UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MOISÉS ALBERTO HAMESTER

Método de Reconhecimento de Gestos Aplicado em Smartphones

Trabalho de Graduação.

Prof. Dr. Dante Augusto Couto Barone
Orientador

Porto Alegre, junho de 2013.

AGRADECIMENTOS

Sem mencionar nomes, a todos que participaram desta jornada.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

LIBRAS	Linguagem Brasileira de Sinais
IDE	Integrated Development Environment
SDK	Sofware Development Kit
ADT	Andoir Development Tools
OpenCV	Open Source Computer Vision Library
SVM	Support Vector Machine
RGB	Red Green Blue
KNN	K Nearest Neighbor
NBC	Naive Bayes Classifier
CV	Cross Validation
FP	Falso Positivo
VP	Verdadeiro Positivo
FN	Falso Negativo
VN	Verdadeiro Negativo

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Nos últimos anos ocorreu uma enorme popularização dos dispositivos *smartphones*, os quais possuem a característica de dispor de um excelente poder de computação e uma câmera agregada, o que abre muitas possibilidades no campo da visão computacional. Uma dessas possibilidades é o reconhecimento de gestos, o qual pode ter muitas aplicações, como por exemplo, permitir uma interação homem computador de maneira mais livre, ou então permitir a comunicação de uma maneira mais natural entre uma pessoa que se comunique usando alguma linguagem de sinais e outra que não.

O trabalho descreve o uso de uma técnica de reconhecimento de gestos aplicada a dispositivos móveis que possuem o sistema Android. E faz um comparativo entre a qualidade de predição entre três técnicas de classificação diferentes, o *Naive Bayes Classifier* (NBC), *K Nearest Neighbor* (KNN) e o *Support Vector Machine* (SVM).

Palavras-Chave: Reconhecimento de Gestos, Espaço de Cores YC_bC_r, Momentos Invariantes de Hu, Support Vector Machine, Naive Bayes Classifier, K Nearest Neighbor, Interação Homem Computador, Visão Computacional, Aplicações Android.

Gesture Recognition Method applied on Smartphones

ABSTRACT

There has been a huge popularization of smartphone devices in recent years, which have the characteristic of having an excellent computing power and a camera attached, which opens up many possibilities in the field of computer vision. One of those possibilities is the gesture recognition, which can have many applications, such as allow human computer interaction in a freer way, or allow communication in a more natural way between a person who communicates using some signal language and another that does not.

This work will discuss the use of a technique for gesture recognition applied to mobile devices running the Android system. And makes a comparison between the quality of prediction of three different classification techniques, the Naive Bayes Classifier, K Nearest Neighbor and Support Vector Machine.

Keywords: Gesture Recognition, YC_bC_r color space, Hu invariants moments, Support Vector Machine, Naive Bayes Classifier, K Nearest Neighbor, Human Computer Interaction, Computer Vision, Android Application.

1 INTRODUÇÃO

1.1 Gestos

Gestos são formas não verbais de comunicação que permitem expressar uma grande variedade de pensamentos e sentimentos, onde usa-se partes do corpo, ou o corpo todo, para fazer a comunicação. Gestos podem ser feitos com qualquer parte do corpo, a mão, a cabeça, os olhos, as pernas ou então partes em conjunto e podem estar combinados com a comunicação verbal ou não. (PEREIRA, s.d.).

Assim como a comunicação falada, a comunicação gestual depende do meio onde ela acontece, podendo um mesmo gesto ter diferentes significados em círculos sociais diferentes. Por exemplo, o gesto americano "OK" feito com a mão, no Brasil seria interpretado de outra maneira, podendo ser considerado até como um gesto ofensivo. (PEREIRA, s.d.)

"A conversação por gestos está na origem de qualquer linguagem, ela é o modelo de qualquer comunicação, já que comporta dois aspectos de qualquer processo social: a reação de adaptação do outro e a antecipação do resultado do ato." (MEAD apud PEREIRA, s.d.)

1.2 Motivação

Desde o surgimento dos computadores modernos a interação entre o homem e a máquina tem sido feita principalmente através do teclado e do mouse. Com o surgimento das telas *touchscreen* e da popularização de dispositivos *smartphones* e *tablets* que possuem tais telas, tornou-se comum também, a interação humano-computador através dessas. Selecionando objetos diretamente na tela, através do toque na mesma, arrastando objetos, e também novos métodos de rotacionar, e aplicar zoom que acabaram se tornando comuns.

Esses métodos de interação, porém não são métodos muito naturais de comunicação, pois é necessário ficar pressionando teclas, movimentando o mouse, ou podem ter alguns inconvenientes, como é o caso das telas *touchscreen* que após um longo período de uso podem causar um desconforto ou fadiga no braço do usuário.

Com o objetivo de explorar outras técnicas de interação, surgiu a ideia de implementar uma aplicação capaz de reconhecer gestos, proporcionando assim um método mais natural de interação humano-computador.

Outra possível aplicação para um sistema de reconhecimento de gestos é o reconhecimento do alfabeto datilológico da Linguagem Brasileira de Sinais (LIBRAS),

o qual é composto por 27 gestos que representam as letras do alfabeto, como mostra a figura 1.1. Esse alfabeto é complementar a LIBRAS, sendo usado para digitar nomes de pessoas, ruas, ou objetos, ou expressões que não possuam um sinal próprio. Podendo tal aplicação ser útil para pessoas com deficiência auditiva comunicarem-se com pessoas que não compreendem a linguagem por exemplo.

Figura 1.1: Alfabeto Datilológico

Fonte: http://escritadesinais.wordpress.com/2010/09/07/alfabeto-manual-ou-datilologia/

1.3 Objetivos do Trabalho

1.3.1 Objetivos Gerais

Esse trabalho tem o objetivo geral de investigar a viabilidade de um aplicativo, desenvolvido para *smartphones*, que seja capaz de fazer o reconhecimento de gestos.

1.3.2 Objetivos Específicos

Este trabalho tem como objetivo específico implementar uma técnica de reconhecimento de gestos aplicada a dispositivos móveis, *smartphones* com o sistema Android. Sendo este aplicativo capaz de reconhecer cinco classes distintas de gestos as quais representam os números de um a cinco. Essas classes de imagens são apresentadas abaixo nas figuras 1.2 a 1.6.

E também avaliar a capacidade do aplicativo em classificar as diferentes classes de gestos utilizando três técnicas de classificação diferentes, sendo elas o *Naive Bayes Classifier*, *K Nearest Neighbor* e o *Support Vector Machine*. E fazer um comparativo entre os resultados alcançados por essas técnicas de classificação.

Figura 1.2: Classe de imagens representando o número um

Fonte: Elaborado pelo autor

Figura 1.3: Classe de imagens representando o número dois

Fonte: Elaborado pelo autor

Figura 1.4: Classe de imagens representando o número três

Fonte: Elaborado pelo autor

Figura 1.5: Classe de imagens representando o número quatro

Fonte: Elaborado pelo autor

Figura 1.6: Classe de imagens representando o número cinco

Fonte: Elaborado pelo autor

1.4 Trabalhos Relacionados

O reconhecimento de gestos é um campo de estudo da visão computacional que gera muitos trabalhos, artigos. Pode-se destacar trabalhos de pesquisa relacionados ao tema, reconhecimento de gestos, como:

O trabalho de Chen, que utiliza *Haar-like features* e o algoritmo de aprendizado *AdaBoost* para fazer o reconhecimento de quatro classes de gestos obtendo uma alta acurácia. (CHEN, 2007)

O estudo de Du, que utiliza o Kinect para extrair a mão do restante da imagem, utilizando a informação da distância de profundidade para tanto. E fazendo a classificação através do número de concavidades e do número de pontos convexos presentes no contorno da mão obtida após a extração. (DU, 2011)

E o trabalho de Chakraborty, que analisa diferentes técnicas de fazer o reconhecimento de gestos, como o uso de gradientes, o método de subtração, um método invariante a rotação e o uso de vetores. (CHAKRABORTY, 2008)

E também o trabalho de Liu que utiliza os momentos invariantes de Hu para classificar três classes de gestos. (LIU, 2012)

2 TECNOLOGIAS UTILIZADAS

Neste capítulo é apresentado o smartphone utilizado nos testes do aplicativo, bem como a *Integrated Development Environment* (IDE) utilizada no desenvolvimento. E também o sistema Android, que é o sistema operacional para o qual o aplicativo foi desenvolvido e a biblioteca Open Source Computer Vision Library (OpenCV) que foi utilizada no desenvolvimento do aplicativo.

2.1 Smartphone

Utilizou-se durante o desenvolvimento e testes do aplicativo um *smartphone* LG E405f, o qual possui um processador Qualcomm, modelo MSM7225A de 800MHz e 512Mb de memória RAM e com uma câmera de 3.2MP conforme informações do fabricante. O smartphone utiliza a sistema Android 2.3.6 e a versão 2.7 da biblioteca OpenCV.

Sendo este um *smartphone* comum, ou seja, não é nenhum top de linha, com um custo razoavelmente baixo, possuindo um preço médio de R$449,00.

2.2 IDE

Para o desenvolvimento do aplicativo utilizou-se a Eclipse IDE que é uma IDE *open source* desenvolvida em sua maior parte em Java e mantida pela Eclipse Foundation. A Eclipse IDE pode ser utilizada para desenvolver aplicações em várias linguagens de programação através do uso de *Software Develop Kits* (SDK) que podem ser instalados individualmente, entre essas linguagens disponíveis estão Java, C, C++, Ruby entre outras.

Juntamente com a Eclipse IDE utilizou-se o plugin Android Development Tools (ADT) que é um plugin desenvolvido para criar um ambiente de desenvolvimento integrado para criar-se aplicativos Android.

2.3 Sistema Android

É um sistema operacional baseado no Linux desenvolvido pelo Open Set Alliance, um consórcio de companhias de hardware, software e telecomunicações liderado pelo Google, para dispositivos móveis como *smartphones* e *tablets*, o primeiro dispositivo a usar o sistema foi vendido em outubro de 2008.

As aplicações são desenvolvidas utilizando a linguagem de programação Java, e são portáveis podendo ser rodadas em diferentes dispositivos, desde que eles tenham os requisitos necessários, sem mudança no código fonte da aplicação.

O sistema Android é o sistema operacional para dispositivos móveis mais utilizado no primeiro trimestre de 2013, tendo uma participação de mercado de 75% contra 59,1% no primeiro trimestre de 2012 contabilizando 162 milhões de novas unidades vendidas no primeiro trimestre de 2013 contra 90 milhões no mesmo período do ano anterior, um aumento nas vendas de 79,5%. Na tabela 2.1 vê-se a participação de

mercado e a quantidade de novas unidades dos principais sistemas operacionais para *smartphones* nos primeiros trimestres dos anos de 2012 e 2013, bem como a taxa de crescimento de cada sistema.

Tabela 2.1: Principais Sistemas Operacionais para *Smartphones*, novas unidades e Market Share no 1º Trimestre de 2013 comparado ao 1º Trimestre de 2012

Sistema Operacional	1º Trim. 2013 Novas Unidades (milhões)	1º Trim. 2013 Market Share	1º Trim. 2012 Novas Unidades (milhões)	1º Trim. 2012 Market Share	Crescimento no ano
Android	162.1	75.0%	90.3	59.1%	79.5%
iOS	37.4	17.3%	35.1	23.0%	6.6%
Windows Phone	7.0	3.2%	3.0	2.0%	133.3%
BlackBerry OS	6.3	2.9%	9.7	6.4%	-35.1%
Linux	2.1	1.0%	3.6	2.4%	-41.7%
Symbian	1.2	0.6%	10.4	6.8%	-88.5%
Others	0.1	0.0%	0.6	0.4%	-83.3%
Total	**216.2**	**100.0%**	**152.7**	**100.0%**	**41.6%**

Fonte: http://www.idc.com/getdoc.jsp?containerId=prUS24108913

Figura 2.1: Logotipo do Sistema Android

Fonte: http://pt.wikipedia.org/wiki/Android

2.4 Biblioteca OpenCV

OpenCV(Open Source Computer Vision Library) é uma biblioteca multiplataforma de funções que surgiu para alavancar o campo da Visão Computacional como uma biblioteca aberta e otimizada, para a disseminação do conhecimento da área permitindo a criação de códigos mais legíveis e também disponibilizar códigos portáveis e de boa performance.

Ela foi primeiramente desenvolvida pela Intel, tendo sua versão alpha sido lançada em janeiro de 1999, atualmente é mantida por Willow Garage e Itseez, ela é desenvolvida em C e C++ porém possui interfaces para outras linguagens como Python,

Matlab e também Java que é a interface utilizada para o desenvolvimento de aplicações para o sistema Android. Sendo compatível também com os sistemas Windows, Linux, Mac OS e iOS.

A biblioteca contém mais de 500 funções, divididas em vários módulos como processamento de vídeo, imagens, calibragem de câmera e também possui um módulo de aprendizado de máquina, visto que muitas aplicações de Visão Computacional necessitam de aprendizado de máquina, como é o caso do reconhecimento de gestos, reconhecimento de padrões, detecção de face.

OpenCV possui uma licença *open source* que permite que o usuário crie aplicações sem que necessite tornar seu produto *open source*. Devido a isso, e aliado à qualidade que a biblioteca possui, ela é usada em uma enorme quantidade de aplicações como, por exemplo, aplicações de segurança, aplicações militares, calibração de câmeras, produção de mapas de satélites, aplicações médicas e também foi utilizada na parte de Visão Computacional do robô "Stanley" da Universidade de Stanford que venceu o desafio Darpa Grand Challenge 2005, que é um desfio criado pelo Departamento de Defesa dos Estados Unidos da América, com o objeto de incentivar o desenvolvimento de veículos autônomos.

Figura 2.2: Logotipo da biblioteca OpenCV

Fonte: http://www.opencv.org

3 PROCESSAMENTO

Neste capítulo é apresentada a técnica utilizada para fazer o reconhecimento de gestos, sendo feita uma descrição do espaço de cores YC_bC_r, o qual é empregado na técnica para fazer a segmentação da mão do restante da imagem, bem como uma descrição da técnica Momentos Invariantes de Hu que é utilizada para fazer a classificação dos gestos.

3.1 Descrição da Técnica Utilizada

Para fazer o reconhecimento de um gesto, o primeiro passo é extrair a mão do restante da imagem. A extração da mão do restante da imagem é feita a partir da detecção da cor da pele humana na imagem. Para tanto, ao adquirir uma imagem, primeiramente é feita a conversão do espaço de cores Red Green Blue (RGB) para o espaço de cores YC_bC_r. Com a imagem já no espaço de cores YC_bC_r é feito o corte da mão do restante da imagem a partir do limiar da cor da pele humana.

O passo seguinte é buscar todos os contornos restantes na imagem, para então calcular-se a área desses. Assume-se que o contorno com a maior área seja o contorno que de fato representa a mão. Caso haja mais de um contorno, os contornos de menor área são descartados. Isto é feito para o caso de haver restado algum ruído após a extração da mão.

De posse do contorno que representa a mão, faz-se o cálculo dos momentos invariantes de Hu, para posteriormente utilizar os quatro primeiros momentos para fazer a classificação do gesto.

3.2 Espaço de Cores YC_bC_r

O espaço de cores YC_bC_r pertence à família dos espaços de cores utilizados na transmissão de sinais digitais de televisão, sendo muito empregado em aplicações de vídeo digital. (ZOU, 2011) Nesse espaço, o componente Y representa a luminância e os componentes C_b e C_r representam a crominância. São armazenados no componente C_b a diferença entre o componente de azul e o valor de referência e em C_r a diferença entre o componente de vermelho e o valor de referência.

O espaço de cores YC_bC_r pode ser convertido a partir do espaço RGB a partir do seguinte cálculo conforme a figura 3.1.

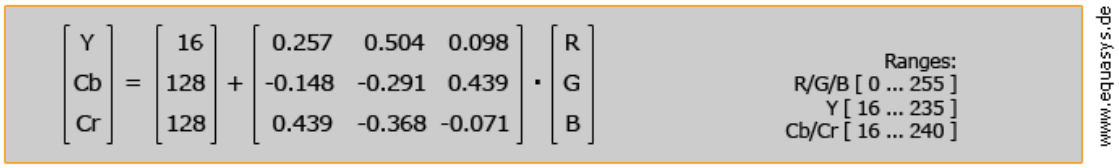

Figura 3.1: Conversão do espaço de cores RGB para o espaço de cores YC_bC_r

Fonte: http://www.equasys.de/colorconversion.html

Esta conversão não ocupa todo o intervalo possível definido por oito bits, pois, como ele é um padrão utilizado para a televisão, existe um pequeno espaço de tolerância nas bordas do intervalo, para permitir a compatibilidade com equipamentos analógicos.

Já em equipamentos digitais como computadores, normalmente pode-se usar todo o intervalo disponível com oito bits, como é o caso de imagens JPEG. Para utilizar todo o intervalo utiliza-se a conversão tal como apresentado na figura 3.2.

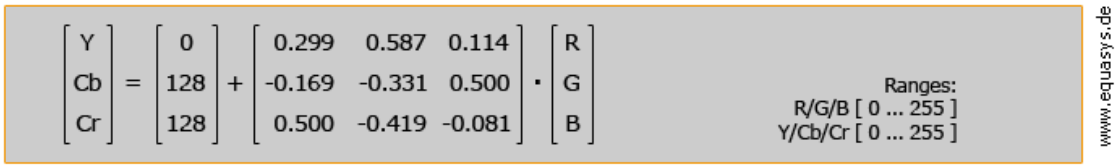

Figura 3.2: Conversão do espaço de cores RGB para o espaço de cores YC_bC_r ocupando todo o espaço de valores possíveis

Fonte: http://www.equasys.de/colorconversion.html

Para converter do espaço de cores YC_bC_r para o espaço RGB faz-se o cálculo apresentado na figura 3.3.

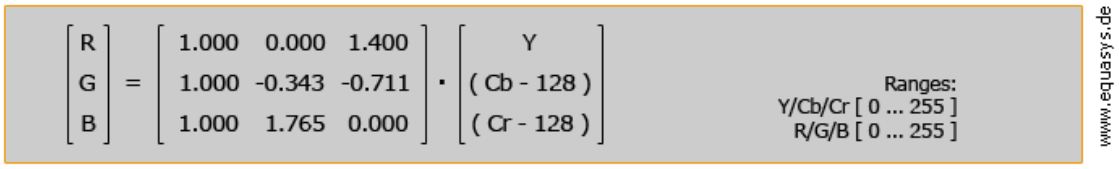

Figura 3.3: Conversão do espaço de cores YC_bC_r para o espaço de cores RGB

Fonte: http://www.equasys.de/colorconversion.html

O espaço de cores YC_bC_r é o mais adequado para fazer a segmentação da mão do restante da imagem, já que ele sofre menor influência devido à iluminação e pelo fato que diferentes cores de pele humana estão distribuídas em um intervalo pequeno e bem definido. (LIU, 2012) Esse intervalo consiste em $70 \leq Cb \leq 130$, $125 \leq Cr \leq 175$, e o componente Y pode ser desprezado, devido a sua não influência. (LIU, 2012)

Na figura 3.4 vê-se o contorno da mão já segmentado do restante da imagem.

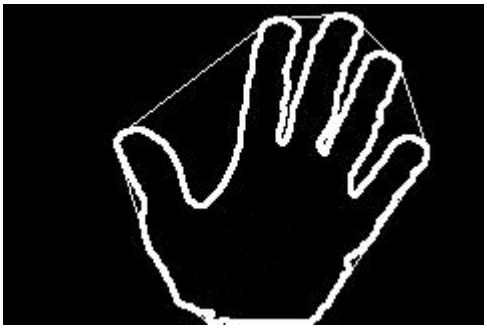

Figura 3.4: Contorno da mão após segmentação do restante da imagem

Fonte: Elaborado pelo autor

3.3 Momentos Invariantes de Hu

A utilização de momentos proporciona um método de descrever as propriedades de um objeto em relação a sua área, orientação, posição e outros parâmetros. (AWCOCK, 1996)

A equação básica que define o momento de um objeto é a dada pela equação 1:

$$M_{ij} = \sum_{x}\sum_{y} x^i y^j a_{xy} \tag{1}$$

Onde a ordem do momento é dada por i + j, x e y são as coordenadas dos pixels, sendo que a origem, geralmente, é o canto superior esquerdo da imagem, e a_{xy} representa o valor do pixel.

O momento de ordem zero, conhecido como superfície da imagem, é a soma do valor dos pixels da imagem. Caso a imagem seja binária, o momento será igual à área da imagem. O momento de ordem zero é dado pela equação 2:

$$M_{00} = \sum_{x}\sum_{y} a_{xy} \tag{2}$$

Os momentos de primeira ordem são representados pelas equações 3 e 4:

$$M_{10} = \sum_{x}\sum_{y} x a_{xy} \tag{3}$$

$$M_{01} = \sum_{x}\sum_{y} y a_{xy} \tag{4}$$

De posse dos momentos de ordem zero e de primeira ordem, é possível obter o centroide da imagem, sendo este definido pelas equações 5 e 6:

$$x' = \frac{M_{10}}{M_{00}} \tag{5}$$

$$y' = \frac{M_{01}}{M_{00}} \tag{6}$$

Já os momentos de segunda ordem M_{20} e M_{02} correspondem aos momentos de inércia da imagem. Nota-se que esses momentos variam de acordo com a posição da imagem, sua escala e orientação, o que diminui a sua utilidade. (AWCOCK, 1996)

É possível calcular momentos sem esta variância, isso é alcançado calculando primeiramente o momento central relativo ao centroide μ, o qual é dado pela equação 7:

$$\mu_{ij} = \sum_{x}\sum_{y}(x - x')^{i}(y - y')^{j}a_{xy} \tag{7}$$

A partir dos quais é possível calcular momentos centrais normalizados conforme a equação 8:

$$\eta_{ij} = \frac{\mu_{ij}}{(\mu_{00})^{\lambda}} \tag{8}$$

Sendo $\lambda = \frac{i+j}{2} + 1$ e (i + j) ≥ 2.

Hu então propôs uma combinação linear dos momentos centrais de segunda e terceira ordem, criando sete momentos invariantes, os quais são invariantes no domínio contínuo da imagem quanto à rotação, translação e escala. (PRATT, 2007) De acordo com Jain (1989), o uso de momentos invariantes pode ser aplicado em problemas de reconhecimento de padrões, como reconhecimento de gestos, por exemplo. Utilizando n momentos para caracterizar uma imagem, ela pode ser representada como um ponto em um espaço n-dimensional, reduzindo o problema de reconhecimento de padrões a um problema de classificação padrão.

Nas equações 9 a 15 são apresentadas as equações dos sete momentos invariantes de Hu.

$$I_1 = \eta_{20} + \eta_{02} \tag{9}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}{}^2 \tag{10}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{11}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{12}$$

$$\begin{aligned} I_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{13}$$

$$\begin{aligned} I_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})^2(3\eta_{21} + \eta_{03}) \end{aligned} \tag{14}$$

$$\begin{aligned} I_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{15}$$

4 MÉTODOS DE CLASSIFICAÇÃO

Um bom método de classificação é uma parte importante de um algoritmo de reconhecimento de gestos, pois ele está diretamente relacionado à qualidade do reconhecimento que o método de reconhecimento de gestos irá alcançar.

Neste capítulo são apresentados três métodos de classificação, os quais foram empregados para fazer as predições das amostras utilizadas para teste.

4.1 Naive Bayes Classifier (NBC)

É um modelo probabilístico de classificação que aplica o método de Bayes, que assume que cada classe de dados é representada por características independentes entre si, ou seja, que nenhuma característica depende da existência ou não de outra. (HASTIE, 2008)

Por exemplo, uma bola pode ser caracterizada como sendo redonda, com um diâmetro de 10cm e cor vermelha. O classificador irá então levar essas três características em consideração de forma independente para calcular a probabilidade de o objeto ser ou não ser uma bola.

O classificador calcula as matrizes de covariância e o vetor principal de cada classe e então, para fazer a predição de uma entrada, ele escolhe a classe com a maior probabilidade de atender as características da amostra que se pretende classificar.

A suposição de independência entre as características que determinam uma classe simplifica a fase de treinamento do algoritmo, pois é possível estimar a densidade de cada característica da classe individualmente.

Essa suposição de independência permite estimar melhor os parâmetros necessários para a classificação com um menor número de amostras para treinamento do que outros classificadores, o que torna o algoritmo eficaz para conjuntos de dados que contenham muitas características.

4.2 K Nearest Neighbor (KNN)

É um algoritmo de classificação baseado em memória que não necessita de modelo. (HASTIE, 2008)

É não paramétrico, ou seja, não é feita nenhuma suposição sobre a distribuição dos dados, o que pode ser interessante, já que no mundo real, nem tudo obedece a uma distribuição teórica presumível.

O algoritmo também não faz nenhuma generalização com os dados de treinamento. Praticamente todos os dados de treinamento são necessários na fase de predição, sendo assim, a fase de treinamento tende a ser bem rápida. Já a fase de predição tende a ser mais custosa quanto ao tempo de execução e à quantidade de memória necessária, já que é necessário armazenar as amostras de treinamento para a fase de predição.

O algoritmo considera as amostras como sendo pontos dispersos no espaço multidimensional de suas características. A partir desse espaço de características, é possível calcular a distância, geralmente euclidiana, entre as diferentes amostras. (MITCHELL, 1997)

Para fazer a predição de uma amostra, buscam-se as k amostras usadas na fase de treinamento que estejam mais próximas à amostra a qual se está classificando. Então, essa amostra é classificada como sendo da mesma classe das amostras que mais aparecem entre as k amostras mais próximas a ela. (HASTIE, 2008)

Na figura 4.1, vê-se um exemplo de predição, onde pretende-se classificar a amostra representada pelo quadrado. Caso utilize-se k=3, vê-se que das três amostras mais próximas àquela que está se fazendo a classificação, duas pertencem à classe representada por triângulos, e uma à classe representada por círculos. Desse modo, a amostra representada pelo quadrado será classificada como pertencente à classe dos triângulos, já que esta classe é a que mais aparece entre as k=3 amostras mais próximas. Já se o valor de k for igual a cinco, a nova amostra será classificada como pertencente à classe dos círculos.

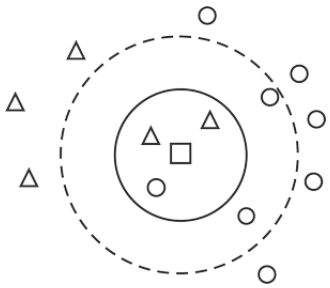

Figura 4.1: Exemplo de classificação usando KNN

Fonte: Elaborado pelo Autor

O valor de k tem influência direta no resultado da predição. A escolha de valores para k muito pequenos pode levar a uma grande influência de ruído na predição e a mínimos locais. Já valores de k muito grandes acabam por tirar a influência do ruído, porém acabam por tornar as fronteiras entre as classes menos distintas.

4.3 Support Vector Machine (SVM)

É um método de aprendizado supervisionado, que ao receber dados de entrada rotulados para o treinamento, trata os mesmos como pontos no espaço de características e então são construídos hiperplanos capazes de fazer a separação entre as diferentes classes. (HASTIE, 2008)

Pode acontecer que as classes não sejam linearmente separáveis no seu espaço de características, nesse caso, é possível fazer o mapeamento das características para um espaço de dimensão maior, isso é feito a partir de uma função de kernel, para lá

encontrar um hiperplano que seja capaz de fazer a separação entre as classes, e então esse hiperplano pode ser mapeado de volta para o espaço original para ser usado como hiperplano de separação das classes. (NORVIG, 2009)

Na figura 4.2, vê-se um exemplo no espaço bidimensional, no qual considera-se como entrada de treinamento para o algoritmo, duas classes, uma composta pelos pontos indicados por círculos e outra pelos quadrados. O algoritmo poderia criar qualquer um dos hiperplanos indicados por linhas tracejadas, para fazer a separação entre as classes.

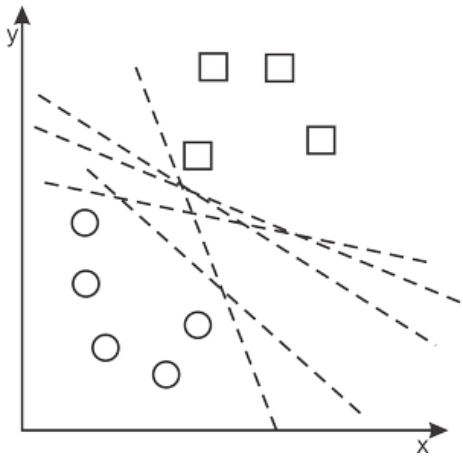

Figura 4.2: Exemplo de hiperplanos gerados pela SVM

Fonte: Elaborado pelo Autor

Entretanto, entre os diversos hiperplanos possíveis, é necessário que se faça a escolha do melhor deles. O melhor hiperplano de separação é o hiperplano que maximiza a margem; ou seja, duas vezes a distância entre o hiperplano e o ponto no espaço mais próximo a ele. (NORVIG, 2009) Quanto maior a margem melhor, pois menos interferência de ruído o algoritmo irá sofrer, e melhor será a generalização dele.

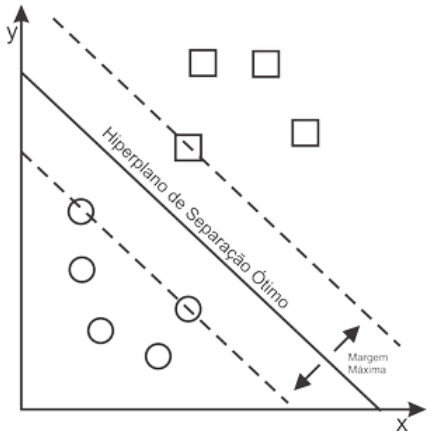

Figura 4.3: Exemplo de hiperplano ótimo

Fonte: Elaborado pelo Autor

5 AVALIAÇÃO

Neste capítulo é apresentada a biblioteca de imagens utilizada para treinamento dos modelos de predição e para sua avaliação. É apresentada também uma técnica de avaliação de modelos, o *K-Fold Cross Validation* (CV). (NORVIG, 2009) Bem como o método de avaliação que aplicou-se para fazer a comparação entre as diferentes técnicas de classificação que foram testadas.

5.1 Biblioteca de Imagens

Montou-se uma biblioteca de imagens composta por cinco classes de gestos, as quais representam os números de um a cinco. Essa biblioteca de imagens é utilizada para fazer o treinamento dos métodos de classificação, bem como sua avaliação.

A biblioteca é constituída de seiscentas imagens para cada classe de gesto, tendo sido adquiridas imagens de ambas as mãos, esquerda e direita, de quatro pessoas, sendo no total cento e cinquenta as imagens adquiridas de cada pessoa.

Optou-se por formar uma biblioteca de imagens composta por cinco classes, pois artigos que versam sobre o assunto geralmente usam de três a seis classes de gestos, como, por exemplo, em Chen (2007), Du (2011), Chakraborty (2008) e Liu (2012).

5.2 K-Fold Cross Validation

Cross Validation é uma técnica simples e muito utilizada para estimar a capacidade de um modelo fazer predições de amostras. O ideal seria ter um conjunto de amostras para treinamento e outro para teste, porém o conjunto de amostras geralmente é escasso, não sendo possível criar um conjunto de testes separadamente do de treinamento. (HASTIE, 2008)

Ainda segundo Hastie, a técnica consiste em dividir o conjunto de amostras em k subconjuntos para então utilizar-se k–1 subconjuntos para treinamento do modelo e o subconjunto que ficou de fora do treinamento é utilizado como conjunto de teste. Repete-se então esse passo até que todos os subconjuntos k tenham sido utilizados como conjunto de teste. Dessa forma, através do uso do *K-Fold Cross Validation*, resolve-se o problema de serem necessários dois conjuntos de amostras independentes.

Em *data mining* e aprendizado de máquina, geralmente utiliza-se o valor k como sendo igual a 10. (REFAEILZADEH, 2008) Para obter resultados ainda mais confiáveis, um número maior de estimativas é melhor, visto que em uma execução do *K-Fold Cross Validation* obtém-se apenas k estimativas. Para tanto, pode-se repetir

várias vezes o *K-Fold Cross Validation*. Sendo assim, pode-se utilizar dez repetições do *K-Fold Cross-Validation* com k igual a dez, que assim irá obter-se o melhor resultado. (REFAEILZADEH, 2008).

5.3 Avaliação dos Métodos de Classificação

Para a avaliação dos métodos de classificação, utilizou-se juntamente à biblioteca de imagens a técnica do *K-Fold Cross Validation* com k igual a dez, e o *Cross Validation* executado dez vezes para obter um número ainda maior de estimativas e assim diminuir a variância dos valores obtidos. Os métodos de classificação testados foram o *Naive Bayes Classifier*, *K Nearest Neighbor* e o *Support Vector Machine*.

As implementações dos algoritmos de classificação utilizados na avaliação foram as que são disponibilizados pela biblioteca OpenCV. Para os testes o KNN foi avaliado com um valor de k igual a 7, e a SVM utilizou a função de kernel *Radial Basis Function* (RBF) a qual é dada pela equação 16 e o valor de gama igual a 1024, valor de C igual a 1024 que é a penalidade para amostras que ficam do lado errado do hiperplano de separação. Estes parâmetros foram definidos após analises iniciais que mostraram que eles obtiveram bons resultados.

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \tag{16}$$

Onde $\gamma > 0$.

Como o valor de k utilizado para a execução do CV é igual a 10, para cada execução do CV, o primeiro passo consiste em dividir a biblioteca de imagens em dez subconjuntos aleatórios. Como a biblioteca de imagens possui 600 imagens de cada classe totalizando 3000 imagens no total, cada um desses subconjuntos gerados possui um total de 300 imagens, sendo 60 imagens de cada classe.

O segundo passo consiste em utilizar nove desses subconjuntos para treinamento do modelo de classificação, totalizando 2700 imagens para treinamento, sendo 540 de cada classe e o subconjunto que ficou de fora do treinamento é utilizado como conjunto de testes, sendo esse conjunto composto por 300 imagens sendo 60 imagens de cada classe. Repete-se esse passo até que todos os subconjuntos gerados no primeiro passo tenham sido utilizados uma vez como conjunto de testes, o que gera as 10 rodadas do CV, visto que o k utilizado é 10.

Como o CV foi executado 10 vezes, ao final obteve-se 100 conjuntos de testes e de treinamento, os quais foram utilizados para avaliar as três técnicas de classificação avaliadas.

Para cada conjunto de testes avaliou-se as três técnicas de classificação e com os resultados obtidos a partir da predição feita por cada método, criou-se matrizes de confusão para cada conjunto conforme o exemplo da tabela 5.1. Onde cada linha da tabela representa a classe correta da entrada e as colunas representam a resposta obtida da predição feita pelo classificador.

Tabela 5.1: Exemplo de matriz de confusão

	Classe 1	*Classe 2*	*Classe 3*	*Classe 4*	*Classe 5*
Classe 1	**59**	1	0	0	0
Classe 2	0	**56**	2	1	1
Classe 3	0	1	**37**	16	6
Classe 4	0	0	3	**45**	12
Classe 5	1	1	3	9	**46**

Fonte: Elaborado pelo autor

A partir da matriz de confusão é possível calcular métricas para a avaliação do desempenho da capacidade de predição do modelo, como por exemplo, a sua acurácia, ou então a precisão e a sensibilidade, as quais são calculadas para cada classe independentemente.

Para comparar-se modelos distintos, quanto a sua qualidade de predição, a acurácia não é indicada, pois ela pode levar a enganos. Para contornar-se esse problema, a solução é a utilização de custos para erros. Para a comparação dos modelos, utiliza-se o seguinte custo: -1 caso a predição seja correta, caso a predição não seja correta o custo é dez. Dessa forma, pode-se comparar diferentes modelos, sendo o modelo de menor custo de qualidade superior a modelos de maior custo.

Tabela 5.2: Exemplo de tabela de custos para uma execução do CV

	Custo
Rodada 0	140
Rodada 1	239
Rodada 2	**85**
Rodada 3	129
Rodada 4	239
Rodada 5	228
Rodada 6	239
Rodada 7	162
Rodada 8	206
Rodada 9	338

Fonte: Elaborado pelo autor

Na tabela 5.2 vê-se um exemplo de tabela de custos, onde o modelo com melhor capacidade de predição a partir do custo é o modelo que foi gerada na segunda rodada do *Cross Validation*, visto que ele minimiza o custo.

6 RESULTADOS

Neste capítulo são apresentadas métricas de avaliação que foram aplicadas aos resultados obtidos por cada técnica de classificação nos testes aplicados. Logo após, são apresentados os resultados obtidos com os testes aplicados aos diversos modelos de treinamento que foram gerados, onde inicialmente apresenta-se os custos de cada modelo gerado nas dez execuções do CV. A partir dos custos é determinado, então, qual o melhor modelo gerado por cada técnica de classificação, ou seja o de menor custo, para então, apresentar a precisão, sensibilidade, acurácia e Medida-F que esses modelos de treinamento obtiveram.

6.1 Métricas de Avaliação

6.1.1 Acurácia

A acurácia é uma métrica do modelo como um todo, ou seja, ela é calculada a partir de todas as classes do modelo. E é igual ao número de predições corretas divididas pelo número de predições feitas.

É necessário tomar cuidado ao utilizar a acurácia para a avaliação de um modelo, pois ela pode ser enganosa. Por exemplo, caso existam duas classes, e no subconjunto de testes existam novecentos e noventa amostras da primeira classe e dez amostras da segunda classe e o modelo predizer que todas as amostras pertençam à primeira classe, o modelo obteria uma acurácia de 990/1000 resultando em uma acurácia de 99%, o que é enganoso, já que o modelo não tem a capacidade de detectar nenhuma amostra da segunda classe.

A acurácia é calculada conforme a equação 17:

$$Acurácia = \frac{VP + VN}{n} \tag{17}$$

Onde n é o número total de predições, VP é o total de verdadeiros positivos e VN é o total de verdadeiros negativos.

6.1.2 Erros (FP)

O número de erros (FP) é o número total de falsos positivos, ou seja, é o somatório de predições feitas como sendo de uma determinada classe, porém a amostra não faz parte dessa classe. Essa é métrica é calculada para cada classe separadamente.

6.1.3 Precisão

Precisão é a fração correta do total previsto como pertencente a uma classe pelo total que foi previsto como pertencente a tal classe.

A precisão é calculada para cada classe conforme equação 18:

$$p = \frac{VP}{VP + FP} \tag{18}$$

Onde VP é o total de verdadeiros positivos da classe que está-se calculando e FP é o total de falsos positivos da classe em questão.

Por exemplo, em uma predição de diagnósticos, a precisão é a fração de pacientes que são realmente portadores de uma doença x pela soma de pacientes que foram classificados como portadores de tal doença.

6.1.4 Sensibilidade

A Sensibilidade, denominada, em inglês, sensitivity ou também recall, refere-se à capacidade do modelo de identificar os resultados positivos. Ela é a proporção das amostras corretamente classificadas como positivas por todas as amostras realmente positivas. Por exemplo, no caso de um diagnóstico médico de câncer, a sensibilidade corresponde à proporção de pacientes classificados como atingidos pela doença pelo número de pacientes que realmente possuam a doença.

A Sensibilidade é calculada para cada classe conforme a equação 19:

$$r = \frac{VP}{VP + FN} \tag{19}$$

Onde VP é o total de verdadeiros positivos da classe que está-se calculando, e FN é o total de falsos negativos da classe em questão.

6.1.5 Medida-F

A Medida-F é uma medida que avalia a capacidade de predição de um modelo. Ela é a média harmônica entre precisão e sensibilidade. Dessa forma quanto melhor o modelo avaliado mais próximo de 1 será o valor da medida, e caso o modelo tenha uma capacidade de predição reduzida ele irá possuir uma Medida-F próxima a 0.

A Medida-F é calculada para cada classe do problema a partir da fórmula 20:

$$F = \frac{2rp}{r + p} \tag{20}$$

Já para calcular-se a Medida-F do modelo como um todo, pode-se utilizar dois diferentes métodos de média, o primeiro método é o Micro-averaged F-Measure, o segundo é o Macro-averaged F-Measure.(ÖZGÜR, 2005)

*6.1.5.1*Medida-F *Micro-averaged*

Nesse método, a Medida-F é calculada globalmente sobre todas as classes dando a ela pesos iguais e tende a ser dominada por classes que possuem mais amostras. (ÖZGÜR, 2005) A sensibilidade e a precisão são obtidas através do somatório da precisão e da sensibilidade de todas as classes conforme as fórmulas 21 e 22:

$$p = \frac{VP}{VP + FP} = \frac{\sum_{i=1}^{M} VP_i}{\sum_{i=1}^{M} VP_i + FP_i} \tag{21}$$

$$r = \frac{VP}{VP + FN} = \frac{\sum_{i=1}^{M} VP_i}{\sum_{i=1}^{M} VP_i + FN_i} \quad (22)$$

Onde M é o número total de classes.

Para então calcular-se a média da Medida-F *Micro-averaged* conforme a fórmula 23:

$$F(Micro - averaged) = \frac{2rp}{r + p} \quad (23)$$

*6.1.5.2*Medida-F *Macro-averaged*

No Macro-averaged F-Measure, a Medida-F é primeiramente calculada localmente em cada classe e então é faz-se o cálculo da média de todas as classes. Então a partir da fórmula 24:

$$F(Macro - averaged) = \frac{\sum_{i=1}^{M} F_i}{M} \quad (24)$$

Onde M é o número total de classes.

Macro-averaged dá um peso igual para cada classe, independentemente de sua frequência, porém esta média acaba sendo mais influenciada pelo desempenho do classificador por classes que não que não possuem tantas amostras. (ÖZGÜR, 2005)

6.2 Resultados do NBC

A partir da análise das tabelas 6.1 e 6.2 percebe-se que o melhor resultado alcançado utilizando o método de classificação NBC foi obtido na rodada de número seis da quinta execução do CV com um custo de 459. E o custo médio dos modelos gerados pelo método de classificação NBC é de 801,98 e o desvio padrão 83,7353.

Tabela 6.1: Custo dos modelos para as cinco primeiras execuções do CV utilizando o NBC

	Custo CV 1	*Custo CV 2*	*Custo CV 3*	*Custo CV 4*	***Custo CV 5***
Rodada 0	811	943	888	734	822
Rodada 1	789	789	800	822	855
Rodada 2	866	833	745	811	712
Rodada 3	866	756	888	778	723
Rodada 4	778	767	822	899	910
Rodada 5	811	877	811	789	811
Rodada 6	932	745	833	844	**459**
Rodada 7	756	844	712	866	910
Rodada 8	767	734	833	690	745
Rodada 9	811	723	657	833	943

Fonte: Elaborado pelo autor

Tabela 6.2: Custo dos modelos para as execuções do CV 6 a 10 utilizando o NBC

	Custo CV 6	*Custo CV 7*	*Custo CV 8*	*Custo CV 9*	*Custo CV 10*
Rodada 0	822	789	844	712	855
Rodada 1	833	910	778	822	833
Rodada 2	976	866	646	679	767
Rodada 3	723	613	833	965	690
Rodada 4	789	646	844	932	833
Rodada 5	800	899	756	822	778
Rodada 6	921	877	789	811	899
Rodada 7	635	723	789	789	657
Rodada 8	767	844	866	767	921
Rodada 9	811	756	822	767	789

Fonte: Elaborado pelo autor

Na tabela 6.3 apresenta-se a matriz de confusão do teste do melhor modelo gerado com o NBC.

Tabela 6.3: Matriz de confusão do teste da sexta rodada da quinta execução do CV para o método de classificação NBC

	Classe 1	*Classe 2*	*Classe 3*	*Classe 4*	*Classe 5*
Classe 1	**58**	0	1	1	0
Classe 2	0	**44**	3	11	2
Classe 3	0	6	**25**	16	13
Classe 4	0	0	1	**51**	8
Classe 5	0	1	1	5	**53**

Fonte: Elaborado pelo autor

A partir da matriz de confusão do modelo é possível calcular a quantidade de falsos positivos, a precisão de cada classe e também a sensibilidade, os quais são apresentados na tabela 6.4.

Tabela 6.4: Métricas do modelo gerado na sexta rodada da quinta execução do CV para o método de classificação NBC

	Classe 1	*Classe 2*	*Classe 3*	*Classe 4*	*Classe 5*
Erros (FP)	0	7	6	33	23
Precisão	1	0,8627	0,8065	0,6071	0,6974
Sensibilidade	0,9667	0,7333	0,4167	0,85	0,8833
Medida-F	0,9831	0,7928	0,5495	0,7083	0,7794

Fonte: Elaborado pelo autor

A acurácia desse modelo é de 77%. A Medida-F utilizando o cálculo da média macro é F(Macro-Averaged) = 0,7626 e calculando a média micro é F(Micro-Averaged) = 0,7790.

6.3 Resultados do KNN

As tabelas 6.5 e 6.6 trazem os custos dos modelos de predição gerados utilizando o método de classificação KNN nas diversas execuções do CV. O KNN obteve um custo médio de 537,98 sendo o desvio padrão dos custos igual a 72,8044.

Percebe-se também, a partir da análise das tabelas, que o melhor resultado obtido pelo KNN foi alcançado na oitava rodada da sétima execução do CV com um custo igual a 338.

Tabela 6.5: Custo dos modelos para as cinco primeiras execuções do CV utilizando o KNN

	Custo CV 1	*Custo CV 2*	*Custo CV 3*	*Custo CV 4*	*Custo CV 5*
Rodada 0	591	525	492	547	503
Rodada 1	404	448	635	657	591
Rodada 2	569	492	602	569	470
Rodada 3	547	492	569	558	525
Rodada 4	481	492	470	536	536
Rodada 5	459	580	536	349	547
Rodada 6	635	503	525	470	481
Rodada 7	448	635	536	811	547
Rodada 8	525	624	481	503	481
Rodada 9	602	591	591	514	646

Fonte: Elaborado pelo autor

Tabela 6.6: Custo dos modelos para as execuções do CV 6 a 10 utilizando o KNN

	Custo CV 6	***Custo CV 7***	*Custo CV 8*	*Custo CV 9*	*Custo CV 10*
Rodada 0	569	624	536	569	679
Rodada 1	514	514	558	591	635
Rodada 2	668	591	382	415	492
Rodada 3	448	602	503	514	503
Rodada 4	580	580	492	613	503
Rodada 5	492	591	569	536	547
Rodada 6	701	591	569	503	514
Rodada 7	470	492	547	558	426
Rodada 8	514	**338**	580	459	503
Rodada 9	525	558	580	470	569

Fonte: Elaborado pelo autor

Na tabela 6.7 apresenta-se a matriz de confusão do melhor modelo de predição gerado com o KNN.

Tabela 6.7: Matriz de confusão do teste da oitava rodada da sétima execução do CV para o método de classificação KNN

	Classe 1	*Classe 2*	*Classe 3*	*Classe 4*	*Classe 5*
Classe 1	**59**	0	0	1	0
Classe 2	0	**50**	9	1	0
Classe 3	0	10	**38**	10	2
Classe 4	1	1	3	**47**	8
Classe 5	0	0	8	4	**48**

Fonte: Elaborado pelo autor

Na tabela 6.8 são apresentadas: a quantidade de falsos positivos, a precisão de cada classe e também a sensibilidade.

Tabela 6.8: Métricas alcançadas pelo modelo da oitava rodada da sétima execução do CV para o método de classificação KNN

	Classe 1	*Classe 2*	*Classe 3*	*Classe 4*	*Classe 5*
Erros(FP)	1	11	20	16	10
Precisão	0,9833	0,8197	0,6552	0,7460	0,8276
Sensibilidade	0,9833	0,8333	0,6333	0,7833	0,8
Medida-F	0,9833	0,8264	0,6441	0,7642	0,8136

Fonte: Elaborado pelo autor

A acurácia atingida por esse modelo de predição é de 80,67%. A Medida-F utilizando a média macro é F(Macro-Averaged) = 0,8063 e calculando a média micro é F(Micro-Averaged) = 0,8067.

6.4 Resultados do SVM

As tabelas 6.9 e 6.10 apresentam os custos dos modelos gerados pelas execuções do CV utilizando para a classificação o método SVM. Com base nessas tabelas pode-se calcular o custo médio dos modelos utilizados para testar a qualidade do SVM, sendo esse custo médio igual a 189,98 com um desvio padrão de 69,4537.

A partir da análise dessas tabelas, pode-se ver também que o melhor resultado do método de classificação SVM é gerado na segunda rodada da oitava execução do CV, atingindo um custo igual a 8.

Tabela 6.9: Custo dos modelos para as cinco primeiras execuções do CV utilizando o SVM

	Custo CV 1	*Custo CV 2*	*Custo CV 3*	*Custo CV 4*	*Custo CV 5*
Rodada 0	305	239	173	151	206
Rodada 1	140	107	217	239	261
Rodada 2	239	206	327	129	30
Rodada 3	195	85	162	118	151
Rodada 4	151	173	206	217	283
Rodada 5	140	239	162	140	195
Rodada 6	261	162	173	206	107
Rodada 7	74	195	184	184	283
Rodada 8	206	151	206	162	107
Rodada 9	217	228	173	195	206

Fonte: Elaborado pelo autor

Tabela 6.10: Custo dos modelos para as execuções do CV 6 a 10 utilizando o SVM

	Custo CV 6	*Custo CV 7*	***Custo CV 8***	*Custo CV 9*	*Custo CV 10*
Rodada 0	217	261	129	195	140
Rodada 1	195	228	173	85	239
Rodada 2	305	173	**8**	118	85
Rodada 3	63	173	261	250	129
Rodada 4	129	96	206	327	239
Rodada 5	151	305	250	184	228
Rodada 6	415	162	140	140	239
Rodada 7	41	151	206	228	162
Rodada 8	118	96	261	107	206
Rodada 9	217	184	173	206	338

Fonte: Elaborado pelo autor

Na tabela 6.11 apresenta-se a matriz de confusão do melhor modelo para o SVM.

Tabela 6.11: Matriz de confusão gerada pelo teste da segunda rodada da oitava execução do CV para o método de classificação SVM

	Classe 1	*Classe 2*	*Classe 3*	*Classe 4*	*Classe 5*
Classe 1	**60**	0	0	0	0
Classe 2	0	**58**	1	1	0
Classe 3	0	0	**50**	8	2
Classe 4	0	0	1	**55**	4
Classe 5	0	1	2	8	**49**

Fonte: Elaborado pelo autor

A partir da matriz de confusão do modelo é possível calcular a quantidade de falsos positivos, a precisão de cada classe e também a sensibilidade, os quais são apresentados na tabela 6.12.

Tabela 6.12: Métricas do modelo 2 da oitava execução do CV para o método de classificação SVM

	Classe 1	*Classe 2*	*Classe 3*	*Classe 4*	*Classe 5*
Erros(FP)	0	1	4	17	6
Precisão	1	0,9831	0,9259	0,7639	0,8909
Sensibilidade	1	0,9667	0,8333	0,9167	0,8167
Medida-F	1	0,9748	0,8772	0,8333	0,8522

Fonte: Elaborado pelo autor

A acurácia atingida por esse modelo é de 90,67%. A Medida-F utilizando a média macro é F(Macro-Averaged) = 0,9075 e calculando a média micro é F(Micro-Averaged) = 0,9067.

6.5 Comparação dos Resultados

A tabela 6.13 apresenta o menor custo atingido por cada técnica de classificação analisada, bem como os seus custos médios e o desvio padrão da média de custo dos modelos gerados por cada técnica. Percebe-se a partir da analise da tabela que o SVM obteve o menor custo e também a menor média de custo.

Tabela 6.13: Menor custo, custo médio e desvio padrão do custo para cada técnica

	NBC	*KNN*	*SVM*
Menor Custo	459	338	**8**
Custo Médio	801,98	537,98	**189,98**
Desvio Padrão	85,7353	72,8044	69,4537

Fonte: Elaborado pelo autor

A tabela 6.14 apresenta a acurácia e as médias Macro-averaged e Micro-averaged obtidas pelos melhores modelos de cada técnica de classificação. A partir da analise da tabela, é possível ver que a técnica SVM obteve os melhores resultados seguida pela técnica KNN e por fim pela NBC.

Tabela 6.14: Acurácia e médias da Medida-F

	NBC	*KNN*	*SVM*
Acurácia	77%	80,67%	**90,67%**
F(Macro-averaged)	0,7626	0,8063	**0,9075**
F(Micro-averaged)	0,7790	0,8067	**0,9067**

Fonte: Elaborado pelo autor

A tabela 6.15 apresenta o total de falsos positivos que o melhor modelo de cada técnica de classificação obteve para cada classe de gestos. A técnica SVM registrou o menor número de falsos positivos em quatro das cinco classes. Já a técnica KNN registrou o melhor resultado na classe 4. E a técnica NBC apenas empatou com a SVM na classe 1, zerando o número de falsos positivos.

Tabela 6.15: Total de Erros (FP) obtida pelo melhor modelo de cada técnica de classificação

	NBC	*KNN*	*SVM*
Classe 1	**0**	1	**0**
Classe 2	7	11	**1**
Classe 3	6	20	**4**
Classe 4	33	**16**	17
Classe 5	23	10	**6**

Fonte: Elaborado pelo autor

A tabela 6.16 apresenta a precisão obtida em cada classe de gestos pelo melhor modelo gerado pelas técnicas de classificação. Percebe-se que a técnica SVM obteve a melhor precisão em todas as classes de gestos, empatando com a NBC na classe 1.

Tabela 6.16: Precisão obtida pelo melhor modelo de cada técnica de classificação

	NBC	*KNN*	*SVM*
Classe 1	**1**	0,9833	**1**
Classe 2	0,8627	0,8197	**0,9831**
Classe 3	0,8065	0,6552	**0,9259**
Classe 4	0,6071	0,7460	**0,7639**
Classe 5	0,6974	0,8276	**0,8909**

Fonte: Elaborado pelo autor

A tabela 6.17 apresenta a sensibilidade que cada técnica de classificação obteve com o melhor modelo gerado em cada uma das classes de gestos. A partir da analise da

tabela, vê-se que a SVM obteve a melhor sensibilidade nas classes 1 a 4, porém obteve o pior resultado na classe 5, classe essa em que a NBC obteve a melhor sensibilidade.

Tabela 6.17: Sensibilidade obtida pelo melhor modelo de cada técnica de classificação

	NBC	*KNN*	*SVM*
Classe 1	0,9667	0,9833	**1**
Classe 2	0,7333	0,8333	**0,9667**
Classe 3	0,4167	0,6333	**0,8333**
Classe 4	0,85	0,7833	**0,9167**
Classe 5	**0,8833**	0,8	0,8522

Fonte: Elaborado pelo autor

A tabela 6.18 apresenta, para cada classe de gestos, a Medida-F obtida pelas técnicas de classificação para o melhor modelo gerada. A técnica SVM obteve os melhores resultados em todas as classes, ficando a KNN em segundo lugar em todas as classes.

Tabela 6.18: Medida-F obtida pelo melhor modelo de cada técnica de classificação

	NBC	*KNN*	*SVM*
Classe 1	0,9831	0,9833	**1**
Classe 2	0,7928	0,8264	**0,9748**
Classe 3	0,5495	0,6441	**0,8772**
Classe 4	0,7083	0,7642	**0,8333**
Classe 5	0,7794	0,8136	**0,8522**

Fonte: Elaborado pelo autor

A partir da análise dos custos apresentados na tabela 6.13 pode-se afirmar que entre as três técnicas de classificação analisadas a que obteve o melhor resultado é a técnica *Support Vector Machine* seguida pelo *K Nearest Neighbor* e a técnica que obteve a pior qualidade de predição foi a *Naive Bayes Classifier*.

Pela analise das 6.14 a 6.18, vê-se que a técnica SVM obteve os melhores resultados em praticamente todas as métricas avaliadas.

A análise dos resultados permite também observar que os três classificadores obtiveram uma boa Medida-F para a classe que representa o número um, fato esse que pode estar relacionado ao fato de essa classe ser a mais distinta entre as classes trabalhadas.

7 CONCLUSÕES E TRABALHOS FUTUROS

O trabalho apresentou a aplicação de uma técnica de reconhecimento de gestos utilizando um *smartphone* capaz de reconhecer cinco classes de gestos. Utilizando o espaço de cores YC_bC_r para fazer a segmentação da mão do restante da imagem através de limiar da cor da pele humana e utilizando os momentos invariantes de Hu para fazer a classificação dos gestos.

Fez a avaliação de três técnicas de classificação distintas, sendo elas o *Naive Bayes Classifier*, *K Nearest Neighbor* e o *Support Vector Machine*. Utilizando uma biblioteca de imagens composta por 3000 imagens de cinco classes, totalizando 600 imagens de cada classe e executando dez vezes o *K-Fold Cross Validation* com um k igual a 10 para então calcular diversas métricas de avaliação como acurácia, precisão, sensibilidade, medida-F, erros de falsos positivos.

Após o calculo das métricas de avaliação foi feito um comparativo entre as diferentes técnicas de classificação e concluiu-se que a utilização do *Support Vector Machine* para fazer a classificação dos gestos é a técnica que obtém o melhor resultado dentre as três analisadas obtendo resultados satisfatórios de classificação.

7.1 Trabalhos Futuros

Um trabalho futuro a ser realizado é aumentar o número de classes reconhecidas pelo sistema, fazendo com que reconheça o alfabeto datilológico da LIBRAS, o qual é apresentado na figura 1.1. Alfabeto este que é um suplemento das linguagens de sinais. Em que a língua escrita serve de base para que se escreva através de sinais. Esse alfabeto é utilizado para digitar nomes, objetos ou palavras que não possuam sinais. Podendo fazer com que o aplicativo possa ser útil para deficientes auditivos, por exemplo.

As classes de gestos utilizadas no trabalho são visualmente bem distintas, sendo necessário analisar o comportamento dos quatro primeiros momentos invariantes de Hu como descritores das imagens ao trabalhar-se com classes de gestos que sejam visualmente parecidas. Caso o resultado não seja satisfatório, faz-se necessário aumentar o número de descritores das imagens. Uma opção seria avaliar a utilização de outras técnicas de momentos como os momentos de Tchebichef, Legendre, Zernik ou Krawtchouk. Os quais poderiam ser utilizados em combinação com os momentos invariantes de Hu.

REFERÊNCIAS

PEREIRA, A. C. C. Gesto. Belo Horizonte. Disponível em: <http://psicolinguistica.letras.ufmg.br/wiki/index.php/Gesto >. Acesso em: jun. 2013.

CHEN, Q.; GEORGANAS, N. D.; PETRIU, E. M. Real-time Vision-based Hand Gesture Recognition Using Haar-like Features. **Instrumentation and Measurement Technology Conference – IMTC 2007**, Ottawa, IEEE, 2007.

DU, H.; TO, T. **Hand Gesture Recognition Using Kinect**, Boston, Boston University: Department of Electrical and Computer Engineering, 2011.

CHAKRABORTY, P.; SARAWGI, P.; MEHROTRA, A.; AGARWAL, G., PRADHAN, R. Hand Gesture Recognition: A Comparative Study. **Proceedings of the International MultiConference of Engineers and Computer Scientists 2008**, Hong Kong, ISBN, 2008.

LIU, Y.; YIN, Y.; ZHANG, S. Hand Gesture Recognition Based on HU Moments in Interaction of Virtual Reality. **4th International Conference on Intelligent Human-Machine Systems and Cybernetics**, Qingdao, IEEE, 2012.

ZOU, Z. **Computer human interaction using hand gestures**. Tese (Mestrado em Engenharia) – School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong. 2011

AWCOCK, G. W.; THOMAS, R. Feature Extraction. In: **Applied Image Processing**. 1.ed. Londres: Department of Electrical and Electronic Engineering University of Brighton, 1995. p.162-165.

PRATT, W. K. Shape Analysis. In: **Digital Image Processing**. 4.ed. Los Altos: Wiley, 2007. p. 631-642.

JAIN, A. K. Image Analysis and Computer Vision. In: **Fundamentals of Digital Image Processing**. 1.ed. Davis: University of California, 1989. 569 p. 377-381.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. Model Assessment and Selection. In: **The Elements of Statistical Learning Data Mining, Inference, and Prediction**. 2.ed. Standford: Springer, 2008. p. 241-248.

MITCHELL, T. M. Instance-Based Learning. In: **Machine Learning**. 1.ed. McGraw-Hill Science/Engineering/Math, 1997. p. 231-236.

NORVIG, P.; RUSSEL, S. Learning from Examples. In: **Artificial Intelligence A Modern Approach**. 3.ed. Prentice Hall, 2009. p. 744-748.

REFAEILZADEH, P.; TANG, L; LIU, H. **Cross Validation**. Arizona State University, 2008.

ÖZGÜR, A.; ÖZGÜR, L; GÜNGÖR, T. **Text Categorization with Class-Based and Corpus-Based Keyword Selection**. Istambul, Springer-Verlag, 2005.