo

Luiz Henrique Brito Rios

ROBÔ COBRA: Sistema para Identificação de Organismos Vivos em Estruturas Colapsadas

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como requisito parcial para a obtenção do título de Engenheiro de Software.

Comissão Examinadora

Prof. Me. Marcelo Indio dos Reis
Universidade Católica do Salvador
Orientador

Prof. Me. André Brasil Vieira Wyzykowski
Universidade Católica do Salvador

Prof. Me. Osvaldo Requião Melo
Universidade Católica do Salvador

Salvador, 28 de julho de 2020

Dedicamos este trabalho a nossos familiares, que nos deram suporte emocional nos momentos mais difíceis durante essa pesquisa.

Agradecimentos

Agradecemos primeiramente a Deus, por nos inspirar em nosso trabalho, a nossos colegas e amigos da equipe McGyvertronics, que estiveram conosco no início das discussões sobre o robô cobra, ao nosso orientador Me Marcelo Indio dos Reis, por sua paciência e didática ao nos guiar com sua sabedoria, ao nosso coorientador Me André Brasil Vieira Wyzykowski por toda a sua ajuda durante a construção do trabalho, e ao Coordenador do curso, Osvaldo Requião Melo, pelo seu incentivo e por acreditar em nossa solução quando a mesma era apenas uma ideia.

"Inteligência é a capacidade de se adaptar a mudanças."
(Stephen Hawkings)

Resumo

Quando ocorre um colapso de estrutura, causado por desmoronamento ou até mesmo por outro motivo, para se obter um tempo de resposta otimizado na busca e salvamento dos sobreviventes, a tecnologia passa a ser uma importante aliada na manutenção da vida. Nesse contexto, vemos a robótica como otimizador do processo. Este trabalho trata-se da construção de um Robô Cobra, um sistema de identificação de organismos vivos em estruturas colapsadas, que tem como objetivo a identificação de seres vivos nestas estruturas utilizando machine learning, contemplando a implementação de um sistema de locomoção autônomo, além de uma classificação de imagens térmicas. Para tal, foi gerado um dataset de imagens térmicas, onde usamos data augmentation para acresce-lo em volume e variedade, e a partir dele gerar um modelo preditivo. Esse modelo preditivo em conjunto com a detecção de objetos, permite que o robô cobra funcione de forma autônoma durante a detecção/classificação dos objetos em seres vivos ou não-vivos. Na construção da solução foram utilizados sensores, microcontrolador Arduino, além da biblioteca Tensorflow com o Keras. Com os experimentos de prática simulada realizados, o robô teve seu funcionamento testado e validado, detectando com precisão num campo de visão de 180º em distâncias entre 10 à 150 cm, obtendo uma acurácia média de 97,5 % para classificação.

Palavras-Chave: 1. Robô Cobra. 2. Machine Learning. 3. Arduino.

Abstract

When a structure collapse occurs, caused by collapse or even for another reason, to obtain an optimized response time in the search and rescue of survivors, technology becomes an important ally in maintaining life. In this context, we see robotics as a process optimizer. This work deals with the construction of a Snake Robot, a system for the identification of living organisms in collapsed structures, which aims to identify living beings in these structures using machine learning, contemplating the implementation of an autonomous locomotion system, in addition to a classification of thermal images. For this, a thermal image dataset was generated, where we use data augmentation to add it in volume and variety, and from it generate a predictive model. This predictive model, in conjunction with object detection, allows the snake robot to function autonomously during the detection / classification of objects in living or non-living beings. In building the solution, sensors, an Arduino microcontroller were used, in addition to the Tensorflow library with Keras. With the simulated practice experiments carried out, the robot had its operation tested and validated, accurately detecting in a 180º field of view at distances between 10 to 150 cm, obtaining an average accuracy of 97.5 % for classification.

Keywords: 1. Snake Robot. 2. Machine Learning. 3. Arduino.

Lista de figuras

Lista de Siglas e Abreviaturas

API	*Application Programming Interface*
IA	Inteligência Artificial
RISC	*Reduced instruction set computer*
PWM	Modulação por largura de pulso
RNA	Redes Neurais Artificiais
CNN	Rede Neural Convolucional
LIDAR	*Light Detection and Ranging*
EVA	Espuma vinílica acetinada
USB	*Universal Serial Bus*
IDE	Ambiente de desenvolvimento integrado
LCD	Display de cristal líquido
GPS	Sistema de posicionamento global
TCP	Protocolo de Controle de Transmissão
IP	Protocolo IP
RNA	Rede neural artificial
TI	Tecnologia da Informação

Sumário

1 Introdução

"Com o crescimento populacional constante e de forma desordenada, houve nos grandes centros urbanos uma proliferação de edificações irregulares, as quais muitas vezes colapsam causando acidentes e vítimas"(ROCHA, 2005). As pessoas menos favorecidas social e economicamente tendem a construir suas moradias em áreas de encostas, mais suscetíveis a deslizamentos e desmoronamentos. Segundo Ribeiro (2011) apud Silva (2018) desde o último século, o número de registro de desastres naturais em várias partes do mundo vem aumentando consideravelmente, tendo como principal causa o aumento populacional, à ocupação desordenada, o processo de urbanização intenso, além das mudanças climáticas, seguido da expansão industrial.

Com um tempo de resposta relativamente pequeno para procurar e localizar sobreviventes, a tecnologia passa a ser uma importante aliada para a manutenção da vida. Segundo Marques et al. (2007) apud Silva e Freitas (2018) após 48h do desastre a probabilidade de sobrevivência é baixa. "O constante avanço tecnológico vem apresentando inovações que podem diminuir o desgaste das equipes e favorecer uma resposta célere, contribuindo de sobremaneira para o sucesso do resgate e salvamento de vítimas. É preciso adquirir maiores conhecimentos sobre novos recursos, estudá-los, e empregá-los quando possível"(NUNES, 2017).

"No intuito de resguardar os agentes envolvidos no resgate e salvamento de vítimas, toda e qualquer tecnologia que avance nesse sentido, deve sempre ser estimulada, desde que seus benefícios sejam claramente apresentados e justificados"(NUNES, 2017). Nesse contexto enxergamos o papel da robótica como um otimizador do processo.

Segundo Fonseca (2018), a robótica tem um caráter multidisciplinar, ou seja, ela abrange desde conceitos fundamentais, como o estudo do comportamento cinemático e dinâmico de uma estrutura, à análise e tratamento de sinais captados através dos sensores, além da parte de acionamento de atuadores mediante estrutura de decisão e controle. Na robótica móvel podemos adicionar aos conceitos supracitados, as estratégias de localização, mapeamento e navegação.

Segundo Parker (1994) apud Fonseca (2018) o objetivo principal do desenvolvimento de robôs móveis é a preservação da integridade e da vida humana, pois reduzimos a exposição a tarefas insalubres e periculosas, tais como limpeza de lixo tóxico, desativação de usinas de energia nuclear, exploração espacial, missões de busca e salvamento, dentre outras atividades.

Em seus estudos, Parker (1994) apud Fonseca (2018) aborda que robôs autônomos inteligentes, mais especificamente no que se refere a sistemas onde se faz o uso de uma inteligência Artificial(IA), é uma linha de implementação da robótica muito utilizada em

resgates e salvamentos, preservando o “homem” de se expor numa estrutura colapsada, e buscando vítimas dentre os escombros de forma mais eficiente. "Inteligência Artificial (IA) pode ser definida como o ramo da computação que se ocupa da automação do comportamento inteligente ou até mesmo como uma coleção de problemas e metodologias estudadas pelos pesquisadores de IA"(LUGGER, 2013).

"*Snake Robots* são especialmente valiosos em operações de busca e resgate. Eles vêm em uma variedade de formas e tamanhos, e atuando em diveros tipos de atividades, como os *snake robots* usados dentro do corpo humano para procedimentos cirúrgicos"(CRAWFORD, 2018). A idéia de um robô entrar em estruturas colapsadas para auxiliar a equipe de resgate na identificação e localização de vítimas já é algo possível, mas pensando num processo de melhoria contínua, seria possível otimizar essa atuação do robô cobra tal qual possa, de forma autônoma, entrar nas estruturas colapsadas, identificar e localizar vítimas?

1.1 Objetivos

O objetivo deste trabalho é implementar um robô autônomo para identificação de organismos vivos em estruturas colapsadas utilizando *machine learning*.

1.2 Objetivos Específicos

- Gerar uma base de dados de imagens térmicas para treinamento dos algoritmos;
- Gerar os modelos preditivos com base no treinamento dos algoritmos;
- Construir um robô que movimente-se tal como uma serpente, num escopo de duas dimensões (2D);
- Criar uma aplicação que utilize as saídas da rede neural e as leituras advindas do sensor ultrasonico para fazer o robô se movimentar de forma autônoma;
- Validar os modelos preditivos desenvolvidos em situação prática simulada.

2 Fundamentação Teórica

Neste capítulo são abordados conceitos pertinentes a toda a construção e revisão bibliográfica desta pesquisa, permeando assuntos desde o conceito de robótica, inteligência artificial, a parte de *hardware* associado, além dos trabalhos correlatos com o desta pesquisa. Na primeira seção deste capítulo os conceitos do que é robótica, e sobre os *snake robots*, são trabalhados. Na sessão 2.2 levantamos conceitos refetente a arduino, shields, sensores e atuadores. A parte de inteligência artificial e *machine learning*, englobando as redes neurais, tensorflow e o keras são trabalhados na sessão 2.4. A última sessão deste capítulo aborda os trabalhos correlatos a esta pesquisa.

2.1 A Robótica e os Snake Robots

2.1.1 O que é Robótica

A Robótica pode ser definida como "a ciência dos sistemas que interagem com o mundo real com pouca ou mesma nenhuma intervenção humana" (ARSCONSULT, 1995). Segundo Zilli (2009), "pode ser definida como uma área multidisciplinar, que integra disciplinas como Matemática, Engenharia Mecânica, Engenharia Elétrica, Inteligência Artificial, entre outras".

Zilli (2009) define um robô como qualquer máquina ou equipamento mecânico, que funcione automaticamente, simulando habilidades humanas, ou como uma máquina que na aparência ou comportamento imita uma pessoa de forma total ou parcial, como um jeito de andar ou um um movimento corporal mais específico.

> O estudo da robótica móvel é um tema bastante relevante e atual, onde esta área de estudos, pesquisas e desenvolvimento apresentou um grande salto em seu desenvolvimento nas últimas duas décadas. Aplicações de segurança e defesa civil e militar (resgate e exploração em ambientes hostis), demonstram a grande gama de aplicações atuais dos robôs (WOLF et al., 2009).

2.1.2 Snake robots

Shmakov (2006) define *Snake robots* como todos os robôs hiper redundantes, ou seja, aqueles constituídos de módulos conectados por juntas ativas ou passivas. Dentre essa classe de robôs, os "verdadeiros" *snake robots* são aqueles que usam os movimentos senóides das cadeias de módulos de forma sincrona, para locomoção em superfícies duras e em líquidos.

Segundo Shmakov (2006), a motivação geral para a construção de *snake robots* são ambientes apertados, passagens inferiores longas e estreitas, ou passagens sobre materiais e terrenos soltos, onde robôs tradicionais são impedidos devido ao seu tamanho ou forma e onde apêndices como rodas ou pernas causam aprisionamento ou falha. Na figura 1 podemos ver um protótipo de um robô cobra.

Figura 1 – Robô Cobra fonte: autor

"Pesquisas em biologia experimental revelaram quatro modos principais de locomoção em serpentes: serpentina (ou rasteira), concertina, rectilínea e lateral"(SKONIECZNY; D'ELEUTERIO, 2009). Para o escopo deste trabalho, iremos considerar a movimentação baseada apenas no movimento serpentina.

Durante seus experimentos para modelar o movimento de uma cobra real, Hirose (1976) estimou que durante o movimento de serpentina, o corpo da cobra muda conforme uma curva senóide, e para tal ele criou a equação da curva serpenoid, conforme descrita nas equações eq. 2.1 e eq. 2.2.

$$x(s) = sJo(\alpha) + \frac{4l}{\pi} \sum_{m=1} \frac{-1^{2m}}{2m} J2m(\alpha) sin(m\pi \frac{s}{l}) \tag{2.1}$$

$$y(s) = (\sum_{m=1} -1^{m-1} J \frac{(2m-1(\alpha))}{2m-1} sin(\frac{2m-1}{2} \pi \frac{s}{l})) \frac{4l}{\pi} \tag{2.2}$$

Onde:

- J = Momento de inércia de cada módulo;
- m = A massa de cada módulo;
- l = Distância do centro de gravidade de cada junta;
- α = Âgulo entre o módulo e o eixo;
- s = Segmento infinitesimal ao longo do corpo da cobra;

2.2 Hardware Envolvido

2.2.1 Arduino

Evans, Noble e Hochebaum (2013) define o Arduino como placas de circuito impresso, que contam com um conjunto de recurso, como *driver*, regulador de tensão, entradas e saídas digitais/analógicas e comunicação serial. Todos os modelos de placa Arduino são baseadas em um microprocessador de 8 bits Atmel AVR, que utiliza a arquitetura do tipo reduced instruction set computer (RISC). As primeiras placas Arduino eram baseadas no ATmega8, mas versões mais recentes como o Duemilanove e Uno, utilizam o ATmega328, com memória flash de 32KB, podendo comutar automaticamente entre USB e corrente contínua.

"Dependendo da versão do Arduino há também uma saída USB, possibilitando a conexão do mesmo a um computador para fazer o upload ou recuperar os dados contidos na placa"(MCROBERTS, 2011). "A placa possui também pinos digitais que podem ser codificados como saída ou entrada e os mesmos podem propiciar saídas de modulação por largura de pulso (PWM) caso sejam programados para tal. Há na placa vários protocolos de comunicação e um botão de *reset*"(EVANS; NOBLE; HOCHEBAUM, 2013).

McRoberts (2011) explica que a programação de um Arduino, figura 2, é feita principalmente através de sua IDE padrão, e que a linguagem utilizada para construção dos códigos fonte é toda baseada em linguagem C, onde através de sketches, o desenvolvedor pode fazer o upload para a memória flash do ATMega, microcontrolador do Arduino, através da porta USB, permitindo a interação do Arduino com qualquer periférico conectado.

Figura 2 – Arduino UNO fonte: autor

2.2.2 Shields

"Existem placas especializadas chamadas shields, que podem expandir a funcionalidade básica do Arduino"(EVANS; NOBLE; HOCHEBAUM, 2013). Segundo McRoberts (2011), os shields são circuitos que se encaixam no arduino, e apresentam uma grande variedade de

componentes, como *drivers*, *displays* de LCD, receptores GPS, teclados, sensores, dentre outros *gadgets*, incrementando as funcionalidades do mesmo. Na figura 3 podemos ver um exemplo de um sensor shield, para conexão de servo motores e sensores.

"O Arduino Sensor Shield é um extensor de entradas e saídas com diversos conectores, e pode ser usado como fonte para servo motor ou micro servo motor. Eles podem incorporar diversos outros equipamentos de forma rápida e prática como sensores, módulos de comunicação e câmeras"(TECNOTRONICS, 2018).

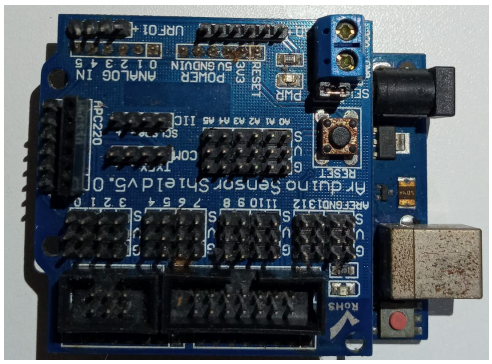

Figura 3 – Sensor Shield fonte: autor

Um tipo de componente importante à construção de um robô são os componentes necessários a sua comunicação, pois através deles podemos enviar comandos aos atuadores do robô, transmitir os dados coletados pelos sensores, ou seja, trocar informações relevantes para o funcionamento adequado do autômato.

> O módulo WiFi ESP8266, figura 4, possui uma pilha de protocolos TCP/IP integrada que pode dar a qualquer microcontrolador acesso a sua rede WiFi. O ESP8266 é capaz de hospedar um aplicativo ou todas as funções de rede Wi-Fi de outro processador de aplicativos. Cada módulo ESP8266 vem pré-programado com um firmware do conjunto de comandos AT, ou seja, você pode simplesmente conectar isso ao seu dispositivo Arduino e utilizar, toda a capacidade de WiFi que o WiFi Shield oferece de forma imediata (BAIG; CHITAY; SALAHUDDIN, 2016).

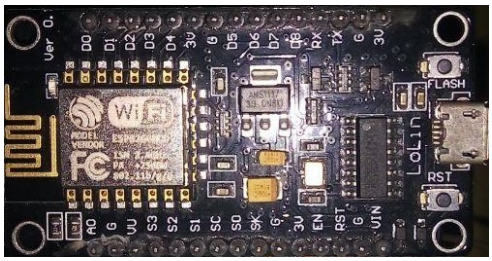

Figura 4 – Módulo ESP8266 fonte: autor

2.2.3 Sensores e Atuadores

"Um robô autônomo necessita de sensores e atuadores para interagir com o meio ambiente de forma eficiente. Essa interação com o ambiente pode ser deliberativa, reativa,

híbrida ou hierárquica"(WOLF et al., 2009). Segundo Braga (2019), sensores ultrassônicos são bastante utilizados para detectar objetos em seu raio de ação, ou a distância entre o objeto detectado e o mesmo. Suas aplicações são diversas, desde a detecção da presença de pessoas, à medição do nível de um silo. Eles podem operar detectando variações, de milímetros à vários metros, sem a necessidade de contato direto com os objetos detectados. Seu princípio de operação é exatamente o mesmo do sonar, usado pelo morcego para detectar objetos e presas em seu voo cego. Na figura 5, podemos ver o sensor ultrassônico HC-SR04. "O HC-SR04 é um sensor ultrassônico de pequenas dimensões e baixo custo, que funciona a uma frequência de 40 Hz. Ele tem alcance de 4 m e precisão de 3 mm"(ELECFREAKS, 2010).

Figura 5 – Sensor HC-SR04 fonte: ElecfreakesDatasheat, 2010

"Servomotor é um circuito fechado, um servo mecanismo que utiliza realimentação de posição final, permitindo o controle preciso da posição angular medida, como também, a velocidade do eixo de saída pelo envio de um sinal na entrada"(MEDEIROS, 2011).

Conforme definição de Weg (2003), os servomotores são atuadores empregados em aplicações onde há a necessidade de se ter um controle de alta precisão, de velocidade e(ou) posicionamento, e em dimensões reduzidas. Na figura 6 temos exemplo de um micro-servo de posição.

"Este tipo de atuador pode ser encontrado em sistemas de automação, na robótica e em auto e aeromodelismo, onde o posicionamento preciso do servomotor é essencial para o funcionamento destes sistemas"(EPUSP, 2014).

Figura 6 – Micro Servo Sg90 fonte: autor

Um outro componente fundamental a construção desse trabalho é a câmera de imagem térmica. Segundo Baltazar, Gokhale e Bansod (2011), ela detecta luz infravermelha

e converte para uma faixa visível do espectro, baseado numa onda de luz dependente da temperatura emitida pelos objetos. A energia encontrada na luz é diretamente proporcional ao seu comprimento de onda. Cores não são nada além de luz visível. Luzes infravermelhas são as luzes que estão abaixo de vermelho ou mais escuro que o vermelho que não pode ser visto a olho nu.

> A câmera térmica Adafruit AMG8833, conforme figura 7, é uma matriz de 8x8 de sensores térmicos infravermelhos. Quando conectado ao seu microcontrolador (ou raspberry Pi), ele retornará uma matriz de 64 leituras individuais de temperatura infravermelha sobre o protocolo I2C. Funciona tal como as sofisticadas câmeras térmicas, sendo mais compactas e simples o suficiente para facilitar a integração. O AMG8833 medirá temperaturas que variam de 0 ° C a 80 ° C, com uma precisão de + - 2,5 ° C e pode detectar um ser humano a uma distância de até 7 metros com uma taxa de quadros máxima de 10 Hz (MILLER, 2017).

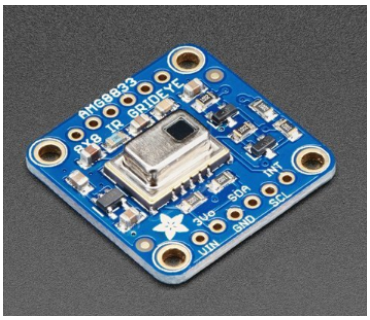

Figura 7 – Câmera termica AMG8833 fonte: (MILLER, 2017)

2.2.4 Mecânica Envolvida

Tão necessários a concepção do robô cobra quanto os sensores, atuadores e *drivers*, abordados anteriormente, alguns conceitos de mecânica são pilares, tal como engrenagens, buchas e acoplamentos. Segundo Cunha (2008), os acoplamentos ou uniões de veios são utilizados quando é necessário ligar dois veios onde seus eixos estejam alinhados, ou levemente desalinhados, na forma axial ou angular.

> As engrenagens são, pois, mecanismos compostos por rodas dentadas rígidas que transmitem movimento entre veios afastados e/ou quando se pretende reduzir, ou aumentar a velocidade ou o momento do veio motor. Podem ser consideradas como uma evolução ou aperfeiçoamento das rodas de atrito. São utilizadas para transformar o movimento de um veio rotativo num movimento de rotação ou de translação (FLORES; GOMES, 2014).

Ferreira e Gordo (2018) explicam que para suportar os eixos, reduzir o atrito, ou amortecer choques e vibrações em estruturas mecânicas, são utilizados tanto mancais quanto as buchas, e quanto ao tipo, eles podem ser de deslizamento ou rolamento.

2.3 Inteligência Artificial e Machine Learning

"Inteligência Artificial (IA) pode ser definida como o ramo da computação que se ocupa da automação do comportamento inteligente ou até mesmo como uma coleção de problemas e metodologias estudadas pelos pesquisadores de IA"(LUGGER, 2013).

Mitchell (1997) classifica *machine learning* como uma subárea da inteligência artificial, onde ao abranger uma série de algorítmos adaptativos, permite o aprendizado mediante dados fornecidos ao sistema.

"Qualquer mudança em um sistema que melhore o seu desempenho na segunda vez que ele repetir a mesma tarefa ou outra tarefa tirada do mesmo conjunto de dados pode ser considerado como aprendizado"(SIMON, 1983).

Segundo Mitchell (1997), os algoritmos de *machine learning* provaram ser de grande valor prático em uma variedade de aplicações., onde baseiam-se em ideias de um conjunto diversificado de disciplinas, incluindo inteligência artificial, probabilidade e estatística, complexidade computacional, teoria da informação, psicologia e neurobiologia, teoria do controle e filosofia. Nesse contexto, o aprendizado envolve a generalização da solução a partir da experiência, ou seja, o desempenho melhora a cada repetição da mesma tarefa, e mais ainda nas repetições de tarefas semelhantes no mesmo domínio.

> Como o aprendizado envolve uma interação entre a máquina e o ambiente, pode-se classificar tarefas de aprendizado de acordo com a natureza dessa interação. Uma delas é entre aprendizado supervisionado e não supervisionado. No aprendizado supervisionado se tem um cenário onde o treinamento contém informações significativas e complementares que os dados de testes não tem, algo como rótulos. Já no aprendizado não supervisionado, não há distinção entre dados de teste, durante o aprendizado, se processam dados de entrada com o objetivo de chegar a algum resumo ou versão compactada desses dados (SHALEV-SHWATZ; BEN-DAVID, 2014).

2.3.1 Redes Neurais

Quando tratamos de um sistema computacional que se utiliza de IA, a arquitetura da rede neural é muito peculiar a solução que se está buscando. "As redes neurais artificiais (RNA) são como sistemas computacionais formados por unidades de processamento simples interligadas e paralelamente dispostas, denominadas neurônios, que armazenam o conhecimento e o generalizam para um conjunto de dados desconhecidos"(COSTA-FILHO et al., 2019).

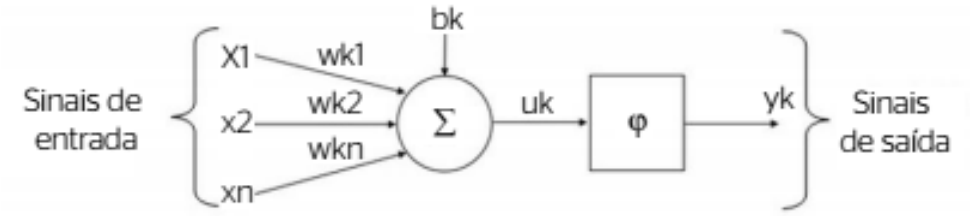

Figura 8 – Neurônio Artificial fonte: Dias, Librantz e Santos (2020)

"A RNA mais simples é aquela composta de apenas um neurônio, chamado perceptron, onde o mesmo possui diversas entradas com seus respectivos pesos, um valor limite para decidir se a sua saída será igual a 1, e o valor de saída. Ao juntar muitos perceptrons formamos uma rede RNA"(TANAKA, 2018). Na figura 8, é mostrado a estrutura típica de um neurônio artificial, onde equações que o compõem eq.2.3 e eq.2.4 são descritas a abaixo.

$$uk = \sum_{j=1}^{n}(wkj \times xj) + bk \tag{2.3}$$

$$yk = \phi(uk) \tag{2.4}$$

Onde:

- uk = estado de ativação do neurônio artificial;
- n = número total de entradas do neurônio artificial;
- j = índice de uma entrada do neurônio artificial;
- wkj = peso sináptico de uma entrada do neurônio;
- xj = entrada do neurônio;
- bk = sinal de bias do neurônio artificial;
- yk = saída do neurônio artificial;
- φ = função de ativação do neurônio artificial.

A escolha da arquitetura de uma RNA é definida por 4 hiperparâmetros principais: número de camadas da rede, número de neurônios em cada camada, o tipo de conexão entre neurônios e a topologia da rede. As redes neurais de múltiplas camadas possuem mais de um neurônio entre uma entrada e uma saída da rede. Dentre as redes de múltiplas camadas temos a do tipo Perceptrons de Múltiplas Camadas (SILVA; LEAL; LIMA, 2019).

Ao treinar um perceptron, devemos comparar o resultado obtido com o resultado esperado, onde alteramos os pesos sinápticos das entradas a fim de minimizar o erro na saída, apurando assim a acurácia da rede neural. Costa-Filho et al. (2019) nos fala que durante o treinamento das RNA, um algoritmo de aprendizado ajusta os pesos sinápticos que ponderam os valores das variáveis de entrada e dos neurônios que compõem as camadas ocultas.

Tanaka (2018) conceitua Rede Neural Convolucional (CNN) como um tipo específico de rede neural destinada ao processamento e análise de imagens, a qual foi proposta inicialmente por LeCun (1998), com o objetivo de reconhecer dígitos manuscritos, e possuía a acurácia de 99,2% . "A arquitetura da CNN simula clusters de neurônios para detectar atributos daquilo que foi visto, organizados hierarquicamente e de forma abstrata o suficiente para generalizar independentemente de tamanho, posição rotação, etc"(TANAKA, 2018).

Segundo Tanaka (2018), uma matriz de três dimensões é a entrada de uma rede CNN, onde para uma imagem em RGB, cada dimensão representa uma camada de cor, tendo em cada dimensão um conjunto de pixels. Sendo assim, uma imagem quadrada de 250 pixels, por exemplo, pode ser representada por três matrizes quadradas de 250 pixels, onde o valor de cada pixel em cada dimensão varia entre 0 e 255

"*Deep Learning* é uma técnica de aprendizado de máquina na qual o programa computacional, que constitui uma CNN, aprende a distinguir entre imagens diferentes tal como humanos fazem: com a experiência, onde, para uma CNN aprender a diferenciar objetos, é necessário fornecer imagens classificadas"(SOUZA et al., 2020). O programa vai aprender diretamente a partir de uma variedade de imagens, de várias classes diferentes.

Simonovsky e Komodakis (2017), nos fala que a popularidade e notoriedade das Redes Neurais Convolucionais (CNNs) são atribuídas as tarefas como processamento de imagens, classificação e segmentação de imagens, ou análise de vídeos, tarefas que possuem uma estrutura de grade para representação de dados subjacentes.

Segundo Ferreira et al. (2017), as CNN's possuem uma sequência de camadas, cada qual com sua função específica ao se propagar o sinal de entrada da rede, tal como na figura 9 . Na CNN, as camadas convolucionais, de *pooling* e totalmente conectadas, são consideradas as principais, onde as camadas convoluvionais são responsáveis por extrair atributos das entradas. As camadas de *pooling* reduzem a dimensão das resultantes convolucionadas, e as camadas de totalmente conectadas, responsáveis por propagar o sinal através da multiplicação ponto-a-ponto e das funções de ativação, tal como a RELU. Na saída da CNN temos a probabilidade de pertencimento da imagem de entrada à alguma classe treinada anteriormente pela rede.

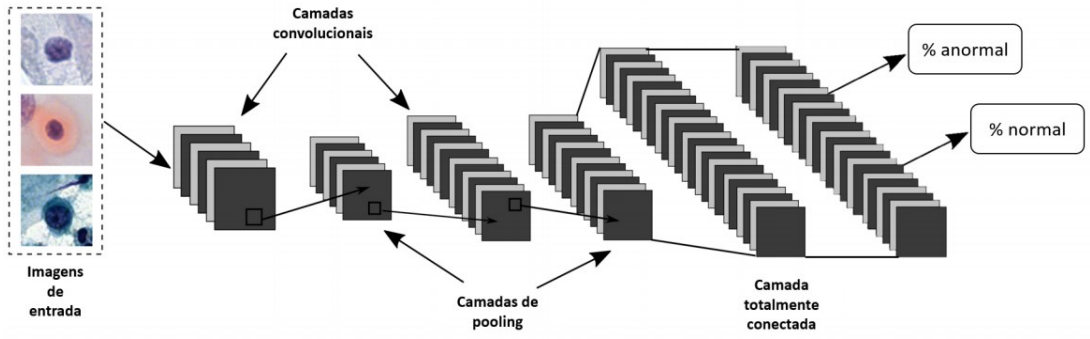

Figura 9 – Arquitetura de uma CNN fonte: Ferreira et al. (2017)

Ng (2018) afirma que convolução pode ser definida como a operação de produto composto da matriz de entrada da imagem pelo filtro (ou kernel) referente a característica que se deseja extrair da sua imagem através da aplicação de um filtro específico, e é representada pelo símbolo "*"(asterisco). Segundo Karpathy (2017) apud Ferreira et al. (2017) , filtros são matrizes que recebem como entrada, através de convolução, um arranjo 3D denominado volume, e que durante o treinamento da CNN, são ajustados de forma que sejam ativados perante características desejadas, como bordas ou mancha de cores. "Cada um desses filtros dá origem a uma estrutura conectada localmente que percorre toda extensão do volume de entrada"(FERREIRA et al., 2017).

Segundo Ng (2018), ao realizar operações de convolução entre a matriz de entrada quadrada, de dimensão $n \times n \times c$, e o filtro específico de dimensão $f \times f \times c$, por exemplo, obtemos uma matriz de saída com as características desejadas extraídas, e com dimensão $\frac{(n-f+2p)}{s} \times \frac{(n-f+2p)}{s} \times c$, onde n é o número de linhas/colunas da entrada, f o número de linhas/colunas do filtro, p o valor do *padding*, e s o valor do *stride*.

Miyazaki (2017) define *stride* como o passo em que os filtros da operação de convolução percorrem toda a matriz de forma sequencial. Ou seja, percorre toda a matriz de entrada pixel a pixel conforme o passo pré determinado. Quando o passo é igual a um, o filtro percorre cada posição uma a uma, se o passo for dois, o filtro irá percorrer duas posições por vez. Na figura abaixo, podemos ver um exemplo de *stride* igual a um, e igual a dois. A figura 10 ilustra o passo com $stride = 1$ e a figura 11 o passo com $stride = 2$.

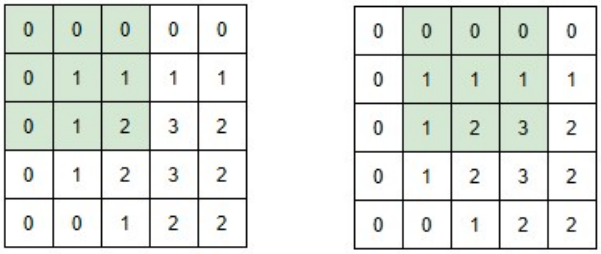

Figura 10 – passo unitário $stride = 1$ fonte: autor

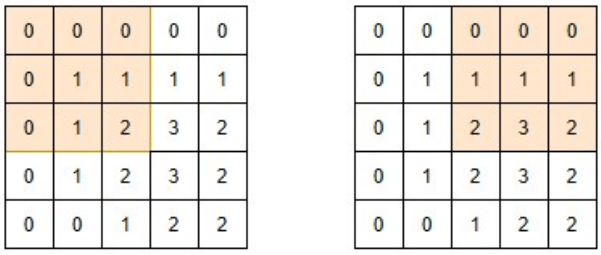

Figura 11 – passo duplo $stride = 2$ fonte: autor

Ng (2018) define *padding* ou preenchimento, como uma outra forma de transformar a CNN, contornando o *shrink output* decorrente de uma operação de convolução para detectar alguma característica, em que a matriz original de dimensão $n \times n$ e um filtro $f \times f$ resultam em uma matriz de saída $(n - f + 1) \times (n - f + 1)$, quando os pixels das bordas são menos utilizados que os pixels mais centrais da matriz de entrada. Ao realizar um *padding* $P = \frac{(f-1)}{2}$, o tamanho $n \times n$ se preserva na saída.

"Os tipos mais utilizados de *padding* são: *valid padding* ou *same padding*. No *valid padding*, as bordas do filtro não ultrapassam as bordas da imagem, enquanto no *same padding*, as fronteiras da imagem são preenchidas com 0"(MIYAZAKI, 2017). Na figura 12 podemos ver um exemplo de *same padding*.

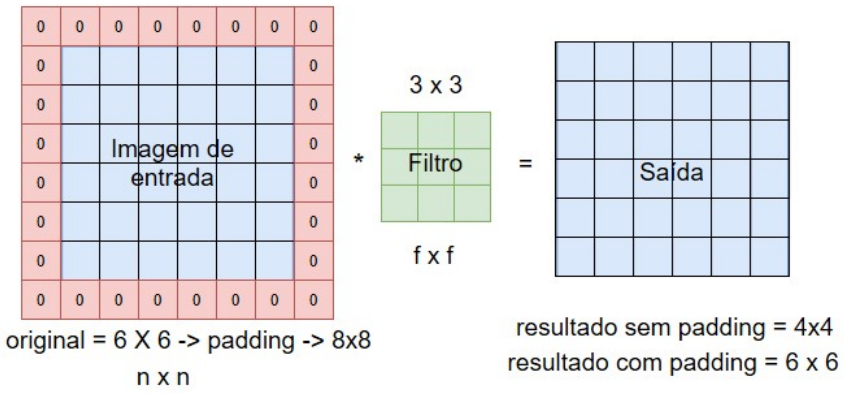

Figura 12 – Exemplo conceitual de same padding fonte: autor

Segundo Miyazaki (2017), "as funções de ativação são funções não-lineares conectadas ao final da estrutura de um neurônio artificial, tendo sua saída baseada nos dados de

entrada e no limiar de ativação. Dentre elas temos a função tangente hiperbólica, sigmoide e a função ReLU".

Ng (2018) costuma definir a função ReLU, ou unidade linear de comutação, através da equação $\phi = max(0, z)$, onde z representa a matriz de dados. Além de ser a função de ativação mais utilizada, é uma função não linear onde a derivada é 1 quando z é positivo, e é 0 quando z é negativo. Apesar da derivada para $z = 0$ não estar definida, ao se implementar de forma computacional, a probabilidade de z ser exatamente 0 é muito baixa, de forma que podemos assumir que a derivada é igual a 1 ou 0 quando z = 0. Na figura 13 podemos ver o gráfico da função ReLU.

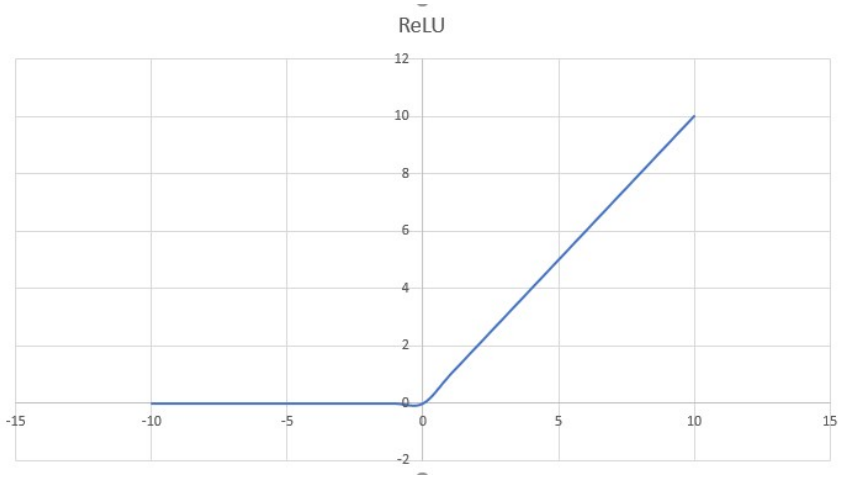

Figura 13 – Função de ativação ReLU fonte: autor

Ainda seguindo seu raciocínio acerca das funções de ativação, Ng (2018) fala que, em casos em que a camada de saída da classificação é binária, usa-se a função sigmoide, mas em outros casos normalmente se usa ReLU. Ela possui vantagens em relação a função sigmoide e tangente hiperbólica pelo fato de que a inclinação é muito diferente de zero para a maioria dos valores de z, ou seja, permitindo as redes neurais aprenderem muito mais rápido, impedindo que o gradiente da função, causa do aprendizado lento, se aproxime de zero.

> O algoritmo de batch propagation, criado por Sergy Oiffe e Christian Szegedy, não apenas facilita a navegação por hiperparâmetros, mas também reduz a correlação entre as redes neurais e os hiperparâmetros. O bacth propagation é realizado através da Normalização da média e da variância de z, após a função de ativação, e antes do treinamento, acelerando o aprendizado para uma quantidade maior de hiperparâmetros, e podendo ser feita em pelo menos uma camada da rede neural(NG, 2018) .

"O processo de treinamento da CNN pode ser dividido em *forward* e *backward propagation*. Enquanto a *forward propagation* é responsável pela transmissão de informações, a *backward propagation* é responsável pela atualização dos parâmetros"(YANG; XU; HONG, 2018).

2.3.2 Tensor Flow e o Keras

"O TensorFlow é uma plataforma de código aberto de ponta a ponta para aprendizado de máquina. Possui um ecossistema flexível e abrangente de ferramentas, bibliotecas e recursos da comunidade, que permite que os pesquisadores desenvolvam o que há de mais moderno em *Machine Learning* (ML), e os desenvolvedores construam e implantem facilmente aplicativos"(TENSORFLOW.ORG, 2020).

Segundo datascienceacademy.com.br (2020) ele possui camadas de API's mais simples que as comumente utilizadas em modelos *Deep Learning*, além de ser multiplataforma, podendo inclusive ser implementada em python. Além disso, possui camadas de API's de alto nível também, como o Keras.

> O Keras é uma API de redes neurais de alto nível, escrita em Python e capaz de executar sobre TensorFlow. Ela foi desenvolvida com foco em permitir experimentação rápida. Ela coloca a experiência do usuário na frente, ao reduzir a carga cognitiva fornecendo feedback claro, além de ser modularizado, podendo ser configurado com o mínimo de restrições possíveis (KERAS.IO, 2019).

2.4 Trabalhos Correlatos

2.4.1 Os snake robots de Gavin Miller

Um dos trabalhos correlatos que nos inspirou desde a idealização do projeto robô cobra para resgate e salvamento, foram os *snake robots* de Dr. Gavin Miller, mais especificamente a versão S5, figura 14, que se apresenta como base para nosso protótipo físico. Miller (2002) dizia que ao tentar construir robôs que imitam e talvez igualem as capacidades de serpentes reais, é possível que criemos ferramentas úteis capazes de transportar sensores, coletar amostras e efetuar mudanças físicas em uma ampla variedade de ambientes. Seus trabalhos se baseiam nos estudos de Hirose (1976), que conseguiu pela primeira vez modelar e implementar a locomoção de robôs baseada em movimentos de serpentes.

Com o propósito de otimizar o trabalho de resgate e salvamento em desmoronamentos, Miller (2002) desenvolveu protótipos de robôs cobra controlados remotamente que se deslocavam dentre os escombros. "Sua movimentação era baseada na movimentação de serpentes reais, e os robôs tinham diversos sensores, desde uma câmera localizada na cabeça, que era utilizada para transmitir imagens para o operador, a sensores de temperatura, distâncias e envio de coordenadas"(MILLER, 2002).

Segundo (MILLER, 2002), a localização é mostrada em relação a um modelo 3-D reconstruído do caminho feito pelo robô cobra, junto com as superfícies que ele detectou. A partir dessas coordenadas, os bombeiros recebem permissão para se aproximarem cuidadosamente do prédio enquanto o operador fala com o sobrevivente através de um alto-falante

presente no robô cobra, deixando-o saber que ajuda em seu caminho e tentando descobrir a extensão de seus ferimentos.

Figura 14 – Snake S5 fonte: Miller (2002)

Os robôs desenvolvidos por Miller (2002) tinham compartimentos internos onde se localizam células de baterias distribuídos pelas estruturas modulares, e sistemas com microcontroladores, a fim de evitar interferências na comunicação com o robô, permitindo comunicação por rádio frequência. O modelo S5 possui rodas, e foi desenvolvido com uma pequena seção transversal e 32 juntas, otimizando a movimentação horizontal ondulatória ao formar uma curva mais senoidal em relação aos modelos anteriores, realizando com eficiência o movimento serpentina.

2.4.2 Robô Autônomo para Detecção de Humanos Baseada em Sensor Laser e Câmera Térmica

Um trabalho correlato muito interessante e pertinente a área de estudo é o projeto de Nin e Osório (2011), que se trata de um sistema de vigilância autônomo, onde o objetivo era fazer um robô móvel detectar humanos em ambientes escuros fazendo o uso de uma câmera térmica e um LIDAR (Light Detection and Ranging). Apesar de não ter relação com busca e salvamento de vítimas em estruturas colapsadas, o estudo acerca do tema levantado pelos autores tem correlação científica no que diz respeito ao desvio de obstáculos de forma autônoma e a identificação de humanos. Na figura 15 podemos ver como funciona a solução de Nin e Osório (2011)

> A detecção do ser humano é feita com base na captura de calor, ou seja, sabendo que o corpo humano possui temperatura relativamente superior ao resto do ambiente, usa-se esta característica como fator principal para processar a imagem. Contudo, há no ambiente alguns objetos como lâmpadas e outros dispositivos eletrônicos que também geram calor. Devido a essa questão, não é trivial a distinção entre um objeto e um corpo humano. Para resolver este problema, é necessário que o algoritmo receba

> como entrada uma imagem do local gerada pela câmera térmica quando não há a presença humana, para efeitos de calibração do sistema. A partir desta imagem é gerado um histograma que é utilizado como base para a análise dos histogramas feitos a partir das novas imagens capturadas em tempo real (NIN; OSóRIO, 2011).

"No projeto a captura em tempo real da câmera é feita utilizando as funções da biblioteca OpenCV", afirmam Bradski Kaehler, (2008) apud Nin e Osório (2011).

> Com essas funções, é possível obter uma imagem digitalizada de 3 camadas de 8 bits no formato RGB com a resolução de 640 x 480 pixels. A digitalização das informações gerada pelo sensor laser é feita utilizando as funções do Player-Stage, onde estes dados são retornados em forma de um vetor contendo para cada ângulo a distância entre o sensor e um objeto (NIN; OSóRIO, 2011).

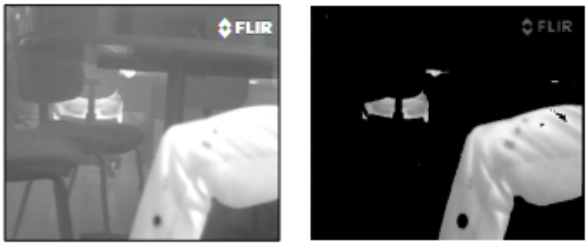

Figura 15 – Aplicação em tempo real fonte: Nin e Osório (2011)

2.4.3 O U-Snake robot da Universidade de Pittsburgh

Um outro trabalho bastante relevante a nosso propósito, em que os objetivos se assemelham com os propostos neste trabalho, é o *snake robot* da biorobotics, da Universidade de Pittsburgh, figura 16. Segundo Ferworn (2012) Trata-se de um modular de laboratório consiste em 16 Módulos conectados em série, com cada junta deslocada 90 graus em relação a junta anterior.

> Isso dá ao robô 8 graus de liberdade rotativos dorsais, e 8 graus de liberdade rotativos laterais. O robô se move ondulando seu corpo através do movimento de coordenação de todas as juntas. Ele funciona mediante controle remoto do operador, e é conectado por um cabo contendo energia, dados e vídeo (FERWORN, 2012).

"O mesmo foi utilizado durante os resgates na Cidade do México, onde o grupo de bombeiros retornava periodicamente a uma base de operações para aguardar atribuições e organizar logística entre as equipes de resgate, e com as informações obtidas pelo robô, puderam realizar as operações de resgate de forma a mitigar os erros durante o processo"(Whitman et al., 2017).

Figura 16 – U-Snake Robot fonte: Whitman et al. (2017)

2.4.4 Relevância dos Trabalhos Correlatos

Os trabalhos correlatos serviram como referência durante o processo de pesquisa e desenvolvimento do nosso robô cobra. Como nossa proposta é de que o robô seja um sistema autônomo de localização de vítimas em estrutura colapsadas, os trabalhos mencionados têm relevância do ponto de vista técnico. Os *snake robots* de Miller (2002) são nossa referência de modelo mecânico, com algumas adaptações. Já o robô de Nin e Osório (2011), trabalha com câmera térmica e LIDAR, além de algoritmos de tratamento e classificação de imagens, alguns dos recursos mais modernos quanto a aquisição de sinais para a área de pesquisa abordada. Já o U-Snake robot da Universidade de Pittsburgh, Whitman et al. (2017), é o que mais se assemelha a nossa proposta, divergindo nos aspectos da aquisição de dados, no tipo de imagem, e no modo de controle proposto. Nos três trabalhos correlatos falta a parte de inteligência artificial, tanto para tornar o robô cobra de Miller (2002) e o de Whitman et al. (2017) autônomo, quanto para a classificação preditiva no robô vigilante de Nin e Osório (2011), e no U-Snake de Whitman et al. (2017) . Nosso trabalho se diferencia dos correlatos pela implementação de um sistema de locomoção autônomo, além de uma classificação de imagens térmicas inteligente, usando IA.

3 Metodologia

Neste capítulo é descrita toda a metodologia utilizada para a construção do robô cobra, do sistema de controle, de como o sistema funciona de forma detalhada, ou seja, como a solução foi construída. Na sessão 3.1 descrevemos toda a construção mecânica, além dos circuitos eletrônicos e suas conexões. A sessão 3.2 aborda toda a parte de sistema de controle, onde toda a construção da solução é modularizada. Para tal abordamos a descrição geral da solução, a comunicação do sistema, a movimentação do robô, além do uso dos sensores e classificação das imagens térmicas.

3.1 Construção do Robô

3.1.1 Parte Mecânica

A construção do robô cobra se deu baseada no protótipo do *snake* S5 de Miller (2002), onde a estrutura permite a montagem em módulos, de forma a otimizar a locomoção do robô. Assim como o *snake* S5, o robô cobra conta com rodas em cada falange. Como sua estrutura foi dimensionada para um ambiente em 2D, no que se refere ao material utilizado na construção da parte mecânica, foram utilizados módulos *pan tilt* usinados em alumínio, diferente do estipulado inicialmente, onde cada peça seria impresso em polímero de alta resistência na impressora 3D. Em cada junta, se fez necessário a utilização de um conjunto de peças mecânicas, dentre elas, mancais, eixos, engrenagens e união de veios, para dar mais resistência e centralizar as rotações de eixo. Na figura 18 podemos ver um módulo *pan tilt* desmontado e um conjunto de falanges montadas.

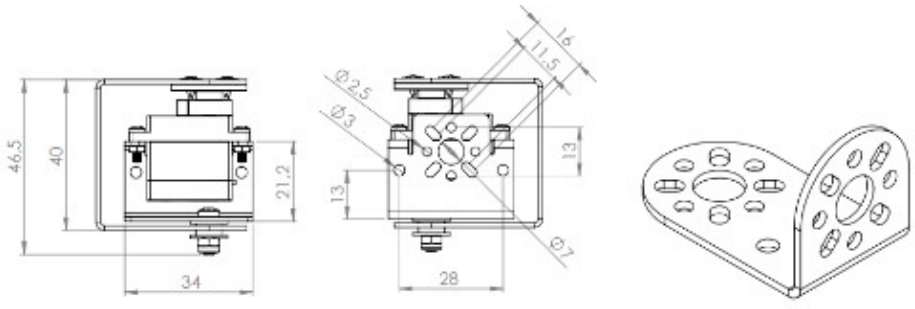

Figura 17 – Dimensões Pan Tilt fonte: autor

A montagem completa da estrutura mecânica do robô cobra foi realizada de forma modular, onde foram utilizados 12 micro servos SG90 acoplado nos suportes *pan tilt*s, tendo sua transmissão de movimento realizada pelas engrenagens, e estabilizadas pelos acoplamentos e mancais de deslizamentos, além de 12 pares de rodas do tipo LEGO

micro, que foram fixadas em cada módulo. Os sensor ultrassônico e a câmera térmica foram fixados na cabeça do robô cobra, para facilitar a identificação de obstáculos e de vítimas nas estruturas colapsadas.

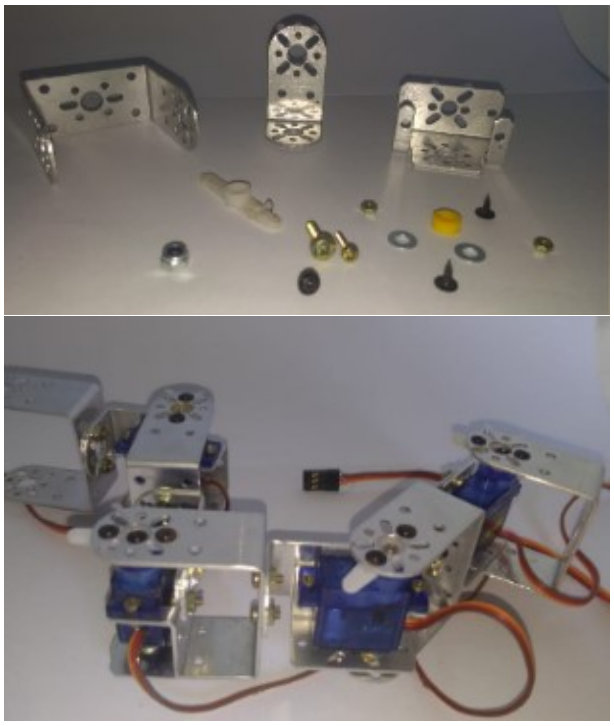

Figura 18 – Módulos do robô cobra fonte: autor

A escolha por esse modelo de protótipo físico foi devido a uma modelagem das movimentações, que se são num ambiente simulado, cartesiano 2D. Esse modelo possui uma bio-mecânica mais simples, baseada em movimentos rotativos modulares, onde cada módulo torna-se parte do eixo do seu módulo antecessor.

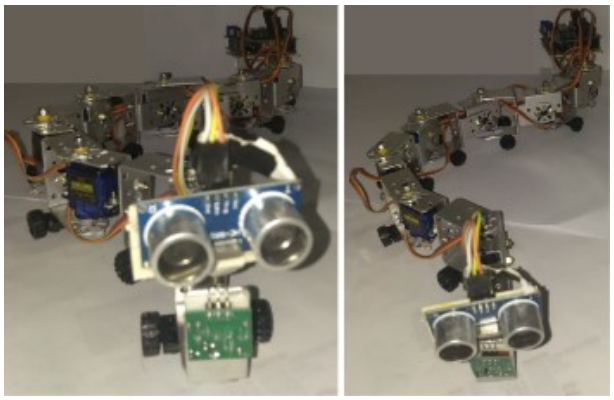

Figura 19 – Parte mecânica montada fonte: autor

3.1.2 Circuitos Eletrônicos

Para a utilização devida do protótipo mecânico, além do sistema de controle, os circuitos eletrônicos são parte fundamental deste conjunto funcional. Para tal foram utilizados

dois circuitos distintos, que posteriormente foram unidos em um circuito único. Devido a modularidade do Arduino e seus componentes, não foi preciso para a montagem desse protótipo projetar nenhum componente de *hardware*, apenas juntar os módulos, conforme descrito nesta seção. Na figura 20, podemos ver o esquema de montagem dos micro-servos com o Arduino UNO.

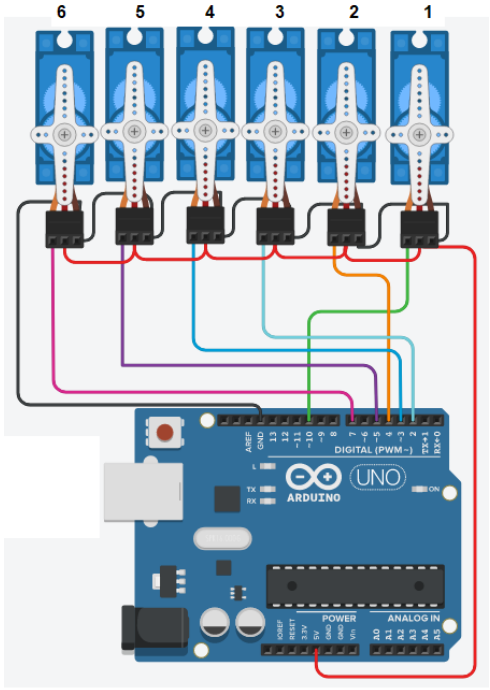

Figura 20 – conexões do arduino com os micro-servos fonte: autor

Para realizar todas as comunicações TCP/IP utilizando Wifi 802.11, foi utilizado um módulo ESP8266, onde o sensor térmico AMG8833, e o sensor Ultrassônico HR-S04 são conectados nas portas do ESP8266 conforme "esquema abaixo". Esse módulo se conecta com o Arduino através das portas seriais, Tx, e Rx, onde o ESP8266 envia informações relativo aos comandos de movimentação para o Arduino, conforme figura 21.

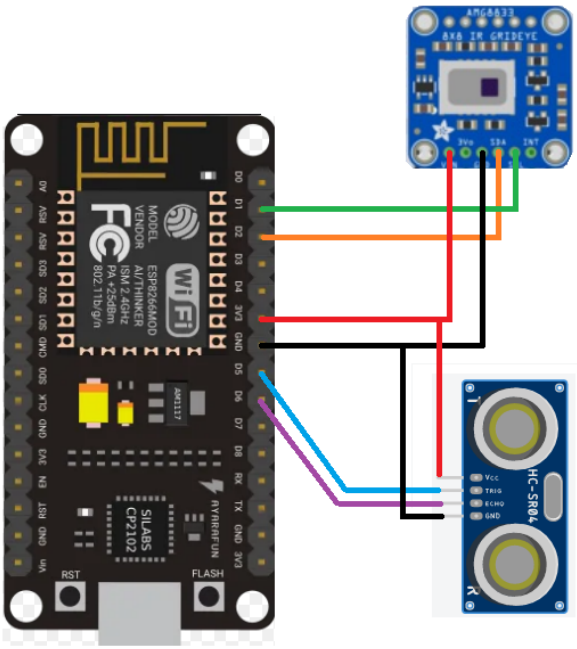

Figura 21 – conexões feitas no ESP8266 fonte: autor

No Arduino estão conectados todos os micro-servos SG90, responsáveis pela movimentação modular do robô cobra, posicionando cada módulo de acordo com os ângulos gerados via software. Cada micro-servo é conectado a uma entrada digital através do sensor shield, figura 22, onde em cada conector digital dispõe de um VCC(5v), Sinal(sinal referente ao posicionamento do servo) e o GND (Negativo ou Ground).

Figura 22 – conexões feitas no sensor shield fonte: autor

A conexão entre o Arduino e o ESP8266, se dá através do Tx, Rx de cada um conforme figura 21. A alimentação do circuito completo, para uma maior mobilidade, deve ser feito com baterias 5v 4A. Na fgura 23 podemos ver o esquema de montagem do circuito completo.

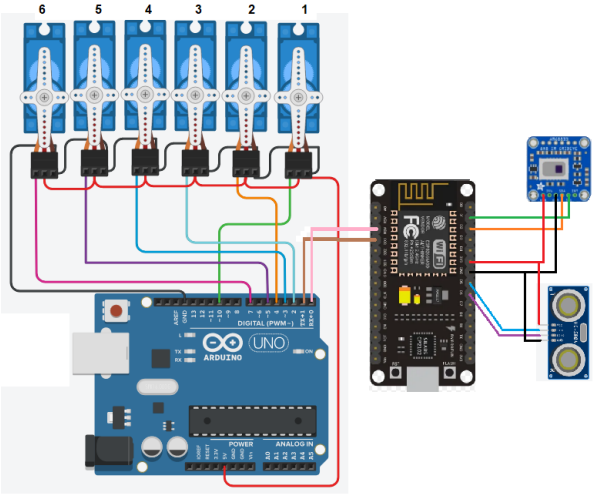

Figura 23 – Circuito completo fonte: autor

3.2 Sistema de Controle

3.2.1 Descrição Geral da Solução

O sistema de controle é constituído por um conjunto de softwares que se comunicam através de um protocolo TCP/IP via wifi, e Serial. A comunicação Wifi foi implementada através de um socket entre o módulo Wi-Fi ESP8266, e a central de controle. O módulo ESP8266 faz a comunicação serial Rx/Tx direta com o arduino presente no robô. Basicamente o módulo ESP8266, presente na cabeça do robô cobra, transmite os sinais advindos do sensor ultrassônico e da câmera térmica AMG8833. Dessa forma, basicamente o ESP8266 orquestra a comunicação entre os programas e dispositivos, transmitindo via Wifi para central os dados advindo dos sensores, depois recebendo a decisão de movimentação da central e repassando os comandos de movimentação para o arduino via comunicação serial.

Baseado no padrão das imagens térmicas advindas do sensor AMG8833, foi gerada uma base de dados de imagens térmicas, para fins de criação do modelo de rede neural, e treinamento da mesma, utilizando TensorFlow com Keras. A saída da rede neural é um fator crucial para a tomada de decisão na movimentação do robô cobra, pois o mesmo ao encontrar algo dentro do raio de 15cm do sensor ultrassônico, busca essas imagens térmicas para classificá-las. Devido a nossas imagens terem sido geradas pelo sensor AMG8833, o volume e variedade de imagens não se mostrou suficiente para gerar e treinar o modelo de uma forma mais eficiente, sendo necessário fazer *data augmentation*.

"O termo *data augmentation* refere-se a métodos para construir otimização iterativa ou algoritmos de amostragem através da introdução de dados não observados ou variáveis latentes"(MAESTRE, 2018).

Toda a construção dos softwares foi realizada fazendo-se o uso de duas linguagens de

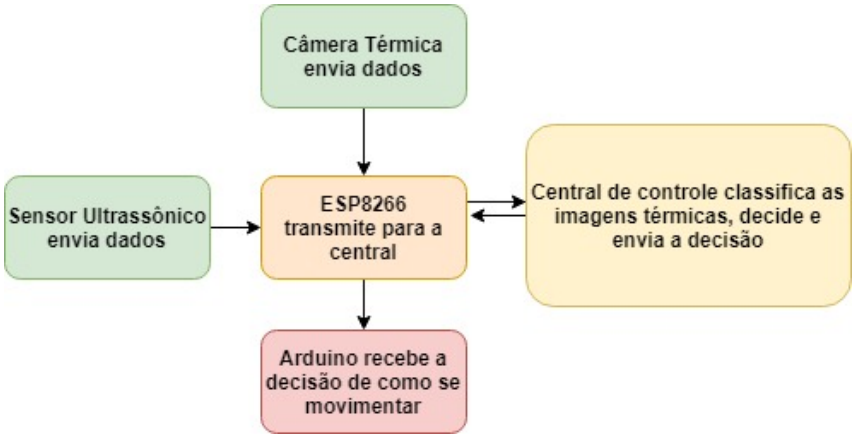

Figura 24 – Fluxo de dados do sistema fonte: autor

programação, Python, e C. Nos embarcados, arduino e ESP8266, foram utilizadas C para implementação. Na central de controle, onde são classificadas as imagens e tomadas as decisões de movimentação, foi utilizado Python 2.7 na implementação. A parte referente a classificação preditiva das imagens térmicas, utilizando IA, foi implementada utilizando TensorFlow com Keras.O fluxo de dados do sistema funciona conforme o diagrama descrito na figura 24.

3.2.2 Comunicação do Sistema

A comunicação de todo o robô foi elaborada em protocolo TCP e Serial. Entre o ESP8266 e o arduino, é utilizada comunicação Serial, onde através do Tx e Rx de cada um, conectados de forma inversa, são transmitidos dados ao arduino relacionado a execução de comandos, como avançar ou recuar. Entre o ESP8266 e a central de controle é utilizada comunicação em TCP/IP, utilizando um cabeçalho simples para manter a estrutura completa dos pacotes.

Como os sensores, ultrassônico e câmera térmica, estão captando dados do ambiente a todo momento e transmitindo pelas portas analógicas e digitais ao ESP8266, esses dados são armazenados em um pacote. Através da comunicação TCP/IP via WiFi 802.11 ponto a ponto, o ESP8266 manda esse pacote de 3 bytes a central, informando o tamanho do pacote de dados, que inclui o *array* de temperaturas obtidas pelo sensor AMG8833, e a distância do objeto detectado, proveniente do sensor ultrassônico.

3.2.3 Movimentação do robô cobra

A movimentação do robô cobra se dá de forma modularizada, onde cada micro-servo recebe um sinal do arduino referente ao ângulo que o mesmo deve se posicionar. Para que os micro-servos se movimentem de forma síncrona e similar a uma movimentação

serpentina, ou rasteira, tal como uma cobra, seguimos um modelo de movimentação que se baseia nas equações de Hirose (1976). A equação utilizada é a seguinte:

$$\theta v = 90 + \gamma + \alpha \times cos(\frac{f \times c \times \pi}{180 + 8 - 4(v-1) \times \varphi}) \tag{3.1}$$

Nessa equação, o angulo θ varia conforme a ordem do servo acionado, 1 à 6, gerando uma variação dentro do cosseno entre -12 e +8 através da variável v. Os demais itens da equação são descriminados a seguir:

- α - Controla a amplitude da curva do robô durante a movimentação;
- f - Ajusta a frequência da curva de movimentação, deixando os movimentos mais curtos ou longos;
- c - Necessário para incrementar o valor que cada servo recebe de acordo com o método de mover-se;
- φ - controla o atraso de fase da curva de movimentação do robô

Além dessas, utilizamos também o gama, que pode ser o rightOffset, ou leftOffset, onde são responsáveis por alterar sentido da inclinação, fazendo o robô virar a direita ou a esquerda. Quando o objetivo é apenas se deslocar para frente ou para trás, a variável = 0. Ao se mover para a frente, o c é incrementado de 0 á 360, para trás, a variável c, é decrementada de 360 a 0.

Para que o robô cobra se movimente adequadamente, ele recebe um comando da central de controle, através do ESP8266, via porta serial, que pode ser "avance", "recue", "direita", ou "esquerda", além do comando de "verificar", que faz uma rotação de 180º da cabeça do robô, verificando a presença de objetos no seu raio de ação. Ao se fazer uma leitura, se o sensor ultrassônico encontrar um objeto a menos de 15cm de distância a sua frente, o classificador verifica a imagem térmica desse objeto. Sendo um organismo vivo, ele alerta ao monitor, senão, ele apenas marca a coordenada e busca caminho a direita. Caso o caminho a direita esteja obstruído, ele busca a esquerda. Caso esteja obstruído também, ele recua e busca novamente. Dessa forma, o robô se movimenta de forma autônoma em um ambiente controlado. Na figura 25 podemos ver um fluxograma referente a movimentação autônoma do robô cobra.

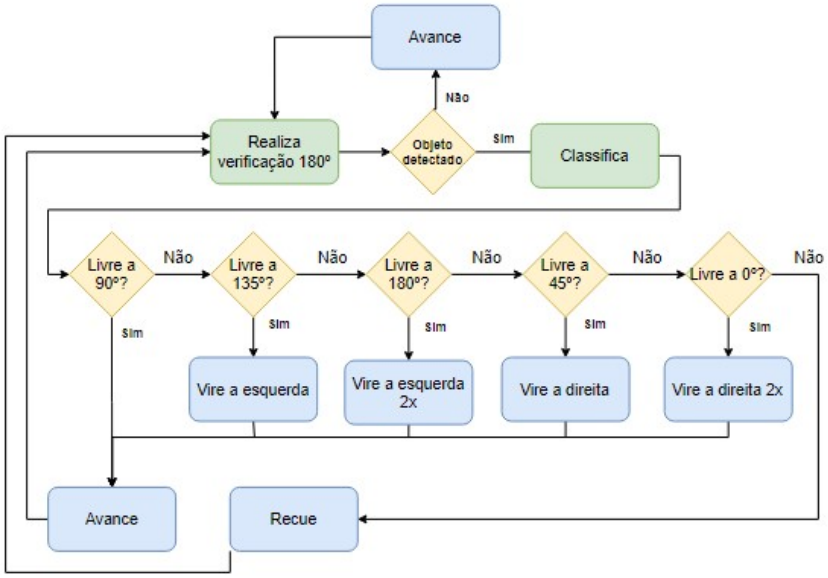

Figura 25 – Fluxograma de movimentação autônoma do robô cobra fonte: autor

3.2.3.1 Sistema de Navegação Primário

Em todo momento um processo ocorre em paralelo, que são, a verificação da solicitação de comandos manuais, e a marcação de coordenadas. Caso o robô venha a receber comando manual de um operador, todo processo automático é interrompido. As coordenadas são marcadas de acordo com a distância percorrida pelo robô em cada movimento, as quais, além de serem mostradas no monitor da central, são armazenadas num csv para posterior consulta. Esse é o sistema de navegação primário, parte do sistema de controle e monitoramento do robô cobra. Na figura 26 podemos ver o robô cobra avançando e virando a esquerda.

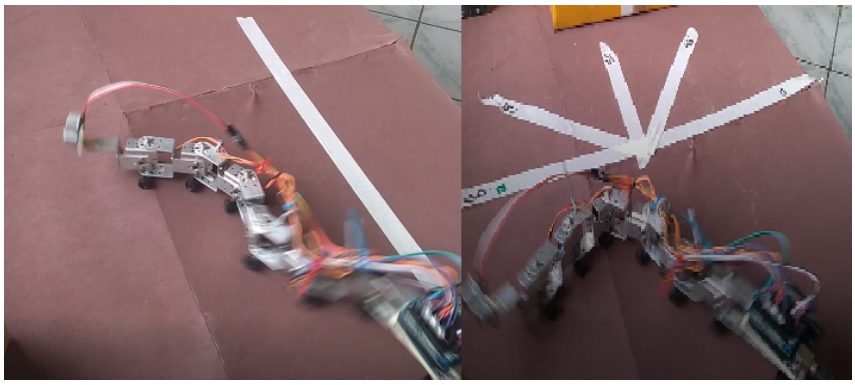

Figura 26 – robô cobra se movimentando fonte: autor

Cada coordenada armazenada segue o padrão (a,b), onde a é o quanto avançou no eixo x,e b o quanto avançou no eixo y. Como exemplo, se o robô andar para frente uma vez partindo do (0,0), ele vai marcar a coordenada (14,0), onde ele percorreu um passo de 14 cm. Se andar a direita em seguida, ele marcará (14,-14), andando a esquerda em

seguida o mesmo marcará a coordenada (28,-14). O gráfico gerado por essa movimentação é mostrado abaixo:

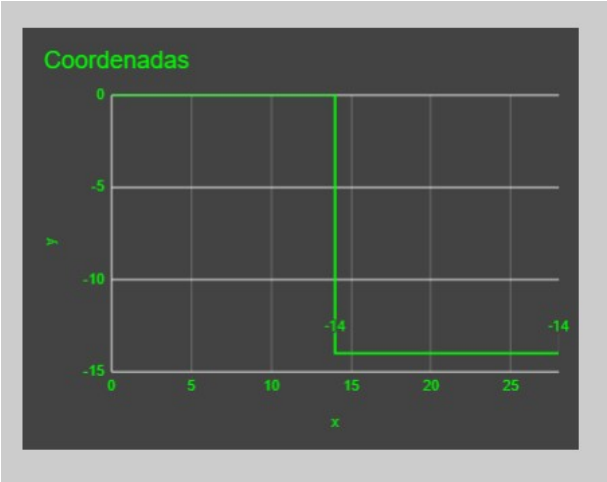

Figura 27 – Gráfico gerado pelas coordenadas conforme exemplo fonte: autor

3.2.4 Aquisição de Sinais dos Sensores

Os sensores são parte fundamental nesse projeto, pois fazem o papel de olhos e ouvidos do nosso robô, coletando dados do ambiente e de organismos presentes. Além disso, caso se utilize o controle manual, as imagens térmicas geradas e mostradas no monitor da central de controle, além das distâncias relatadas pelo sensor ultrassônico, ajudariam o operador no modo manual. O sensor ultrassônico HR-S04 detecta objetos em seu raio de ação, mais especificamente a frente do sensor, onde as medidas válidas para nossa central de controle são objetos detectados a uma distância menor que 15 cm. Para tal, fazemos uso de duas portas digitais, uma como trigger, e outra como echo.

O pino trigger do sensor HR-S04 é uma saída, ele emite um ultrassom que bate no objeto, e volta para o sensor. Quando o ultrassom refletido retorna, ele é captado pela entrada digital echo que recebe esse sinal para ser tratado via software. O valor recebido em echo, é o tempo que demora para o ultrassom bater no objeto e voltar. Esse tempo em milissegundos é tratado para nos dar a distância em centímetros (cm).

O sensor térmico AMG8833 capta temperaturas do objeto detectado, em forma de matriz 8X8, convertida em uma imagem renderizada de 224X224, que será o input da CNN, no tamanho padrão de input do modelo da que estamos utilizando. Para tal, o sensor realiza cada leitura de um objeto fazendo uso da biblioteca do arduino Adafruit-tAMG88xx, onde é instanciada em amg, e através do comando readPixels, o programa pega todos os pixels gerados pelo sensor térmico em um *array*. No momento seguinte, é construído um JSON, com a *array* das temperaturas e a distância obtida pelo sensor ultrassônico, e então é enviado para a central de controle via TCP/IP, na porta 23.

As imagens geradas não são muito distinguíveis a olho humano, porém, como um dos objetivos específicos deste trabalho é classificar imagens térmicas extraídas de um sensor, onde as possibilidades de ocorrências nos direcionam ao uso de vários padrões de imagens a serem identificados, além das variações das mesmas, utilizamos uma Rede Neural Convolucional.

3.2.5 Dataset de Imagens Térmicas

Um dos objetivos específicos deste trabalho foi a criação de uma base de dados de imagens térmicas, para que pudéssemos gerar nosso modelo preditivo e treina-lo. Para tal, utilizamos a própria câmera térmica do robô cobra, o AMG8833. Tal escolha se deu devido a peculiaridade do formato e tamanho das imagens geradas a partir de uma matriz de temperaturas adquirida.

Cada leitura realizada pelo AMG8833 capturava uma matriz de temperaturas de dimensão 8x8, que através da biblioteca pygame, do python, através da definição do *width* e *height*, transformamos cada matriz de temperaturas 8x8 em uma imagem colorida de dimensões 224 x 224 x 3, figura 28, onde os valores de 224 correspondem ao *width* e *height* de uma imagem quadrada, de tamanho compatível com a CNN do tipo MobileNet, utilizada em nosso modelo, e o número 3 as camadas de cores RGB.

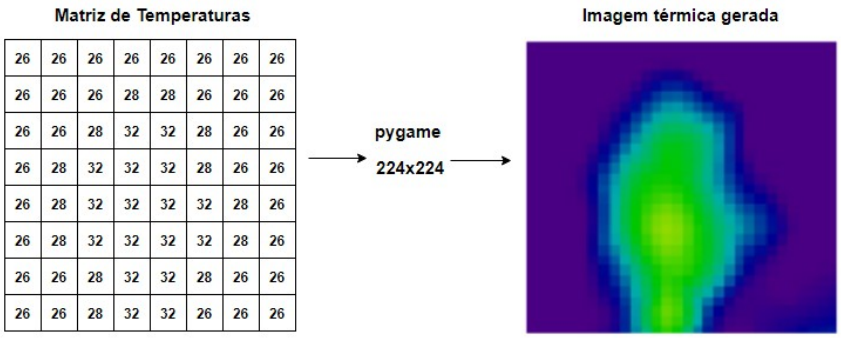

Figura 28 – Processo de obtenção de imagens térmicas fonte: autor

A partir desta etapa, todas nossas imagens do *dataset* foram capturadas e geradas para compor duas classes distintas, vivo ou não-vivo. Na classe de vivo, escolhemos colocar imagens termicas geradas a partir de cães, gatos, pessoas, e partes dos seres citados, como patas, braços ou cabeças. Na classe de não-vivos optamos por capturar imagens térmicas de parede, travesseiros, livros, blocos cerâmicos, além de chamas. A escolha pelos objetos e seres vivos supracitados se deu pela fácil disposição para a aquisição dos dados de forma experimental.

Nessas especificações conseguimos um *dataset* relativamente pequeno, algo em torno de 200 imagens por classe, inadequado para a geração de um modelo de classificação eficiente, com acurácia superior a 96 %. Para enriquecer o *dataset*, geramos variações das

nossas 200 imagens capturadas através de *data augmentation*, onde conseguimos gerar duas classes com um número de 32.000 imagens térmicas. Durante a implementação do *data augmentation* utilizamos operações de rotações, *zoom in*, *zoom out*, deslocamento de altura e largura, cisalhamento e giro horizontal. Ao reduzir o número de variação por imagens para 10/1, conseguimos um dataset com cerca de 11% do obtido com *data augmentation* originalmente, o qual usamos para gerar o modelo preditivo. A partir dai, utilizamos parte das imagens residuais das 32.000 geradas anteriormente, para poder treinar o modelo preditivo.

> Para criar modelos úteis de Deep Learning, o erro de validação deve continuar diminuindo com o erro de treinamento. *Data augmentation* é um método muito poderoso de conseguir isso, onde os dados gerados a partir do mesmo representarão um conjunto mais abrangente de possíveis pontos de dados, minimizando assim a distância entre o conjunto de treinamento e validação, bem como qualquer conjunto de testes futuros (SHORTEN; KHOSHGOFTAAR, 2019).

"Existem várias maneiras de se fazer um *data augmentation*. Nas imagens, a imagem original pode ser girada, alterar as condições de iluminação, cortá-las, para que uma imagem possa gerar diferentes subamostras, aumentando em volume e diversidade o *dataset*"(MAESTRE, 2018). Podemos ver abaixo variações da mesma imagem térmica como exemplo.

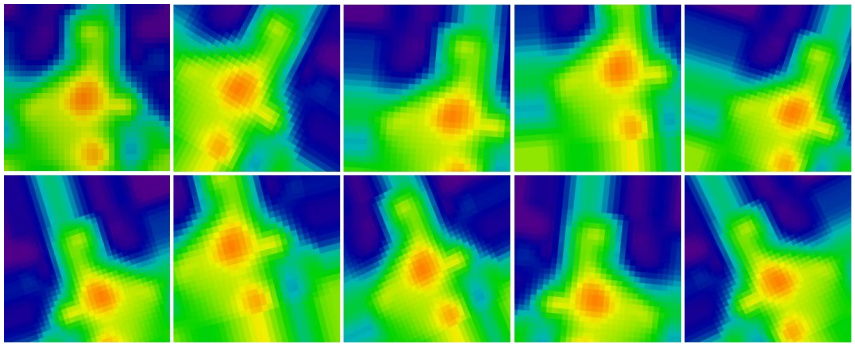

Figura 29 – Imagem térmica original e suas variações através de data augmentation fonte: autor

Com um *dataset* robusto, tivemos problemas computacionais ao tentar gerar nosso modelo preditivo, o que nos levou a reduzir o volume de imagens, refatorando o *dataset* até que o mesmo chegasse a um tamanho de 3.497 imagens na classe de organismos vivos, e 1.775 imagens na classe de não vivos. Dentre as imagens que compunham o *dataset* de imagens térmicas, 525 imagens de vivos, e 267 imagens de não vivos, foram destinados a treinamento e validação do modelo preditivo gerado.

3.2.6 Classificação das Imagens Térmicas

No sistema de classificação das imagens térmicas do robô cobra, foi-se gerado um modelo de CNN a partir do TensorFlow com Keras, fazendo uso da ferramenta Teachable Machine da Google, onde adicionamos as classes de imagens térmicas para construção do modelo, e alteramos parâmetros referentes a Epochs, determinando a quantidade de vezes que cada amostra do conjunto de dados de treinamento é alimentada pelo modelo, e Batch Size, selecionando a quantidade de amostras para cada interação de treinamento.

"O Teachable Machine é uma API da google que facilita e cria rapidamente modelos de aprendizado de máquina para seus projetos, possibilitando o reconhecimento de imagens, sons e poses, e disponibilizando a exportação do modelo para uso em sites e outras aplicações"(LAB, 2019).

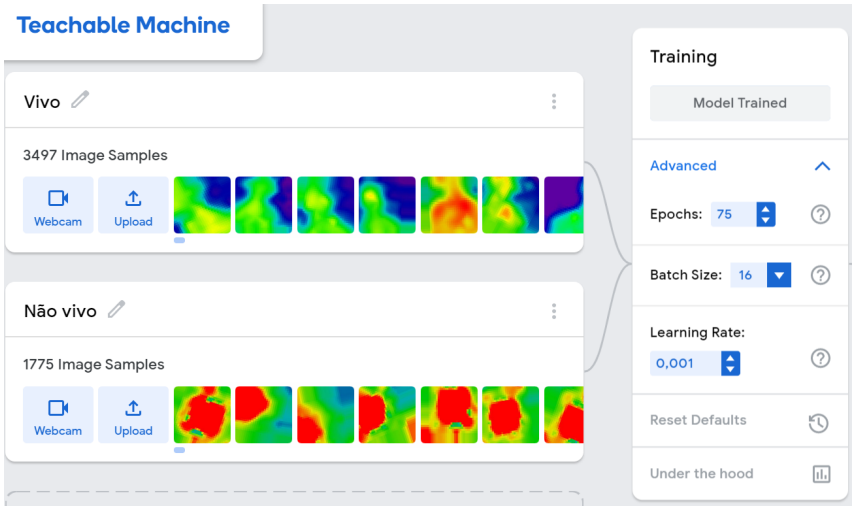

Figura 30 – Criação das classes e treinamento do modelo fonte: autor

Após inserir criar as classes no modelo, realizamos o treinamento, onde para se chegar num modelo com a maior acurácia, repetimos esse processo algumas vezes, alterando os valores de batch e epochs e analisando o comportamento do mesmo. Através dos valores obtidos de acurácia por classe, pudemos chegar aos valores ótimos para o modelo, que foram, epochs=75, e batch size=16, conforme figura 30. A taxa de aprendizado foi mantida em 0.001.

Como a API teachable machine disponibiliza a exportação do modelo gerado, assim o fizemos, optando pela exportação do mesmo, em formato .h5, para linguagem python, a mesma em que o sistema de classificação foi escrito. Sua implementação foi fluida devido ao uso das bibliotecas supracitadas. Analisando o *backend* do teachable machine, percebemos que o mesmo utiliza estrutura CNN MobileNet, tendo como padrão de entrada imagens em 224 x 224 RGB, mesmo padrão adotado na geração de nossas imagens térmicas.

Segundo (HOWARD et al., 2017) a estrutura MobileNet é construída em convoluções separáveis em profundidade, exceto pela primeira camada que é uma convolução completa. Através da rede podemos explorar facilmente diversas topologias até alcançar uma boa configuração. Na arquitetura MobileNet Todas as camadas são seguidas por um batch normalization e uma função de ativação ReLU, com exceção da camada final totalmente conectada que não possui linearidade e alimenta uma camada softmax para classificação.

Conforme abordado anteriormente, nossas imagens para modelagem e treinamento da CNN foram geradas no próprio sensor do robô cobra de forma experimental, onde capturamos imagens de cão, gato, pessoa, e objetos inanimados, a fim de criar duas grandes classes, vivos ou não vivos. Porém, para construir um modelo robusto e que necessitasse de poucos ajustes, além de treina-lo da forma devida, precisamos de classes de imagens com um volume maior, e com diversidade também. Para isso utilizamos *data augmentation*, conforme abordado na sessão 3.2.5.

O modelo preditivo e o métodos de classificação estão sendo utilizados na central de controle do robô, onde o modelo exportado pelo teachable machine é carregado na aplicação e utilizado através do tensorflow com keras. No método de classificação, carregamos o modelo, criando em seguida o *array* no tamanho padrão MobileNet $224 \times 224 \times 3$, comportando uma imagem por vez. Antes de classificar a imagem do objeto detectado, o sistema normaliza a imagem para em seguida inseri-la no *array* criado anteriormente e classificá-la em vivo ou não-vivo. Na figura 31 podemos ver o sistema captando a imagem térmica de um gato e classificando a mesma como vivo.

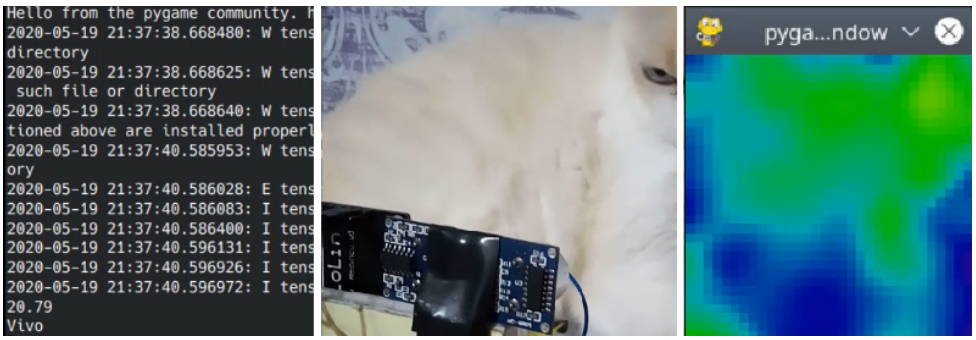

Figura 31 – Sistema classificando fonte: autor

4 Experimentos e Coleta de Dados

Para validar todo o sistema de movimentação do robô cobra, além de toda a parte de aquisição de sinais dos sensores, e classificação das imagens térmicas, foram realizados experimentos destinados a comprovar ou não as hipóteses sobre funcionamento do robô. Este capítulo apresenta todos os experimentos realizados nesta pesquisa, bem como a metodologia da experimentação, variáveis, hipóteses nulas e alternativas, além dos resultados e discussão. Na sessão 4.1, são realizados experimentos relacionados com o sistema de movimentação do robô cobra, como avançar, recuar, ou virar a direita ou esquerda. Na sessão 4.2, os experimentos tratam-se da validação do funcionamento devido dos sensores, além da classificação de imagens. As sessões 4.3 e 4.4, apresentam os resultados de cada experimento, além da discussão acerca dos mesmos.

4.1 Movimentação

Uma das principais funcionalidades do robô cobra é a sua movimentação, um trabalho conjunto do Arduino, que recebe comandos do módulo ESP8266 baseado na decisão da central de controle. Esses comandos são referentes a direção e sentido, se o robô avança, recua, vira a direita ou esquerda. Nesse contexto, os atuadores devem trabalhar de forma sincronizada, tal como descrito anteriormente no capítulo 3, seção 3.2.3 . Esse experimento é dividido em 3 partes que são, distância percorrida ao avançar, distância percorrida ao recuar, e virar a direita e esquerda, tendo como objetivo além de validar a movimentação do robô cobra, aferir empiricamente a distância percorrida ao avançar ou recuar, e a inclinação realizada ao se deslocar para a direita ou a esquerda.

4.1.1 Distância Percorrida ao Avançar

Nesse experimento tínhamos o objetivo de medir a distância percorrida pelo robô ao receber comandos únicos para avançar. Para tal, foi utilizada como variável independente a execução do método avançar, e como variável dependente a média das distâncias percorridas durante o experimento, como em (4.1). Na figura 32 podemos ver o momento de uma dos testes do comando avançar.

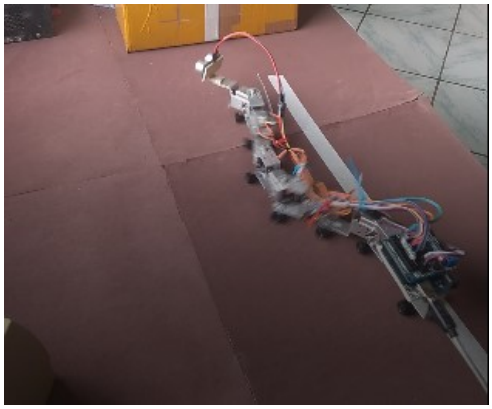

Figura 32 – Testando o comando avançar fonte: autor

As métricas do experimento foram de extrema importância para determinar a distância percorrida em cada avanço do robô, servindo inclusive como valor base para o desenvolvimento do Sistema de Navegação Primário, como descrito no capítulo 3, seção 3.2.3, subseção 3.2.3.1. As métricas foram as medidas em cm de cada passo S, figura 33, e a média dos S em cada tentativa. O espaço amostral de testes realizados nesse experimento foi de 20.

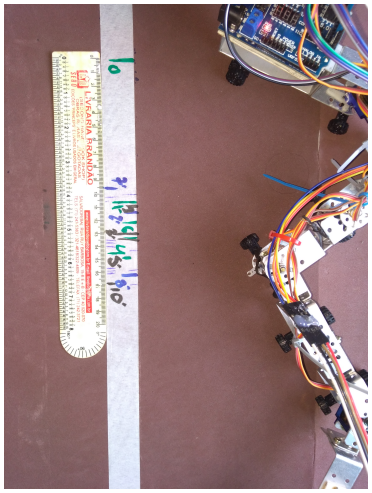

Figura 33 – Extraindo medidas fonte: autor

Os procedimentos realizados para esse experimento seguem abaixo:

- O descolamento foi realizado numa bancada apropriada para tal, forrada com EVA, facilitando o atrito das rodas com o piso.
- Na direção do deslocamento, foi colocada uma fita, onde havia um marco inicial, e em cada tentativa foi marcada a distância entre a ponta da cauda do robô em relação ao marco inicial
- Ao final de cada experimentação, a distância entre o marco inicial e o final era medida com uma régua e computada.

Nossa hipótese alternativa nesse experimento era de que o robô cobra ao avançar, ele se deslocasse numa média de distância entre 10 à 15cm (4.2), e como hipótese nula (4.3), que o mesmo não se deslocasse, gerando uma média de deslocamentos nula. Ao final dos procedimentos descritos, todos os dados foram tabulados para extrair a média, variância e desvio padrão acerca dos dados coletados.

$$\Delta S = \frac{(S1 + S2 + .. + Sn)}{n} \tag{4.1}$$

$$10cm <= \Delta S <= 15cm \tag{4.2}$$

$$\Delta S = 0 \tag{4.3}$$

4.1.2 Distância Percorrida ao recuar

Nesse experimento tínhamos o objetivo de medir a distância percorrida pelo robô ao receber comandos únicos para recuar. Para tal, foi utilizada como variável independente a execução do método recuar, e como variável dependente a média das distâncias percorridas durante o experimento, como em (4.4). Na figura 34 podemos ver o momento de uma dos testes do comando recuar.

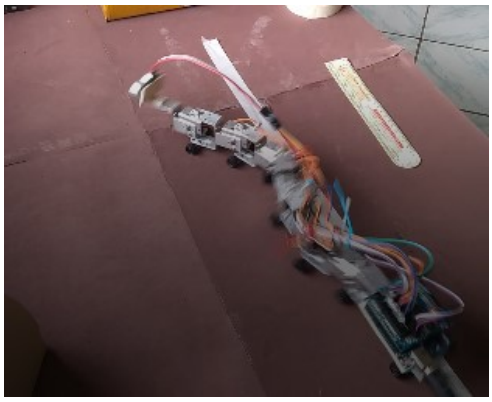

Figura 34 – Testando o comando recuar fonte: autor

As métricas desse experimento foram fundamentais ao se determinar a distância percorrida em cada recuo do robô, servindo inclusive como valor base para o desenvolvimento do Sistema de Navegação Primário, tal como o experimento anterior. As métricas foram as medidas em cm de cada passo D, figura 35, e a média dos D em cada tentativa. O espaço amostral de testes realizados nesse experimento foi de 20.

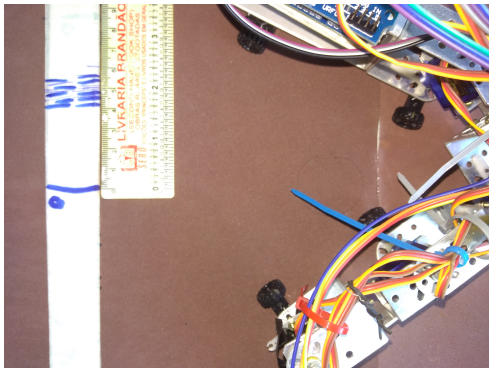

Figura 35 – Extraindo medidas do recuo fonte: autor

Os procedimentos realizados para esse experimento seguem abaixo:

- O descolamento foi realizado numa bancada apropriada para tal, forrada com EVA, facilitando o atrito das rodas com o piso.
- Na direção do deslocamento, foi colocada uma fita, onde havia um marco inicial, e em cada tentativa foi marcada a distância entre a ponta da cauda do robô em relação ao marco inicial
- Ao final de cada experimentação, a distância entre o marco inicial e o final era medida com uma régua e computada.

Nossa hipótese alternativa nesse experimento era de que o robô cobra ao recuar, ele se deslocasse numa média de distância entre 10 à 15cm (4.5), tal como no avanço, e como hipótese nula (4.6), que o mesmo não se deslocasse, gerando uma média de deslocamentos nula. Ao final dos procedimentos descritos, todos os dados foram tabulados para extrair a média, variância e desvio padrão acerca dos dados coletados.

$$\Delta D = \frac{(D1 + D2 + .. + Dn)}{n} \tag{4.4}$$

$$10cm <= \Delta D <= 15cm \tag{4.5}$$

$$\Delta D = 0 \tag{4.6}$$

4.1.3 Virar a Direita e Esquerda

O experimento proposto nessa etapa tinha o objetivo de medir a inclinação de deslocamento do robô ao receber instruções únicas para virar a direita, ou para virar a esquerda. Para tal, foram utilizadas como variáveis independentes as execuções dos métodos virar a

direita, ou virar a esquerda, e como variáveis dependentes, as médias das inclinações realizadas pelo robô durante o experimento, como em (4.7). Nas figuras 36 e 37 podemos ver o momento em que são testados os movimentos de virar a direita e equerda respectivamente.

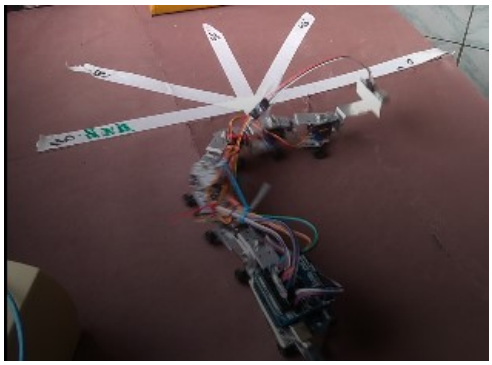

Figura 36 – Testando o comando Virar a direita fonte: autor

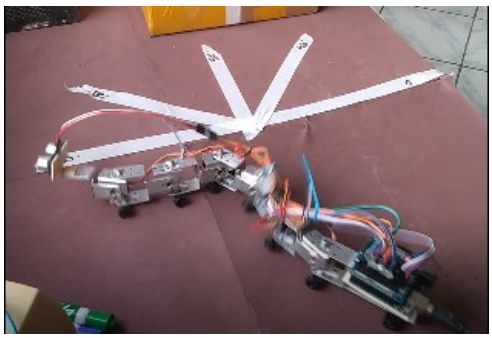

Figura 37 – Testando o comando Virar a esquerda fonte: autor

As métricas desse experimento foram fundamentais ao se determinar a inclinação realizada em cada deslocamento a direita, e a esquerda do robô, servindo como valor base para o desenvolvimento do Sistema de Navegação Primário, tal como o experimento anterior. As métricas foram as medidas em graus º de cada inclinação a, figura 38 e 39, e a média das tentativas. O espaço amostral de testes realizados nesse experimento foi de 20, para cada sentido preterido.

Figura 38 – Extraindo inclinação a direita fonte: autor

Figura 39 – Extraindo inclinação a esquerda fonte: autor

Os procedimentos realizados para esse experimento seguem abaixo:

- O deslocamento foi realizado numa bancada apropriada para tal, forrada com EVA, facilitando o atrito das rodas com o piso.
- Na bancada foi fixado um conjunto de fitas, com os ângulos de 0º, 45º, 90º, 135º, 180º, a frente de onde o robô se deslocaria, tal como um transferidor.
- Seguir em frente significa manter-se em 90º, a direita inclinar-se em ângulos abaixo de 90º, e a esquerda inclinar-se em ângulos acima de 90º.
- Em cada tentativa foi marcado nas fitas a inclinação realizada.
- Ao final de cada experimentação, a inclinação era medida e computada.

Nossa hipótese alternativa nesse experimento era de que o robô cobra ao virar a direita, ele se inclinasse numa média entre 0º à 45º (4.8), e ao se deslocar a esquerda, se inclinasse

numa média entre 135º e 180º (4.9). Como hipótese nula (4.10), que o mesmo seguisse em frente para os dois casos, gerando uma média de inclinação de 90º e ambos sentidos preteridos. Ao final dos procedimentos descritos, todos os dados foram tabulados para extrair a média, variância e desvio padrão acerca dos dados coletados.

$$\Delta\alpha = \frac{(\alpha 1 + \alpha 2 + .. + \alpha n)}{n} \tag{4.7}$$

$$0^o <= \Delta\alpha dir < 45^o \tag{4.8}$$

$$135^o <= \Delta\alpha esq < 180^o \tag{4.9}$$

$$\Delta\alpha = 90^o \tag{4.10}$$

4.2 Os sensores e a Classificação

4.2.1 Identificação de Obstáculos num raio de 180 º

O experimento proposto nessa sessão tinha o objetivo de atestar o funcionamento do comando "verificar", responsável por realizar o giro em 180º da cabeça do robô cobra, a fim de detectar objetos a serem classificados antes de se locomover. Para tal, foram utilizadas como variáveis independentes a execução do método "verificar"nas distâncias de 10, 20 e 30 cm, entre a cabeça do robô e o objeto, conforme figuras40, 41 e 42. Como variáveis dependentes, as médias das distâncias detectadas pelo sensor ultrassônico do robô nos ângulos 0, 45, 90, 135, e 180º, durante o experimento.

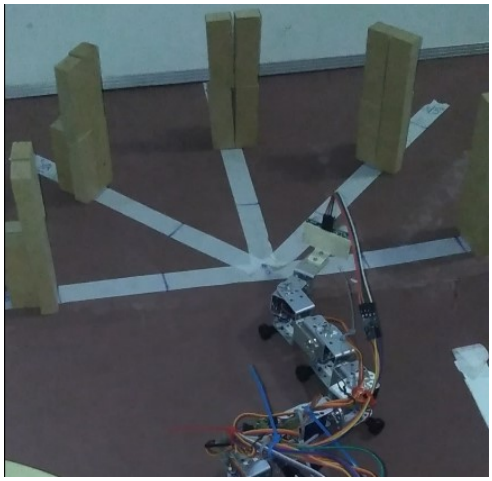

Figura 40 – Testando o comando verificar fonte: autor

As métricas desse experimento serviram como base para calibração para o método de movimentação e para calibração do Sistema de Navegação Primário. As métricas foram as

medidas em cm de cada distância detectada pelo sensor para cada ângulo específico, além da média das distâncias detectadas nas tentativas. O espaço amostral de testes realizados nesse experimento foi de 20, para cada distância e ângulo, conforme supracitado.

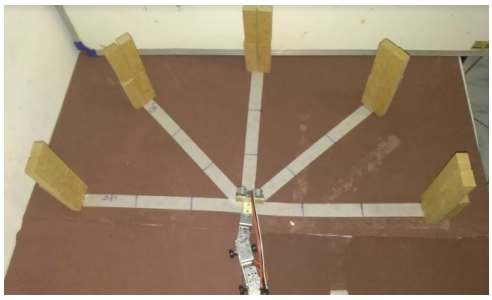

Figura 41 – Bancada preparada para o experimento fonte: autor

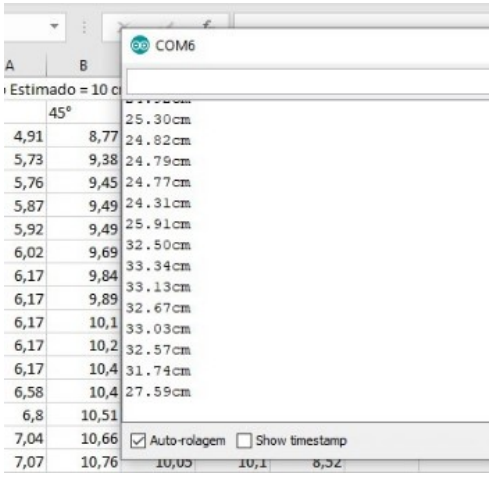

Figura 42 – Extraindo dados do experimento fonte: autor

Os procedimentos realizados para esse experimento seguem abaixo:

- O experimento foi realizado numa bancada apropriada para tal, forrada com EVA.
- Na bancada foi fixado um conjunto de fitas, com os ângulos de 0º, 45º, 90º, 135º, 180º, a frente da cabeça do robô, tal como um transferidor.
- Nas fitas foram marcadas distâncias a partir do ponto central, onde a cabeça do robô se localizava, com as distâncias de 10, 20, e 30 cm para cada inclinação.
- Os objetos para detecção foram colocados nessas três distâncias em cada experimentação.
- Verificar significa realizar um giro de 180º da cabeça do robô, partindo de 0º e girando até chegar em 180º.
- Em cada tentativa foram printados na porta serial as distâncias detectadas, para cada ângulo predefinido.

- Ao final de cada experimentação os dados eram computados e tabulados por distância e por ângulo.

Nossa hipótese alternativa (4.12) nesse experimento era de que o robô cobra ao verificar, ele conseguiria detectar os objetos num campo de 180º, com uma variação de 2 cm da média das distâncias(4.11), para mais ou para menos, para cada ângulo supracitado em relação a medida real. Como hipótese nula (4.13), que a detecção para cada ângulo fosse igual a medida real. Ao final dos procedimentos descritos, todos os dados foram tabulados para extrair a média, variância e desvio padrão acerca dos dados coletados.

$$\Delta d = \frac{(d1 + d2 + .. + dn)}{n} \tag{4.11}$$

$$\Delta d - 2cm <= DistReal <= \Delta d + 2cm \tag{4.12}$$

$$\Delta d = DistReal \tag{4.13}$$

4.2.2 Experimentado a Detecção e Classificação em Vivo e Não-Vivo

Detectar e classificar objetos como vivo ou não-vivo é uma das premissas principais desse projeto. Sendo assim, esse experimento tinha o objetivo de verificar o percentual de erros e acertos na detecção e classificação dos seres vivos, validando o sistema de classificação de forma empírica, conforme figuras 43 e 44. Para tal, foram utilizadas como variáveis independentes a execução do método de detecção e classificação nas distâncias de 20, 50, 100 e 150 cm, entre os sensores da cabeça do robô e o objeto. Como variáveis dependentes, as médias das distâncias detectadas pelo sensor ultrassônico, e a taxa de acerto médio decorrente da classificação em vivo e não-vivo dos objetos detectados.

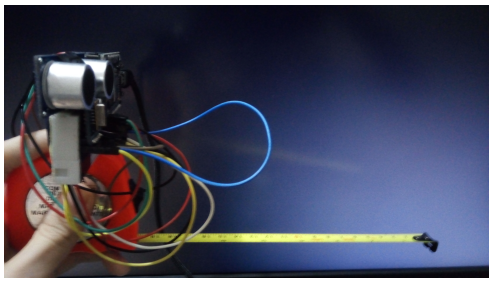

Figura 43 – Dispositivo de testes fonte: autor

As métricas desse experimento foram as medidas em cm de cada distância detectada pelo sensor para tentativa, além da classificação em vivo ou não-vivo. O espaço amostral de testes realizados nesse experimento foi de 10 para cada distância supracitada com objetos diversos.

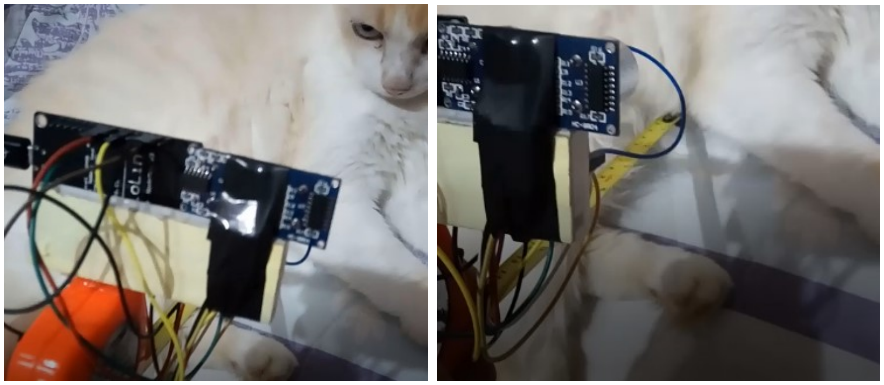

Figura 44 – Detectando um Gato fonte: autor

Os procedimentos realizados para esse experimento seguem abaixo:

- O experimento foi realizado utilizando-se um dispositivo composto pelos sensores presentes na cabeça do robô, mais uma trena fixada junto aos sensores.
- Cada experimentação era realizada encostando a ponta da trena, no objeto a ser classificado, na distância experimentada prevista.
- As distâncias estipuladas na experimentação foram de 20, 50, 100, e 150cm, fixadas na trena em cada teste realizado.
- Os objetos de testes escolhidos para a experimentação foram: ventilador, vidro, pessoa, monitor, celular, geladeira, fogo, perna de pessoa, cabeça de pessoa, torso de pessoa, almofada, parede, e gato.
- Em cada tentativa foram printados no terminal as distâncias detectadas, e a classificação dos objetos.
- Ao final de cada experimentação os dados eram computados e tabulados.

Nossa hipótese alternativa (4.15) nesse experimento era de que o robô cobra ao detectar e classificar, ele conseguiria classificar nas distâncias supracitadas os objetos, tendo a acurácia média ($\Delta\alpha\%$) (4.14) de 97,0% de acurácia, uma variação de 1,5 em relação a taxa de acerto calculada pelo modelo preditivo. Como hipótese nula (4.16), que a classificação para as distâncias supracitadas tivesse valor médio de acurácia igual ao previsto no modelo preditivo. Ao final dos procedimentos descritos, todos os dados foram tabulados para extrair a média, variância e desvio padrão acerca dos dados coletados.

$$\Delta\alpha\% = \frac{(\Delta\alpha 1\% + \Delta\alpha 2\% + .. + \Delta\alpha n\%)}{n} \tag{4.14}$$

$$\Delta\%\alpha - 1,5 <= \alpha Model <= \Delta\%\alpha + 1,5 \tag{4.15}$$

$$\Delta\%\alpha = \alpha Model \tag{4.16}$$

4.3 Resultados e Discussão

4.3.1 Resultados de Movimentação

4.3.1.1 Distância ao Avançar

Após a realização do experimento para aferir a distância percorrida ao avançar, foram coletados dados referentes a 20 distâncias distintas dentro da amostra. A amostra foi disposta no gráfico abaixo, onde o eixo horizontal equivale a experimentação, e o vertical a distância percorrida em cada uma.

Figura 45 – Gráfico da amostra do Experimento fonte: autor

As distâncias variam entre 9.5 cm de distância percorrida à 16,5, tendo em como elemento com maior ocorrência no histograma gerado, figura 46, o intervalo (12.4, 15.3] em cm. O desvio padrão gerado pela amostra foi de 2,27, com uma variância de 5,14.

Figura 46 – Histograma do Experimento fonte: autor

De acordo com as premissas do experimento, á média das distâncias percorridas foi de Delta S = 13,9 cm, ficando dentro do intervalo proposto na hipótese alternativa para esse experimento. Sendo assim, foi confirmada a hipótese, e determinada para o nosso sistema de navegação primário, que a cada comando de avançar, o robô cobra se desloca 13,9 cm.

4.3.1.2 Distância ao Recuar

Após a realização do experimento foram coletados dados referentes a 20 distâncias distintas dentro da amostra, onde a amostra foi disposta no gráfico abaixo. Nesse gráfico o eixo horizontal equivale a experimentação, e o vertical a distância percorrida em cada teste.

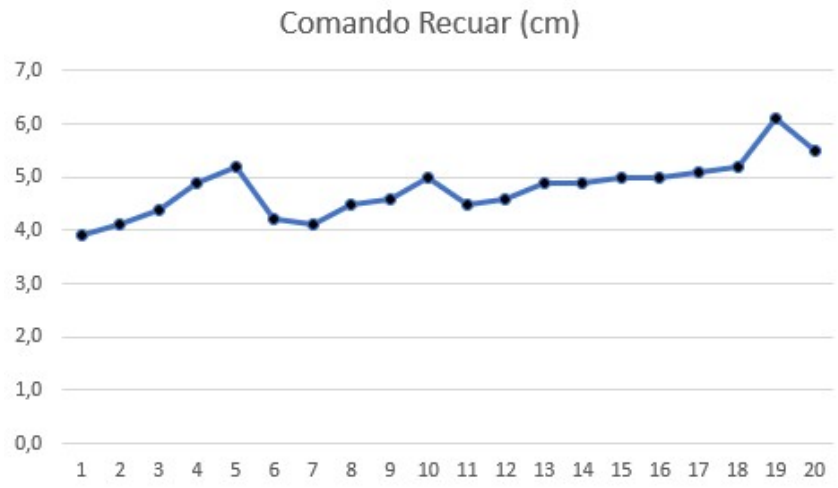

Figura 47 – Gráfico da amostra do Experimento fonte: autor

As distâncias variam entre 3,9 cm de distância percorrida à 6,1 cm, tendo em como elemento com maior ocorrência no histograma gerado, figura 48, o intervalo (4,6, 5,3] em cm. O desvio padrão gerado pela amostra foi de 0,53, com uma variância de 0,28. Isso evidencia uma maior regularidade de valores na distância de recuo em relação aos valores encontrados no experimento do comando avançar.

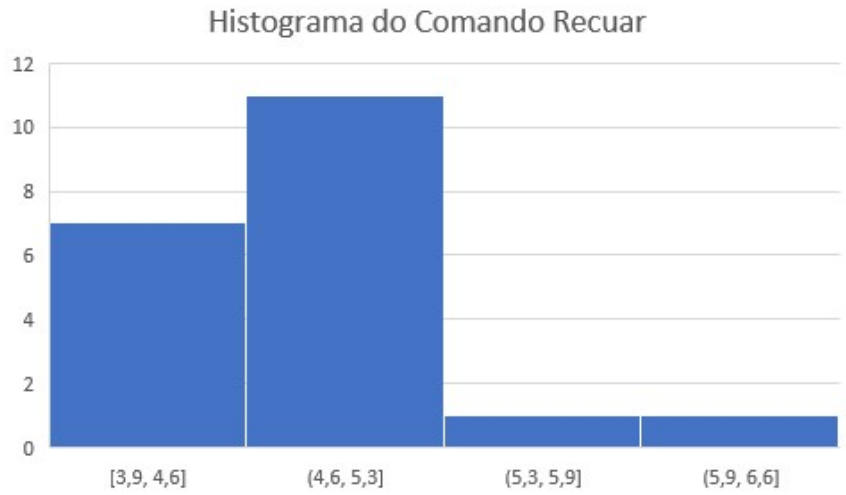

Figura 48 – Histograma do Experimento fonte: autor

Nesse experimento, á média das distâncias percorridas foi de Delta S = 4,9 cm, ficando dentro do intervalo proposto na hipótese alternativa para o mesmo, confirmando assim a hipótese. Como o valor médio para recuar é de 4,9 cm, cerca de 20% da medida de um avanço, cada comando de recuo será repetido três vezes no sistema, e a distância adotada para o sistema de navegação primário ficará em 14,7cm para cada recuo.

4.3.1.3 Direita e Equerda

Nesse experimento foram coletados dados referentes a ângulos para cara comando, direita ou esquerda, gerando uma amostra de tamanho 20 para cada comando supracitado. No experimento de virar a esquerda, todas as ocorrências da amostra foram de 180º, gerando uma média de 180º e com desvio padrão e variância, ambos iguais a zero, conforme figura 49. Evidenciando que o movimento de virar a esquerda está calibrado de forma a virar o robô em 180º da sua direção inicial, confirmando assim nossa hipótese alternativa para esse experimento, de que a inclinação ficaria entre 135º e 180º. Para o sistema de navegação primário, adotamos o comando de virar a direita como um giro a direita, mais um comando de avançar, associando os dois comandos.

Para comando de virar a direita, tivemos um *outline*, onde um dos valores coletados foi de 45º, divergindo dos demais dezenove valores, que foram 0º. Devido a esse único valor, desvio padrão cresceu significativamente para 10,06 em relação aos demais experimentos, e a variância foi de 101,25. Para efeitos de análise estatística, esse valor não foi considerado um valor válido, porém ainda assim confirma nossa hipótese alternativa, que estimava inclinações entre 0º e 45º ao virar a direita.

Figura 49 – Gráfico de amostra virar a direita fonte: autor

Para uso no sistema de navegação primário, utilizamos o comando virar a esquerda com uma inclinação de 0º em relação a sua direção em linha reta, 90º, mais um comando de avançar. O *outline* presente na amostra, foi resultado de uma deformidade no emborrachado da bancada, figura 50, que desviou o curso do robô durante o experimento, como nosso robô se encontra no escopo de 2D, tal dado foi descartado nessa análise.

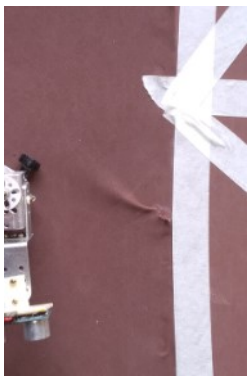

Figura 50 – Deformidade no piso fonte: autor

4.4 Sobre os Sensores e Classificação

4.4.1 Identificação de Obstáculos pelo sensor

Neste experimento foram coletados dados referentes a 20 valores detectados nos raios de 10, 20 e 30 cm, para os ângulos de 0, 45, 90, 135 e 180º dentro da amostra. As amostras foram dispostas nos gráficos abaixo, figuras 51, 52 e 53. Nesses gráficos o eixo vertical equivale a distância detectada em cada teste e as cores das linhas aos respectivos ângulos.

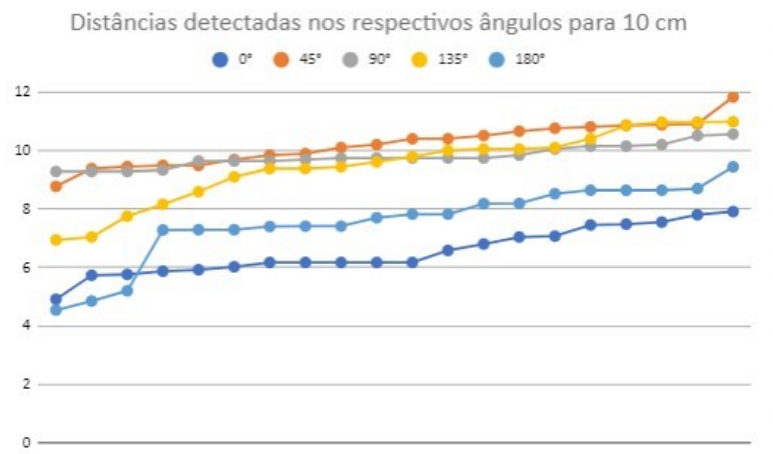

Figura 51 – Gráfico da amostra do Experimento em 10cm fonte: autor

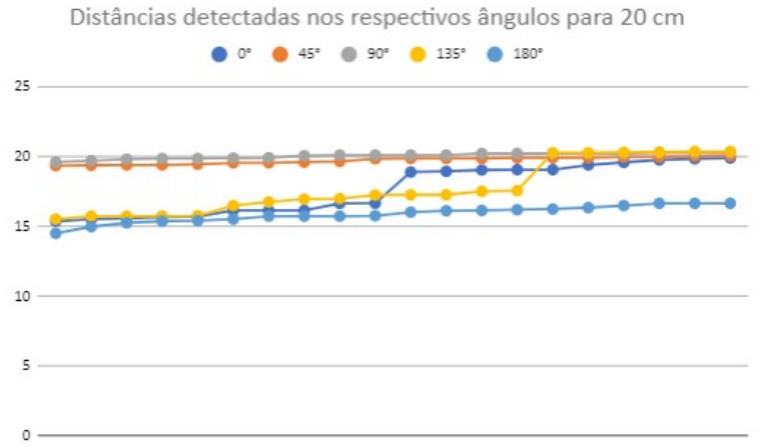

Figura 52 – Gráfico da amostra do Experimento em 20cm fonte: autor

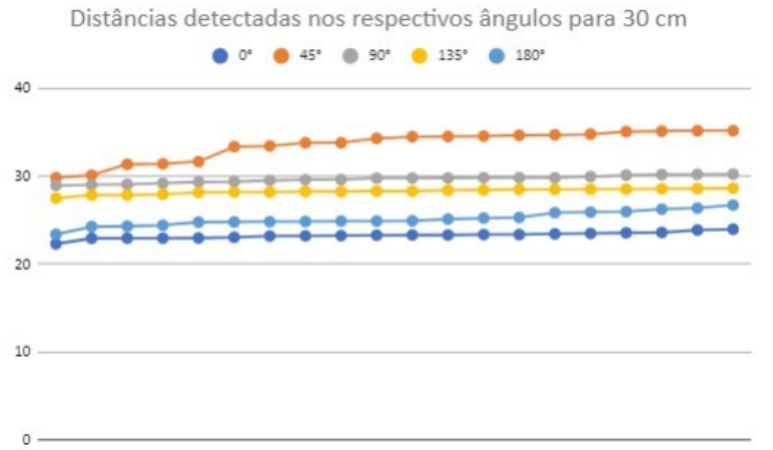

Figura 53 – Gráfico da amostra do Experimento em 30 cm fonte: autor

Os valores detectados para os ângulos supracitados, para um raio de 10cm, ficaram numa média de 9,7 cm, com um desvio padrão de 1,70 e uma variância de 2,91. Para a distância de 20 cm, os valores detectados ficaram em torno de 17,75, tendo um desvio padrão de 1,92 e uma variância de 3,7. No raio de 30 cm, a média dos valores detectados fica em 28,29 cm, com um desvio padrão de 3,7, e uma variância de 13,76.

Nesse experimento, para o raio de 10cm, o $\Delta d = 9,7$ confirma nossa hipótese alternativa. Para o raio de 20 cm, o $\Delta d = 17,75$ se aproxima do valor esperado para confirmar a hipótese alternativa, mas ficando 0,15 abaixo do esperado. No raio de 30 cm, o $\Delta d = 28,29$ confirma a hipótese alternativa, mesmo tendo um aumento significativo do desvio padrão e da variância. Pode-se concluir inclusive a partir desses resultados, figura 54, que o sensor ultrassônico diminuiu demais a precisão das aferições com o aumento da distância até o objeto detectado, pois nas especificações do HC-SR40 estimava uma distância máxima de 4 metros, com precisão de 3 mm. No gráfico abaixo podemos verificar o aumento do desvio padrão em relação à distância de detecção do sensor.

Figura 54 – Desvio padrão por distância fonte: autor

4.4.2 Classificação de Imagens

Após a realização do experimento foram coletados dados referentes a 10 amostras de distâncias e classificações distintas, sendo as distâncias padrão das amostras, 20, 50, 100 e 150cm, e as classificações em vivo e não-vivo. Foi possível obter 100% de acurácia na classificação, nas distâncias 30, 50 e 100cm, e 90% de acurácia na distância de 150cm, figura 54.

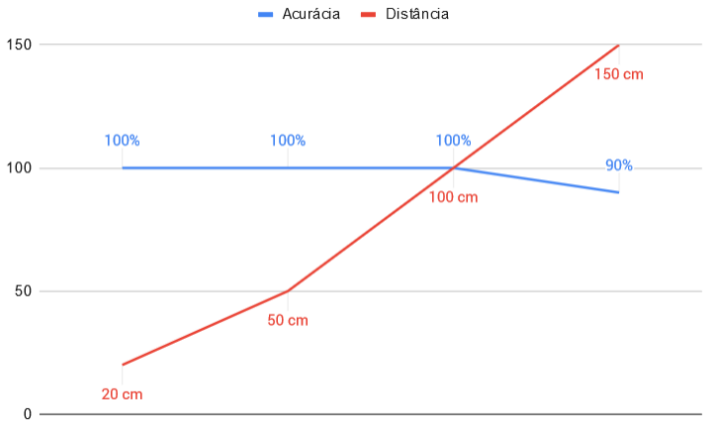

Figura 55 – Acurácia por distância fonte: autor

Nesse experimento, a acurácia média foi de $\Delta\% = 97,5\%$, confirmando a hipótese alternativa com relação a eficiência do sistema de classificação quando submetido a experimento em ambiente real. O desvio padrão ficou na média de 0,0433, tendo uma variância de 0,00187. Esse valor médio da acurácia foi calculado a partir da acurácia média para as distâncias de detecção padrão adotadas no experimento.

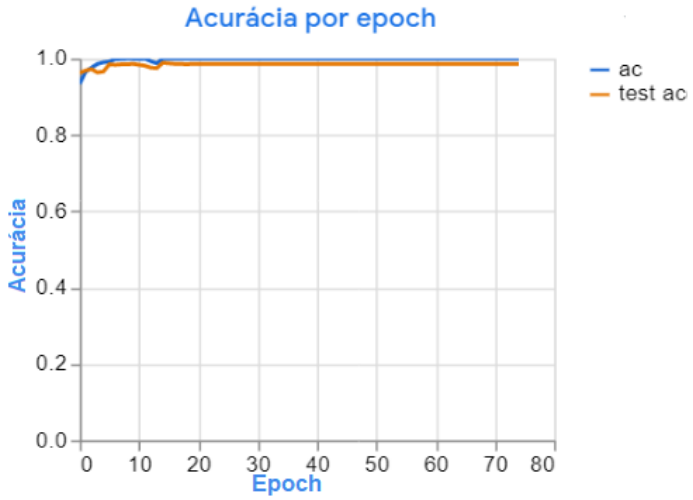

Figura 56 – Acurácia por epochs fonte: autor

O teachable machine disponibiliza após o treinamento do modelo preditivo o resultado das taxas de acurácia por classe, acurácia por época, figura 56, e perda por época, figura 57. Nesses gráficos, fica evidente também a precisão do modelo preditivo ao classificar em vivo e não-vivo, onde para um batch size de 16, com um epochs de 75, conseguimos a acurácia por classe de 100% para vivos, e 97% para não-vivos, conforme os gráficos de acurácia (figura 57) e perda por época (figura 58).

Figura 57 – Loss por epochs fonte: autor

5 Conclusões

O crescimento populacional desordenado, aliado a uma imensa desigualdade social contribui significativamente para a construção em locais inapropriados como área de encostas, suscetíveis à desabamentos, além dos fatores climáticos e os relativos a desastres naturais. Nesse contexto, ocorrendo os desabamentos, e ou deslizamentos, uma boa estratégia no resgate se faz necessária. Ao realizar uma operação de busca e salvamento em estruturas colapsadas, a equipe de resgate deve estar preparada para fazê-lo num tempo resposta menor possível, tendo como objetivo primordial salvar vidas.

Apropriando-se de alguns conceitos de robótica, e inteligência artificial, esse trabalho propôs uma solução para otimizar o processo de resgate, um robô cobra, para identificar vítimas dentro de escombros. O protótipo apresenta funcionalidades necessárias a identificação de vítimas em escombros, tal como o uso de *machine learning*, a classificação das imagens térmicas, além da capacidade de se locomover nas estruturas colapsadas de forma autônoma enviando coordenadas durante o trajeto.

Para tal, foram geradas durante a construção da solução, uma base de dados com imagens térmicas, advindas de um sensor térmico localizado na cabeça do robô cobra, o qual foi construído conforme supracitado. No *dataset*, foi usado *data augmentation* para acrescê-lo em tamanho e diversidade, separando parte das imagens para criação do modelo preditivo, e parte para treinamento do modelo, servindo como base para estrutura de decisão da central de controle, que junto com os dados advindos do sensor ultrassônico, enviava comandos ao robô para que o mesmo se movimentasse de forma autônoma.

Após a construção do robô, bem como toda sua parte sistêmica, foram realizados experimentos, a fim de validar o funcionamento adequado da solução, conforme proposto nos objetivos desse trabalho. Sobre o sistema de movimentação, pôde-se atestar seu bom funcionamento, testando de forma simulada os métodos associados, tal como avançar, com distância média de 13,9 cm, recuar, com distância média de 4,9 cm, direita e esquerda, contornando em 180º e 0º respectivamente, e verificar, com 93,3% de precisão nas distâncias detectadas em um arco de 180º durante a varredura. Quanto aos sensores, também funcionaram adequadamente em situação simulada, conseguindo detectar os objetos de forma eficiente, em várias distâncias experimentadas, com uma acurácia média de 97,5% com relação a classificação entre vivo e não-vivo.

Podemos concluir então, que esse trabalho conseguiu cumprir seus objetivos, conforme especificado previamente, de forma eficaz, ampliando a gama de soluções pré-existentes na área de resgate em estrutura colapsadas, propondo um protótipo de robô cobra, autônomo, que identifica organismos vivos em escombros.

5.1 Trabalhos futuros

Como trabalhos futuros, com relação ao sistema de movimentação, pode-se realizar um aprimoramento primeiramente à dimensão da solução, trazendo o escopo para 3D, aprimorando inicialmente a estrutura mecânica, ampliando o circuito de controle, além da evolução na complexidade dos métodos de movimentação, possibilitando assim que o robô realize movimentos como subir escadas, ou esgueirar-se por um tronco de árvore por exemplo. Esse aprimoramento traria a solução para um ambiente mais próximo de uma situação real.

Outro aprimoramento que é vislumbrado para um trabalho futuro está diretamente relacionado ao sistema de identificação/classificação, onde através da adição de uma câmera com visão noturna, e do uso de uma sensor térmico de maior resolução, possibilitar ao sistema usar um *dataset* mais robusto. Tais adições de *hardware* no projeto possibilitariam fazer uso de alguns *datasets* de alta resolução disponíveis na internet, além do uso de um modelo CNN mais robusto, permitindo inclusive classes mais específicas como perna, cabeça, ou braço por exemplo.

Por último, o uso da solução proposta utilizando uma tecnologia de rede de alcance amplo, mas de baixa potência, com triangulação de sinal para controle, otimizaria todo o sistema de comunicação, amenizando as interferências das estruturas dos escombros, além de reduzir o consumo de amperagem durante a comunicação sem fio entre o robô e a central de controle. Tais melhorias seriam significativas e possibilitariam trabalhos de pesquisa com resultados pertinentes a diversas áreas, ampliando a fronteira da solução além do resgate e salvamento.

Referências

ARSCONSULT. Apostila de introdução a robótica. *Recife*, v. 12, n. 1, 1995. 15

BAIG, I.; CHITAY, M.; SALAHUDDIN, D. Home automation using arduino wifi module esp8266. *A PROJECT REPORT*, 2016. Disponível em: https://core.ac.uk/download/pdf/55305294.pdf. Acesso em: 11 maio 2020. 18

BALTAZAR, P.; GOKHALE, J.; BANSOD, U. Infrared thermography and ir camera. *International Journal of Research In Science & Engineering*, 2011. Disponível em: https://pdfs.semanticscholar.org/2a24/b0240376c3c5a1dfd213284ff8fdaf647674.pdf. Acesso em: 30 maio 2019. 19

BRAGA, N. C. Como funcionam os sensores ultrassônicos. 2019. Disponível em: https://www.newtoncbraga.com.br/index.php/como-funciona/5273-art691. Acesso em: 04 jun. 2019. 19

COSTA-FILHO, S. V. S. d. et al. Configuração de algoritmos de aprendizado de máquina na modelagem florestal: um estudo de caso na modelagem da relação hipsométrica. *CiÃFlorestal*, scielo, v. 29, p. 1501 – 1515, 12 2019. ISSN 1980-5098. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1980-50982019000401501&nrm=iso>. 21, 23

CRAWFORD, M. *Snake Robots Crawl to the Rescue*. first. [S.l.]: ASME, 2018. ([S. l.], p. 1-1). Disponível em: https://www.asme.org/engineering-topics/articles/robotics/snake-robots-crawl-rescue-part-1. Acesso em: 23 maio 2019. 14

CUNHA, L. V. Desenho técnico. *Fundação Calouste Gulbenkian*, v. 14ª Ed, 2008. 20

DATASCIENCEACADEMY.COM.BR. 2020. Disponível em http://datascienceacademy.com.br/blog/o-que-e-o-tensorflow-machine-intelligence-platform/, acesso em 22/05/2020. 27

DIAS, C. G.; LIBRANTZ, A. A. F. H.; SANTOS, F. A. C. R. d. Modelagem e simulação de um sistema inteligente para controle de dosagem da pós-cloração em estações de tratamento de água. *Eng. Sanit. Ambient., Rio de Janeiro*, 2020. 8, 22

ELECFREAKS. Ultrasonic ranging module hc-sr04. 2010. Disponível em: http://users.ece.utexas.edu/ valvano/Datasheets/HCSR04b.pdf. Acesso em: 04 jun. 2019. 19

EPUSP. Controle de um servo motor. 2014. Disponível em: https://www2.pcs.usp.br/ labdig/pdffiles$_2$014/$controle - servo - semestral.pdf.Acessoem$: 18mai.2019. 19

EVANS, M.; NOBLE, J.; HOCHEBAUM, J. Aduino em ação. *São Paulo, Novatec*, 2013. 17

FERREIRA, J.; GORDO, N. Elementos de máquinas. *SENAI et. al. Elementos de Máquinas*, v. 1, 2018. 20

FERREIRA, M. et al. Redes neurais convolucionais com tensorflow: Teoria e prática. *Escola Regional de Informática do Piaui. Livro Anais - Artigos e Minicursos*, v. 1, n. 1, p. p. 382–406, 2017. 8, 23, 24

FERWORN, A. Dog and snake marsupial cooperation for urban search and rescue deployment. *Biorobotics Papers, [S. l.]*, p. p.1–5, nov 2012. 29

FLORES, P.; GOMES, J. Cinemática e dinâmica de engrenagens. *Aspectos gerais sobre engrenagens*, v. 1, p. p.1–45, 2014. 20

FONSECA, J. Redes de petri de alto nível e pnrd invertida associadas ao controle de robôs móveis:. *Uma Abordagem para Operações de Busca e Salvamento em Trilhas e Travessias*, p. p.1, 2018. Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia. 13

HIROSE, U. Y. Kinematic control of active cord mechanism with tactile sensors. *In 2nd International CISM-IFTMM Symposium on Theory and Practice of Robots and Manipulators*, p. 241, 1976. 16, 27, 37

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. Disponível em: <http://arxiv.org/abs/1704.04861>. 43

KERAS.IO. Keras: The python deep learning library: Keras. *Documentation, [S. l.]*, p. p.1, 2019. Disponível em: https://keras.io/. Acesso em: 1 abr. 2020. 27

LAB, G. C. Teachable machine. *Experiments with google*, 2019. Disponível em: https://experiments.withgoogle.com/teachable-machine. Acesso em: 23 maio 2020. 42

LUGGER, G. F. *Inteligência Artificial*. first. [S.l.]: Pearson, 2013. v. 1. 14, 21

MAESTRE, U. G. *Effective Techniques of the Use of Data Augmentation in Classification Tasks*. 2018. 35, 41

MCROBERTS, M. Arduino básico. *1 Ed. São Paulo*, Novatec, 2011. 17

MEDEIROS, R. L. P. Desenvolvimento e aplicação de motores de corrente contínuas virtuais aplicadas nas aulas laboratoriais de controle de sistemas,. *Universidade Federal do Pará*, 2011. Faculdade de Engenharia Elétrica, Belém. 19

MILLER, D. Overview. adafruit amg8833 8x8 thermal camera sensor. *Cambridge/London: MIT Press Cambridge, MA, USA*, Snake Robots for Search and Rescue, p. p. 1–2, jun 2017. Disponível em: https://learn.adafruit.com/adafruit-amg8833-8x8-thermal-camera-sensor/overview. Acesso em: 30 maio 2019. 8, 20

MILLER, G. Neurotechnology for biomimetic robots. *Cambridge/London: MIT Press Cambridge, MA, USA*, Snake Robots for Search and Rescue, p. pp. 271–284, 2002. 8, 27, 28, 30, 31

MITCHELL, T. M. Machine learning. *McGraw-Hill Science/Engineering/Math*, p. 432 pages, mar 1997. ISBN.0070428077. 21

MIYAZAKI, C. K. Redes neurais convolucionais para aprendizagem e reconhecimento de objetos 3d. *USP*, p. p.1–79, 2017. 24, 25

NG, A. Y.-T. Convolutional neural networks. *Programa de cursos integrados Aprendizagem profunda - Coursera*, jun 2018. Disponível em: https://www.coursera.org/learn/convolutional-neural-networkssyllabus, Acesso em: 23 mai. 2020. 24, 25, 26

NIN, M.; OSóRIO, F. Navegação de robôs móveis autônomos e detecção de humanos baseada em sensor laser e câmera térmica. *MNR Mostra Nacional de Robótica*, 2011. Disponível em: http://osorio.wait4.org/publications/2011/MNR-Artigo-Matheus(BIC-TCC).pdf. Acesso em: 30 maio 2019. 8, 28, 29, 30

NUNES, T. AplicaÇÃo da tecnologia atravÉs de drones no corpo de bombeiros militar de santa catarina. *Santa Catarina: Universidade Federal de Santa Catarina*, 2017. 13

ROCHA, G. Riscos ambientais: Análise e mapeamento em minas gerais. *Ed.UFJF. Juiz de Fora*, p. 126 p, 2005. 13

SHALEV-SHWATZ, S.; BEN-DAVID, S. Entendendo machine learning da teoria aos algorítmos. *Cambridge UK: Cambridge University Press*, 2014. 21

SHMAKOV, O. Snakelike robots locomotions control. mechatronics - foundations and applications. *[S. l.], p. 1-26*, 2006. 15, 16

SHORTEN, C.; KHOSHGOFTAAR, T. A survey on image data augmentation for deep learning. *Journal of Big Data*, v. 6, 12 2019. 41

SILVA, l.; FREITAS, T. AnÁlise de tecnologias da iot para uso em logÍstica humanitÁria e busca e salvamento de pessoas uma revisÃo da literatura recente. *São Paulo SP, p. 1-4, 5*, p. p.1, 2018. 13

SILVA, R. Estruturas colapsadas: O emprego de pessoal e material de engenharia em operaÇÕes de busca e resgate. *Revista Agulhas Negras, Resende*, 2018. 13

SILVA, R. M.; LEAL, M.; LIMA, F. Predico do câncer de mama com aplicação de modelos de inteligência computacional. *TEMA (São Carlos), São Carlos*, v.20, n. n.2, p. p. 229–240, aug 2019. 22

SIMON, H. A. The sciences of the artificial. *Cambridge, MA; MIT Press*, v. 2. ed, 1983. 1981, Why should machines learn? In: MICHALSKY, R. S.; CARBONELL, J. G.; MITCHELL, T. M.(Eds.) Machine Learning: An Artificial Intelligence Approach, v. 1. Palo Alto, CA; Tioga,. 21

SIMONOVSKY, M.; KOMODAKIS, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017. Disponível em: <http://arxiv.org/abs/1704.02901>. 23

SKONIECZNY, K.; D'ELEUTERIO, G. M. T. Modeling friction for a snake-like robot. *Advanced Robotics 22*, 2009. 16

SOUZA, E. et al. Aplicações do deep learning para diagnóstico de doenças e identificação de insetos vetores,. *Saúde debate, Rio de Janeiro*, v. 43, n. n.2, p. p. 229–240, feb 2020. Disponível em: http://www.scielo.br/scielo.php?script=sci$_a rttext\&pid = S0103 - 11042019000600147\&lang = pt.Acessoem$: $1abr.2020.23$

TANAKA, M. Classificação de imagens com deep learning e tensorflow: O que é machine learning e como funciona. *Imasters, [S. l.]*, p. p. 1–2, jun 2018. Disponível em: https://imasters.com.br/back-end/classificacao-de-imagens-com-deep-learning-e-tensorflow. Acesso em: 1 abr. 2020. 22, 23

TECNOTRONICS. Expansor de entradas e saídas v5.0. v. 5, 2018. Disponível em: <https://www.tecnotronics.com.br/expansor-de-servo-entradas-e-saidas-v5-0.html>. Acesso em: 04 jun. 2019. 18

TENSORFLOW.ORG. apr 2020. Https://www.tensorflow.org/. 27

WEG. Variação de velocidade. *Jaraguá do Sul*, 2003. Weg. 19

Whitman, J. et al. Snake robot urban search after the 2017 mexico city earthquake. *Biorobotics Papers, [S. l.]*, p. p.1–6, dec 2017. 8, 29, 30

WOLF, D. F. et al. Robótica móvel inteligente: Do simulado às aplicações no mundo real. *Congresso da SBC – Sociedade Brasileira de Computação, São Carlos - SP*, 2009. Disponível em: http://inct-sec.icmc.usp.br/actrep/sites/default/files/highlights/Tutorial-JAI.pdf. Acesso em: 14 maio 2019. 15, 19

YANG, G.; XU, N.; HONG, Z. Identification of navel orange lesions by nonlinear deep learning algorithm. *Eng. Agríc., Jaboticabal*, v. 38, n. n.5, p. p. 783–796, sep 2018. 26

ZILLI, S. d. R. A robótica educacional no ensino fundamental: Perspectivas e prática. *Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis*, 2009. 89. 15