UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Evaristo José do Nascimento

RECONHECIMENTO DE GESTOS EM IMAGENS UTILIZANDO UM SENSOR DE PROFUNDIDADE PARA O CONTROLE DE UM ROBÔ MÓVEL

Santa Maria, RS
2017

Evaristo José do Nascimento

RECOHECIMENTO DE GESTOS EM IMAGENS UTILIZANDO UM SENSOR DE PROFUNDIDADE PARA O CONTROLE DE UM ROBÔ MÓVEL

Monografia apresentada ao curso de Graduação em Bacharelado em Engenharia de Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Engenheiro de Computação**.

Orientador: Prof. Dr. Daniel Fernando Tello Gamarra

Santa Maria, RS
2017

Evaristo José do Nascimento

RECOHECIMENTO DE GESTOS EM IMAGENS UTILIZANDO UM SENSOR DE PROFUNDIDADE PARA O CONTROLE DE UM ROBÔ MÓVEL

Monografia apresentada ao curso de Graduação em Bacharelado em Engenharia de Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Engenheiro de Computação**.

Aprovado em 19 de dezembro de 2017

Daniel Fernando Tello Gamarra, Dr. (UFSM)
(Presidente/Orientador)

Osmar Marchi do Santos, Dr. (UFSM)

Andrei Piccinini Legg, Dr. (UFSM)

Santa Maria, RS
2017

RESUMO

RECONHECIMENTO DE GESTOS EM IMAGENS UTILIZANDO UM SENSOR DE PROFUNDIDADE PARA O CONTROLE DE UM ROBÔ MÓVEL

AUTOR: Evaristo José do Nascimento
ORIENTADOR: Prof. Dr. Daniel Fernando Tello Gamarra

Foi realizado um estudo para desenvolvimento de um sistema de reconhecimento de comandos gestuais aplicados a um robô móvel. Para leitura dos gestos foi utilizado o sensor *Kinect*, o qual permite obter dados das articulações do corpo de uma pessoa e obter dados mais precisos dos movimentos. Iniciou-se a pesquisa fazendo um estudo da viabilidade do desenvolvimento do sistema. Para isso foram pesquisados diferentes algoritmos de agrupamento de dados para que fossem utilizados na pesquisa. Utilizaram-se os algoritmos *K-means* e *Fuzzy C-means* e foram obtidos resultados semelhantes entre os dois algoritmos, apesar de que com dados indefinidos o algoritmo *Fuzzy C-means* conseguiu trabalho com maior eficiência. Foi concluído que utilizando a técnica de agrupamento de dados não seria possível trabalhar com gestos que possuíssem maior complexidade. Para solução deste problema foi utilizada a técnica de redes neurais, a qual permitiu uma maior robustez do sistema podendo trabalhar com movimentos gestuais mais completos. Para trabalhos futuros estuda-se o uso de técnicas como *Deeplearning* para permitir que o sistema consiga maior desempenho e que consiga tratar semelhanças entre diferentes gestos, evitando possíveis erros de decisão, dificuldade a qual o sistema desenvolvido com redes neurais de uma camada enfrenta.

Palavras Chave: Sensor Kinect, Reconhecimento Gestual, Redes Neurais.

ABSTRACT

GESTURE RECOGNITON IN IMAGES USING A DEPTH SENSOR FOR A MOBILE ROBOT CONTROL

AUTHOR: Evaristo José do Nascimento
ADVISOR: Prof. Dr. Daniel Fernando Tello Gamarra

A study was realized to develop a system of recognition of gesture commands applied to a mobile robot. To read the gestures was used the Kinect sensor, which allows to obtain data of the joints of a person's body and to obtain more precise data of the movements. The research was started making a study of the feasibility of the development of the system. For this, different data grouping algorithms were searched for them to be used in the research. We used K-means and Fuzzy C-means algorithms and similar results were obtained between the two algorithms, although with undefined data the Fuzzy C-means algorithm was able to work with higher efficiency. It was concluded that using the data grouping technique it would not be possible to work with gestures that had greater complexity. To solve this problem, the neural networks technique was used, which allowed a greater robustness of the system, being able to work with more complete gestures. For future work, the use of techniques such as Deep Learning is used to allow the system to achieve greater performance and to be able to treat similarities between different gestures, avoiding possible decision errors, which the system developed with neural networks of a layer faces.

Keywords: Kinect Sensor, Gesture Recognition, Neural Networks.

LISTA DE FIGURAS

Figura 1 - Exemplos Robôs Móveis. 11
Figura 2 - Comparação entre os dados em seus grupos reais e os grupos definidos pelo kmeans.....17
Figura 3 - Esquema de funcionamento de um perceptron..... 20
Figura 4 - Exemplo genérico de uma rede neural..... 22
Figura 5 - Esquema básico de funcionamento do sistema de reconhecimento gestual.....23
Figura 6 - Sensor Kinect. 25
Figura 7 - Robô utilizado para realização do estudo 27
Figura 8 - Motor EMG 49, utilizado no robô móvel. 29
Figura 9 - Driver MD49, utilizado no robô móvel 29
Figura 10 - Modulo ArduinoMega 30
Figura 11 - Interface de desenvolvimento *Processing*..... 31
Figura 12 - Imagem lida pelo sensor *Kinect* 33
Figura 13 - Interface de Treinamento de Redes Neurais do MATLAB.....39
Figura 14 - Comparação de resultados de diferentes execuções do algoritmo K-means, aplicado em dados indefinidos. 41
Figura 15 - Gestos utilizados para classificação por agrupamento de dados. 41
Figura 16 - Resultados de execução do algoritmo K-means utilizando os dados dos comandos gestuais. 42
Figura 17 - Comparação de resultados de diferentes execuções do algoritmo Fuzzy C-means, aplicado em dados indefinidos. 43
Figura 18 - Resultados de execução do algoritmo Fuzzy C-means utilizando os dados dos comandos gestuais. 43
Figura 19 - Comparação entre os resultados dos diferentes algoritmos. 44
Figura 20 - Dados de Regressão de treinamento das três redes neurais. 45
Figura 21 - Resultado de execução das redes neurais para um gesto específico..... 46

LISTA DE QUADROS

Quadro 1 - Características físicas dos motores utilizados no robô 28
Quadro 2 - Valores de acurácia de cada rede neural e do sistema em geral 46

SUMÁRIO

1 INTRODUÇÃO 9
1.1 ROBÓTICA MÓVEL 9
1.2 REVISÃO BIBLIOGRÁFICA 11
1.3 OBJETIVOS 13
1.3.1 Objetivo Geral 13
1.3.2 Objetivos Específicos 13
2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 ALGORITIMOS DE AGRUPAMENTO DE DADOS 14
2.1.1 Algoritmo *K-means* 14
2.1.1.1 Limitações do Algoritmo K-means 15
2.1.2 Algoritmo Fuzzy C-means 16
2.1.2.1 Clusterização Fuzzy C-means 17
2.1.2.2 Propriedades do algoritmo Fuzzy C-means 18
2.1.2.3 Limitações do algoritmo Fuzzy C-means 18
2.2 REDES NEURAIS ARTIFICIAIS 18
2.2.1 O Perceptron 19
2.2.2 Redes Neurais Multicamadas (*feedforward*) 20
2.2.2.1 Treinamento de uma Rede Neural 21
3 MATERIAIS E MÉTODOS 23
3.1 O MICROSOFT KINECT 24
3.1.1 Sensor de Profundidade 24
3.1.2 Câmera RGB 25
3.1.3 Ferramenta *OpenNI* 25
3.2 ROBÔ MÓVEL 26
3.3 AQUISIÇÃO DE IMAGENS PELO *MICROSOFT KINECT* 30
3.4 IDENTIFICAÇÃO E ENVIO DOS COMANDOS AO ROBÔ MÓVEL 32
3.4.1 Utilizando o Algoritmo *K-means* e o Algoritmo *Fuzzy C-means* 34
3.4.2 Utilizando Redes Neurais *Feedforward* 35
3.4.2.1 Treinamento da Rede Neural 36
3.4.2.2 Execução da Rede Neural 38
4 RESULTADOS 40

4.1 RESULTADOS UTILIZANDO OS ALGORITMOS DE AGRUPAMENTO DE DADOS 40
4.1.1 Utilizando o algoritmo *K-means* 40
4.1.2 Utilizando o algoritmo *Fuzzy C-means* 41
4.1.3 Utilizando Redes Neurais 44
CONCLUSÃO 48
APENDICE A – CÓDIGO PROCESSING PARA OBTENÇÃO DE DADOS DOS MOVIMENTOS GESTUAIS 51
APENDICE B – CÓDIGO MATLAB DE RECONEHCIMENTO GESTUAL UTILIZANDO O ALGORITMO K-MEANS 53
APENDICE C - CÓDIGO MATLAB DE RECONEHCIMENTO GESTUAL UTILIZANDO UMA REDE NEURAL 54
APENDICE D – ARTIGOS PUBLICADOS DURANTE A PESQUISA 56

1 INTRODUÇÃO

O avanço da tecnologia nos permite hoje utilizar novas técnicas e conhecimentos aos quais o engenheiro tem que se adaptar e atualizar constantemente. Isso nos possibilita desenvolver sistemas que trás maiores capacidades e facilidades em tarefas que realizamos tanto na nossa vida profissional como no nosso dia-a-dia.

Com isso se busca nesse trabalho iniciar uma pesquisa utilizando técnicas que nos permita, em um futuro próximo, desenvolver ferramentas que possam facilitar nossas tarefas e atividades, tornando-as fáceis e práticas possibilitando vencer dificuldades e realiza-las em menos tempo. Em nossa pesquisa está sendo desenvolvido um sistema de reconhecimento de comandos gestuais, os quais controlam um robô móvel e a partir de um comando específico este robô realiza uma tarefa específica.

O estudo inicia-se pesquisando técnicas de agrupamento de dados, utilizadas com bastante frequência na área de aprendizado de máquina, que seria como uma ramificação da área de inteligência artificial. São utilizados dois algoritmos, *K-means* e *Fuzzy C-means*, para discriminação dos dados. O Algoritmo *K-means* possui dificuldades de trabalhar com dados que pertencem a *clusters* não radiais, onde esses dados possuem incertezas em suas características podendo se tornarem outliers, (PRASS, 2013). Essa dificuldade é resolvida com a utilização do algoritmo *Fuzzy C-means.* O trabalho também explorou o uso de uma rede neural. Com essa técnica é possível utilizar comandos gestuais mais complexos, o que não seria possível com algoritmos de agrupamentos de dados devido as suas limitações.

Após uma breve introdução, é apresentado, no Capitulo 2, um embasamento teórico explicando as técnicas e ferramentas utilizadas apresentando uma explicação teórica de como é seu funcionamento. No Capitulo 2 é descrito como que as técnicas e ferramentas vistas no Capitulo 2 foram utilizadas no projeto, assim como problemas encontrados. Também são apresentadas as ferramentas de software utilizadas no desenvolvimento do sistema. Já no Capitulo 3 são apresentados os resultados obtidos, realizando um análise de cada resultado obtido com as diferentes técnicas utilizadas. E por fim é realizadas uma breve conclusão sobre o trabalho além de apresentar os códigos gerados durante a pesquisa.

1.1 ROBÓTICA MÓVEL

A palavra robô é derivada da palavra tcheca *robota,* a qual é livremente traduzida como ‘trabalhador humilde’ (Romero, Prestes, Osório e Wolf (2014). Isso nos traz que um

robô não é nada mais que um “escravo” para os seres humanos, uma ferramenta para realizar tarefas do dia-a-dia ou auxiliar na realização das mesmas.

Romero, Prestes, Osório e Wolf(2014) também citam momentos em que podemos utilizar um robô:

a) Realizar tarefas que ofereçam risco ao ser humano;
b) Tarefas que sejam tediosas para uma pessoa;
c) Tarefas diárias;
d) Auxilio com pessoas necessitadas;
e) Como forma inovadora de ensino na educação.

Entre essas e outras utilidades que pode ter um robô, pode-se dizer que é uma ferramenta que possui uma capacidade de nos permitir praticidade, segurança e eficiência em muitos momentos de nosso dia-a-dia. A pesquisa na área de robótica cresce aceleradamente com o avanço da tecnologia e já está sendo utilizada em muitas áreas de conhecimento, desde a área médica até a área militar e de segurança.

Na área médica, por exemplo se utilizam robôs para auxilio em processo cirúrgicos que necessitam de precisão por exemplo, em situações em que determinado procedimento cirúrgico seja em pequenas regiões do corpo humano, em que seja muito difícil o acesso com as próprias mão do médico, o que se torna possível graças ao avanço da tecnologia atual que nos permite desenvolver máquinas com dimensões e movimentos muito precisos para realização de atividades que necessitam desta habilidade.

Existem muitos tipos de robôs: robôs industriais, robôs militares, robôs domésticos, entre outros. A área da robótica móvel é uma dessas áreas da robótica que está em desenvolvimento. Um robô móvel permite acessar lugares que ofereçam risco a um ser humano ou de difícil acesso. Exemplos disso são o uso de robôs por equipes antibombas em lugares de confronto militar, ou também pesquisas submarinas onde se torna difícil acessar área muito profunda de um oceano, ou ainda na procura de sobreviventes em acidente desastrosos.

Existem vários tipos de robôs móveis, em desenvolvimento. Os diferentes tipos de robôs móveis são classificados em relação ao seu modo de locomoção: Terrestres, aquáticos e aéreos.

Entre os robôs terrestres existem dois principais tipos, os robôs com rodas e robôs que se locomovem por meio de pernas (bípedes ou quadrúpedes, por exemplo). Os robôs terrestres realizam tarefas em solo. São exemplos, robôs de expedições militares, anti-minas, ou robôs domésticos como aspiradores de pó autônomos. Para acesso a locais onde a superfície possui

muitas irregularidades, existem robôs que se locomovem por pernas. O número de pernas varia, existem robôs bípedes (humanoides), existem robôs quadrupedes, entre outros. Os robôs bípedes são muito utilizados em pesquisas que buscam imitar certas características dos seres humanos, como robôs que praticam esportes, por exemplo.

Os robôs aquáticos são utilizados em processos de pesquisa submarinos em áreas oceânicas muito profundas. Geralmente são telecomandados e funcionam como se fossem extensão do corpo humano, não possuindo muita automatização. Também utilizam robôs aquáticos como submarinos não tripulados, buscando executar missões de alto risco de não sobrevivência de uma pessoa.

Também são desenvolvidos robôs móveis de mobilidade aérea. Funcionam basicamente como aeronaves não tripuladas e geralmente são telecomandados. Os robôs aéreos são muito utilizados em operações de busca e resgate, também são utilizados em missões militares de reconhecimento de alguma área, são utilizados como arma de combate em conflito militares, como equipamento de transporte em casos específicos e em aplicações na área civil e comercial.

Na Figura 1 são apresentados alguns exemplos de robôs móveis.

Figura 1 - Exemplos Robôs Móveis.

Fonte: Google Imagens.

1.2 REVISÃO BIBLIOGRÁFICA

A pesquisa relacionada a reconhecimento gestual tem uma grande abrangência em relação ao numero de áreas de conhecimento interessadas. Partindo da área da saúde, passando pela área comercial e chegando a área de engenharia realizam-se estudos

relacionados a reconhecimento de gestos de uma pessoa. Antes de apresentar o método de como foi realizado nosso estudo, vamos á partir desta sessão apresentar o estado da arte na pesquisa de reconhecimento gestual e no uso de técnicas na área de processamento de imagens, trazendo aqui trabalhos que se relacionam com nosso estudo e trabalhos que serviram como base para desenvolvimento da pesquisa.

Na área da saúde foi encontrado um trabalho voltado para identificação de sintomas relacionados a doenças ósseas e musculares. Pal, Saha e Konar (2014) utilizam o algoritmo *Fuzzy C-means* para classificar dados de movimentos e identificar possíveis dificuldades que pacientes possam possuir para realiza-los, identificando assim uma possível doença ou o estagio em que essa doença se encontra.

Já na área de pesquisa em engenharia foram encontrados vários trabalhos voltados ao reconhecimento de objetos como de gestos. Existem trabalhos que buscaram desenvolver técnicas de reconhecimento de objetos com câmeras RGB–D (BO, REN E FOX, 2012). Voltados ao reconhecimento gestual foram encontrados trabalhos que buscam utilizar as mais diversas técnicas e algoritmos em suas pesquisas. Kajan, Pernecký e Hammad (2015) utilizam redes neurais multicamadas, com dados obtidos pelo sensor Microsoft Kinect. Também temos Patsadu, Nukoolkit e Watanapa (2012), um trabalho que busca comparar técnicas de mineração de dados como SVM (*Support Vector Machine*), Redes Neurais, arvore de decisão e redes beiseianas, onde é verificada qual a técnica mais eficiente para aplicação com o sensor *Kinect*. Ainda temos Rimkus (2013) onde é aplicada a técnica de redes neurais para reconhecimento de movimentos das mãos de uma pessoa, baseado em informações das articulações do corpo, pesquisa a qual também utiliza recursos do sensor *Kinect* para obtenções dos dados de movimento. Ainda na pesquisa de reconhecimento gestual, existem trabalhos que utilizam técnicas de *clustering* para desenvolvimento do sistema. Panchal e Kandoriya (2013) desenvolveram um trabalho que utiliza o algoritmo *k-means* para realização do processo de classificação. Temos também Zhang (2015) que buscou desenvolver um sistema de reconhecimento gestual usando técnicas de *Unequal-probabilities background difference* combinada com o algoritmo *Fuzzy C-means*.

Além de trabalhos relacionados a pesquisa de Reconhecimento gestual, também foram encontradas pesquisas que exploram as técnicas estudadas neste estudo, utilizadas em reconhecimento de objetos. Gamarra e Quadros (2015) aplicaram técnicas de agrupamento de dados, através do algoritmo *K-means*, SVMs e *Bag of Words* para desenvolvimento de um sistema de reconhecimento de *objetos* à partir de dados da câmera RGB-D do sensor *Kinect*. Correa (2012) explorou o uso de redes neurais na aplicação de movimentação autônoma de

um robô móvel, utilizando técnicas de processamento de imagens e redes neurais para identificação de obstáculos e áreas de passagem livre em um ambiente. Lowe (1999) desenvolve um sistema de reconhecimento de objetos com a técnica K-NN (*Nearest-neighbor*) aplicada em regiões de uma imagem lida que seja a "chave" para identificação de um objeto, ou seja, uma característica específica de um determinado tipo de objeto. Finalmente Matas e Obdrzáek (2004), realiza um estudo onde se buscam verificar características geométricas e de aparência para identificação de um objeto.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Este trabalho tem como objetivo geral desenvolver um sistema de reconhecimento gestual. Cada gesto reconhecido funciona como um comando a um robô móvel que, ao receber informações de um comando gestual recebido, realizará uma tarefa especifica referente ao respectivo comando.

1.3.2 Objetivos Específicos

a) Aplicar técnicas e bibliotecas relacionadas ao uso do *Microsoft Kinect*;
b) Estudar e aplicar algoritmos de agrupamentos de dados, com o algoritmo de *K-means* na área de reconhecimento gestual;
c) Estudar e aplicar algoritmos de agrupamentos de dados, *Fuzzy C-means* na área de reconhecimento gestual;
d) Estudar e aplicar técnicas relacionadas a redes neurais na área de reconhecimento gestual.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ALGORITIMOS DE AGRUPAMENTO DE DADOS

Para o desenvolvimento do trabalho foi necessário explorar algumas técnicas da área de *machine learning* (também conhecido como aprendizado de máquina). Entre as técnicas da área de *machine learning* (ML) existe uma técnica onde se usam algoritmos para classificarem dados recebidos por um sistema. Esses algoritmos são conhecidos como **Algoritmos de Agrupamento de Dados**.

Nesta seção apresentaremos dois desses algoritmos usados em nossa pesquisa: o algoritmo ***K-means*** e o algoritmo ***Fuzzy C-means***.

2.1.1 Algoritmo *K-means*

O algoritmo *K-means* classifica os dados obtidos pelo sistema em grupos distintos respeitando as diferentes características que definem cada grupo. Para verificar as semelhanças entre o novo dado de entrada obtido pelo sistema e o grupos que são compostos pelos dados já classificados, é calculada a distância quadrática entre o novo dado obtido pelo sistema e os centroides de cada grupo definido. O grupo que possuir o centroide mais próximo quadraticamente ao dado a ser classificado, é o grupo ao qual o novo dado pertence.

O algoritmo recebe inicialmente o numero de grupos a serem utilizados na classificação dos dados, conforme vão sendo obtidos pelo sistema. Os dados podem ser obtidos um-a-um ou o sistema pode entregar um vetor ou matriz de dados para que o algoritmo possa classifica-los, isto depende de como o sistema, ao todo, foi desenvolvido.

A partir do numero de grupos definidos previamente (numero o qual vamos chamar de K) o algoritmo seleciona aleatoriamente K valores para formarem os centroides iniciais para realizar a classificação dos dados. Após cada novo dado classificado, são calculados os novos centroides a partir da média aritmética entre os dados que pertencem ao grupo. Após o recálculo dos centroides, os dados são realocados nos grupos conforme as novas distâncias quadráticas entre os dados e os novos centroides. Os centroides não são necessariamente um dado real, e sim o valor central em relação aos dados do grupo ao qual representa, por exemplo, em um grupo que possua três elementos numéricos, e seus valores sejam 3, 5 e 8, o centroide deste grupo será a média aritmética entre esses valores, onde neste exemplo a média é igual a 5,33.

A equação utilizada para o cálculo das distancias, entre os componentes de um grupo e os centroides, é a seguinte:

$$||x_i^{(j)} - c_j||^2 \quad (2.1)$$

Na equação, $\boldsymbol{x_i}$ é um dado pertencente ao conjunto $\boldsymbol{j}$, e $\boldsymbol{c}$ é o centroide do mesmo conjunto.

Abaixo mostramos um resumo baseado no artigo deAlizadeh (2014) que mostra o funcionamento do algoritmo de *K-means*:

a) Localizar aleatoriamente os K centroides iniciais utilizando os primeiros K elementos obtidos pelo sistema;
b) Adicionar cada elemento em seu respectivo grupo pertencente, baseado na distancia entre esse elemento e os centroides;
c) Recalcular os centroides e realocar os elementos nos grupos;

Repetir os passos 3 e 4 até que o local dos centroides não se alterem mais.

Podemos, antecipadamente, definir para que o algoritmo repita todo esse processo mais de uma vez, para que a precisão dos locais dos centroides seja melhor. Onde em cada reinicio da execução do algoritmo é tomada como posição inicial dos centroides a posição final dos centroides na execução anterior, exceto na primeira execução.

Buscando uma precisão no sistema podemos, também, definir a posição inicial dos centroides. Permitindo ao sistema "conhecer" previamente os grupos que classificarão os elementos do sistema. Alizadeh (2014) nos diz que o algoritmo de *K-means* é de aprendizado não supervisionado. O procedimento de definição prévia dos centroides pode ser chamado de treinamento do algoritmo. Treinando o algoritmo podemos **aproximá-lo** a um algoritmo supervisionado, lembrando que os valores pré-definidos dos centroides são apenas **iniciais**, ou seja, durante o processo de classificação dos elementos os locais dos centroides são reajustados conforme os dados são classificados.

2.1.1.1 Limitações do Algoritmo K-means

O *K-means* é um dos algoritmos mais simples e práticos quando se trata do assunto de agrupamento de dados. Mas como a maioria dos algoritmos que existem, ele possui algumas limitações relacionadas à posição inicial dos centroides, formatos dos grupos de dados e densidade dos dados.

Alizadeh (2014,) afirma que o algoritmo *K-means* é sensível aos centroides iniciais de cada grupo e os resultados podem ser diferentes com vários centroides iniciais de cada grupo.

Reforçando o que dissemos anteriormente, também nos diz que “uma solução para o problema é executar o algoritmo múltiplas vezes para obter os melhores resultados”.

Outra limitação é relacionada à densidade dos dados a serem agrupados. Dados com densidades menores tendem a serem divididos em diferentes grupos, onde podem na realidade pertencer a um mesmo. Pode ocorrer também o fato de que, se houver mais de um grupo mais denso que outros estes mais densos podem ser inseridos em um mesmo grupo. Esse fenômeno pode ocorrer devido a possível divisão de um grupo em dois ou mais, tendendo a agrupar dois grupos mais densos buscando atender o numero de grupos definidos no inicio do algoritmo.

Também se observa uma dificuldade do algoritmo *K-means* em trabalhar com grupos não circulares. Dados que na realidade pertencem a um determinado grupo pode ser adicionado a outro, pelo motivo de que certa região de um grupo possa estar muito próxima de outra, e a distância dos dados que pertencem a essa região e o centroide de outro grupo pode ser menor que a distância em relação ao centroide do grupo ao qual o dado realmente pertence. Alizadeh (2014) diz que uma solução para isso seria adicionar mais centroides ao processo de agrupamento buscando subdividir grandes grupos não globulares em menores grupos globulares, evitando assim que dados sejam classificados erroneamente em grupos distintos.

A Figura 2 exibe um exemplo de resultado do algoritmo com dados de grupos não circulares.

2.1.2 Algoritmo Fuzzy C-means

Existem casos, no processo de clusterização (agrupamento) de dados em que pode haver incertezas causadas por imprecisão nos atributos dos dados ou até mesmo devido à sobreposição de grupos distintos. Para solucionar esses problemas de incerteza, podemos fazer uso da Teoria de Conjuntos *Fuzzy* (TCF) (YONAMINE, SPECIA, CARVALHO E NICOLETTI, 2002).

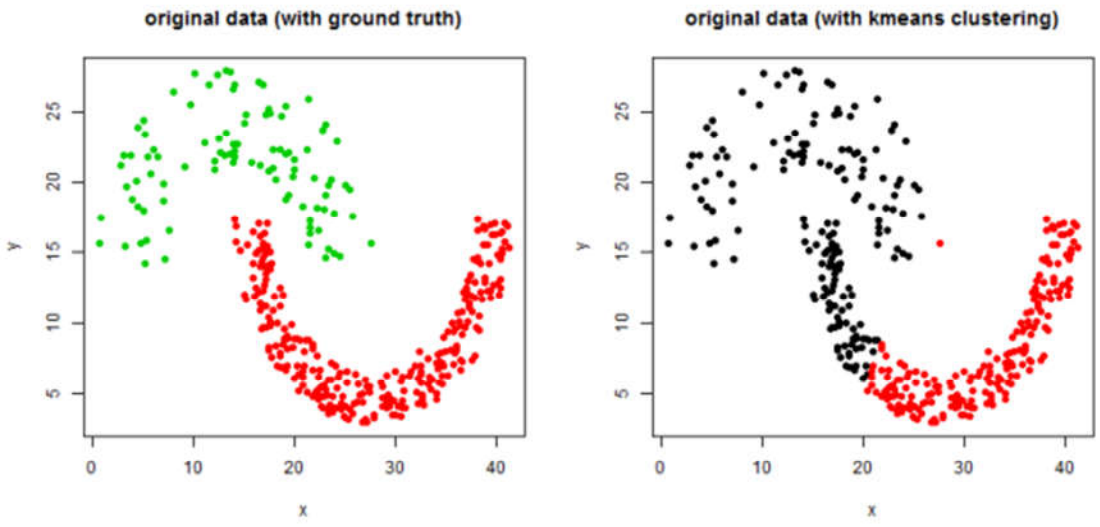
Quando um problema possui certas incertezas, uma maneira boa de trata-las é utilizando a probabilidade de determinado fato ocorrer. Pela teoria *fuzzy* podemos, no momento de fazer uma classificação de dados, verificar a probabilidade de determinado dado pertencer a certo grupo ou família, a partir de suas características. Então após o processo de agrupamento acabar, temos a informação da probabilidade de cada dado pertencer a cada grupo do sistema.

2.1.2.1 Clusterização Fuzzy C-means

O algoritmo de clusterização *Fuzzy C-means* é uma algoritmo de agrupamento de dados adaptado do algoritmo *K-means*, proposto por James C. Bezdeck em 1973 (BEZDEK, 2005 apud YONAMINE, SPECIA, CARVALHO E NICOLETTI, 2002). O algoritmo, na maioria das vezes que é utilizado, é aplicado em sistemas de reconhecimento de padrões.

Basicamente o algoritmo calcula, além da distância entre um dado e o centroide de um grupo, a probabilidade que um determinado dado tem de pertencer a um determinado grupo característico, utilizando as informações dos elementos daquele grupo. Obtendo as probabilidades de pertencer a cada grupo, o algoritmo insere o dado no grupo em que o dado possui a maior probabilidade de ser semelhante a maioria dos elementos desse grupo. Esse método resolve problemas de dados que se situam em fronteiras entre grupos, pois esses dados possuem caraterísticas próximas de dados pertencentes a esses diferentes grupos causando maiores incertezas a qual grupo realmente pertence.

Figura 2 - Comparação entre os dados em seus grupos reais e os grupos definidos pelo k-means.



Fonte: Google Imagens.

A probabilidade de um determinado elemento pertencer a um grupo é conhecida por **grau de pertinência** (YONAMINE, SPECIA, CARVALHO E NICOLETTI, 2002).

2.1.2.2 Propriedades do algoritmo Fuzzy C-means

O algoritmo *Fuzzy C-means* possui duas prorpiedades:

a) A soma dos graus de pertinência de todos os elementos de todos os grupos deve ser igual a 1, como mostra a equação abaixo:

$$\sum_{i=1}^{c} A_i(x_k) = 1 \quad (2.2)$$

Onde A_i seria um dos c grupos e x_k seria um elemento do grupo;

b) A soma dos graus de pertinência dos elementos de um grupo deve ser menor que soma do numero de elementos de todos os grupos, como mostra a equação abaixo:

$$0 < \sum_{k=1}^{n} A_i(x_k) < n \quad (2.3)$$

Onde n é a soma dos elementos de todos os grupos.

2.1.2.3 Limitações do algoritmo Fuzzy C-means

O algoritmo *Fuzzy C-means* possui dificuldades em lidar com dados ruidosos e também possui limitações na identificação de grupos de diferentes formas (JUNIOR, PALHANO, FELIPPE, MENEZES, SIMÕES E GARCIA, 2012).

Da mesma forma que o algoritmo *K-means*, o *Fuzzy C-means* possui dificuldades de trabalhar com grupos não circulares, por existir a possibilidade de dados de certa região de um grupo estar mais próximo do centroide de outro grupo do que o centroide do grupo que deveria pertencer.

Dados ruidosos, característica de dados pertencentes a grupos menos densos, por exemplo, também geram dificuldades para uma melhor execução do *Fuzzy C-means*. O algoritmo, ao deparar com ruído, acaba classificando-o de forma errônea por se encontrar junto de dados de um grupo, o que acaba interferindo na homogeneidade do respectivo grupo.

2.2 REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais são uma aproximação do cérebro humano, e são utilizadas para a solução de sistemas mais complexos. O cérebro humano é formado por uma rede de neurônios que processam sinais elétricos recebidos gerando uma nova informação. Em uma rede neural não é diferente, ela é formada por uma série de nós chamados de *perceptrons* (MONOLITO NIMBUS, 2017) que recebem informações em suas entradas, realizam um

processamento nesse dado através de uma função de ativação e disponibilizam a informação processada em sua saída.

2.2.1 O Perceptron

O *perceptron* não é nada mais que o neurônio da rede neural. Em uma rede neural eles estão interligados em camadas, onde os nós (*perceptron*) de uma camada têm suas saídas ligadas as entradas de todos os nó da camada posterior e suas entradas ligadas as saídas de todos os nós da camada anterior.

Cada *perceptron* possui n entradas onde cada entrada possui um "peso" inicializado aleatoriamente. Ele também possui uma entrada especial chamada de "*bias*" que define um grau de liberdade para o aprendizado do *perceptron*. O peso de cada entrada multiplica o valor que é recebido nessa entrada, após isto, todos os valores das entradas multiplicados por seus respectivos pesos são somados entre eles e também o *bias*, disponibilizando o resultado na saída do nó.(TISSOT, CAMARGO E POZO, 2012) A equação abaixo descreve matematicamente o funcionamento de um *perceptron*:

$$X_j = \sum_i W_{ij} X_i + \theta_j \tag{2.4}$$

Na equação acima temo que W_{ij} representa o peso da entrada i no neurônio j, X_i seria o dados da entrada i e θ_j o *bias* do neurônio j.

Porém o dado da saída somente será disponibilizado se atingir um *threshold* mínimo. Este *threshold* é definido a partir de uma função de ativação. A função de ativação insere uma não linearidade ao resultado, para isso ela precisa ser levemente derivável para que facilite o processo de treinamento da rede neural. Ela também define se o dado na saída do *perceptron* está correto ou não. Matematicamente o resultado na saída no *perceptron* fica da seguinte maneira:

$$X_j = \phi\left(\sum_i W_{ij} X_i + \theta_j\right) \tag{2.5}$$

Onde a função $\boldsymbol{\phi}$ seria a função de ativação do neurônio. Normalmente as funções mais utilizadas são a função Unidade Linear Retificada (ReLu) (FACURE, 2017) que édefinida pela seguinte equação:

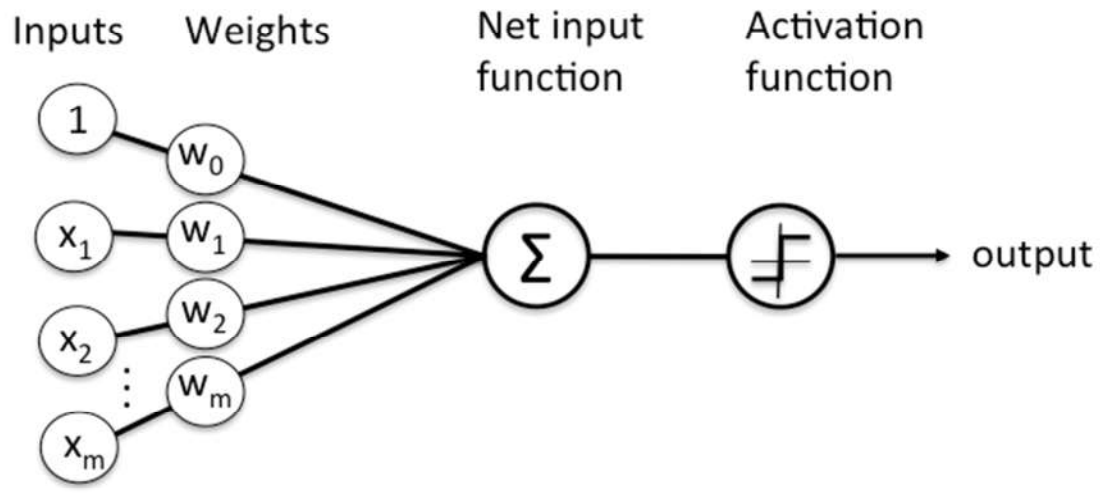
$$y = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2.6}$$

A função de ativação também pode ser defnida pela função sigmoide que é definida pela seguinte equação (FACURE, 2017):

$$Y = \frac{1}{1+ e^{-X}} \tag{2.7}$$

A figura 3 apresenta um esquemático básico da estrutura de um peréptron.

Figura 3 - Esquema de funcionamento de um perceptron.



Fonte: Google Imagens.

2.2.2 Redes Neurais Multicamadas (*feedforward*)

Como dito no inicio da sessão, as redes neurais são formadas por múltiplas camadas de *perceptrons* que são completamente interligadas, ou seja, todos os nós de uma camada são ligados a todos os nós da camada anterior e da camada posterior. Essas camadas são classificadas em três diferentes tipos.

O primeiro tipo é a camada de entrada. Esta camada é composta pelos neurônios que recebem os dados de entrada da rede. O numero de neurônios que compõem essa camada é igual o numero de entradas da rede, isso significa que cada neurônio da camada é responsável por uma entrada da rede, logo cada neurônio tem apenas uma entrada.

O segundo tipo é a camada de saída. Esta camada é composta pelos nós que disponibilizam os dados resultantes do processamento realizado pela rede. Igualmente a camada de entrada, o numero de neurônios que compõem a camada de saída é igual ao

numero de saídas que a rede neural possui. Então cada neurônio da camada de saída é responsável por uma saída da rede.

O terceiro tipo de camada de uma rede neural e não menos importante é a camada oculta. Podendo ser mais de uma, a camada oculta (*hidden*) é responsável pelo processamento dos dados inseridos na rede. Essas camadas trabalham interligadas totalmente umas com as outras propagando os dados que recebem da camada de entrada da rede (*forwarding*). O numero de camadas ocultas a serem adicionados em uma rede neural é definido de uma forma empírica, o que faz com que, no momento de desenvolvimento da rede neural, tenha que verificar a necessidade de haver mais camadas ou não para adicionar na rede.

2.2.2.1 Treinamento de uma Rede Neural

O treinamento de uma rede neural é o processo de aprendizado (*learning*) da rede para que ela consiga apresentar resultados corretos a partir de padrões recebidos em suas entradas. O treinamento consiste em ajustar os pesos do *perceptrons* que compõem a rede neural buscando minimizar o erro entre os resultados apresentados pela rede e os resultados esperados. Os métodos para realização são variados, e aplicáveis nos mais diversos tipos de redes neurais. Um método bastante conhecido é o *backpropagation*, onde o algoritmo começa a correção dos pesos da rede nas saídas e propaga pelas camadas ocultas até a entrada.

Inicialmente, pelo método *backpropagation*, os pesos dos nós da rede neural são inicializados aleatoriamente. Então é apresentado um vetor com combinações de valores de entrada e um vetor de combinações de valores de saída correspondente a cada valor de entrada. Adiciona-se uma combinação de valores de entrada na rede neural e se verifica o resultado obtido pela mesma. Compara-se o resultado obtido com a combinação de valores no vetor de saída correspondente a entrada que foi adicionada, então se encontra um erro conforme a seguinte equação:

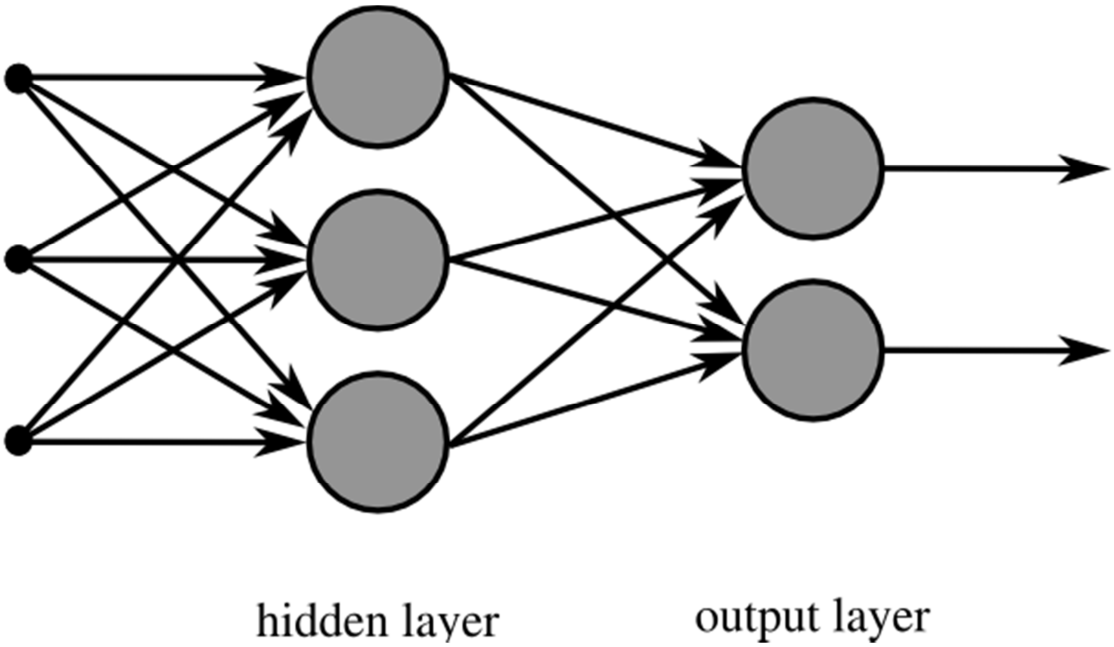
$$E = \sum_p \sum_i \left(V_{i,p} - V_{ip}^{desejado}\right)^2 \tag{2.8}$$

Onde V é uma combinação de dados de saída contida no vetor de padrão de saída p.

Após encontrar o erro, ele é retropropagado pelas camadas ocultas da rede e esse valor de erro servirá como parâmetro para atualização dos pesos nos nós da rede neural que possuem a maior “culpa” pelo erro ocorrido. Isso ocorre calculando o gradiente da função de ativação nos nós da rede e multiplicando-o pelo erro, onde o valor encontrado funcionará como fator multiplicativo no calculo do novo peso de cada nó.

A Figura 4 apresenta a estrutura de uma rede neural.

Figura 4 - Exemplo genérico de uma rede neural.



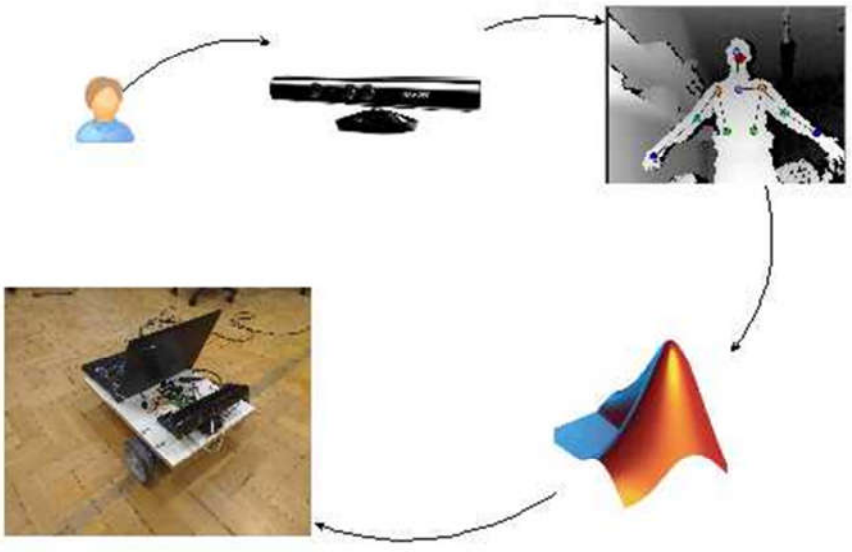
Fonte: Google Imagens.

3 MATERIAIS E MÉTODOS

O sistema de reconhecimento de gestos foi desenvolvido e aplicado em um robô móvel que utiliza rodas para sua locomoção. Para reconhecimento gestual foi utilizado o sensor *Microsoft Kinect,* para o qual foi desenvolvida uma estrutura física no robô, para que fosse possível a sua fixação no mesmo. O Kinect realiza a leitura do movimento gestual realizado, esses dados são preparados e enviados pelo *Procesing,* para o software MATLAB. O processing é um ambiente de programação com bibliotecas prontas onde se permite, com maior praticidade, trabalhar com técnicas de processamento de imagem e processamento e dados obtidos por sensores em geral (REAS E FRY, 2014). O Matlab realiza a identificação desse gesto verificando a qual classe pertence e a partir do gesto identificado envia o respectivo comando para o robô, o qual realiza a tarefa referente ao comando enviado. A Figura 5 apresenta um simples esquema do fluxo de funcionamento do sistema.

A seguir serão descritos os elementos que fazem parte deste processo e como cada um exerce sua função no sistema.

Figura 5 - Esquema básico de funcionamento do sistema de reconhecimento gestual.



Fonte: Google Imagens e Arquivo Pessoal.

3.1 O MICROSOFT KINECT

O *Microsoft Kinect* é um dispositivo fabricado pela *Microsoft* com o intuito de captar dados de movimento de uma pessoa que se encontra realizando movimentos a sua frente. Foi desenvolvido para o videogame *Xbox*, pela própria *Microsoft*, buscando maior interatividade e imersão em seus jogos. O *Kinect* consiste em um sensor que permite captar dados de uma pessoa, ou objeto, que está sendo filmado por ele, em três dimensões. Isso é possível graças a uma câmera que por onde ele obtém dados de imagem e um projetor infravermelho (IR) que trabalha junto com uma câmera IR para obter dados de profundidade do local que está sendo lido pelo sensor.

A Figura 6 exibe os principais componentes do sensor.

3.1.1 Sensor de Profundidade

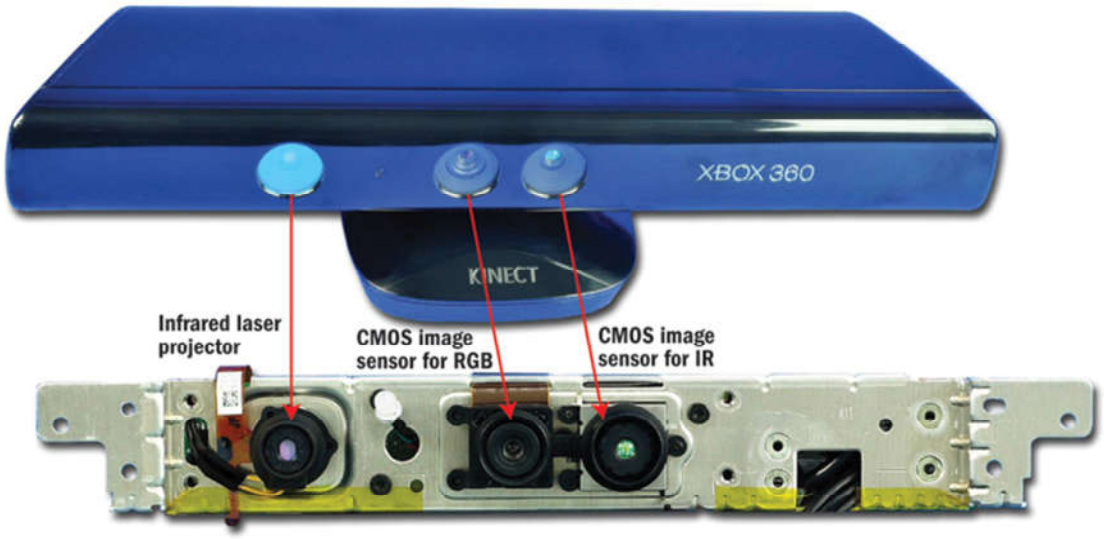
O sensor de profundidade é constituído de um emissor infravermelho e uma câmera infravermelho, os quais permitem obter dados de profundidade nos permitindo obter valores em três dimensões sobre a posição de um objeto ou uma pessoa, ou ainda dados de algum movimento executado por essa pessoa.

O emissor infravermelho emite uma matriz de raios infravermelhos, fazendo um mapa em três dimensões de um ambiente. No momento em que cada raio é refletido por um objeto ou uma pessoa, esse raio é captado pela câmera IR e assim é possível obter a distancia em que esse objeto se situa a partir do *Kinect*. O sensor calcula o tempo que passou entre o momento em que o raio foi emitido e o momento em que foi captado pela câmera, com isso ele consegue aproximar a distância do objeto.

Cada ponto da matriz de raios infravermelhos emitida pelo dispositivo equivale a um pixel. Então no momento em que determinado raio é refletido por um corpo podemos obter a coordenada, referente à matriz infravermelha gerada, do corpo ou objeto que está sendo captado por aquele raio. Juntando as informações adquiridas por todos os pontos da matriz, se torna possível montar a imagem em três dimensões do que está sendo lido pelo sensor *Kinect*.

Cruz, Lucio e Velho (2012,) apontam que A câmera IR do Kinect opera em 30 Hz e produz imagens com 1200x960 pixels. Estas imagens são amostradas para 640x480 pixels com 11 bits.

Figura 6 - Sensor Kinect.



Fonte: Google Imagens

Também mencionam que o campo de visão no sistema é de 58 graus horizontal, 45 graus vertical, 70 graus diagonal, e o intervalo de operação é entre 0.8 metros a 3.5 metros.

3.1.2 Câmera RGB

Além do sistema infravermelho composto por um emissor de raios e uma câmera infravermelha, o *Kinect* possui também uma câmera RGB. A câmera RGB tem a função de obter a imagem RGB pelo *Kinect*, possibilitando montar a imagem colorida do ambiente que está sendo trabalhado.

Segundo Cruz, Lucio e Velho (2012)a câmera RGB opera a30 Hz e pode produzir imagens em 640x480 pixels com 8 bits por canal" A câmera RGB, junto com o sistema IR, permite também o reconhecimento de um indivíduo identificando diferenças de cores e proximidades do mesmo.

3.1.3 Ferramenta *OpenNI*

Para que o Microsoft *Kinect* funcione corretamente, além dos drivers de funcionamento, é necessária uma calibração do dispositivo para que a câmera RGB e o

sistema infravermelho do *Kinect* funcionem de forma sincronizada. Esta calibração é realizada através de ferramentas que possibilitam, além da calibração, trabalhar com os dados obtidos com o *Kinect*, podendo obter dados de posição de um objeto ou pessoa, obter dados das juntas do corpo de uma pessoa ou ainda identificar os movimentos realizados por ela.

Uma das ferramentas utilizadas para se trabalhar com o *Kinect* é o *OpenNI*. Ele permite que possamos trabalhar com dados do corpo de uma pessoa, com as principais juntas, e possamos, com isso, identificar e manipular dados obtidos por movimentos realizados por essa pessoa. Além do *OpenNI* existem outras ferramentas como o *OpenKinect* e o *Microsoft Kinect for Windows*.

O *OpenNI* é uma plataforma multilinguagem *open source* compatível com os mais conhecidos sistemas operacionais existentes (CRUZ, LUCIO E VELHO, 2012). A plataforma e dividida em módulos que nos permitem trabalhar de diferentes formas com dados obtidos de uma pessoa.

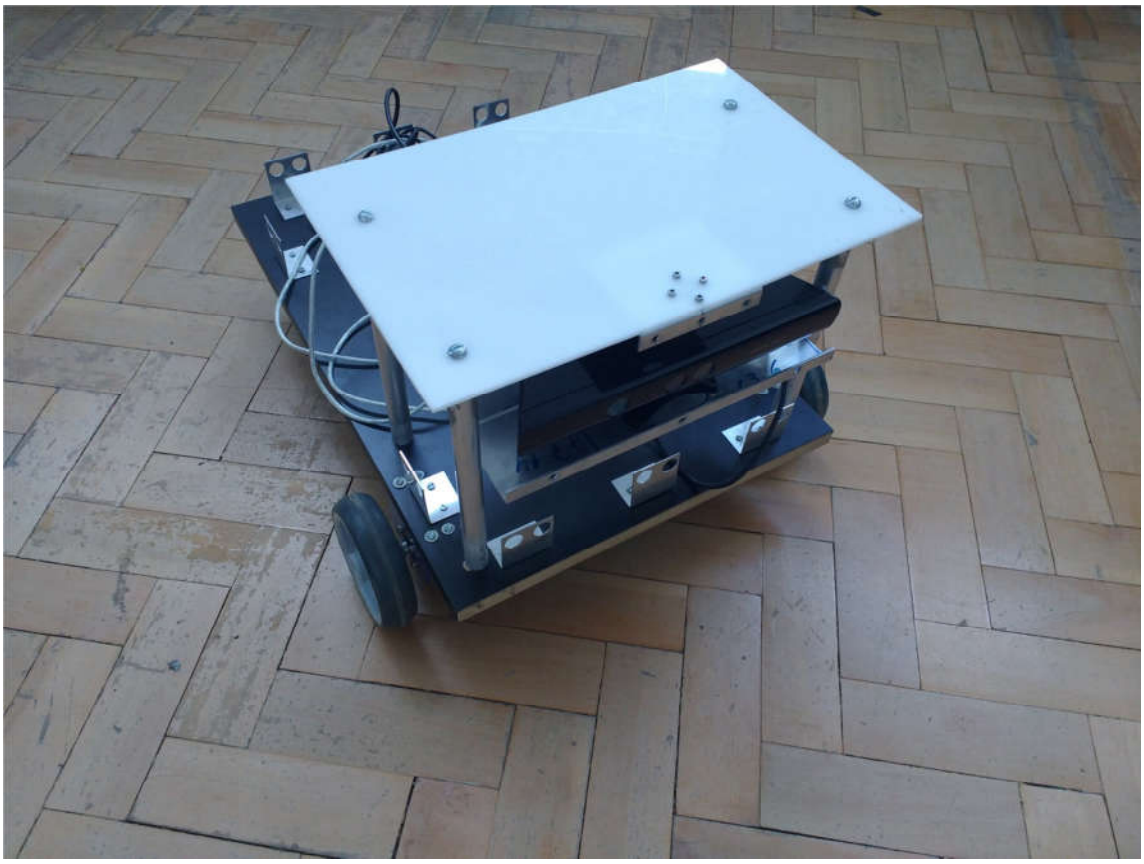
O *OpenNI* possui um módulo que permite obter a posição da mão esquerda do interator, permitindo-nos obter dados de posição da pessoa para utilização no sistema. A plataforma possui também um módulo que permite obter dados das principais juntas do corpo de uma pessoa. Chamado de *skeleton*, ele permite identificar uma ou mais pessoas que estão interagindo com o dispositivo. Um terceiro módulo permite captar movimentos realizados pela mão direita da pessoa. Esse módulo é conhecido como *scene analizer* (CRUZ, LUCIO E VELHO, 2012). Ele permite que possamos trabalhar com movimentos gerados pelo interator podendo desenvolver um sistema interativo entre o dispositivo que utiliza o *Kinect* e uma pessoa.

Outra forma de identificar movimentos é, utilizando o *skeleton*, captar as variações de posição de qualquer uma das juntas que estão sendo lidas pelo sensor. Assim podemos montar um vetor de movimento, que seria um vetor com as coordenadas X, Y e Z das posições em que determinada junta esteve e que formaram determinado movimento.

3.2 ROBÔ MÓVEL

O robô móvel, exibido a Figura 7, que recebeu o sistema de reconhecimento de gestos consiste em um robô movido por rodas, onde ele possui duas rodas dianteiras, as quais realizam a tração para exercer o movimento do robô e uma terceira roda traseira independente com a função de dar equilíbrio e estabilidade para o robô.

Figura 7 - Robô utilizado para realização do estudo



Fonte: Arquivo Pessoal.

Os motores que fazem as rodas dianteiras girarem são dois motores servo, com *encoders* embutidos, do modelo EMG49, controlados por uma placa drive do modelo md49. O controle do robô é realizado pelo modulo controlador *Arduino Mega* 2560 que utiliza o controlador *ATMega* 2560 da *Atmel*.

Os motores EMG49 são indicados para utilização com o *driver* md49. Isto se deve ao fato de que o md49 foi desenvolvido para trabalhar com esse modelo de motores. Os motores trabalham a um valor de tensão de 24 volts, com um consumo de corrente que chega a 13 amperes e um torque avaliado em 16 kg/cm.

O Quadro 1 contém as especificações físicas dos motores utilizados no robô móvel.

Quadro 1 - Características físicas dos motores utilizados no robô

Especificação	Valores
Tensão Nominal	24 V
Torque Nominal	16 kg/cm
Velocidade Nominal	122 rpm
Corrente Nominal	2100 mA
Velocidade sem carga	143
Corrente sem carga	500 mA
Corrente em torque máximo	13 A
Potencia Nominal de saída	34,7 W
Contagem do *Encoder* por volta do eixo	980

Fonte: Robot Eletronics - https://www.robot-electronics.co.uk/htm/emg49.htm.

A Figura 8 é uma imagem do motor utilizado no robô.

O *driver* md49 possui a capacidade de controlar dois motores de forma independente ao mesmo tempo. Ele permite determinar as velocidades, a energia enviada para os motores além de obter informações como dados de corrente, tensão dos motores e dados de movimento dos motores como posição e velocidade, obtidos através dos *encoders*. A tensão de trabalho do md49 é de 24 volts, o que exigiu que usássemos uma fonte regulável, possibilitando limitar a corrente e a tensão de entrada do driver.

Podemos ver na Figura 9 uma imagem do driver que foi utilizado no robô móvel.

Como citado anteriormente, o robô utiliza o modulo de controle *Arduino Mega* 2560. O microcontrolador que opera neste módulo é o *ATMega* 2560 da *Atmel*. O *ATMega* 2560 é um microcontrolador de 8 bits, desenvolvido em arquitetura RISC e utiliza 32 registradores de propósito geral com 8 bits cada.

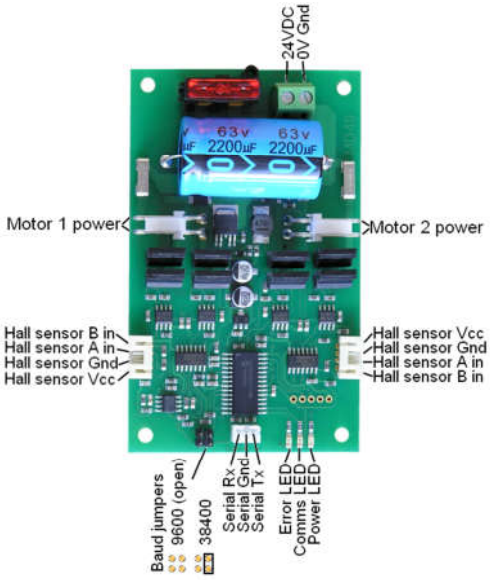
A Figura 10 mostra uma imagem do módulo controlador *Arduino Mega.*

Está sendo implementado no robô móvel um sistema de autonomia de movimento, o qual permite uma mobilidade independente para o robô. Com este sistema ele terá capacidade de identificar obstáculos e definir uma nova rota de movimento buscando desviar este obstáculo.

Figura 8 - Motor EMG 49, utilizado no robô móvel.

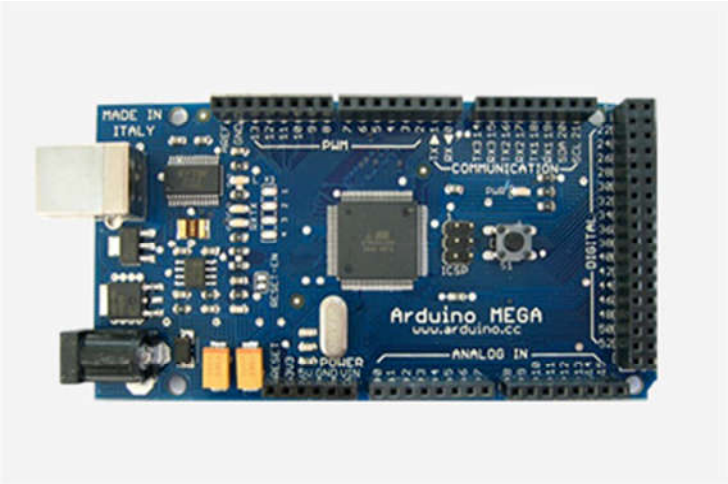
Fonte: Robot Eletronics - https://www.robot-electronics.co.uk/htm/emg49.htm.

Figura 9 - Driver MD49, utilizado no robô móvel



Fonte: Robot Eletronics - http://www.robot-electronics.co.uk/htm/md49tech.htm.

Figura 10 - Modulo ArduinoMega



Fonte: Google Imagens.

Esse sistema está sendo desenvolvido utilizando pelo menos três sensores de distância ultrassônicos, os quais são fixados nas laterais do robô e em sua frente, fazendo a leitura lateral e frontal da distância em que o robô está de um possível obstáculo, ou uma parede, por exemplo.

3.3 AQUISIÇÃO DE IMAGENS PELO *MICROSOFT KINECT*

O processo para aquisição de imagens pelo *Kinect* foi realizado usando uma ferramenta de software chamada *Procesing*. O *Processing*, exibido na figura 11, é um ambiente de programação desenvolvido pelo MIT no ano de 2001 que permite desenvolvimento de programação gráfica de uma forma facilitada. O *Processing* permite também a obtenção e processamento de dados com ferramentas como microfones e câmeras de video, possibilita também estabelecer comunicações via serial RS232 e via *Ethernet*. O *Processing* também nos possibilita trabalhar com dados obtidos por dispositivos como *eyetracking sensor*, *Leap Motion* e *Microsoft Kinect*. Toda essa gama de possibilidades se da ao fato de sua linguagem de programação ser baseada em JAVA. A sintaxe de programação é idêntica à sintaxe de JAVA. Uma vantagem é que se podem importar bibliotecas JAVA para desenvolvimento, o que aumenta o nível de possibilidades de desenvolvimento de uma forma simplificada.

Figura 11 - Interface de desenvolvimento *Processing*

Fonte: Arquivo Pessoal.

O *Processing* possui uma gama de ferramentas disponibilizadas, para as mais diversas utilidades, como por exemplo, processamento de imagens, processamento de áudio, programação visual, desenvolvimento de sistemas inteligentes, utilização de protocolos de comunicação e obtenção de dados por sensores.

O *Processing* disponibiliza também outros modos de programação utilizando as linguagens de programação *Python*, *Javascript* e também desenvolvimento para sistemas *Android*.

Para leitura dos dados do *Microsfot Kinect* foi utilizada a biblioteca para *Processing, SimpleOpenNI*. Desenvolvida para comunicação com a ferramenta *OpenNI*, ela possibilita

que, via *Processing,* se possa utilizar os recursos disponíveis pela *OpenNI* para trabalhar com dados obtidos pelo *Kinect*.

O software desenvolvido pelo *Processing* obtém os dados de posição do individuo que está sendo lido e as coordenadas de sua mão esquerda (direita na visão do *Kinect*). Essas informações são armazenadas em um objeto[3] especifico da biblioteca *SimpleOpenNI* e disponíveis para processamento posterior.

O primeiro passo para se obter dados do *Kinect* é montar a imagem obtida pelo mesmo, exibindo na tela a imagem do ambiente em profundidade que está sendo gerada pelo *Kinect*. Com isso podemos preparar o *Kinect* para captar dados de algum usuário que possa estar sendo lido pelo *Kinect*. Então se busca por algum usuário, caso tenha, insere-se em uma lista para o caso de houver mais de um usuário (funcionalidade oferecida pelo *Kinect* onde, mesmo que não seja utilizada, é preciso considera-la), para que assim seja possível organizar os dados de cada um. A partir de então se verifica se a lista contém algum usuário identificado e se inicia a comunicação com ele. Esta comunicação permite obter os dados de posição, dados das juntas e de movimentos que o usuário possa estar realizando.

Com o usuário identificado o próximo passo é, estando com a comunicação estabelecida, obter dados de posição dele dentro da imagem gerada pelo *Kinect* e, no nosso caso, ao mesmo tempo em que obtemos os valores das coordenadas x, y e z do usuário, podemos inserir uma circunferência de cor magenta para manter a identificação visual do usuário na imagem. Com esta etapa pronta, já podemos identificar a mão esquerda do usuário, para que seja possível identificar os gestos que serão realizados por ele.

Identifica-se a mão utilizando a ferramenta *skeleton* da *OpenNI*. Depois de identificada já podemos obter as coordenadas da mão esquerda do usuário e assim, igualmente como foi feito para a identificação do usuário, é adicionada um circunferência na cor azul, na imagem gerada pelo *Kinect*, no local onde seria a mão esquerda do usuário.

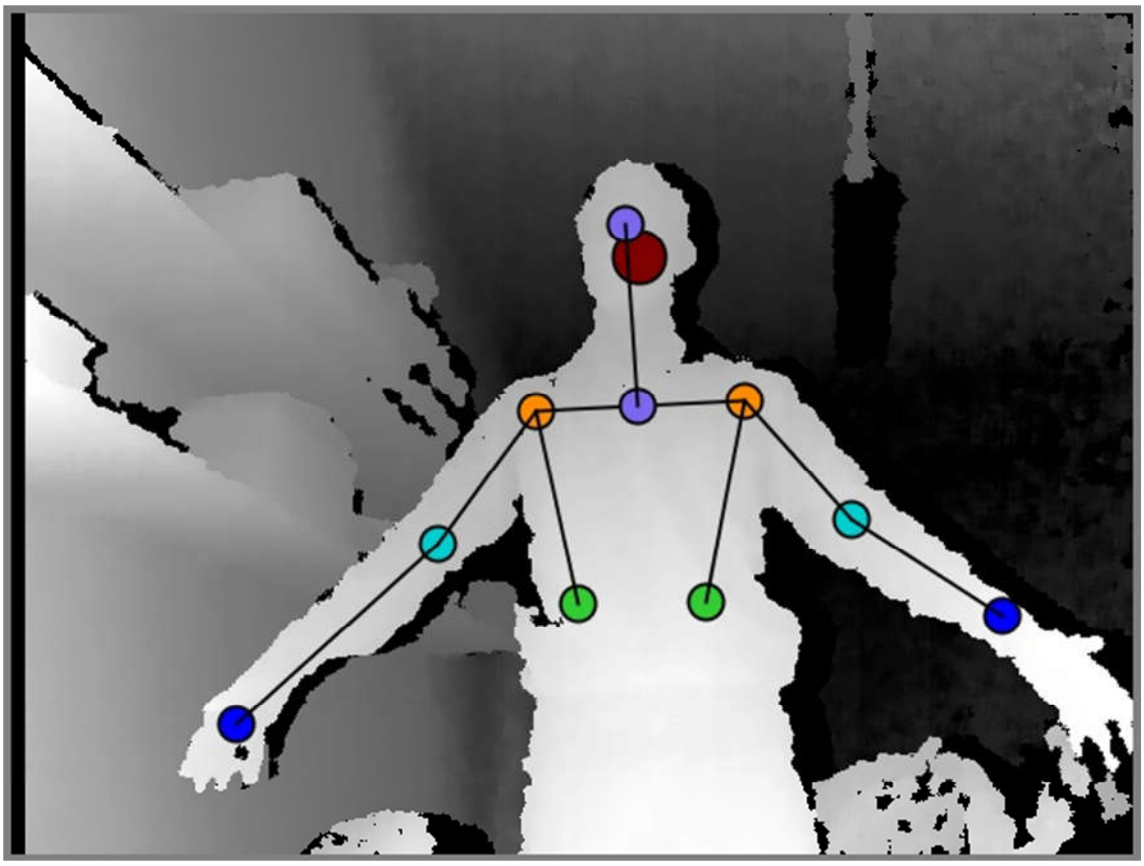
A Figura 12 mostra a imagem lida pelo kinect.

3.4 IDENTIFICAÇÃO E ENVIO DOS COMANDOS AO ROBÔ MÓVEL

Para processamento dos dados recebidos do *Kinect* foi utilizada a ferramenta MATLAB da empresa Mathworks, pelo motivo de proporcionar praticidade e eficiência no uso das técnicas para identificação dos dados gestuais captados pelo *Kinect.*

[3] Estrutura desenvolvida e interpretada pelo paradigma de programação orientado a objetos.

Figura 12 - Imagem lida pelo sensor *Kinect*



Fonte: Arquivo Pessoal.

O MATLAB é uma ferramenta utilizada no ramo da engenharia para projeto, ensaios e simulação. Além das mais variadas possibilidades que o MATLAB proporciona, com ele, podemos utilizar técnicas de *machine learning* para que possamos desenvolver nossa pesquisa da melhor forma.

Inicialmente é preciso falar de como foi implementado o processo de transferência de dados entre o *Processing* e o MATLAB. Para que fosse permitindo a leitura de dados pelo *Kinect* e em tempo real os comandos gestuais captados já pudessem ser enviados ao robô, foi preciso criar uma conexão *socket*[4] onde o servidor foi implementado no código de obtenção de dados, no *Processing* e o código de reconhecimento dos gestos, implementado no MATLAB, foi transformado em um cliente *socket*, onde ele recebe dados via a conexão e já realiza o processamento dos mesmos. Montada a conexão o *Processing* envia os dados para o

[4] Primitivas de transporte que permitem acessar vários protocolos da camada de transporte do modelo TCP, dentre eles TCP e UDP. Assim é possível realizar uma comunicação bidirecional entre duas máquinas em uma mesma LAN (FACULDADE DE INFORMÁTICA PUC-RS, 1998).

endereço de *loopback*[5] e o cliente criado no MATLAB que está configurado para receber dados do servidor que está também configurado no endereço de *loopback*, recebe os dados para serem processados.

Foram exploradas três diferentes técnicas para o reconhecimento dos gestos. Para cada técnica de processamento dos dados adotada, foram utilizados diferentes métodos para a realização do projeto do software e envio dos comandos ao robô.

3.4.1 Utilizando o Algoritmo *K-means* e o Algoritmo *Fuzzy C-means*

Os algoritmos desenvolvidos com a utilização dos algoritmos *K-means* e o que utiliza o algoritmo *Fuzzy C-means* são muito semelhantes, não necessitando realizar muitas modificações de uma para o outro.

Utilizou-se o algoritmo *K-means* buscando obter a viabilidade de uma possível solução do problema em questão. Com o intuito de se obter um sistema simples e ao mesmo tempo eficaz no processo de classificação dos gestos. Basicamente para chegar à solução adicionam-se os dados obtidos do *Kinect* a uma matriz de entrada para que o algoritmo possa realizar a sua classificação e depois de classificado se identifica a que grupo de comandos pertencem os dados classificados e então se envia o respectivo comando ao robô para que realize a ação correspondente.

Além da configuração do sistema como cliente socket, precisou-se também a configuração de uma conexão via serial, por onde é feita a comunicação do robô para envio do comando identificado.

Todos os passos, desde a obtenção dos dados enviados pelo *Kinect*, até a classificação e envio dos comandos ao robô, são implementados dentro de uma estrutura de repetição para que o sistema execute vários comandos em tempo real conforme ele vai recebendo mais dados. Primeiro ele recebe os dados da conexão socket, se disponíveis, e os adiciona em uma matriz. Esta matriz funciona como uma matriz de entrada de dados que será enviada como parâmetro de entrada para o algoritmo, para que ele realize a classificação dos dados que estão na matriz. Após sua execução o algoritmo retorna as coordenadas finais dos centroides de cada *cluster* e também um vetor que indica a qual grupo cada dado da matriz pertence. A partir deste vetor se identifica a qual grupo o gesto recebido pertence e então é enviado, via

[5] Endereço referente ao dominio de *localhost* (127.0.0.1) utilizados para realização de testes e pesquisa em redes de computadores.

comunicação serial, o comando correspondente ao grupo em que o gesto lido pelo *Kinect* pertence.

A implementação do código utilizando o algoritmo *Fuzzy C-means* não muda até o momento de iniciar a execução da classificação. Após a classificação ser realizada, é preciso, por características do algoritmo, fazer uma separação, por grupos, dos dados para identifica-los. Com isso é possível verificar a qual grupo pertence o gesto lido e enviar o comando correspondente ao robô móvel.

3.4.2 Utilizando Redes Neurais *Feedforward*

Com o objetivo de obter uma melhor robustez do sistema e uma possibilidade de aumenta-lo posteriormente, foi desenvolvido um código utilizando redes neurais *feedforward* para interpretação dos gestos e classificação dos mesmos. Utilizando redes neurais se permite uma maior capacidade para se trabalhar com um volume maior de dados, além de usar dados mais complexos e não lineares, permitindo aumentar a capacidade do sistema em relação a desempenho e confiança.

No processo de desenvolvimento do sistema é necessário decidir qual é o método de treinamento da rede, obter os dados de treinamento, como eles serão apresentados, como será o resultado que queremos obter da rede, para que, a partir de então se possa definir as dimensões da camada de entrada da rede neural, a camada de saída, o numero de camadas ocultas (*hidden layers*) e as dimensões de cada camada oculta.

Antes de começar o desenvolvimento do código que realizará o treinamento da rede, é preciso coletar amostras de movimentos para montar os dados de treinamento. Os dados que formam o movimento consistem de uma matriz n X 3, onde n é o numero de pontos lidos pelo sensor *Kinect,* e são 3 coordenadas (X, Y e Z) que descrevem a localização de cada ponto*,* onde esses pontos descrevem a trajetória do movimento captado.

Uma dificuldade encontrada para captar os pontos que descrevem os gestos realizados é a distância do *Kinect* para quem está realizando o gesto. Quanto mais longe do *Kinect* está o usuário, considerando que esteja dentro do intervalo de alcance do *Kinect*, menor é área ocupada por quem está executando os movimentos dentro da imagem captada pelo *Kinect*. Logo essa diferença faz com que sejam lidos valores diferentes das coordenadas, principalmente no eixo Z o qual informa os valores de distancia, e fazendo com que a rede neural menor probabilidade de informar os resultados corretamente.

Para solucionar esse problema mudamos o referencial para o centro de massa de quem está executando o gesto. Então basta verificar a de cada coordenada com cada coordenada do centro de massa para a pessoa, fazendo com que a distância entre as juntas que estão sendo utilizadas para realizar o movimento e seu referencial não mude. (Heickal, Zhang e Hasanuzzaman, 2013).

A equação a seguir descreve o cálculo das coordenadas em relação ao centro de massa do usuário:

$$(X_r, Y_r, Z_r) = (X_1 - X_c, Y_1 - Y_c, Z_1 - Z_c) \quad (2.9)$$

Onde X_ré a coordenada da junta que está sendo lida em relação ao centro de massa do usuário; X_1 é a coordenada da junta em relação ao *Kinect* e X_c é a coordenada do centro de massa do usuário.

Para realizar o treinamento da rede neural esses dados necessitam ser adicionados em uma matriz de entrada para serem aplicados na rede neural. Porém nenhum dado é idêntico a outro, ou seja, algumas amostras de movimento vão ter um maior numero de pontos captados do que outras. Com isso os dados precisam ser alinhados utilizando um algoritmo chamado *Dynamic Time Warping* (DTW) (MATHWORKS, 2016)para que então, seja possível adicioná-los na matriz de dados para treinamento da rede neural. Após serem alinhados, os dados são normalizados, e, a partir de então, estão prontos para serem utilizados no processo de treinamento da rede neural.

A seguir mostraremos como foi desenvolvido o processo de treinamento da rede neural e a execução dela conforme o sistema recebe novos dados do *Kinect*.

3.4.2.1 Treinamento da Rede Neural

Para que a rede neural nos apresente resultados corretos e que possua um mínimo de acertos é necessário realizar o treinamento da rede, aonde ela irá "conhecer" os diferentes tipos de dados que irá receber, e saber qual gesto está sendo realizado pelo usuário.

O processo de treinamento da rede neural consiste basicamente em carregar os dados pré-coletados para realizar o treinamento, montar as matrizes de treino e de teste, montar o vetor de saída com os resultados desejados, montar a rede neural, realizar seu treinamento e testá-la para verificar se está funcionando como o desejado ou se precisa realizar alguma modificação.

Inicialmente é montado o vetor de saídas com os respectivos resultados desejáveis, mapeados pelos índices do vetor. O vetor é de dimensões 1 x n, onde n é o numero de

amostras que compõem a matriz de dados de treinamento e teste da rede neural. Simplificando o entendimento, o elemento que está no índice i do vetor de saída, ou vetor de resultados desejados, é o resultado esperado para os valores de entrada i, da matriz de dados de entrada. No nosso problema os dados estão acondicionados em um vetor pelo motivo de que a camada de saída da nossa rede neural terá apenas um neurônio (uma saída), o que nada impede de ter uma rede com mais saídas, o que exige então, que os dados de saída sejam acondicionados em uma matriz $n\ X\ m$, onde n são o numero de saídas da rede e m o numero de amostras a serem utilizadas no treinamento/teste da rede neural. Montado o vetor de saída, os dados são normalizados.

Seguindo o processo de treinamento da rede neural, o próximo passo é montar a matriz de entrada do treinamento, de saída desejada do treinamento e a matriz de teste da rede neural. Para montagem da matriz de entrada, é selecionado, aleatoriamente, dois terços das amostras para realizar o treino e o restante é utilizado para realização dos testes da rede neural após o treinamento realizado. Os resultados desejados, referentes aos dados de entrada que serão usados para treinar a rede, também são separados porque são utilizados para comparação no processo de treinamento da rede.

Os dados que foram utilizados para o treinamento foram organizados em três matrizes. Buscando uma melhor praticidade na sua manipulação, cada matriz armazena uma coordenada dos movimentos, ou seja, uma matriz para os valores das coordenadas X das amostras, outra matriz para os valores das coordenadas Y e a terceira para os valores das coordenadas Z das amostras. A matriz de entrada da rede neural possui dimensões como mostra a seguinte equação:

$$n \text{ X } m$$

Onde n representa o triplo do tamanho de um vetor que representa um movimento. No nosso projeto estamos utilizando movimento com 67 valores (tamanho da maior amostra obtida), onde da linha 1 até 67 da matriz ficam as coordenas X, da linha 68 até 134 são adicionadas as coordenas Y e da linha 135 até 201 são adicionados os valores das coordenadas Z das amostras. O numero m de colunas representa o numero de amostras utilizadas no treinamento, onde em cada coluna da matriz está contida uma amostra.

Antes de iniciar o treino da rede neural, precisamos cria-la. Para criar a rede neural precisamos informar a configuração da camada oculta da rede neural e depois configurá-la informando os dados que serão utilizados no processo de treinamento para que possa configurar os dados de entrada e saída de rede neural. Para definir a configuração das camadas ocultas devemos informar um vetor de n posições, onde n é o numero de camadas

ocultas que serão criadas e cada elemento deste vetor define quantos neurônios irão compor cada camada. No processo de configuração da rede neural também são inicializados os pesos de cada *perceptron* de forma aleatória, os quais serão corrigidos durante o processo de treinamento.

A partir de então já pode ser realizado o processo de treinamento da rede neural. Durante a execução do processo é exibida uma ferramenta pelo MATLAB, chamada de *nntraintool*. Ela permite que possamos acompanhar o processo e treinamento e depois de terminado, verificar a variação e comportamento do erro da rede, durante o treino.

A Figura 13 exibe a interface do *nntraintool.*

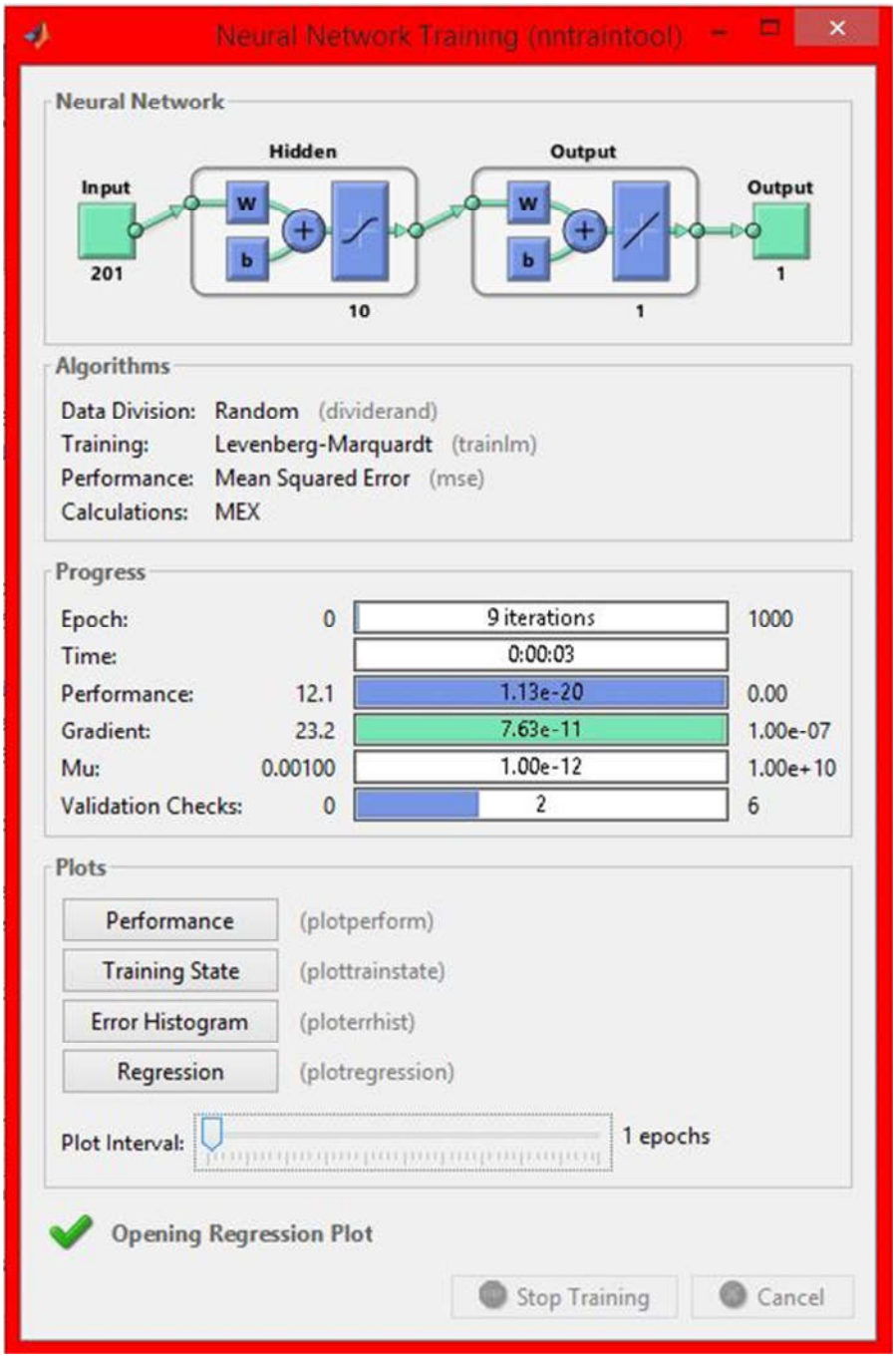
Finalmente a rede neural é testada inserindo os dados de teste (dados os quais a rede não conhece) para verificar se desempenho e verificar a necessidade de realizar algumas modificações para corrigir algum problema na rede neural.

3.4.2.2 Execução da Rede Neural

Para execução da rede neural primeiramente são carregado os dados lidos pelo *Kinect* que descrevem um gesto executado. Então estes dados são organizados em um vetor de entrada igualmente aos dados de entrada de treinamento, do elemento 1 ao 67 as coordenadas X dos pontos que compõem o movimento, do 68 ao 134 as coordenas Y e do 135 ao 201 as coordenas Z.

Após os dados serem organizados, são inseridos na rede neural para que seja identificado qual o gesto que foi realizado. O valor de saída apresentado pela rede é comparado com os valores que representam cada tipo de gesto reconhecível pelo sistema, os quais foram utilizados no processo de treinamento da rede, e então é calculada a distância entre o resultado obtido e os resultados possíveis, onde o resultado possível mais próximo ao obtido indica que o gesto o qual é representado por si é o gesto realizado pelo usuário. Após o gesto ser identificado, envia-se o respectivo comando ao robô para que realize a tarefa desejada.

Figura 13 - Interface de Treinamento de Redes Neurais do MATLAB.



Fonte: Arquivo Pessoal.

4 RESULTADOS

Neste ultimo capítulo apresentaremos os resultados obtidos utilizando as diferentes técnicas aplicadas para desenvolvimento da pesquisa. Serão apresentados detalhes dos resultados comparando-os e avaliando dificuldades e vantagens de cada técnica.

4.1 RESULTADOS UTILIZANDO OS ALGORITMOS DE AGRUPAMENTO DE DADOS

4.1.1 Utilizando o algoritmo *K-means*

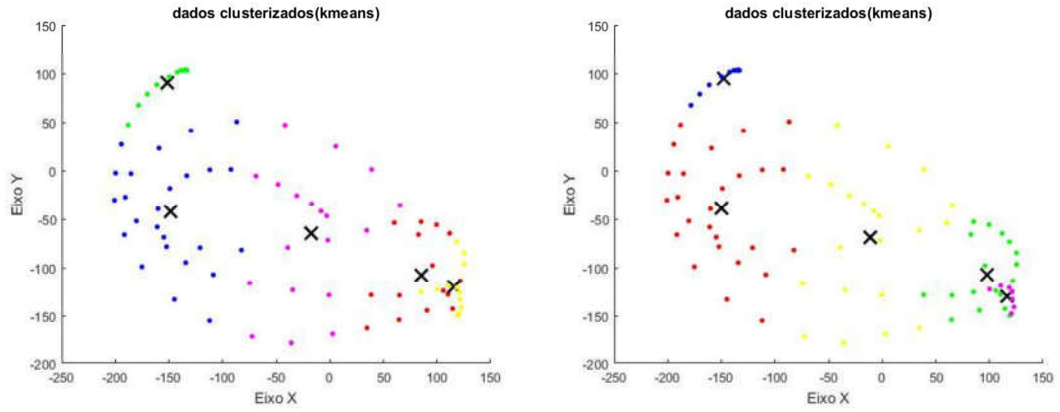
O sistema utilizando o algoritmo *K-means* possui limitações consideráveis no processo de reconhecimento gestual. Com o desenvolvimento da pesquisa foi descoberto que o algoritmo, proveniente do MATLAB, não trabalha com matrizes cúbicas de dados, ou seja, não é possível agrupar dados de uma matriz de vetores com o algoritmo utilizado.

Com isso o sistema capta a posição, na imagem captada pelo *Kinect,* da articulação utilizada no movimento gestual realizado pelo usuário e identifica em que região da imagem está sendo realizado o gesto. Conforme a região identificada o respectivo comando é enviado para o robô. Essa limitação possibilita trabalhar apenas com gestos mais simples como erguer a mão ou colocar a mão pra frente, por exemplo.

Na Figura 14 é apresentado um exemplo de agrupamento, realizados duas vezes com o mesmo grupo de dados, utilizando o algoritmo *K-means*. Pode-se notar que com regiões bem distintas o algoritmo trabalha de uma forma bem eficiente, já que é uma característica do *K-means* ter facilidade de trabalhar com grupos de dados distintos, diferente de quando encontra dados em regiões de incerteza como fronteiras entre *clusters* ou quando se tem grupos quê não são bem definidos.

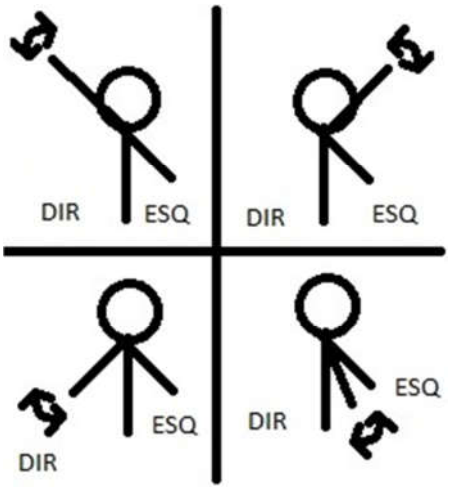
Os movimentos gestuais utilizados para realização dos testes são mostrados na Figura 15, lembrando que os movimentos foram realizados com a mão esquerda a qual é reconhecida como direita pelo *Kinect* pelo motivo de a imagem resultante do sensor ser espelhada em relação à imagem lida. A Figura 16 apresenta os dados coletados de cada movimento gestual e como foram agrupados pelo algoritmo.

Figura 14 - Comparação de resultados de diferentes execuções do algoritmo K-means, aplicado em dados indefinidos.



Fonte: Arquivo Pessoal.

Figura 15 - Gestos utilizados para classificação por agrupamento de dados.



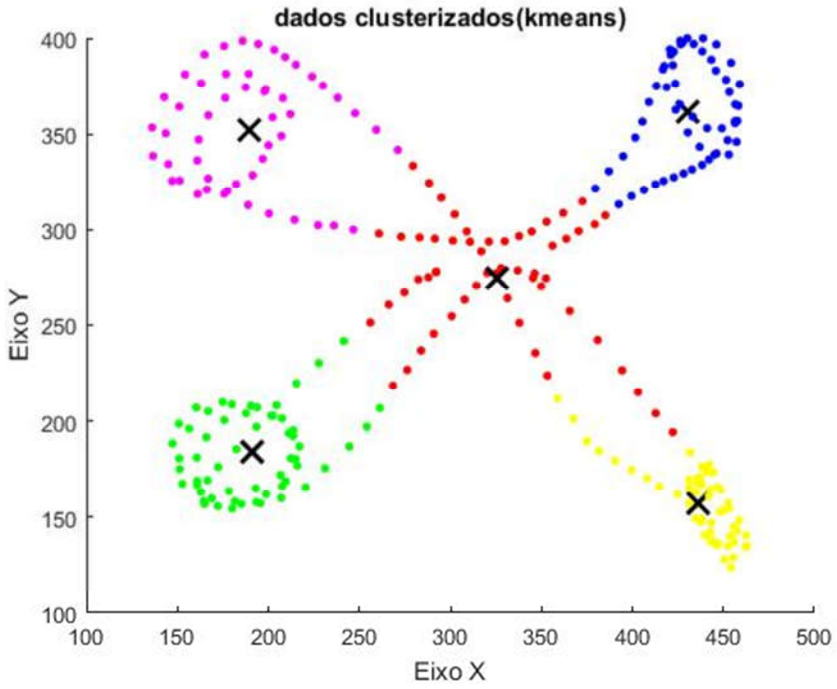
Fonte: Arquivo Pessoal.

4.1.2 Utilizando o algoritmo *Fuzzy C-means*

A semelhança entre os algoritmos *Fuzzy C-means* e *K-means* fazem com que as limitações do projeto realizado utilizando os algoritmos são as mesmas. Porém o *Fuzzy C-means* consegue tratar dificuldades com dados que se encontram em regiões de incerteza. Com o *Fuzzy C-means* consegue-se agrupar os dados que causam incerteza no momento do *clustering*, graças ao uso do grau de pertinência que define a qual grupo determinado valor

pertence, mesmo que graus de pertinência à diferentes grupos sejam muito próximos, um sendo maior que o outro, o mínimo que seja, é o que define a qual grupo o dado pertence.

Figura 16 - Resultados de execução do algoritmo K-means utilizando os dados dos comandos gestuais.



Fonte: Arquivo Pessoal.

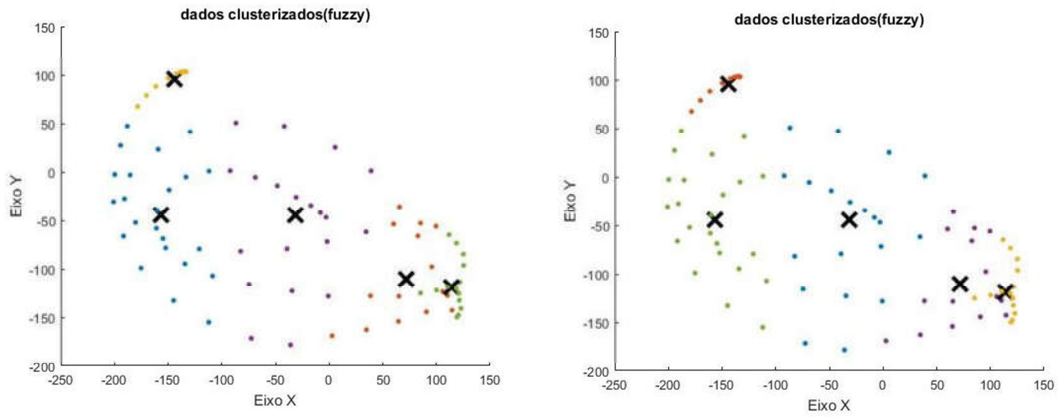
Duas execuções do algoritmo com o mesmo grupo de dados é apresentado na Figura 17. É visível a consistência dos resultados, os quais são muito semelhantes entres as duas execuções com grupos menos distintos.

A Figura 18 apresenta os dados resultantes dos movimentos utilizados já agrupados pelo algoritmo *Fuzzy C-means.*

Comparando os algoritmos *Fuzzy C-means* e o *K-means*, nota-se uma semelhança no desempenho dos dois algoritmos. O Fuzzy C-means leva vantagem em dados incertos, porém se for trabalhado com movimento bem distintos essa característica deixa de ser presente. Pode-se concluir que os dois algoritmos se comportam da mesma forma e que não há um melhor que o outro para nossa aplicação. Essa conclusão é reforçada pela figura 19 onde vimos poucas diferenças na comparação de seus resultados. Nesse gráfico pode-se notar

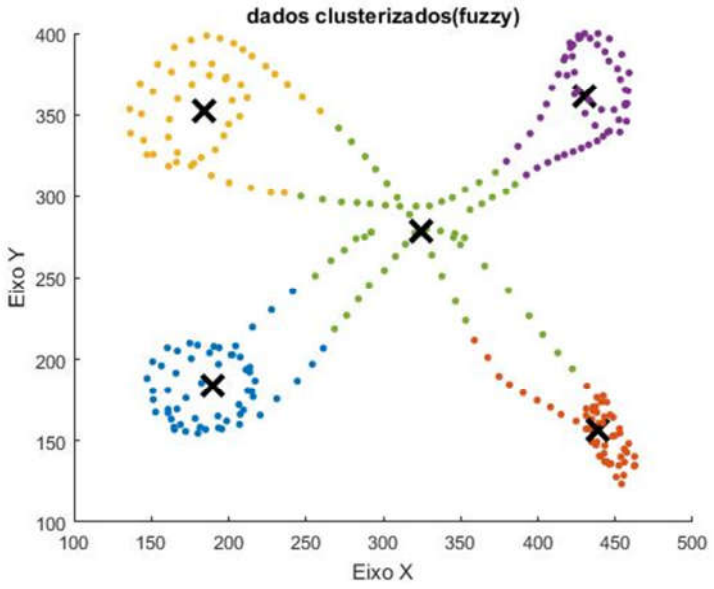
pouca variação na distribuição dos dados dentro os grupos. Os grupos 3 e 5 apresentam diferenças por serem grupos vizinhos, onde dados próximos a fronteiras entre esses dois grupos são adicionados ao grupo 3 pelo algoritmo *Fuzzy C-means* e esses mesmos dados são adicionados ao grupo 5 pelo algoritmo *K-means*.

Figura 17 - Comparação de resultados de diferentes execuções do algoritmo Fuzzy C-means, aplicado em dados indefinidos.



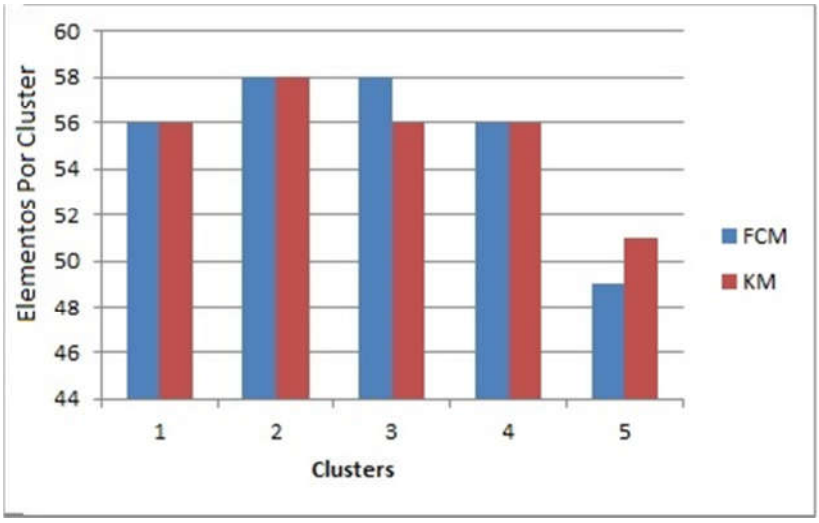
Fonte: Arquivo Pessoal.

Figura 18 - Resultados de execução do algoritmo Fuzzy C-means utilizando os dados dos comandos gestuais.



Fonte: Arquivo Pessoal.

Figura 19 - Comparação entre os resultados dos diferentes algoritmos.



Fonte: Arquivo Pessoal

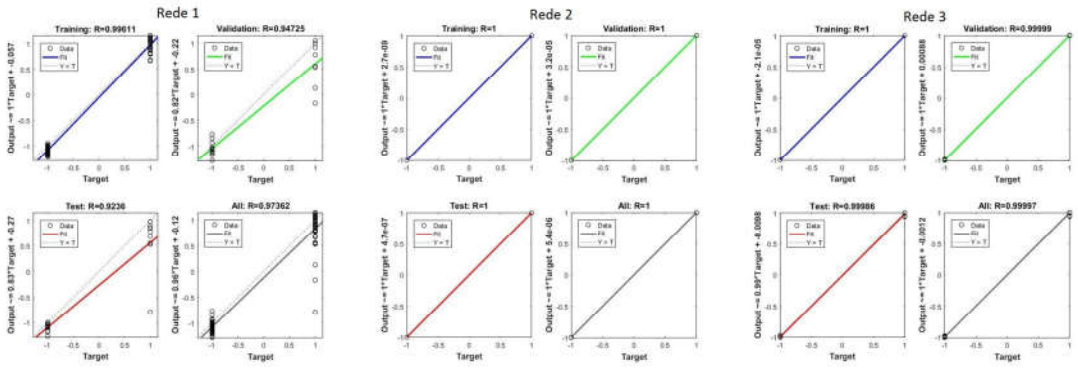
4.1.3 Utilizando Redes Neurais

Para realizar o treinamento de uma rede neural foi preciso definir o numero de amostras para realizar o treinamento da rede neural e o numero de amostras para realizar a validação dos resultados apresentados pela rede. A forma de definir esses parâmetros é empírica, ou seja, foi necessário escolher um numero inicial de amostras para, á partir dos resultados, foi definido se era necessário mais dados para treinamento ou se o numero de mostras escolhido já era o suficiente, com o erro médio quadrático e velocidade da rede neural para apresentação de seus resultados, foi possível verificar se seria necessária uma maior quantidade de amostras ou menor para melhorar o desempenho da rede neural. O valor definido de amostras foi de 210 onde 138 são para treinamento e 72 para validação da rede neural.

Para desenvolvimento da pesquisa foram definidos três gestos distintos, sendo 70 amostras de cada gesto. Para se obter uma melhor performance do sistema, foram treinadas três redes distintas para cada gesto, ou seja, cada rede foi treinada somente com um tipo de gesto. No momento que se adicionam dados para reconhecimento, duas redes irão identificar como gesto desconhecido e um irá identificar o gesto corretamente e enviar o respectivo comando para o robô. Essa maneira de desenvolver o sistema resolveu um problema de identificação que era causado por usarmos uma rede somente.

A Figura 20 apresenta os valores de regressão do processo de treinamento das três redes neurais.

Figura 20 - Dados de Regressão de treinamento das três redes neurais.



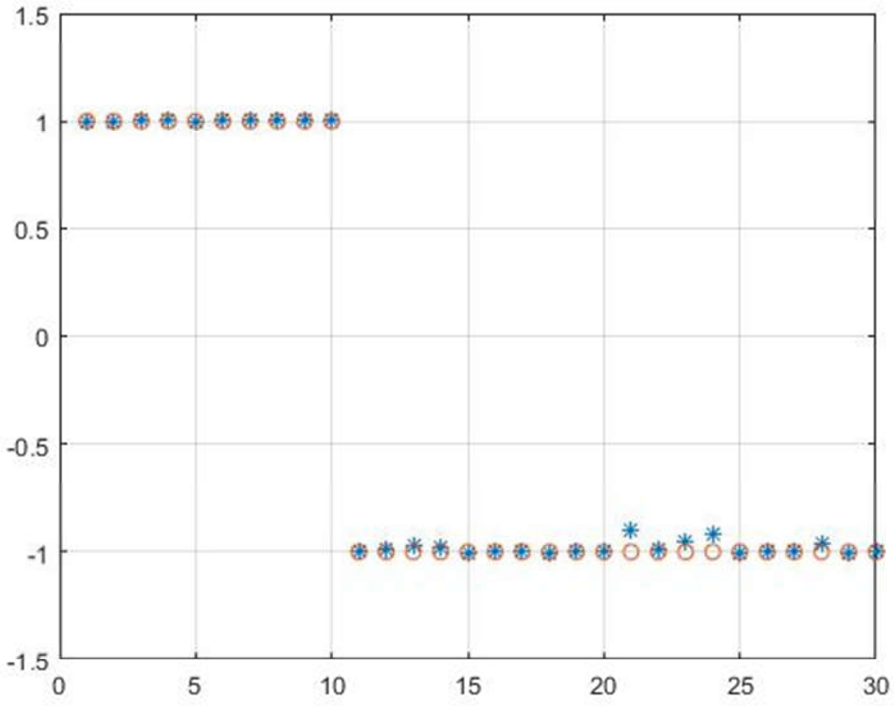
Fonte: Arquivo Pessoal.

A partir de então foi realizado um novo teste com novas amostras dos gestos utilizados para verificar o desempenho do sistema e obter o valor de acurácia do mesmo. Foram coletadas mais 15 amostras, 5 de cada gesto e aplicados no sistema para serem identificados.

A Figura 21 apresenta um exemplo de resultado vindo de uma das redes neurais. No seguinte gráfico podemos identificar no eixo X os 15 gestos aplicados à rede. No eixo Y podemos verificar a qual classe ele pertence. As circunferências representam os valores reais de cada gesto, ou seja, a qual classe ele realmente pertence e os asteriscos representam o resultado da rede neural, ou seja, a qual classe cada gesto pertence, segundo a rede neural em questão.

Para verificar o valor de acurácia das redes neurais foram feitas dez execuções do sistema. A partir de então foi verificado o valor de acurácia de cada rede neural a partir da taxa de acertos de cada rede neural. Para obter o valor final de cada rede foi feita a média aritmética entre as taxas de acerto das dez execuções. Tendo esses valores foi verificado o valor de acurácia do sistema realizando a média aritmética entre os valores de acurácia de cada rede neural.

Figura 21 - Resultado de execução das redes neurais para um gesto específico.



Fonte: Arquivo Pessoal.

O Quadro 2 apresenta os valores e acurácia do sistema.

Quadro 2 - Valores de acurácia de cada rede neural e do sistema em geral

Rede Neural	**Acurácia**
Rede 1	78,8%
Rede 2	86,5%
Rede 3	92,1%
Sistema	85,8%

Fonte: Arquivo Pessoal.

A pesquisa realizada neste TCC resultou na escrita de um artigo apresentado no Workshop Escola de Informática Teórica (WEIT), realizado em Santa Maria – RS, em 2017 e também na escrita de um resumo expandido apresentado no Congresso Regional de Iniciação Cientifica em Engenharia (CRICTE), realizado em Ijuí – RS, em 2017. Os dois trabalhos

publicados trataram do desenvolvimento do sistema de reconhecimento gestual utilizando os algoritmos de agrupamento de dados *K-means* e *Fuzzy C-means*.

CONCLUSÃO

A partir deste trabalho foi desenvolvido um sistema de reconhecimento gestual aplicado a um robô móvel. Primeiramente foi realizado um estudo sobre algoritmos de agrupamento de dados visando verificar a viabilidade do sistema verificando as necessidades para se ter um sistema eficiente e confiável. Os algoritmos utilizados foram o *K-means* e *Fuzzy C-means*. Posteriormente foi estudada a utilização da técnica de redes neurais no sistema com o intuito de se obter um sistema com mais robustez e capacidades, assim podendo utilizar gestos mais complexos.

No processo de desenvolvimento foi possível conhecer ferramentas que permitissem maior praticidade na manipulação do sensor Kinect e utilização de suas bibliotecas. Foi possível aprimorar o conhecimento e aplicação em redes neurais, extendendo conteúdos apresentados em aula, permitindo dar a capacidade de extender os estudos com técnicas mais atuais na área de aprendizado de maquina e redes neurais.

O sistema gerou resultados satisfatórios. O algoritmo *Fuzzy C-means* apresentou resultados com uma maior consistência, comparando varias execuções com os mesmos dados, fenômeno que não ocorre com o algoritmo *K-means*. Porém durante o estudo foram utilizados gestos bem definidos, com isso os algoritmos apresentaram resultados semelhantes, onde concluímos que, nestas condições, nenhum algoritmo se sobressai em relação ao outro em desempenho. A dificuldade de se trabalhar com gestos mais complexos foi resolvida com a aplicação de redes neurais para reconhecimento do gesto, isso possibilitou também um a maior robustez e capacidade.

Também se pode concluir que o sistema encontra dificuldades em gestos com características semelhantes o que aumenta as chances de erro no momento da tomada de decisão do sistema. Esse problema possui um potencial de se ser resolvido com estudos futuros, onde se pretende utilizar técnicas de *Deep learning*.

REFERÊNCIAS

ALIZADEH, A. Gesture Recognition based on Hidden Markov Models from Joints' Coordinates of a Depth Camera for Kids age of 3-8. Helsinki, 2014

BO, L.; REN, X.; FOX, D. Unsupervised Feature Learning for RGB-D Based Object Recognition.[S.l.], 2012.

CORREA, D. S. O. Mobile Robots Navigation in Indoor Using Kinect Sensor. São Carlos – SP, 2012.

CRUZ, L.; LUCIO, D.; VELHO, L. Kinect na RGB Images: Challenges and Applications. Rio de Janeiro, 2012. p.4.

FACURE, M. Funções de Ativação: Entendendo a Importância da Ativação Correte nas Redes Neurais. [S. l.], 2017. Disponível em: https://matheusfacure.github.io/2017/07/12/activ-func/. Acesso em: 08 dez. 2017.

GAMARRA, D. F. T.; CUADROS, M. A. DE S. L. Comparissom of Fuzzy C Means, K-means and K-medoids for Clustering in the Bag of Words Algorithm. [S.l.], 2015.

HEICKAL, H.; ZHANG, T.; HASANUZZAMAN. Real-time 3D Full Body Motion Gesture Reconition. Shenzhen, China, 2013.

JUNIOR, A. C. Os algoritmos Fuzzy C-means, Robust-Prototypes e Unsupervised Robust C-Prototypes aplicados à uma base de dados das bacias hidrográficas da Região de Criciúma. [S.l.], 2012.

KAJAN, S.; PERNECKÝ, D.; HAMMAD, A. Hand Gesture Recognition Using Multilayer Perceptron Network. Bratislava, 2015.

LOWE, D. G. Object Recognition from Local Scale-Invariant Features. Vancouver, 1999.

MATAS, J.; OBDRZÁLEK, S. Object Recognition Methods Based on Transformation Covariant Features. Praga, 2004.

MATHWORKS. Dinamic Time Warping. [S. l.], 2016. Disponível em: https://www.mathworks.com/help/signal/ref/dtw.html. Acesso em: 08 dez. 2017.

MONOLITO NIMBUS. Perceptron – redes neurais. [S.l.], 2017. Disponível em: https://www.monolitonimbus.com.br/perceptron-redes-neurais/. Acesso em: 22 dez. 2017.

PAL, M.; SAHA, S.; KONAR, A. A Fuzzy C-means Clustering Approach for Gesture Recognition in Healthcare. India, 2014.

PANCHAL, J. B.; KANDORIYA, K. P. Hand Gesture Recognition Using Clustering Based Technique. Gujarat, 2013.

PATSADU, O.; NUKOOLKIT, C.; WATANAPA, B. Human Gesture Recognition Using Kinect Camera. Bangkok, 2012.

PONTÍFICIA UNIVERSIDADE CATÓLICA. Faculdade de Informática.Sockets. [Porto Alegre], 1998. Disponível em: http://www.inf.pucrs.br/~fldotti/redes/982/sockets.htm. Acesso em: 22 nov. 2017.

PRASS, F. Algoritmo de K-means. [S. l.], 2013. Disponível em: http://fp2.com.br/blog/index.php/2013/algoritmo-de-k-means/ . Acesso em : 08 dez. 2017.

REAS, C.; FRY, B. Processing: A Programming Handbook for Visual Designers, Second Edition. The MIT Press. [S. l.], 2014.

RIMKUS, K. 3D Human Hand Motion Recognition System. Polônia, 2013.

ROMERO, R. A. F. Robótica Móvel. 1 ed. [S.l.]: LTC, 2014.

TISSOT, H. C.; CAMARGO, L. C.; POZO, A. T. R. Treinamento de Redes Neurais Feedforward: Comparativo dos Algoritmos Backpropagation e Diffenrential Evolution. Curitiba, 2012.

YONAMINE, F. S. Aprendizado não supervisionado em domínios fuzzy – algoritmo fuzzy c-means. São Carlos, 2002.

ZHANG, Q. Dynamic Hand Gesture Segmentation Method Based on Unequal-probabilities Background Difference and Improved FCM Algorithm. China, 2015.

APENDICE A – CÓDIGO PROCESSING PARA OBTENÇÃO DE DADOS DOS MOVIMENTOS GESTUAIS

```
import SimpleOpenNI.*;
import processing.net.*;

int alvo = 0;
SimpleOpenNI kinect;
PrintWriter output;
Server s;
int amostras = 1;

void setup() {
  size(650, 490);
  kinect = new SimpleOpenNI(this);
  kinect.enableDepth();
  kinect.enableUser();
  output = createWriter("posicao.txt");
  frameRate(12);
  s = new Server(this, 12345);
}

void draw() {
  //---------- MONTAGEM DA IMAGEM ----------
  strokeWeight(2);
  stroke(5);
  smooth();
  kinect.update();
  background(#808080);
  image(kinect.depthImage(), 5, 5);
  //---------- OBTENÇÃO DO JOGADOR ----------
  IntVector userList = new IntVector();
  kinect.getUsers(userList);
  if (userList.size() > 0) {
    //---------- OBTENÇÃO DA POSIÇÃO DO JOGADOR ----------
    int userId = userList.get(alvo);
    PVector position = new PVector();
    kinect.getCoM(userId, position);
    kinect.convertRealWorldToProjective(position, position);
    fill(#800000);
    ellipse((position.x), (position.y), 30, 30);
    if ( kinect.isTrackingSkeleton(userId)) {
      //---------- OBTENÇÃO DA POSIÇÃO DA MÃO DIREITA ----------
      PVector rightHand = new PVector();
      PVector rightElbow = new PVector();
      PVector rightShoulder = new PVector();
      float confidencehand = kinect.getJointPositionSkeleton(userId, SimpleOpenNI.SKEL_LEFT_HAND, rightHand);
      float confidenceelbow = kinect.getJointPositionSkeleton(userId, SimpleOpenNI.SKEL_LEFT_ELBOW, rightElbow);
      float confidanceshouder = kinect.getJointPositionSkeleton(userId, SimpleOpenNI.SKEL_LEFT_SHOULDER, rightShoulder);
      PVector convertedRightHand = new PVector();
      PVector convertedRightElbow = new PVector();
      PVector convertedRightShoulder = new PVector();
      kinect.convertRealWorldToProjective(rightHand, convertedRightHand);
      kinect.convertRealWorldToProjective(rightElbow, convertedRightElbow);
      kinect.convertRealWorldToProjective(rightShoulder, convertedRightShoulder);
      line(convertedRightHand.x, convertedRightHand.y, convertedRightElbow.x, convertedRightElbow.y);
      line(convertedRightElbow.x, convertedRightElbow.y, convertedRightShoulder.x, convertedRightShoulder.y);
      fill(#00FF00);
```

```
      ellipse(convertedRightShoulder.x, convertedRightShoulder.y, 20, 20);
      fill(#0000FF);
      ellipse(convertedRightHand.x, convertedRightHand.y, 20, 20);
      fill(#FFFF00);
      ellipse(convertedRightElbow.x, convertedRightElbow.y, 20, 20);
      line(convertedRightHand.x, convertedRightHand.y, convertedRightElbow.x, convertedRightElbow.y);
      line(convertedRightElbow.x, convertedRightElbow.y, convertedRightShoulder.x, convertedRightShoulder.y);

      output.println((convertedRightHand.x - position.x)+"\t"+(convertedRightHand.y - position.y)+"\t"+(convertedRightHand.z - position.z));
      println((convertedRightHand.x - position.x)+"X"+(convertedRightHand.y - position.y)+"X"+(convertedRightHand.z - position.z));

      //---------- ENVIO DOS DADOS VIA ETHERNET----------
      s.write((short)(convertedRightHand.x*0.39));
      println((short)(convertedRightHand.x*0.39));
    }
   }
}

void keyPressed(){
   if(key == 'c'){
     output = createWriter("amostrasKmeFCMII.txt");
     amostras++;
   } else if(key == 'f'){
     output.flush();
     output.close();
   }
}

void onNewUser(SimpleOpenNI curContext, int userId)
{
  println("onNewUser - userId: " + userId);
  println("\tstart tracking skeleton");

  kinect.startTrackingSkeleton(userId);
}

void onLostUser(SimpleOpenNI curContext, int userId)
{
  println("onLostUser - userId: " + userId);
}

void onVisibleUser(SimpleOpenNI curContext, int userId)
{
  println("onVisibleUser - userId: " + userId);
}
```

APENDICE B – CÓDIGO MATLAB DE RECONEHCIMENTO GESTUAL UTILIZANDO O ALGORITMO K-MEANS

```
clear all; close all; clc
m = ([42.5; 127.5; 212.5; 0]);
t = tcpclient('localhost', 12345);
s = serial('COM4');
fopen(s);

while(true)
    if t.BytesAvailable ~= 0
        comando = read(t, 1);
        m(4) = comando;
        [idx,C] = kmeans(m,3,'Start', [42.5; 127.5; 212.5]);
        if idx(4) == 1
             fprintf(s,'D');
             disp(comando);
             disp('um');
        elseif idx(4) == 2
             fprintf(s,'S');
             disp(comando);
             disp('dois');
        elseif idx(4) == 3
             fprintf(s, 'L');
             disp(comando);
             disp('três');
        end
    end
end
```

APENDICE C - CÓDIGO MATLAB DE RECONEHCIMENTO GESTUAL UTILIZANDO UMA REDE NEURAL

```
clear all;
close all;
clc;

mX = load('matX.mat');
mY = load('matY.mat');
mZ = load('matZ.mat');

ccg = [321.39 297.78 1644.32];

idx = randperm(210);
idxtr = idx(1:138);
idxte = idx(139:210);

traindata = zeros(201, 138);
traindata(1:67, :) = mX.matX(:, idxtr) - ccg(1);
traindata(68:134, :) = mY.matY(:, idxtr) - ccg(2);
traindata(135:201, :) = mZ.matZ(:, idxtr) - ccg(3);
ntraindata = zeros(201, 138);
[ntraindata(1:67, :), nminx, nmaxx] = premnmx(traindata(1:67, :));
[ntraindata(68:134, :), nminy, nmaxy] = premnmx(traindata(68:134, :));
[ntraindata(135:201, :), nminz, nmaxz] = premnmx(traindata(135:201, :));

output = zeros(1, 210);
output(1, 71:140) = ones(1, 70);
output(1, 141:210) = ones(1, 70)*2;
[noutput, omin, omax] = premnmx(output);

outputtrain = noutput(1, idxtr);

testdata = zeros(201, 72);
testdata(1:67, :) = mX.matX(:, idxte) - ccg(1);
testdata(68:134, :) = mY.matY(:, idxte) - ccg(2);
testdata(135:201, :) = mZ.matZ(:, idxte) - ccg(3);
ntestdata = zeros(201, 72);
ntestdata(1:67, :) = tramnmx(testdata(1:67, :), nminx, nmaxx);
ntestdata(68:134, :) = tramnmx(testdata(68:134, :), nminy, nmaxy);
ntestdata(135:201, :) = tramnmx(testdata(135:201, :), nminz, nmaxz);

outputtest = noutput(1, idxte);

net1 = feedforwardnet(10);
net1 = configure(net1, ntraindata, outputtrain);

net1 = train(net1, ntraindata, outputtrain);
y1 = sim(net1, ntraindata);

y2 = sim(net1, ntestdata);

p = 1:1:138;
figure(1)
plot(p, outputtrain, 'o', p, y1, '*');
%
```

```
q = 1:1:72;
figure(2)
plot(q, noutput(1, idxte), 'o', q, y2, '*');
```

APENDICE D – ARTIGOS PUBLICADOS DURANTE A PESQUISA

Utilização dos Algoritmos de *K-means* e *Fuzzy C-means* para Controle de um robô móvel por meio de um sensor *Kinect*

Evaristo J. do Nascimento[1], Marco A. S. L. Cuadros[2], Daniel F. T. Gamarra[3],

[1]Engenharia de Computação – Universidade Federal de Santa Maria (UFSM), CEP 97.105-900 – Santa Maria – RS – Brazil

[2]Instituto Federal do Espirito Santo (IFES), CEP 29.173-087 – Serra – ES - Brasil

[3]Departamento de Processamento de Energia Elétrica – Universidade Federal de Santa Maria (UFSM)

evaristo.nascimento@ecomp.ufsm.br, marcoantonio@ifes.edu.br, daniel.gamarra@ufsm.br

Abstract. *This work presents a gestural recognition system development using the Kinect sensor, applied for the control of a mobile robot. The system receives gestures information and send commands to a robot to realize specific movements desired by a user. A user realize predetermined gestures the robot receives that gestural information through Kinect sensor, the robot has a system that should interpret this gesture using clustering data algorithms such as fuzzy CMeans and KMeans, after recognize the gesture read, the robot executes the task that corresponds to the realized gesture. The results present a same resemblance about both algorithms, which present a high accuracy of hits of the system.*

Resumo. *O trabalho apresenta o desenvolvimento de um sistema de reconhecimento de gestos utilizando o sensor Kinect, aplicado em um robô móvel. O sistema é capaz de receber informações gestuais de um usuário, e enviar comandos para um robô para que realize movimentos especificos desejados por um usuário. Este usuário realiza gestos predeterminados, o robô recebe essa informação gestual através do sensor Kinect, o robô possui um sistema capaz de interpretar esse gesto utilizando algoritmos de agrupamento dados de Fuzzy C Means e K Means, após reconhecer o gesto lido, o robô executa a tarefa referente ao comando gestual recebido. O artigo descreve a arquitetura implementada com resultados experimentais.*

1. Introdução

O uso da tecnologia nos permite facilidades em muitas situações onde temos dificuldades. Dificuldades essas causadas por alguma deficiência, ou problema físico gerado por algum problema de saúde ou um acidente. Para ajudar-nos nessas situações, estudam-se muitas ferramentas que possam nos trazer mais facilidade e praticidade em atividades que precisam ser realizadas.

O trabalho desenvolvido neste artigo visa a desenvolver um sistema inteligente que permita o controle de um robô móvel à partir de comandos gerados por gestos de uma pessoa, os quais são obedecidos pelo robô fazendo com que ele execute movimentos específicos dependentes de qual comando foi recebido.

A técnica a ser empregada para a classificação dos gestos de uma pessoa é o agrupamento ou clusterização de dados através do algoritmo de *K-means* e do algoritmo *Fuzzy C-means*. Os referidos algoritmos vão permitir classificar os gestos captados por um sensor de movimento, verificar qual comando foi recebido e qual é a tarefa, associada ao comando, que deverá ser realizada pelo robô. A utilização de algoritmos de agrupamento de dados também possibilita desenvolvimento de um sistema mais simples e uma alta eficiência devido a simplicidade dos algoritmos. O trabalho também tem como objetivo comparar as técnicas que serão utilizadas na pesquisa e verificar a viabilidade do uso de cada uma.

Na literatura existem diversas aplicações utilizando as técnica de agrupamento de dados de *K-Means* e *Fuzzy CMeans* para reconhecimento de gestos. Q. Zhang (2015) utiliza o algoritmo de *Fuzzy C-means* para resolver problemas de reconhecimento gestual em local onde o fundo do ambiente, dificulta o contraste entre a pessoa e o ambiente, dificultando a análise do processo pelas técnicas de processamento de imagens; Pal (2014), aplica a técnica para reconhecimento de gestos de pessoas com deficiência muscular, possibilitando, a partir destas informações, que um profissional de saúde possa realizar diagnósticos de uma possível evolução de uma doença que cause problemas físicos em seus pacientes; Panchal e Kandoriya (2013) utilizam a técnica para reconhecimentos gestuais de uma mão fixa, identificando as posições dos dedos e classificando-os em diferentes gestos e Gamarra (2015), utiliza a técnica de agrupamento de dados, utilizando os algoritmos *K-means, Fuzzy C-means* e *K-medoids*, comparando o desempenho de diversos algoritmos de clusterização na aplicação da técnica de *Bag of Visual Words* para reconhecimento de objetos.

2. Fundamentação Teórica

2.1 Algoritmo *K-means*

O Algoritmo de K-means é muito utilizado na área de *machine learning* e é classificado como algoritmo de agrupamento de dados. Ele é utilizado para separação de dados em grupos, buscando classifica-los. Inspirando-se na descrição de Gamarra (2015), o algoritmo funciona basicamente definindo os centroides que identificam cada grupo a ser formado e, utilizando a equação (1), são calculadas as distancias entre cada dado a ser classificado e os centroides, onde o grupo em que o centroide é o mais próximo do dado é o grupo ao qual este dado pertence.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - v_k\|^2 \quad (1)$$

Na equação acima r_{nk} é uma variável binária que indica que determinado valor x_n pertence a determinado *cluster* k, onde recebe 1 quando a distância entre o valor e um centroide é menor que a distância com os outros centroides. Caso contrário r_{nk} é 0.

Após verificar a qual cluster determinado valor pertence, os centroides são recalculados usando a seguinte equação:

$$v_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \quad (2)$$

2.2 Algoritmo *Fuzzy C-means*

O algoritmo *Fuzzy C-means*, é uma variação do algoritmo de *K-means*, onde ele possibilita tratar incertezas, ou seja, dados que possuem características de dois diferentes grupos são classificados a partir de um "grau de pertinência" em cada grupo (Yonamine, 2002). O grau de pertinência descreve quanto de semelhança ou proximidade, determinado elemento tem de cada grupo. Indicando que o grupo ao qual este determinado dado pertence é aquele o qual possui maior grau de pertinência.

Uma característica muito importante que dever ser verificada é que a soma dos graus de pertinência de um dado a cada grupo, tem de ser igual a 1.

A equação (3) descreve o processo de classificação:

$$J_{FCM} = \sum_{n=1}^{N} \sum_{k=1}^{c} u_{ik}^{m} \|x_i - v_k\|^2 \quad (3)$$

Onde u_{ik}^{m} representa o grau de pertinência do i-ésimo elemento em relação ao k-ésimo cluster. E m é o define a taxa de evolução do algoritmo até chegar a um valor de erro tolerante.

Para recalcular os centroides após cada interação usamos a seguinte equação:

$$v_{ik}^{l} = \frac{\sum_{k=1}^{N} \left(u_{ik}^{l-1}\right) x_k}{\sum_{n=1}^{N} \left(u_{ik}^{l-1}\right)}, 1 \leq i \leq c \quad (4)$$

3. Robô Móvel

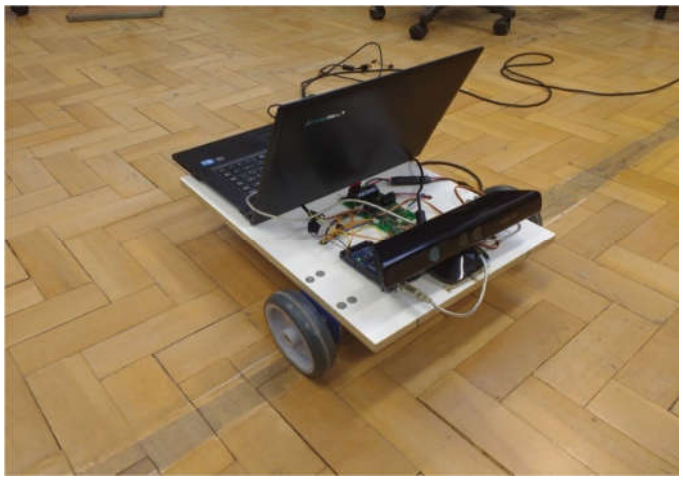


Figura 1: robô móvel utilizado

A pesquisa utilizou como plataforma experimental um robô móvel com três rodas, duas rodas são tracionadas por um motor em cada roda, e uma terceira roda independente auxilia o equilíbrio do robô. O robô utiliza dois servomotores que trabalham a uma tensão aproximadamente de 24 Volts. Eles são comandados por um driver md49 o qual envia os comandos de direção e velocidade aos motores. Os movimentos do robô móvel são controlados por um microcontrolador *Arduino* Mega 2560 (Atmega 2560) o qual é

programado através da linguagem C pela IDE de desenvolvimento do *Arduino*. Um sensor Kinect também é fixado na parte superior do robô. A figura 1 mostra o robô móvel utilizado durante os experimentos.

3.1. O sensor *Microsoft Kinect*

Para a leitura dos gestos que serão interpretados e posteriormente traduzidos em comandos para o robô móvel será utilizado o sensor de movimentos da *Microsoft Kinect*. O sensor *Kinect*, como é popularmente chamado permite que possamos capturar movimentos de uma pessoa obtendo as coordenadas X, Y, Z das juntas de uma pessoa, assim como também sua imagem.

Figura 2: Sensor Kinect

Para a leitura de movimento, o *Kinect* possui um sensor de profundidade e uma câmera RGB. O sensor de profundidade consiste em um emissor infravermelho que emite uma matriz de pontos que, através do princípio da reflexão, permite que seja possível obter-se a distância (coordenada Z) de algum corpo que esteja a sua frente. A figura 2 mostra o sensor Kinect.

4. Setup Experimental

O sistema está constituído além do robô móvel e um sensor Kinect, de um ambiente de programação *Processing*, criado pelo MIT (*Massachusetts Institute of Technology*). O *Procesing* é um software que possibilita o desenvolvimento de projetos de processamento de imagens tanto para leigos em programação como para programadores avançados [Reas e Fry 2007]. Também foi utilizado o software MATLAB, da empresa *Mathworks* para o desenvolvimento do sistema inteligente para a classificação dos gestos baseado nas técnicas de *clustering*. O ambiente *Processing* é baseado em linguagem Java e uma de suas utilidades é permitir manipulação de sensores, e fazer relações entre os dados recebidos com sons e imagens. Um dos sensores que se pode trabalhar no *Processing* é o *Microsoft Kinect*, sendo então uma ótima ferramenta a ser

utilizada na pesquisa. No *Processing* foi desenvolvida a obtenção de dados do *Kinect*, e a preparação dos dados para serem enviados para o MATLAB. O sistema inteligente de tomadas de decisão baseado nos algoritmos *Kmeans* e *Fuzzy C-means*, que foi implementado no MATLAB, onde é realizada a clusterização dos dados enviados e, sua associação com os respectivos comandos do robô móvel mediante o microcontrolador *Arduino*. A Figura 3 mostra o resultado da captura de uma imagem com as juntas do usuário com o sensor *Kinect* feitas pelo processing.

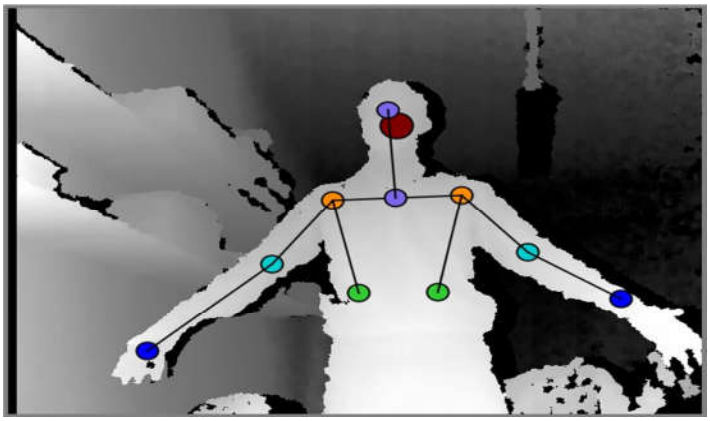


Figura 3: Imagem com as juntas do usuário capturada pelo Kinect

A figura 4 apresenta os gestos que foram utilizados para a realização da pesquisa, os quais podem ser vinculados a diversos comandos através de programação, assim, sendo livre no momento do desenvolvimento a tarefa a ser definida para cada comando gestual.

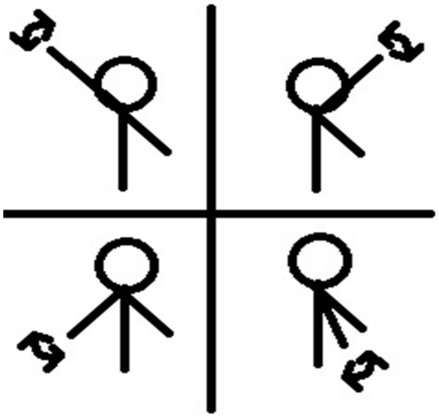


Figura 4: Gestos utilizados para realização do estudo.

A figura 5 mostra o funcionamento do sistema como um todo. Primeiramente o *Kinect* reconhece o indivíduo e monta o mapa das principais juntas do corpo da pessoa. A partir de então ele obtém as informações de posição das juntas (no nosso experimento a mão

esquerda) e envia as informações para o MATLAB. O MATLAB realiza o processo de classificação do gesto recebido e então envia o respectivo comando ao robô para que realize a atividade desejada.

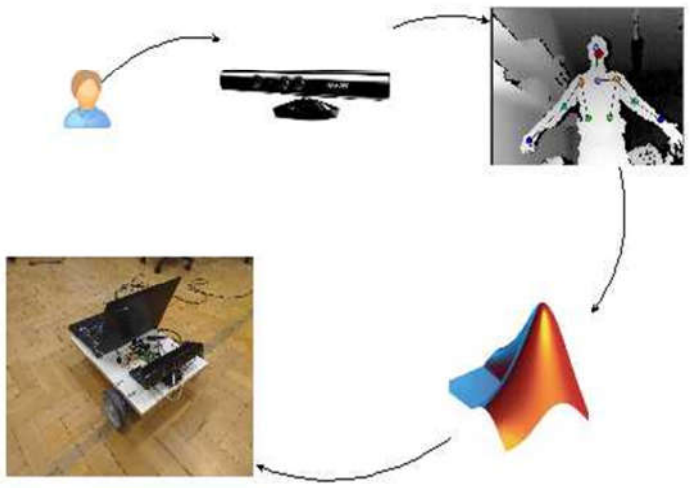


Figura 5: Esquema de Funcionamento do Sistema com o robô móvel.

5. Resultados

O setup experimental foi montado com o robô móvel, e foi feita a integração com o sensor Kinect e o software de classificação de gestos utilizando os algoritmos de clustering no Matlab. Representando este sistema integrado, o primeiro resultado tangível deste projeto.

O sistema reconhece a região da imagem lida pelo *Kinect.* A região da imagem é identificada e associada a um determinado movimento que será executado pelo robô. Os comandos referentes para cada região de leitura do Kinect foram definidos na implementação do sistema. Os algoritmos trabalham com matrizes bidimensionais de dados, onde o número de colunas define o número de coordenadas de cada posição lida de uma junta.

O algoritmo de *clustering* recebe as coordenadas X, Y e Z da posição onde se encontra a mão esquerda do usuário (direita na visão do kinect), e essas informações são enviadas ao sistema construído no MATLAB para serem classificados. As figuras 6 e 7 apresentam os resultados de classificação utilizando os algoritmos de *K*-means e o *Fuzzy C*-means respectivamente.

As imagens mostram um processo de *clustering* dos dados das imagens do *kinect* em cinco grupos. Cada região de dados mais estrema significa um comando gestual diferente e o grupo central representa um gesto nulo. As marcações em "X" representam os valores centroide de cada grupo.

Os dois algoritmos se comportam de uma forma muito semelhante apresentando variações somente em regiões de fronteira entre os grupos. Neste espaço de amostra os algoritmos apresentam algumas dificuldades de interpretação dos dados nas fronteiras entre o grupo central e o inferior direito, o que pode ocasionar erros no momento de execução do sistema. Isso se deve ao fato de que o sistema está levando em conta

variações em profundidade (Eixo "Z") e esses dados apresentam características de profundidade semelhantes aos dados do seu grupo vizinho. A figura 8 apresenta a comparação do resultado apresentado por cada algoritmo.

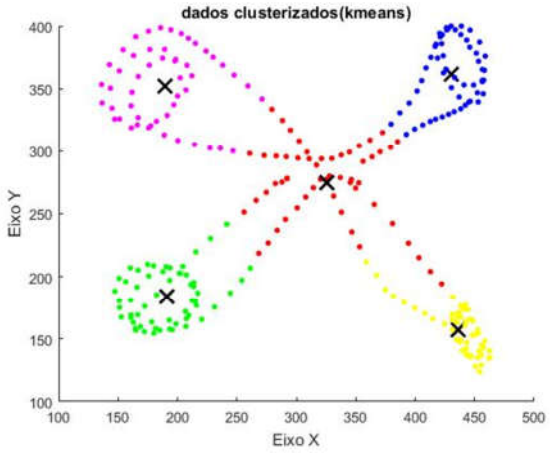


Figura 6: Resultados do *clustering* obtidos utilizando *K-means*

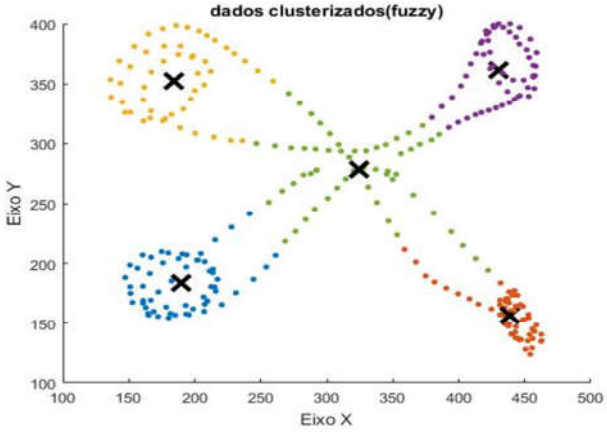


Figura 7:Resultados do *clustering* obtidos utilizando o algoritmo *Fuzzy C-means*

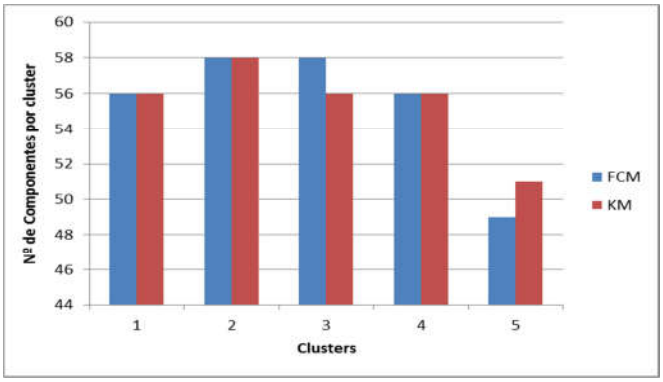


Figura 8: Comparação entre os resultados dos dois algoritmos.

6. Conclusão

O trabalho desenvolvido neste artigo tem como principais resultados, a criação de um sistema que integra o comando dos movimentos do robô mediante o uso de um sensor Kinect para reconhecimento de gestos de um usuário. O referido sistema utilizou diversas tecnologias, as quais foram integradas para o controle de movimentos do robô. Um segundo aporte do artigo é a implementação de um sistema que mediante o uso de técnicas de clustering como o *K-means* e *Fuzzy C-means. O sistema* consegue processar e discriminar gestos diferentes de uma pessoa baseado na extração das informações das juntas do usuário. Foi possível também obter dados sobre as diferentes técnicas utilizadas na pesquisa possibilitando uma análise de desempenho de cada algoritmo e verificando a viabilidade do uso dos mesmos. Os resultados possuem uma característica satisfatória contando com a viabilidade de utilizar algoritmos de agrupamentos de dados para identificação de comandos gestuais. Porém a técnica nos limita a leituras de pontos ou regiões onde determinado gesto é executado na imagem recebida pelo *Kinect* e em trabalhos futuros poderiam incluir o uso de redes neurais para as tarefas de classificação com maior complexidade.

Referências

Zhang, Q., Lu, J., Wei, H., Zhang, M., Duan, H., "Dynamic hand gesture segmentation method based on unequal-probabilities background difference and improved FCM Algorithm", In: International Journal of Innovative Computing, Information and Control, 2015.

Pal, M., Saha, S., Amit, K., "A Fuzzy C Means Clustering Approach for Gesture Recognition in Healthcare", In: International Journal of Enhanced Research in Science Technology & Engeneering, 2014.

Panchal, B. J., Kandoriya, K. P., "Hand Gesture Recognition Using Clustering Based Technique", In: International Journal of Science and Research, 2013.

Gamarra, D. F. T., Cuadros, M. A. de S. L., "Comparisson of Fuzzy C Means, K-Means and K-MEdoids for Clustering in the Bag of Visual Words Algorithm", In: Biblioteca Digital Brasileira de Computação, 2015.

Cruz, L., Lucio, D., Velho, L. (2012) "Kinect and RGBD Images: Challenges and Applications", In: SIBGRAPI-T, Conference on Graphics, Patterns and Images Tutorials. p 36-49.

Alizadeh A. (2014) "Gesture Recognition based on Hidden Markov Models from Joints' Coordinates of a Depth Camera for Kids age of 3–8", University of Helsinki.

Yonamine, F. S., Specia, L., Carvalho, V. O., Nicoletti, M. C. (2002) "Aprendizado não supervisionado em domínios fuzzy – algoritmo fuzzy c-means", Universidade Federal de São Carlos.

Reas, C., Fry, B., "Processing: A programming Handbook for Visual Designers and Artists", Massachusetts Institute of Technology, 2007.

CRICTE 2017
XXVIII Congresso Regional de Iniciação Científica e Tecnológica em Engenharia
UNIJUÍ UNIVERSIDADE REGIONAL

UTILIZAÇÃO DOS ALGORITMOS DE K-MEANS E FUZZY C-MEANS PARA CONTROLE DE UM ROBÔ MÓVEL MEDIANTE O USO DE UM SENSOR DE PROFUNDIDADE

Evaristo J. do Nascimento
Acadêmico do curso de Engenharia de Computação, Universidade Federal de Santa Maria
evaristo.nascimento@ecomp.ufsm.br
Daniel F. T. Gamarra
Professor do Departamento de Processamento de Energia Elétrica, Universidade Federal de Santa Maria
daniel.gamarra@ufsm.br

Resumo. *O trabalho apresenta o desenvolvimento de um sistema de reconhecimento de gestos utilizando o sensor de profundidade Kinect, aplicado em um robô móvel. O sistema é capaz de receber informações gestuais de um usuário, e enviar comandos para um robô para que realize movimentos especificas desejada por um usuário. A informação gestual é capturada medianteo sensor Kinect, o robô possui um sistema capaz de interpretar esse gesto utilizando algoritmos de agrupamento dados de Fuzzy C-Means e K-Means, após reconhecer o gesto lido, o robô executa a tarefa referente ao comando gestual recebido. O artigo descreve a arquitetura implementada com resultados experimentais.*

Palavras-chave: *Kinect, Reconhecimento Gestual, Agrupamento de Dados.*

1. INTRODUÇÃO

O trabalho desenvolvido neste artigo visa desenvolver um sistema inteligente que permita o controle de um robô móvel a partir de comandos gerados por gestos de uma pessoa, os quais são obedecidos pelo robô fazendo com que ele execute movimentos específicos dependentes de qual comando foi recebido.

As técnicas a ser empregada para a classificação dos gestos de uma pessoa é o agrupamento de dados através do algoritmo de *K-means* e do algoritmo *Fuzzy C-means*. As referidas técnicas vão permitir classificar os gestos captados por um sensor de movimento, verificar qual comando foi recebido e qual é a tarefa, associada ao comando, que deverá ser realizada pelo robô. O trabalho também tem como objetivo comparar as técnicas que serão utilizadas na pesquisa e verificar a viabilidade do uso de cada uma.

Na literatura existem diversas aplicações utilizando as técnica de agrupamento de dados de *K-Means* e *Fuzzy C Means* para reconhecimento de gestos. Q. Zhang [1] utiliza o algoritmo de *Fuzzy C-means* para resolver problemas de reconhecimento gestual em local onde o fundo do ambiente, dificulta o contraste entre a pessoa e o ambiente, dificultando o analise do processo pelas técnicas de processamento de imagens; Pal [2], aplica a técnica para reconhecimento de gestos de pessoas com deficiência muscular, possibilitando, a partir destas informações, que um profissional de saúde possa realizar diagnósticos de uma possível evolução de uma doença que cause problemas físicos em seus pacientes; Panchal e Kandoriya [3] utilizam a técnica para reconhecimentos gestuais de uma mão

fixa, identificando as posições dos dedos e classificando-os em diferentes gestos e Gamarra [4], utiliza a técnica de agrupamento de dados, utilizando os algoritmos *K-means*, *Fuzzy C-means* e *K-medoids*, comparando o desempenho de diversos algoritmos de clusterização na aplicação da técnica de *Bag of Visual Words* para reconhecimento de objetos.

2.FUNDAMENTAÇÃO TEÓRICA

2.1 Algoritmo *K-means*

O Algoritmo de *K-means* é muito utilizado na área de *machine learning* utilizado para separação de dados em grupos, buscando classifica-los. Tomando como referência o trabalho de Gamarra em [4], o algoritmo funciona basicamente definindo os centroides que identificam cada grupo a ser formado recalculando as distancias entre cada dado a ser classificado e os centroides, onde o grupo em que o centroide é o mais próximo do dado é o grupo ao qual este dado pertence.

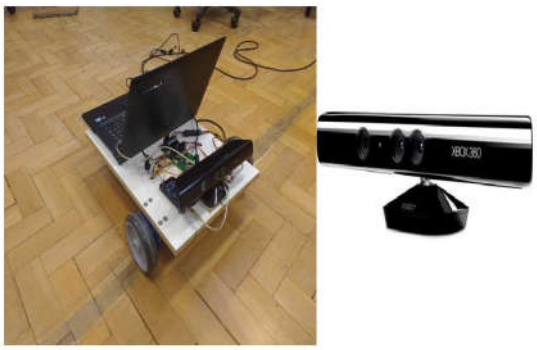
2.2 Algoritmo *Fuzzy C-means*

O algoritmo *Fuzzy C-means*, é uma variação do algoritmo de K-means, onde ele possibilita tratar incertezas, ou seja, dados que possuem características de dois diferentes grupos. Estes são classificados a partir de um "grau de pertinência" de cada grupo (Yonamine [5]). O grau de pertinência descreve o quanto de semelhança ou proximidade, determinado elemento tem de cada grupo, indicando que o grupo ao qual este determinado dado pertence é aquele que o que possui maior grau de pertinência.

3. SETUP EXPERIMENTAL

O sistema está constituído por um robô móvel com três rodas, duas rodas são tracionadas por um motor em cada roda, e uma terceira roda independente auxilia o equilíbrio do robô; Um sensor de profundidade *Kinect* permite que possamos capturar movimentos de uma pessoa obtendo as coordenadas X, Y, Z das juntas de uma pessoa, assim como também sua imagem. Ambos o robô e o Kinect são mostrados na figura 1.

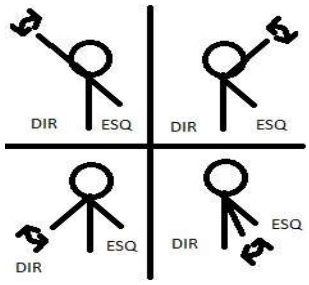
Figura 1. Robô móvel e Sensor *Kinect* utilizados



Para obtenção dos dados do *Kinect* é utilizado o ambiente de programação *Processing*, criado pelo MIT (*Massachusetts Institute of Technology*). O *Processing* é um software que possibilita o desenvolvimento de projetos de processamento de imagens utilizando *Java*, *Python* ou *JavaScript* (Reas e Fry [7]). Também foi utilizado o software MATLAB, da empresa *Mathworks* para o desenvolvimento do sistema inteligente para a classificação dos gestos baseado nas técnicas de *clustering* utilizando os algoritimos *K-means* e *Fuzzy C-means*, e também é feita a associação de cada gesto com cada comando para ser enviado ao robô. No *Processing* foi desenvolvida a obtenção de dados do *Kinect*, e a preparação dos dados para serem enviados para o MATLAB. O processamento dos comandos é realizado mediante o microcontrolador *Arduino* (ATMEGA 2560) fixado no robô.

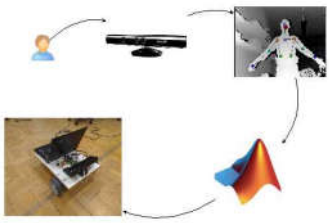
A figura 2 apresenta os gestos que foram utilizados para a realização da pesquisa.

Figura 2: Gestos utilizados para realização do estudo.



A figura 3 mostra o funcionamento do sistema como um todo. Primeiramente o *Kinect* reconhece o indivíduo e monta o mapa das principais juntas do corpo da pessoa. A partir de então ele obtém as informações de posição das juntas (no nosso experimento a mão esquerda) e envia as informações para o MATLAB. O MATLAB realiza o processo de classificação do gesto recebido e então envia o respectivo comando ao robô para que realize a atividade desejada.

Figura 3. Esquema de Funcionamento do Sistema com o robô móvel



4.RESULTADOS

O funcionamento do sistema consiste em reconhecer a região da imagem lida pelo *Kinect*. A região da imagem é identificada e associada a um determinado movimento que será executado pelo robô. Os comandos referentes para cada região de leitura do *Kinect* foram definidos na implementação do sistema. Os algoritmos trabalham com matrizes bidimensionais de dados, onde o número de colunas define o número de coordenadas de cada posição lida de uma junta.

O algoritmo de *clustering* recebe as coordenadas X, Y e Z da posição onde se encontra a mão esquerda do usuário (direita na visão do *kinect*), e essas informações são enviadas ao sistema construído no MATLAB para serem classificados. As figuras 4 e 5 apresentam os resultados de classificação utilizando os algoritmos de *K-means* e o *Fuzzy C-means* respectivamente.

As imagens mostram um processo de *clustering* dos dados das imagens do *kinect* em cinco grupos. Cada região de dados mais estrema significa um comando gestual diferente e o grupo central representa um gesto nulo. As marcações em "X" representam os valores centroide de cada grupo.

Os dois algoritmos se comportam de uma forma muito semelhante apresentando variações somente em regiões de fronteira entre os grupos. Neste espaço de amostra os algoritmos apresentam algumas dificuldades de interpretação dos dados nas fronteiras entre o grupo central e o inferior direito. Nos grupos 1, 2 e 4 os dois algoritmos sem comportam de forma muito semelhante, apresentando os mesmos resultados, compondo esses grupos com 56, 58 e 56 elementos respectivamente. Já com os grupos 3 e 5, devido a algumas incertezas com dados de fronteira, há uma variação na composição dos grupos onde que com *K-means* os grupos possuem 56 e 51 dados respectivamente e com o *Fuzzy C-means* os grupos possuem 58 e 49 elementos respectivamente. Esse fenômeno pode ocasionar erros no momento de execução do sistema. Isso se deve ao fato de que o sistema está levando em conta variações em profundidade (Eixo "Z") e esses dados apresentam características de profundidade semelhantes aos dados do seu grupo vizinho. A figura 6 apresenta a comparação do resultado apresentado por cada algoritmo.

Figura 4. Resultados do *clustering* obtidos utilizando *K-means*

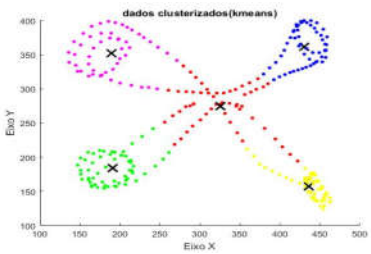


Figura 5. Resultados do *clustering* obtidos utilizando *Fuzzy C-means*

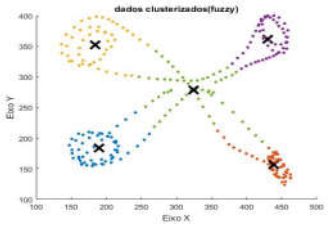
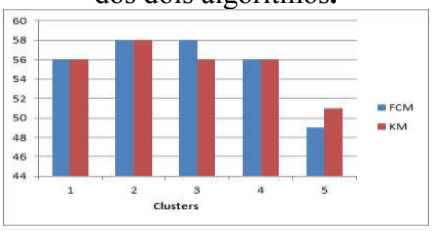


Figura 6: Comparação entre os resultados dos dois algoritmos**.**



5. CONCLUSÃO

O trabalho desenvolvido neste artigo tem como principais resultados, a criação de um sistema que integra o comando dos movimentos do robô mediante o uso de um sensor *Kinect* para reconhecimento de gestos de um usuário. O referido sistema utilizou diversas tecnologias as quais foram integradas para o controle de movimentos do robô. Um segundo aporte do artigo é a implementação de um sistema que mediante o uso de técnicas de *clustering* como o *K-means* e *Fuzzy C-means*. O sistema consegue processar e discriminar gestos diferentes de uma pessoa baseado na extração das informações das juntas do usuário. Foi possível também obter dados sobre as diferentes técnicas utilizadas na pesquisa possibilitando uma análise de desempenho de cada algoritmo e verificando a viabilidade do uso dos mesmos. As técnicas utilizadas para identificação dos gestos tiveram resultados muito semelhantes o que não nos possibilita dizer se um algoritmo é melhor que o outro. Para isso é necessário realizar novos experimentos com dados que possuem diferentes características e comportamento o que está a ser considerado numa futura evolução do estudo. Trabalhos futuros poderiam incluir o uso de redes neurais para as tarefas de classificação.

6. REFERÊNCIAS

[1] Zhang, Q., Lu, J., Wei, H., Zhang, M., Duan, H., "Dynamic hand gesture segmentation method based on unequal-probabilities background difference and improved FCM Algorithm", In: International Journal of Innovative Computing, Information and Control, 2015.

[2] Pal, M., Saha, S., Amit, K., "A Fuzzy C Means Clustering Approach for Gesture Recognition in Healthcare", In: International Journal of Enhanced Research in Science Technology & Engeneering, 2014.

[3] Panchal, B. J., Kandoriya, K. P., "Hand Gesture Recognition Using Clustering Based Technique", In: International Journal of Science and Research, 2013.

[4] Gamarra, D. F. T., Cuadros, M. A. de S. L., "Comparisson of Fuzzy C Means, K-Means and K-MEdoids for Clustering in the Bag of Visual Words Algorithm", In: Biblioteca Digital Brasileira de Computação, 2015.

[5] Yonamine, F. S., Specia, L., Carvalho, V. O., Nicoletti, M. C. (2002) "Aprendizado não supervisionado em domínios fuzzy – algoritmo fuzzy c-means", Universidade Federal de São Carlos.

[6] Cruz, L., Lucio, D., Velho, L. (2012) "Kinect and RGBD Images: Challenges and Applications", In: SIBGRAPI-T, Conference on Graphics, Patterns and Images Tutorials. p 36-49.

[7] Reas, C., Fry, B., "Processing: A programming Handbook for Visual Designers and Artists", Massachusetts Institute of Technology, 2007.