UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI - UFSJ

CAMPUS ALTO PARAOPEBA

ENGENHARIA MECATRÔNICA

DESENVOLVIMENTO DE UM SISTEMA PARA OPERAÇÃO DE UM MANIPULADOR ROBÓTICO BASEADO EM VISÃO COMPUTACIONAL

Autor: José Marques de Oliveira Júnior

Orientador: Michel Carlo Rodrigues Leles

OURO BRANCO/MG, MARÇO DE 2018

RESUMO

Este trabalho propõe o desenvolvimento de um sistema para operação de um manipulador robótico baseado em visão computacional. As imagens são capturadas por meio do sensor *Microsoft Kinect®*. Uma vez adquiridas, as imagens representativas dos gestos são classificadas através de uma técnica de Aprendizado de Máquina conhecida como Máquina de Vetores de Suporte (tradução livre para *Support Vector Machines* - SVMs). Na sequência, os comandos são enviados para o manipulador robótico por comunicação serial, utilizando o MATLAB® como interface. Os resultados obtidos demonstraram que o sensor *Microsoft Kinect®* torna mais fácil a manipulação e o tratamento das imagens e também que o modelo SVM foi eficiente, atingindo o objetivo de classificar corretamente os gestos e gerar comandos para o manipulador.

1. INTRODUÇÃO

Substituir operadores humanos por robôs em tarefas arriscadas ou em ambientes hostis representa uma evolução na automatização de processos industriais e domésticos [1]. Os desafios impostos são inúmeros e as abordagens são fortemente interdisciplinares [2]. Entretanto, a complexidade envolvida em muitas atividades ainda requer a participação ativa de especialistas humanos. Neste caso, robôs controlados em procedimentos de inspeção e monitoramento de ambientes e estruturas de difícil acesso ao ser humano são ainda de fundamental importância. Isso requer robôs que atuem no mundo real possuam autonomia suficiente para execução de tarefas como desvio de obstáculos, controle de trajetória e capacidade de planejamento de movimento, mas ainda respeitando a comandos dos seres humanos [3].

O campo da Telerrobótica visa prover a esses especialistas uma experiência de interação remota com o ambiente por meio de robôs da forma mais natural possível [4]. O grau de autonomia de um sistema robótico deve ser ajustado de acordo com as especificações da missão ou situação considerada [5]. Para tanto, no presente trabalho, são utilizadas duas ferramentas, a saber: i) Visão Computacional; e ii) Aprendizado de Máquinas. Ambos descritos a seguir.

Visão Computacional pode ser definida como a área de conhecimento que estuda como computadores podem processar imagens digitais extraindo informações e tomando algum tipo de decisão [6]. Dentre as suas principais funcionalidades, incluem-se, mas não são restritas a: aquisição, processamento, análise e reconhecimento de algum padrão em imagens digitais. Tang [7] mostra como esse conceito pode ser aplicado no reconhecimento de faces. Pisharady & Saerbeck [8] fornecem uma revisão recente de métodos para reconhecimento de gestos baseado em visão computacional, enquanto Ruffieux et al. [9] discutem um método para segmentação temporal e identificação de gestos. Pillajo & Sierra [10] apresentam uma Interface Homem-Máquina baseada no sensor *Microsoft Kinect*® que permite ao usuário operar um braço robótico SCARA.

Aprendizado de Máquina é uma área da Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática [11]. As aplicações dessas técnicas tem sido frequentes em Visão Computacional. Nolker & Ritter [12] utilizaram Redes Neurais Artificiais para reconhecimento de gestos. Beluco [13] aplicou uma rede Multi-Layer Perceptron para classificação de imagens de sensoriamento remoto. Shiba et al. [14] avaliaram o desempenho de árvores de decisão para rotulação de imagens de satélites. Krizhevsky et al. [15] apresentaram um significativo trabalho de classificação de imagens utilizando *Convolutional Neural Networks.* O uso das Máquinas de Vetores de Suporte (SVMs, tradução livre para *Support Vector Machines*) é recorrente na detecção, segmentação e classificação de objetos em imagens [16] [17] [18].

As SVMs constituem uma técnica que vem recebendo crescente atenção da comunidade de Aprendizado de Máquina (AM) [19]. São embasadas pela teoria do aprendizado estatístico, desenvolvida por Vapnik [20]. Em uma tarefa de segregação, o objetivo da SVM é encontrar um hiperplano que separe as classes pela maior margem possível. Na Figura 1 abaixo é mostrado um exemplo de classificação de dados que possuem duas dimensões.

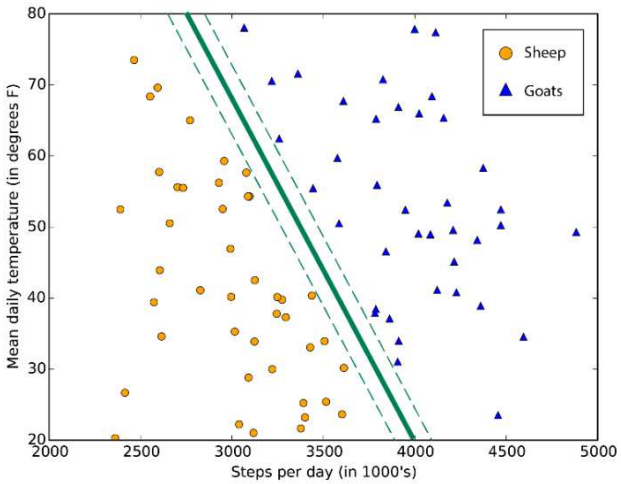


Figura 1 - Exemplo de separação feita por uma SVM objetivando maximizar a margem. Fonte: Como escolher algoritmos de Aprendizado de Máquina[1].

Em problemas de dados de grande dimensão em que outros algoritmos tendem a se tornar super ou sub ajustados (*overfitting* e *underfitting* [21] [22]), as SVMs apresentam robustez visto que são necessários menos parâmetros (também chamados de pesos) para configurá-las. Outra característica atrativa é a convexidade da questão de otimização formulada em seu treinamento, que implica na existência de um único mínimo global [23]. Essa é uma vantagem das SVMs sobre, por exemplo, as Redes Neurais Artificiais MLP [24] [25], em que há mínimos locais na função objetivo minimizada. Além disso, o uso de funções Kernel na não-linearização das SVMs torna o algoritmo eficiente, pois permite a construção de simples hiperplanos em um espaço de alta dimensão de forma tratável do ponto de vista computacional [26].

No tipo de aprendizado supervisionado (utilizado neste trabalho), dado um conjunto de exemplos rotulados na forma (x_i, y_i), em que x_i representa um exemplo e y_i denota o seu rótulo, deve-se produzir um classificador, também denominado modelo, preditor ou hipótese, capaz de predizer precisamente o rótulo de novos dados [23]. Essa parte do processo é denominada treinamento. Após a criação do modelo, deve-se verificar seu desempenho, classificando amostras não utilizadas no treinamento e comparando tais resultados com os rótulos verdadeiros. As SVMs apresentam boa capacidade generalização, isto é, geram bons resultados ao prever corretamente a classe de dados não vistos anteriormente pelo algoritmo.

[1] Disponível em: <https://docs.microsoft.com/pt-br/azure/machine-learning/studio/algorithm-choice>. Acesso em março de 2018.

Este projeto propõe a utilização de visão computacional para operação de um manipulador robótico (efetuando-se alguns movimentos padronizados), a partir do reconhecimento de alguns gestos pré-estabelecidos do operador, que são capturados por um sensor *Microsoft Kinect*®. O mesmo pode, portanto, ser utilizado em situações em que um operador pode (a distância e em segurança) determinar os movimentos que um robô deve efetuar.

2. MÉTODOS

Na Fig. 2 abaixo é mostrado o fluxograma de todo o processo:

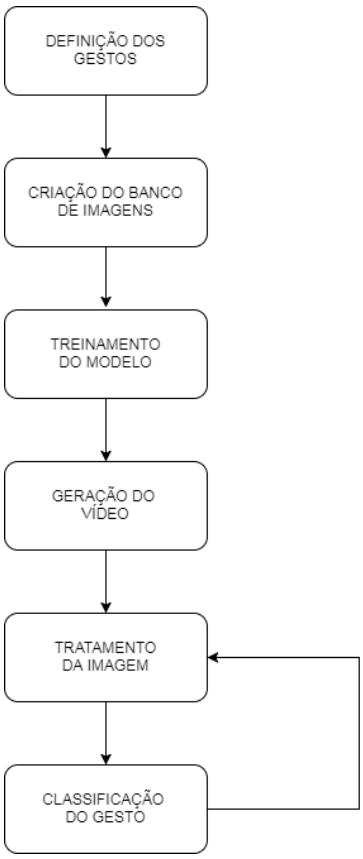


Figura 2 - Fluxograma do algoritmo para reconhecimento de gestos.

O ciclo do programa se inicia com o treinamento do modelo SVM, prosseguindo para a exibição do vídeo, obtenção e tratamento da imagem do gesto e finalmente sua classificação, repetindo os dois últimos passos. Nas subseções seguintes são detalhadas cada etapa desse processo.

O Anexo I - Tutorial para Comunicação - contém as informações necessárias para o funcionamento e acesso aos dados fornecidos pelo *Microsoft Kinect*® por meio do MATLAB®.

2.1 Definição dos gestos

O passo inicial consiste na definição dos gestos e a resposta pretendida. Foram definidos três tipos de movimentos a serem reconhecidos pelo sistema: i) abertura e fechamento da mão; ii) contagem do número de dedos na mão; e iii) ângulo formado entre o braço e o antebraço.

Dessa forma, três tipos básicos são obtidos: i) um booleano (mão aberta ou fechada); ii) um inteiro (número de dedos na mão); e iii) um ponto flutuante (ângulo entre braço e antebraço). Portanto, em possíveis casos de continuidade do projeto, pode-se utilizar estes algoritmos como base, restando apenas a adaptação em cada situação específica. Os métodos para identificação de mão aberta ou fechada e para contagem do número de dedos são os mesmos; já para o cálculo do ângulo entre o braço e o antebraço foi utilizada outra estratégia.

2.2 Criação do banco de imagens

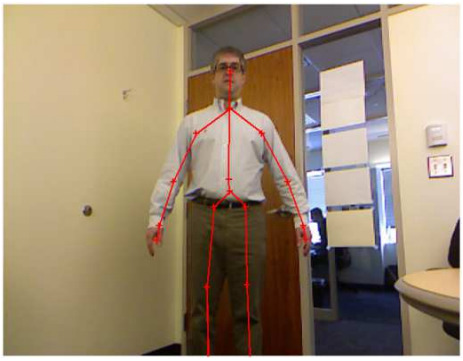
O passo seguinte é a criação do banco de imagens. Visto que a abordagem utilizada com Máquinas de Vetor de Suporte (SVMs) é um tipo de Aprendizado Supervisionado, é necessário a criação de um conjunto de imagens rotuladas (isto é, associadas às classes previamente definidas) para treinamento e teste do modelo. Essa etapa pode ser dividida em duas partes distintas: a obtenção das imagens propriamente ditas e a preparação dos dados para os passos seguintes.

A obtenção do banco de imagens segue o algoritmo mostrado na Figura 3. Basicamente, configura-se uma quantidade de imagens desejadas, inicia-se a comunicação com o Kinect e, durante o laço de repetição *while*, segmenta-se a região de interesse, sendo esta tratada e salva no diretório desejado. O fragmento relativo à segmentação e ao tratamento da imagem é idêntico na classificação em tempo real (mais detalhes na subseção *Tratamento da Imagem* e no Anexo II - Códigos, que contém os códigos utilizados neste trabalho).

Programa *Principal_gerar_imagens:* Criação do banco de imagens

```
qte_imagens = n;    % quantidade de imagens desejadas
iniciar_kinect();   % ver Figura 2 do Anexo II
cont = 0;           % contador auxiliar de imagens
while(true)
    meta_dados = mostrar_video();                          % ver Figura 3 do Anexo II
    if (pessoa_localizada)
        limites = gerar_roi(meta_dados);                   % ver Figura 4  do Anexo II
        imagem = tratar_imagem(limites,meta_dados);    % ver Figura 5 do Anexo II
        salvar_imagem(imagem);                             % ver Figura 6 do Anexo II
        cont = cont + 1;    % incrementando o contador
        if (cont >= qte_imagens)
            break                                          % interrompe o laço while
        end if
    end if
end while
```

Figura 3 - Código-fonte do programa *Principal_gerar_imagens.*



jointIndices =

318	256
317	240
318	143
319	92
271	177
239	243
219	303
216	323
363	177
399	243
421	303
424	322
296	277
286	387
288	492
286	520
340	277
342	384
347	493
350	522

jointCoordinates =

-0.0119	-0.0072	1.9716
-0.0107	0.0545	2.0376
-0.0051	0.4413	2.0680
0.0033	0.6430	2.0740
-0.1886	0.3048	2.0469
-0.3130	0.0472	2.0188
-0.3816	-0.1768	1.9277
-0.3855	-0.2448	1.8972
0.1724	0.3022	2.0449
0.3102	0.0382	2.0304
0.3740	-0.1929	1.9591
0.3786	-0.2625	1.9356
-0.0942	-0.0850	1.9540
-0.1367	-0.4957	1.9361
-0.1356	-0.8765	1.9339
-0.1359	-0.9284	1.8341
0.0683	-0.0871	1.9504
0.0706	-0.4822	1.9293
0.0858	-0.8804	1.9264
0.0885	-0.9321	1.8266

Figura 4 - Dados fornecidos pelo Microsoft Kinect®: índices e coordenadas das articulações. Fonte: *Using the Kinect® for Windows® V1 from Image Acquisition Toolbox™*.[2]

A comunicação entre o Microsoft Kinect® e o MATLAB® oferece algumas facilidades para a programação. Uma delas é o fornecimento das posições das articulações da pessoa localizada no vídeo (Figura 4 acima). Cada par de dados em *jointIndices* (relativos aos pixels na imagem) e cada trio de dados em *jointCoordinates* (relativos à coordenadas globais) referem-se a posição de uma parte do corpo (marcados com uma cruz na imagem), como cabeça, ombros, cotovelos e mãos, dentre outros.

A função *gerar_roi (meta_dados)* é de extrema importância para o funcionamento correto do programa. Ela realiza a segmentação da região de interesse (também chamada de *ROI* - do inglês *Region Of Interest*), ou seja, cria uma sub-imagem contendo somente a região da mão. As informações das partes do corpo fornecidas pelo Microsoft Kinect® são usadas como referência nesta função.

É realizada uma compensação do tamanho da sub-imagem em função da distância da mão ao sensor. Assim, mesmo com diferentes posicionamentos da pessoa a ser localizada, a proporção e o tamanho da região de interesse tenderá a se manter.

Função gerar_roi (meta_dados)*:* Criação da sub-imagem contendo somente a mão

```
pixel_mao = meta_dados.JointImageIndices(12,:);                 %  posição da mão na imagem
coordenada_mao = meta_dados.JointWorldCoordinates(12,:);        % distâncias da mão ao sensor

profundidade = calcular_profundidade(coordenada_mao);           % o raio calculado será
raio = calcular_raio(profundidade);                             % relacionado ao tamanho
                                                                % da sub-imagem criada
ponto1 = pixel_mao_x - raio;                                    % delimitação da sub-imagem
ponto2 = pixel_mao_x + raio;
ponto3 = pixel_mao_y - raio;
ponto4 = pixel_mao_y + raio;
roi = [ponto1 ponto2; ponto3 ponto4];                           % retorno da função
```

Figura 5 - Código-fonte da função *gerar_roi(),* que gera a Região de Interesse. Para maiores detalhes, ver Figura 4 no Anexo II.

[2] Disponível em: <https://www.mathworks.com/help/imaq/examples/using-the-kinect-r-for-windows-r-from-image-acquisition-toolbox-tm.html>. Acesso em Março de 2018.

Na segunda parte dessa etapa (preparação dos dados) a versão MATLAB® R2017a possui um melhor desempenho, já que suas *toolboxes* tornam mais simples a rotulação de imagens, bem como sua manipulação. Para o treinamento e teste do modelo criado a partir da SVM, o conjunto de imagens obtidos deve ser transformado em matrizes (em que cada linha representa os pixels de uma imagem) e os rótulos transformados em vetores, em que cada elemento representa a classe a qual a imagem pertence.

Antes da transformação em vetores e, consequentemente, em uma matriz, é realizado o redimensionamento de todas imagens, padronizando o tamanho dos elementos gerados. Isso anula os efeitos de distorções causadas pela distância da pessoa ao sensor que não foram compensados pela função *gerar_roi()*.

Assim, são criadas a partir do banco de imagens dois grupos distintos: a matriz de treinamento e o vetor de rótulos de treinamento, usados na criação do modelo; e a matriz de teste e o vetor de rótulos de teste, utilizados para verificação do desempenho do modelo.

2.3 Treinamento do modelo

Seguindo o fluxograma da Fig. 2, vemos que a próxima etapa é o treinamento do modelo.

A Fig. 6 abaixo apresenta os conceitos referentes à criação de um classificador de forma simplificada.

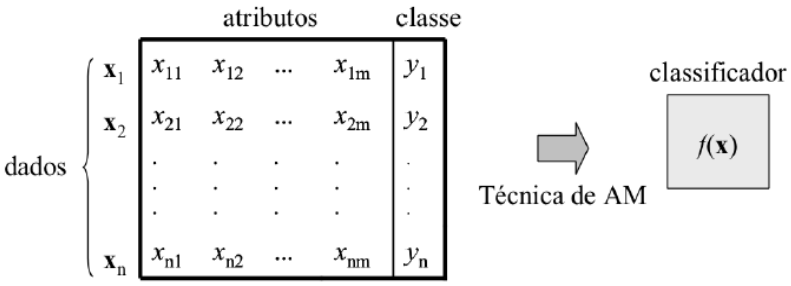


Figura 6 - Criação de um classificador utilizando Aprendizado de Máquina Supervisionado (adaptado de Lorena, A. C. & Carvalho, A. C. P. L. F., 2007).

Podemos observar que os dados são manipulados da seguinte maneira: cada elemento (no nosso caso, cada imagem) é equivalente a uma linha x_i; atrelado a essa linha há uma classe referência y_i, tida como verdadeira. A partir disso, o algoritmo traça um hiperplano separando tais classes, objetivando maximizar a margem entre as classes. Dessa forma é possível a classificação de dados ainda não vistos pelo programa de forma eficiente, formando o conceito de generalização. Isso caracteriza as chamadas Máquinas de Vetor de Suporte (SVMs).

Na classificação com múltiplas classes é utilizada a estratégia chamada *One-Versus-All* [27]. São construídos *k* modelos SVM onde *k* representa o número de classes. O enésimo modelo SVM é

treinado com todos os exemplos da enésima classe com os rótulos positivos e todos os outros exemplos com rótulos negativos [28]. Finalmente, verifica-se se cada novo elemento pertence a cada uma das classes. Abaixo é mostrado o pseudocódigo para o treinamento dos modelos.

```
Função svm_multiclasse (BancoTreinamento, RotulosTreinamento):
numClasses = contar_classes(RotulosTreinamento);          % conta a quantidade de classes
                                                          % diferentes nos dados
for k = 1 : numClasses
    OneVersusAll  = (RotulosTreinamento==u(k));           % treinando k modelos svm
    modelos(k) = treinar_svm(BancoTreinamento,OneVersusAll);
end
```

Figura 7 - Código-fonte da função *svm_multiclasse,* que cria um modelos para classificação. Para mais detalhes, ver Figura 11 do Anexo II.

Logo após a criação do classificador, utiliza-se o banco de testes para verificar a precisão do modelo. O objetivo é calcular o desempenho do modelo em imagens não vistas no treinamento.

A matriz de confusão é a forma de representação da qualidade obtida de uma classificação digital de imagem, sendo expressa por meio da correlação de informações dos dados de referência (compreendido como verdadeiro) com os dados classificados [29]. A precisão do modelo é calculada utilizando a matriz de confusão a partir da divisão entre o somatório das amostras classificadas corretamente (somatório da diagonal principal) e o somatório de todos os elementos da matriz. Inúmeras outras métricas de qualidade podem ser obtidas a partir de tal matriz, como sensibilidade, Medida F1, acurácia, especificidade, entre outras [30].

2.4 Geração do vídeo

Após o treinamento do modelo classificador, o programa segue para a aquisição das imagens em tempo real. Para melhor organização e portabilidade do código foi criada uma função chamada *'iniciar_kinect'* (Figura 2 no Anexo II). Nessa função são realizadas todas as configurações necessárias para obtenção dos dados e dos vídeos fornecidos pelo Microsoft Kinect®.

Com o objeto de aquisição de imagens configurado e inicializado, utiliza-se um laço de repetição '*while*' que se repete a cada imagem obtida pelo Microsoft Kinect®. O vídeo em cores é exibido na tela. Dentro do laço '*while*' serão executadas repetidamente as funções para tratar a imagem e classificar o gesto, conforme o fluxograma da Fig. 2. Para mais detalhes, ver Figura 3 do Anexo II.

2.5 Tratamento da imagem

Uma outra característica que facilita a manipulação dos dados com o Microsoft Kinect® é o fornecimento da imagem de profundidade, além da câmera RGB.

Figura 8 - Imagem de profundidade obtida pelo Microsoft Kinect®.

Para melhor desempenho do classificador e redução do custo computacional, optou-se por binarizar a imagem. Para isso, ao obter a região de interesse (proximidades da mão), procede-se da seguinte forma:

- obtêm-se a região de interesse (proximidades da mão) com a função *gerar_roi()*;
- normaliza-se a imagem em escala de cinza, visando minimizar possíveis interferências do segundo plano;
- utiliza-se como limiar para binarização o valor médio da imagem em escala de cinza normalizada.

Dessa forma podemos segmentar e binarizar a mão, tornando o treinamento e a classificação mais efetivos.

2.6 Classificação da imagem

A partir do vídeo obtém-se a imagem a ser classificada. Essa classificação é executada com os objetos já criados, conforme descrito na subseção *2.3 - Treinamento do modelo.*

A imagem tratada (após passar pelo processo de segmentação, normalização e binarização) é transformada em um vetor, representando o elemento a ser classificado. O resultado é exibido na tela, mostrando a qual classe a imagem obtida pertence.

Portanto, o fluxo do processo se inicia com o treinamento do modelo SVM, prosseguindo para a exibição do vídeo, obtenção e tratamento da imagem do gesto e finalmente sua classificação, repetindo os dois últimos passos. Nas subseções seguintes são detalhadas cada etapa desse processo.

2.7 Análise utilizando PCA

A Análise das Componentes Principais (PCA, do inglês *Principal Component Analysis*) é uma técnica que tem como objetivo extrair importantes informações dos dados em uma tabela multivariável e representá-las em um novo conjunto de variáveis ortogonais chamadas *componentes principais* [31], [32]. Define-se que o primeiro componente principal tem a maior variância possível (ou seja, é responsável pelo máximo de variabilidade dos dados), e cada componente seguinte, por

sua vez, tem a máxima variância sob a restrição de ser ortogonal aos componentes anteriores. Dessa forma, é possível representar os dados obtidos utilizando um número menor de dimensões sem perda de informações.

Visto que os dados obtidos neste trabalho são de alta dimensão (imagens de 70x70 *pixels*, gerando um vetor de 4900 dimensões), a visualização e o tratamento de cada elemento torna-se simplificado aplicando a técnica do PCA. Tal estratégia também foi aplicada na verificação da eficiência do modelo.

2.8 Ângulo do braço

Já para o algoritmo que calcula o ângulo do braço foram utilizadas outras técnicas (ver fluxograma na Figura 9 abaixo).

Os procedimentos para inicialização do Microsoft Kinect® e a geração do vídeo são os mesmos aplicados anteriormente. Entretanto, não há etapas de classificação nesse algoritmo. São utilizados os dados das articulações fornecidos pelo sensor, também chamados de *metadados*, da mesma forma que exemplificado na Figura 4, subseção *2.2 - Criação do banco de imagens.*

Figura 9 - Fluxograma do algoritmo para cálculo do ângulo do braço.

Dada as posições das do ombro, cotovelo e pulso, obtêm-se o ângulo formado entre eles da seguinte maneira (detalhes na Figura 14 do Anexo II - Códigos):

- cria-se dois vetores: um referente ao braço e outro referente ao antebraço, que correspondem, respectivamente, a diferença entre as posições do ombro e do cotovelo e a diferença entre as posições do cotovelo e do pulso;
- calcula-se o ângulo entre os vetores criados fazendo:

$$\theta = arctg\left(\frac{\|\boldsymbol{u} \times \boldsymbol{v}\|}{\boldsymbol{u} \cdot \boldsymbol{v}}\right)$$

onde $\|\boldsymbol{u} \times \boldsymbol{v}\|$ representa o módulo do produto vetorial entre os vetores e $\boldsymbol{u} \cdot \boldsymbol{v}$ o produto escalar entre os mesmos.

Foi criado um buffer com o intuito de suavizar mudanças bruscas na saída. Configura-se o tamanho desse buffer no programa principal e cria-se um vetor com o mesmo tamanho. O funcionamento se assemelha a uma fila: a cada imagem obtida é adicionado ao vetor o novo ângulo calculado e removido o valor mais antigo. A saída final é a média de todos os elementos de tal vetor. Dessa forma, valores espúrios tem menos relevância na saída final da função.

3. RESULTADOS

Os primeiros resultados obtidos foram os da criação do banco de imagens. Para a classificação do estado da mão (aberta ou fechada) foram utilizadas cerca de 100 imagens para cada estado. Já para a classificação dos dedos mostrados o banco de cada classe foi composto por aproximadamente 200 amostras:

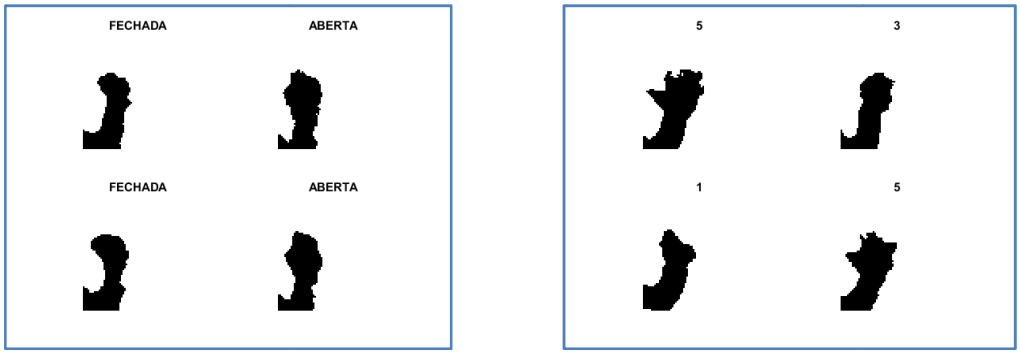


Figura 10 - Exemplos contidos no banco de imagem para classificação do gesto.

A qualidade dos modelos treinados pode ser verificada com a análise das matrizes de confusão:

Tabela 1 - Matrizes de confusão de treinamento e teste do classificador de estado: aberta ou fechada.

		Classe prevista	
		Fechada	Aberta
Classe real	Fechada	62	0
	Aberta	0	76

a) Matriz de confusão de treinamento

		Classe prevista	
		Fechada	Aberta
Classe real	Fechada	27	0
	Aberta	0	32

b) Matriz de confusão de teste

Tabela 2 - Matrizes de confusão de treinamento e teste do classificador de dedos: um, três ou cinco.

		Classe prevista		
		1	3	5
Classe real	1	132	0	0
	3	0	140	0
	5	0	0	140

a) Matriz de confusão de treinamento

		Classe prevista		
		1	3	5
Classe real	1	56	0	0
	3	0	60	0
	5	0	0	60

b) Matriz de confusão de teste

Observa-se que em nenhum dos casos houve classificações erradas, ou seja, para o banco de imagens de teste, correspondente a 30% do total, o modelo fez todas as previsões corretamente. Com isso, alcançou-se a precisão de 100%. Vale ressaltar que, com uma quantidade de dados relativamente baixa (aproximadamente entre 100 e 200 imagens por gesto), atingiu-se um bom funcionamento do sistema.

Para uma melhor análise do desempenho do código de criação dos modelos foram adicionados ruídos aleatórios ao banco de imagens. Esse experimento foi dividido em duas partes: na primeira tanto o banco de treinamento quanto o de teste apresentavam ruídos; já na segunda, apenas o banco de teste apresentava as distorções (isto é, verificou-se o desempenho em classificar imagens ruidosas quando o treinamento não continha ruídos).

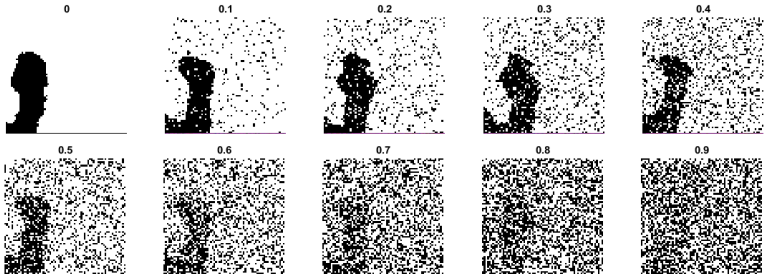


Figura 11 - Exemplos de amostras com vários níveis de ruídos para análise do desempenho.

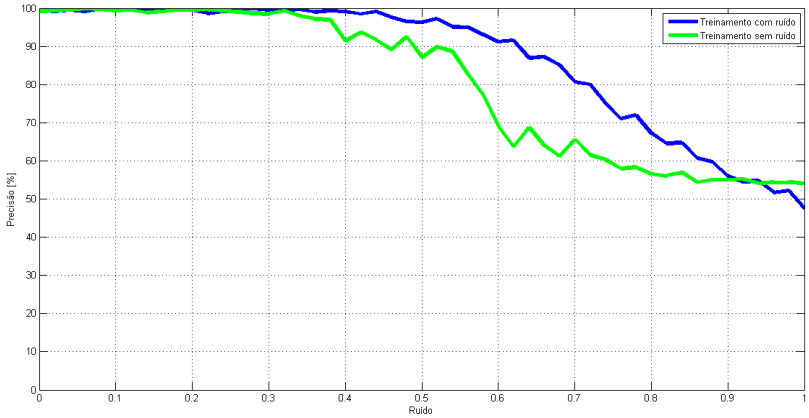


Figura 12 - Precisão obtida para o banco de testes na classificação da mão (aberta ou fechada) nos dois casos: treinamento com ruído (em azul) e sem ruído (em verde).

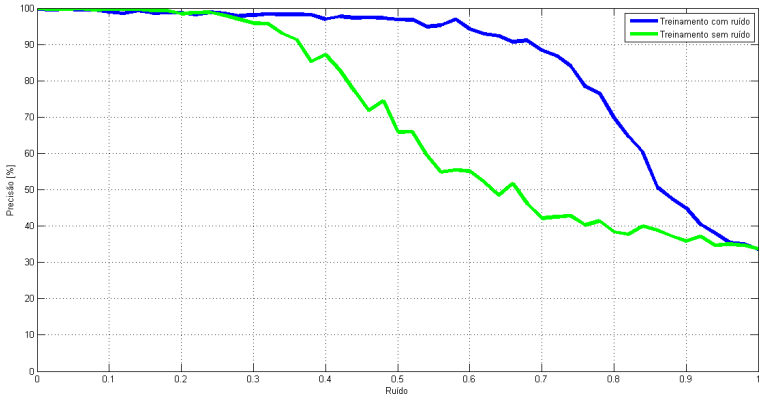


Figura 13 - Precisão obtida para o banco de testes com o classificador de gestos (número de dedos) nos dois casos: treinamento com ruído (em azul) e sem ruído (em verde).

Os dados de precisão dos modelos foram obtidos a cada incremento de 2% no ruído inserido. Para cada quantidade de ruído foram treinados 10 modelos diferentes e calculadas sua precisão. Os pontos nos gráficos representam a média da precisão destes modelos.

Como esperado, verifica-se que o aumento do ruído aplicado às amostras implica na redução do desempenho do classificador. Outra observação válida é que, quando o modelo é treinado já com as imagens distorcidas, seu resultado é superior a quando somente o banco de teste é distorcido.

Nos experimento acima descritos, a precisão para o banco de treinamento se manteve em 100%, mesmo com as imagens totalmente aleatórias, havendo uma grande diferença em relação ao resultado no banco de testes. Esse fenômeno é conhecido como sobreajuste ou *overfitting* [21] [22]. A grande dimensão dos dados tratados torna o modelo mais suscetível a este tipo de problema.

A fim de avaliar o comportamento dos bancos de dados criados, utilizou-se o PCA para redução das dimensões. Serão mostrados a seguir os gráficos de dispersão com duas e três dimensões, tanto para os bancos sem ruídos quanto para os com ruídos. Na Figura 14 são apresentados os Gráficos de dispersão do banco de imagens para classificação de gestos (aberta ou fechada) após o PCA: à esquerda, sem ruídos; à direita, com ruídos. Cada elemento representa uma imagem. Enquanto na Figura 15 os gráficos de dispersão para classificação de gestos referentes ao número de dedos.

Pela análise das Figuras 14 e 15 pode-se observar uma grande diferença na disposição entre os dados com ruídos e sem ruídos tanto no caso com duas quanto no com três classes. Intuitivamente, percebe-se uma separação mais fácil nos casos sem ruídos. Isso é demonstrado nas figuras abaixo, que mostram o desempenho do classificador em função do número de dimensões após passar pelo PCA.

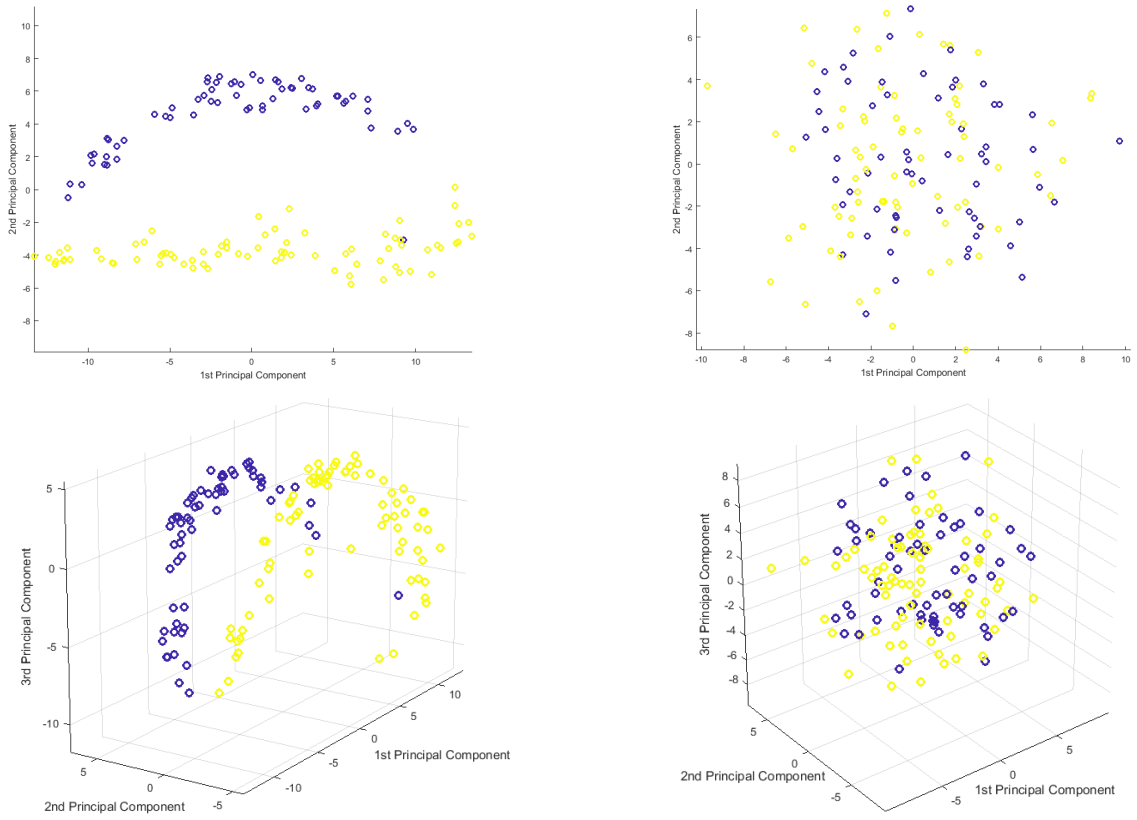


Figura 14 - Gráficos de dispersão do banco de imagens para classificação de gestos (aberta ou fechada) após o PCA: à esquerda, sem ruídos; à direita, com ruídos. Cada elemento representa uma imagem.

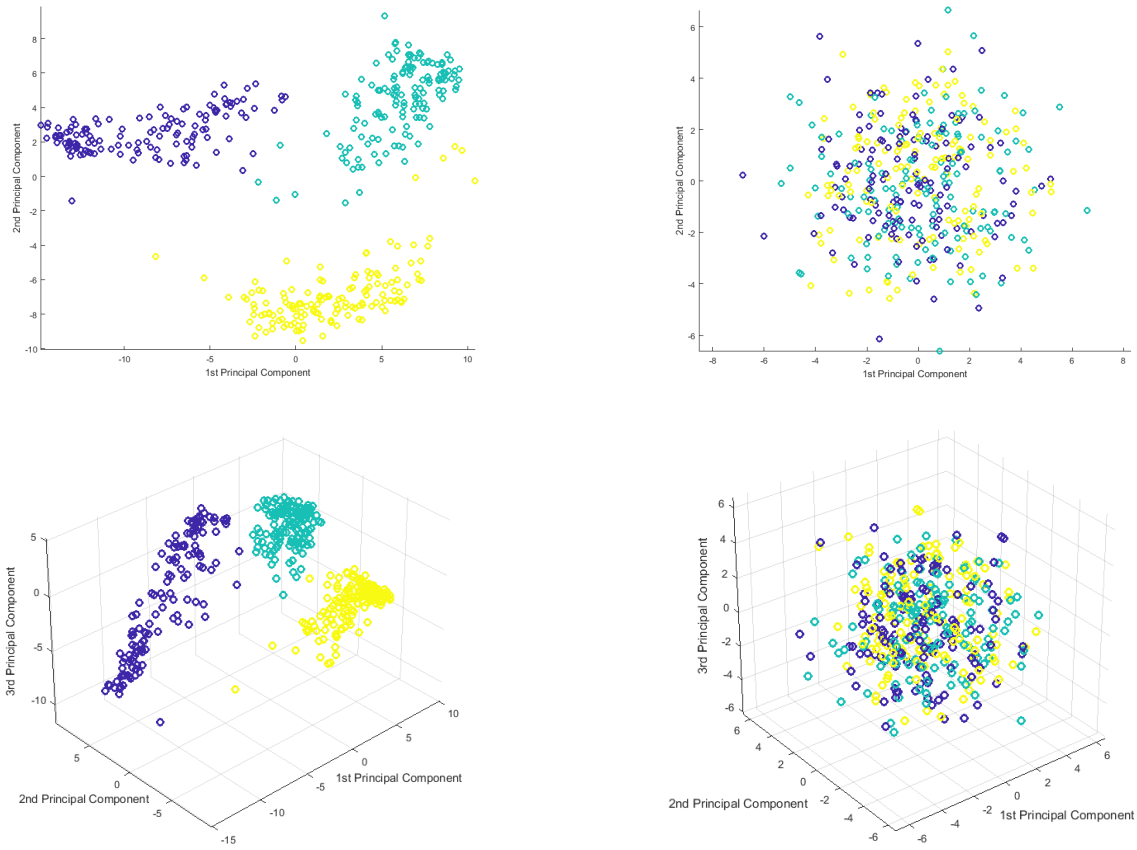


Figura 15 - Gráficos de dispersão do banco de imagens para classificação de gestos (número de dedos) após o PCA: à esquerda, sem ruídos; à direita, com ruídos. Cada elemento representa uma imagem.

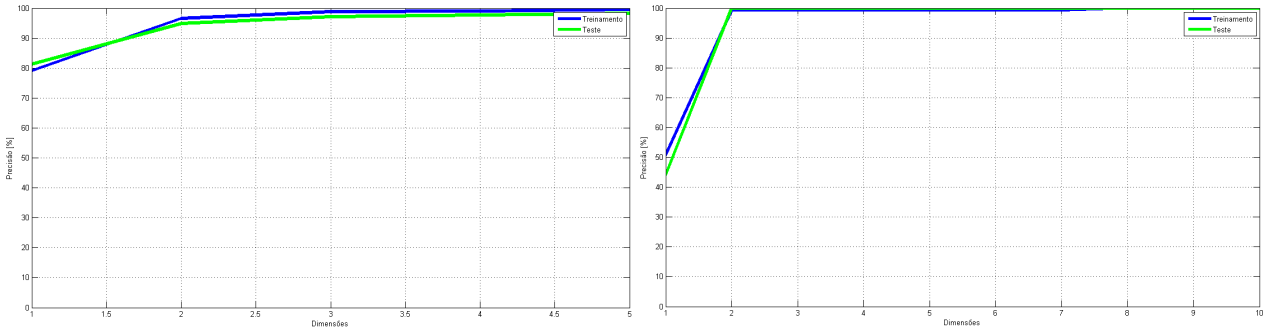


Figura 16 - Precisão do classificador em função do número de dimensões para o banco sem ruídos (em azul, precisão do banco de treinamento; em verde, precisão do banco de testes): à esquerda, o contador de dedos; à direita, o classificador de gestos (aberta ou fechada).

A Figura 16 apresenta a precisão do classificador em função do número de dimensões para o banco com e sem ruído em duas situações: na esquerda o contador de dedos, na direita o classificador de gestos. A precisão sobe rapidamente para 100% mesmo com poucas dimensões (em torno de três). Isso acontece em ambos os bancos de imagens (treinamento e teste).

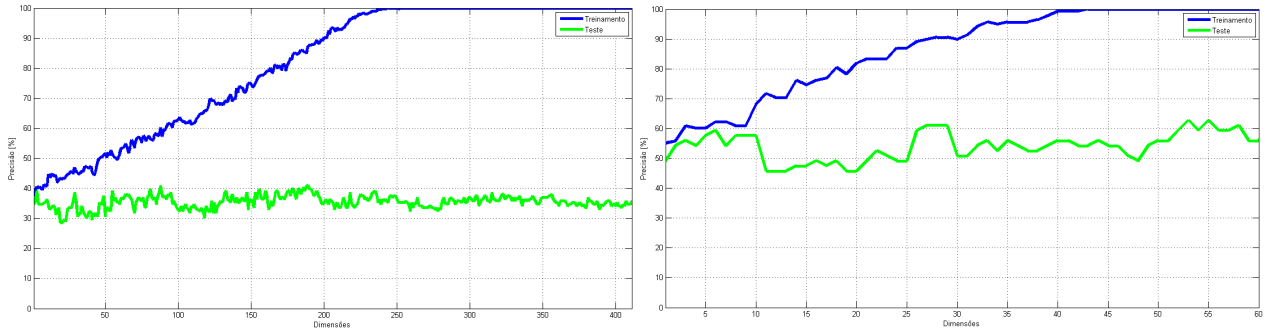


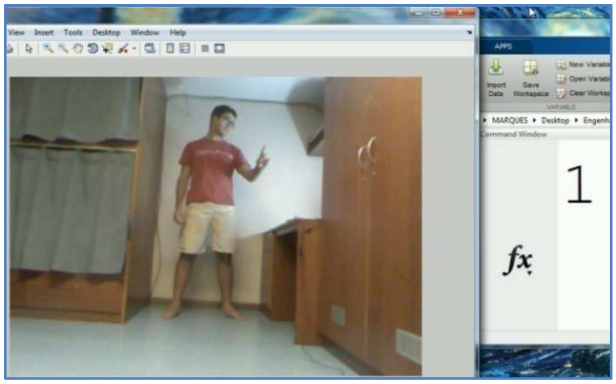
Figura 17 - Precisão do classificador em função do número de dimensões para o banco com ruídos (em azul, precisão do banco de treinamento; em verde, precisão do banco de testes): à esquerda, o contador de dedos; à direita, o classificador de gestos (aberta ou fechada).

Já no banco de imagens com ruídos (Fig. 17) o comportamento é diferente: a precisão no treinamento sobe gradualmente de acordo com o número de dimensões enquanto no teste se mantém em aproximadamente (1/número de classes). Isso caracteriza o chamado *overfitting*. A precisão atinge o máximo para o banco de treinamentos com cerca de 250 dimensões para o contador de dedos e aproximadamente 40 para o classificador de gestos (mão aberta ou fechada).

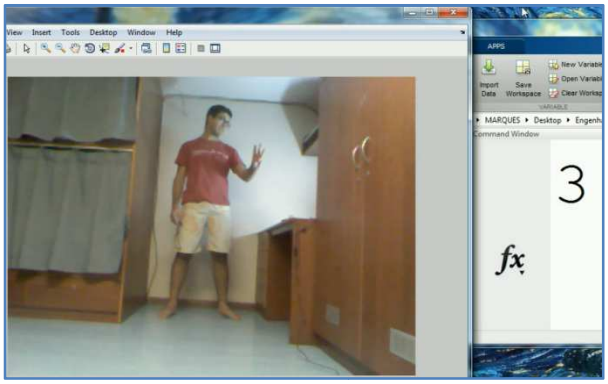
As próximas etapas referem-se à classificação em tempo real. Com ajuda do ponto da mão fornecido pelo *Microsoft Kinect*®, extrai-se a região de interesse (ROI), obtendo uma imagem em escala de cinza (Fig. 18). Na função de tratamento da imagem estabelece-se um padrão de tamanho e realiza-se a binarização. O limiar utilizado é a média dos valores de escala de cinza da imagem extraída, já que a mão ocupa grande área na figura.

Figura 18 - Imagem em escala de cinza obtida pelo *streaming* de profundidade do *Microsoft Kinect*® e imagem após passar pelo processo de binarização.

Os resultados finais são mostrados nas Fig. 19, 20 e 21 a seguir[3]:



(a)



(b)

(c)

Figura 19- *Print screens* do vídeo gerado para o classificador de gestos (número de dedos na mão).

Figura 20 - *Print screens* do vídeo gerado para o classificador de gestos (mão aberta ou fechada).

[3] Os vídeos podem ser encontrados no seguinte link: <https://www.youtube.com/channel/UC2WovW85lxhSQndD97lPaSA>.

No segundo algoritmo observa-se o ângulo calculado à direita e duas linhas representando as posições obtidas do braço (vermelha) e do antebraço (verde).

Figura 21 - *Print screens* do vídeo gerado para medir o ângulo do braço.

Uma possível limitação do sistema é a necessidade de treinamento do algoritmo no local de utilização do sensor. O plano de fundo pode interferir na classificação dos gestos obtidos.

Vale destacar que com esses resultados podemos enviar comandos para o manipulador robótico, como já realizado em trabalhos anteriores por Miranda [33]. Tais trabalhos criaram uma interação entre o MATLAB® e o manipulador usando o movimento ocular como gerador dos comandos.

4. CONCLUSÃO

Os resultados obtidos foram satisfatórios, mostrando como o emprego do sensor *Microsoft Kinect*® torna mais fácil o reconhecimento de gestos. O fornecimento das posições das articulações (ombro, cotovelo e mão, por exemplo) é extremamente útil na obtenção da região de interesse. Outro ponto importante foi a alta eficiência dos classificadores do tipo Máquina de Vetores de Suporte (SVM). Atingiu-se um bom funcionamento do sistema utilizando por volta de 100 a 200 imagens para treinamento, demonstrando o quão poderosa essa ferramenta pode ser.

A base aqui consolidada pode ser aplicada no acionamento de motores, controlando sua velocidade e sentido de rotação, por exemplo. Trabalhos futuros também podem ser realizados com base nesse projeto na área de tecnologia assistiva, auxiliando pessoas em tarefas em que são requeridos maiores esforços físicos, reconhecimento de sinais de LIBRAS (Língua Brasileira de Sinais), além da integração final com o manipulador robótico.

REFERÊNCIAS BIBLIOGRÁFICAS

1. SHERIDAN, T. Telerobotics. **Automatica**, v. 25, n. 4, p. 487-507, 1989.

2. LATOMBE, J. C. Robot Motion Planning. **Kluwer Academic Publishers**, n. 2, 1991.

3. BURDEA, G.; , Z. J. Dextrous telerobotics with force feedback - an overview. Part 1: Human Factors. **Robotica**, v. 9, p. 171-178, Fevereiro 1991a.

4. HALAVAIS, A. Book review: The robot in the garden: Telerobotics and telepistemology in the age of the internet. **Journal of the American Society for Information Science and Technology**, v. 52, 2001.

5. SPRINGER Handbook Robotics. [S.l.]: Springer, 2008. 741-757 p.

6. WEINLAND, D.; RONFARD, R.; BOYER, E. A survey of vision-based methods for action representation, segmentation and recognition. **Computer Vision and Image Understanding**, v. 115, p. 224-241, Fevereiro 2011.

7. TANG, Y.; SUN, Z.; TAN, T. Slice representation of range data for head pose estimation.. **Computer Vision and Image Understanding**, v. 128, p. 18-35, Novembro 2014.

8. PISHARADY, P. K.; SAERBECK, M. Recent methods and databases in vision-based hand gesture recognition A review. **Computer Vision and Imagem Understanding**, v. 141, p. 152-165, Dezembro 2015.

9. RUFFIEUX, S. et al. Gesture recognition corpora and tools: A scripted ground truthing method. **Computer Vision and Image Understanding**, v. 131, p. 72-87, Fevereiro 2015.

10. PILLAJO, C.; SIERRA, J. E. **Human machine interface hmi using kinect sensor to control a scara robot.** Communications and Computing (COLCOM), IEEE Colombian Conference. [S.l.]: [s.n.]. 2013.

11. MONARD, M. C.; BARANAUSKAS, J. A. Conceitos de Aprendizado de Máquina. In: MONARD, M. C.; BARANAUSKAS, J. A. **Sistemas Inteligentes - Fundamentos e Aplicações**. [S.l.]: Manole, 2003. p. 89-144.

12. NOLKER, C.; RITTER, H. **GREFIT:** Visual Recognition of Hand Postures. Lecture Notes in Artificial Intelligence. Gif-sur-Yvette, França: Springer. 1999. p. 61 - 72.

13. BELUCO, A. **Classificação de Imagens de Sensoriamento Remoto Baseada em Textura por Redes Neurais**. Universidade Federal do Rio Grande do Sul. Porto Alegre, RS. 2002.

14. SHIBA, M. H. et al. **Classificação de imagens de sensoriamento remoto pela aprendizagem por árvore de decisão:** uma avaliação de desempenho. Simpósio Brasileiro de Sensoriamento Remoto. Goiânia, GO: [s.n.]. 2005. p. 4319 - 4326.

15. KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **ImageNet Classification with Deep Convolutional Neural Networks**. Annual Conference on Nural Information Processing Systems. Lake Tahoe, EUA: [s.n.]. 2012.

16. HEISELE, B.; HO, P.; POGGIO, T. **Face recognition with support vector machines:** global versus component-based approach. Eigth IEEE Internacional Conference on Computer Vision. Vancouver, Canada: [s.n.]. 2001. p. 688 - 694.

17. MALISIEWICZ, T.; GUPTA, A.; EFROS, A. A. **Ensemble of examplar-SVMs for object detection and beyond**. International Conference on Computer Vision. Barcelona, Espanha: [s.n.]. 2011. p. 89 - 96.

18. OSUNA, E.; FREUND, R.; GIROSIT, F. **Training support vector machines:** an application to face detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Juan, Porto Rico: [s.n.]. 1997. p. 193 - 199.

19. MITCHELL, T. **Machine Learning**. [S.l.]: McGraw Hill, 1997.

20. VAPNIK, V. N. **The nature of Statistical learning theory**. New York: Springer-Verlag, 1995.

21. KARYSTINOS, G. N.; PADOS, D. A. On overfitting, generalization, and randomly expanded training sets. **IEEE Transactions on Neural Networks**, v. 11, n. 5, p. 1050 - 1057, Setembro 2000. ISSN 1045 - 9227.

22. LIU, Z. P.; CASTAGNA, J. P. **Avoiding overfitting caused by noise using a uniform training mode**. International Joint Conference on. Washington, USA: [s.n.]. Julho 1999. p. 1788 - 1793.

23. LORENA, A. C.; CARVALHO, A. C. P. L. F. Uma Introdução às Support Vector Machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43 - 67, 2007.

24. BRAGA, A.; CARVALHO, A. C. P. L. F.; LUDERMIR, T. B. **Redes Neurais Artificiais:** Teoria e Aplicações. [S.l.]: LTC, 200.

25. HAYKIN, S. **Neural Networks - A Compreensive Foundation**. 2. ed. New Jersey, EUA: Prentice-Hall, 1999.

26. BURGES, C. J. C. A tutorial on support vector machines for pattern recognition. **Knowledge Discovery and Data Mining**, v. 2, p. 1 - 43, 1998.

27. BOTTOU, L. et al. **Comparison of classifier methods:** a case study in handwriting digit recognition. International Conference on Pattern Recognition. [S.l.]: [s.n.]. 1994. p. 77 - 87.

28. HSU, C.; LIN, C. A Comparison of Methods for Multi-class Support Vector Machines. **IEEE Transactions on Neural Networks**, Beijing, China, v. 13, n. 2, p. 415 - 425, 2002.

29. PRINA, B. Z.; TRENTIN, R. **GMC:** Geração de Matriz de Confusão a partir de uma classificação digital de imagem do ArcGIS. Simpósio Brasileiro de Sensoriamento Remoto. João Pessoa - PB: [s.n.]. 2015. p. 131-139.

30. MATOS, P. F. et al. **Relatório Técnico - Métricas de Avaliação**. USP, UFSCar, Unimep. São Carlos - SP, p. 11. 2009.

31. WOLD, S.; ESBENSEN, K.; GELADI, P. Principal Component Analysis. **Chemometrics and Intelligent Laboratory Systems**, Amsterdan, p. 37 - 52, fev. 1987.

32. ABDI, H.; WILLIAMS, L. J. Principal component analysis. **WIRES Computacional Statistics**, v. 2, p. 433 - 459, Julho 2010.

33. MIRANDA, V. R. F. et al. **Controle de um Manipulador Robótico via Eletrooculografia:** Uma Plataforma para Tecnologia Assistiva. Simpósio Brasileiro de Automação Inteligente (SBAI). Natal - RN: [s.n.]. 2015. p. 1613 - 1618.

34. MICROSOFT CORPORATION. Getting Started - Kinect for Windows 1.8. **https:** //msdn.microsoft.com. ISSN 1. Disponivel em: <https://msdn.microsoft.com/en-us/library/hh855354.aspx>. Acesso em: 20 Janeiro 2018.

35. MICROSOFT CORPORATION. System Requirements for Kinect for Windows 1.8. **https:** //msdn.microsoft.com. ISSN 2. Disponivel em: <https://msdn.microsoft.com/en-us/library/hh855359.aspx#ID4EW>. Acesso em: 20 Janeiro 2018.

36. MICROSOFT CORPORATION. Download SDK Kinect for Windows v1.8. **https:** //msdn.microsoft.com. ISSN 3. Disponivel em: <https://www.microsoft.com/en-us/download/details.aspx?id=40278>. Acesso em: 20 Janeiro 2018.

37. BURDEA, G.; ZHUANG, J. Dextrous telerobotics with force feedback - an overview. Part 2: Control and Implementation. **Robotica**, v. 9, p. 291-298, Março 1991b.

38. WU, X.; KUMAR, V. **The Top Ten Algorithms in Data Mining**. Boca Raton, EUA: Chapman & Hall, 2009.