

DADOS DA LISTA: Extração de características Surf e Hog

Matéria: Processamento digital de imagens (PDI) **Código:** CMP1084-A01

Alunos:

Higor Alves Ferreira **Matrícula:** 20161003305438

Lucas Macedo da Silva **Matrícula:** 20161003305470

Vitor de Almeida Silva **Matrícula:** 20161003305497

DEFINIÇÕES TEÓRICAS

Características

Uma característica (*features*) é um pedaço de informação, o qual é relevante para solucionar problemas computacionais relativos a certas aplicações. Em relação a processamento de imagens, características podem especificar estruturas tais como pontos, alvos ou objetos (TYAGI, 2019).

As características podem surgir, também, por meio de operações entre vizinhos de pixels, ou da detecção de características em uma imagem. Deste modo, as características podem ser separadas em dois tipos (TYAGI, 2019):

- **Características de locais específicos da imagem:** Também conhecidas como *keypoint features*, essas características são encontradas em locais específicos da imagem como cantos ou picos de montanhas;
- **Características que podem ser combinadas:** São características que podem ser combinadas por meio de sua orientação e aparência local. Tais características, são chamadas de *edges*, podendo ser utilizadas para classificação, detecção de limites e processos de oclusão em objetos de imagens estáticas ou sequenciais.

A partir destes conceitos, surge a ideia de pontos e áreas de interesse. Um ponto de interesse, pode ser tanto uma coordenada onde há uma mudança abrupta nos limites de um objeto, quanto um ponto de interseção entre dois segmentos edge. Já a área de interesse, refere se a um conjunto de pontos de interesse, podendo estes, serem gerados por meio da análise dos vizinhos de um ponto de interesse inicial (TYAGI, 2019).

Para a realização dessa análise, surge a necessidade de se organizar as características de uma imagem em algum tipo de estrutura, para tanto, existem dois tipos mais conhecido segundo Gonzalez, Woods (2010):

- **Vetor de características:** São estruturas vetoriais $N \times 1$ (vetor coluna), que são gerados por meio da extração de características de imagens associando n valores (descritores) a cada índice X_i de linha do vetor. A Figura 1 mostra o formato do vetor.

Figura 1: vetor de características genérico

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Fonte: (GONZALEZ, WOODS, 2010)

- **Matriz de característica:** São estrutura vetoriais $n \times m$, que contém valores de descritores de dados em cada ponto $M_{n,m}$ da matriz. Tais matrizes, por sua vez, podem ser geradas por meio da concatenação em coluna de K vetores de características. A Figura 2 mostra um exemplo de matriz de características genérica.

Figura 2: Matriz de características

$$\mathbf{M}_{n,m} = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} & \dots & M_{1,m} \\ M_{2,1} & . & . & & . \\ M_{3,1} & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ M_{n,1} & M_{n,2} & M_{n,3} & \dots & M_{n,m} \end{bmatrix}$$

Fonte: Autoral

Speeded Up Robust Features (SURF)

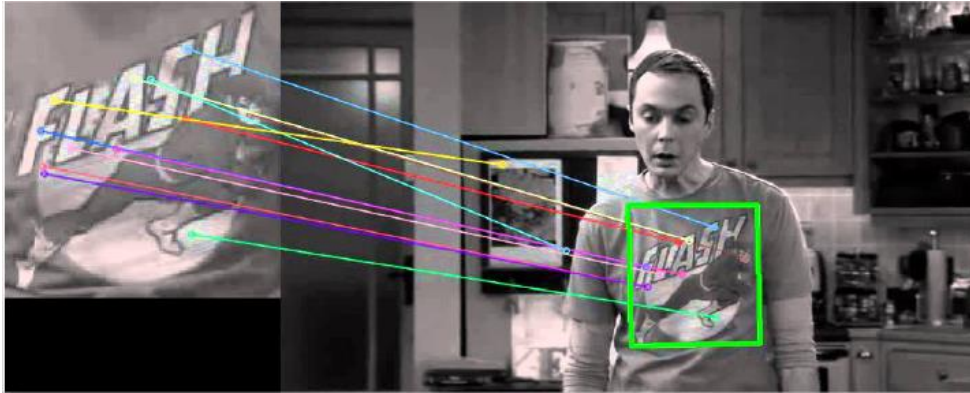
Se aproveitando das definições supracitadas, existem diversos tipos de detectores e extratores de característica, como é o caso do método SURF (points = detectSURFFeatures(I) (MATHWORKS,2020)). Este método é um algoritmo rápido para representação de similaridades invariantes e locais para comparação de imagens. O interesse principal do SURF, é realizar aproximações rápidas de operadores usando *box filters*, deste modo ele possibilita a criação de aplicações de tempo real, tal como *tracking* e reconhecimento de objetos (TYAGI, 2019).

Com isso, o surf é composto por 2 passos básicos segundo Tyagi (2019):

- **Extração de características:** Realiza uma aproximação do ponto de interesse por meio de uma matriz hessiana básica. Essa matriz então, é gerada por meio dos pixels das imagens, a qual, posteriormente irá ser usada para realização de uma convolução com uma derivada gaussiana de segunda ordem. Essa operação exige um baixo custo computacional, porém, gera uma fraqueza em relação a imagens rotacionada em ângulos múltiplos de $\pi/4$;
- **Descrição de características:** essa operação é realizada em dois passos. O primeiro, consiste de se produzir círculos utilizando as informações ao redor dos pontos de interesse. Já o segundo, consiste da construção de um quadrado alinhado com a orientação selecionada e extrair a descrição Surf disso.

A Figura 3 mostra um exemplo da aplicação do método SURF em uma imagem. Nesta, foram utilizadas a extração de característica e a descrição onde foram marcados os pontos de interesse.

Figura 3: Aplicação do método SURF exemplo

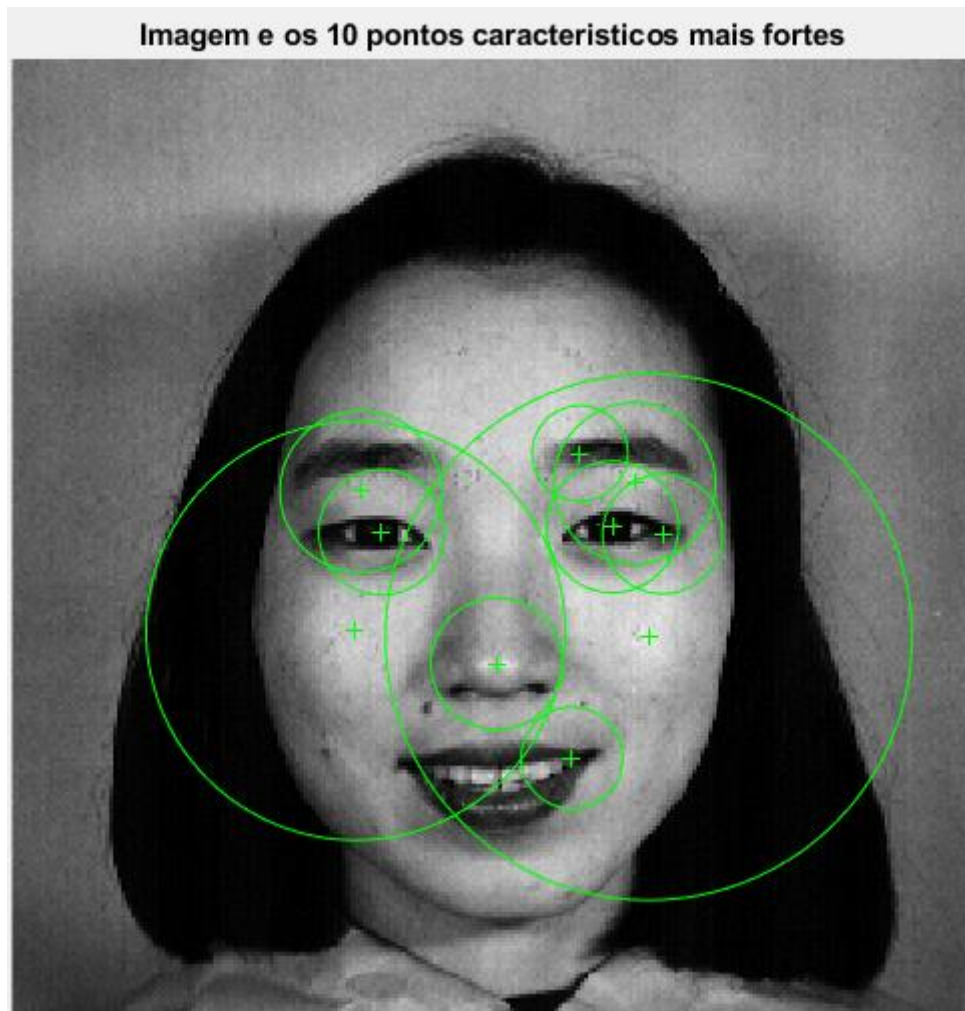


Fonte: (TYAGI, 2019)

Para extrair características utilizando a técnica SURF foi empregada a função `detectSURFFeatures` do Matlab. Essa função recebe como uma imagem em escala de cinza e retorna um objeto contendo informações sobre as características detectadas empregando a técnica SURF. Após foram selecionados os 5 pontos que contém as métricas mais fortes, para tanto foi empregada a função `selectStrongest` do Matlab. Por fim desses pontos foram extraídas as características utilizando a função `extractFeatures` do Matlab.

A figura a seguir ilustra a aplicação do processo supracitado em uma das imagens do dataset. Foram escolhidos os 10 pontos característicos mais fortes nessa imagem.

Figura 4: Exemplo de aplicação da técnica SURF para detectar características nas imagens do dataset



Fonte: Autoral

Histogram of Oriented Gradient (HOG)

O *Histogram of Oriented Gradient* (HOG) é um descritor de características que é utilizado para extrair características de imagens (SINGH, 2020). No campo da computação visual ele é usado para o processamento de imagens para fins de detecção de objetos (AHMAD, 2020). A técnica usa a direção dos gradientes como características. Os gradientes são úteis pois a sua magnitude é grande nas bordas e dos cantos, além disso as bordas e cantos possuem muitas informações do objeto (MALLICK, 2020).

Algumas características do descritor HOG são (SINGH, 2020):

- O descritor foca na forma ou estrutura do objeto. Além de detectar a borda o algoritmo também detecta a direção da borda. Isso é possível devido a detecção do gradiente e da orientação das bordas que o algoritmo faz;
- A imagem é dividida em regiões menores e para cada região, são calculados o gradiente e a orientação;
- Para cada região é criado um histograma. Os histogramas são criados usando os gradientes e orientações dos valores de pixel.

Mallick (2020) apresenta os passos para do HOG são eles:

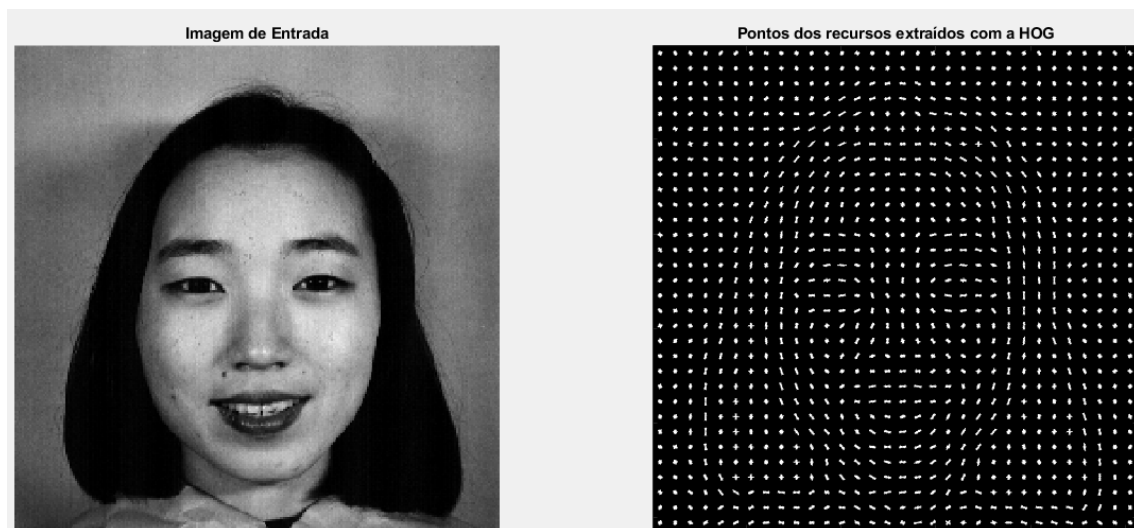
1. Pré-processamento: Aplicação de alguma operação na imagem, como por exemplo, o redimensionamento a conversão para a escala de cinza.
2. Cálculo dos gradientes das imagens: São calculados os gradientes horizontais e verticais da imagem. Para realizar esse procedimento, pode-se utilizar filtros de detecção de bordas. Depois a magnitude e direção do gradiente é calculada. Para cada pixel da imagem o gradiente tem uma magnitude e direção.
3. Cálculo do histograma do gradiente em células 8x8: A imagem é dividida em células 8x8 e para cada célula (de dimensões 8x8) é calculado o gradiente. Após é então calculado o histograma de gradiente. O histograma possui 9 colunas, cada coluna representa um ângulo, 0° , 20° , ..., 180°
4. Normalização em blocos 16x16: Essa etapa busca evitar que histograma seja afetado pela luz. A normalização ocorre em janelas de 16x16 blocos.
5. Cálculo do vetor HOG de características: Os vetores já normalizados são combinados em um único vetor. Para visualizar o HOG em uma imagem os histogramas em blocos 9x1 são plotados 8x8 células.

No presente trabalho para extração das características utilizando a técnica HOG foi empregada a função `extractHOGFeatures` do Matlab. Essa função recebe como parâmetro uma imagem em escala de cinza e retorna um vetor $1 \times n$ com n recursos extraídos usando a técnica HOG e um vetor contendo a localização desses pontos na imagem (MATHWORKS, 2020). Essas

características codificam informações de forma local das regiões dentro de uma imagem.

A Figura a seguir ilustra a aplicação dessa função em uma das imagens do dataset. Na imagem da esquerda a imagem original e na figura da direita a plotagem dos pontos dos recursos extraídos.

Figura 5: Imagem de entrada e plot com os pontos dos recursos extraídos utilizando a técnica HOG



Fonte: Autoral

Definição do problema

Para o desenvolvimento do presente trabalho foi utilizada a base de dados JAFFE (Japanese Female Facial Expression) (LYONS, KAMACHI, GYOBA; 2020) que contém imagens de 10 modelos japonesas expressando 6 sentimentos e a expressão neutra. Essa base de dados na presente versão contém 140 imagens de 10 modelos diferentes.

Sendo assim, para a realização do processo de classificação das emoções na base de dados dada, foram separações das imagens. Tal separação, consistiu da criação de duas pastas principais, descritas a seguir:

- **Pessoas:** Pasta que se os indivíduos por classe se baseando no nome das imagens, foram um total de 10 pastas com 14 imagens

cada. As classes separadas são: KA, KL, KM, KR, MK, NA, NM, TM, UY e YM;

- **Emoções:** Similar a anterior, a pasta de emoções separa as imagens em 7 classes, que consistem de pastas com os respectivos nomes das emoções, cada pasta contém 20 imagens. As emoções separadas foram: Raiva (Angry), Desgosto (Disgusting), Medo (Fear), Felicidade (Happy), Tristeza (Sad) e Surpresa (Surprise), bem como a expressão Neutra (Neutral). Essas foram as classes utilizadas para o processo de classificação.

Algoritmo de Classificação

Para realizar a classificação das imagens, foi utilizado a Análise de Componentes Principais (em inglês Principal Component Analysis – PCA). A PCA, consiste em uma formulação matemática que permite reduzir a dimensionalidade dos dados e permite a identificação de padrões nos dados de maneira que suas semelhanças e diferenças sejam detectadas (SANTO, 2012). Sendo assim, ela pode ser usada como um algoritmo de classificação.

No presente trabalho o código utilizado para a classificação foi o mesmo disponibilizado pelo Professor na plataforma Classroom, exceto por algumas alterações para a aplicação no problema. Desta forma, as funções **Classificar** e **GerarPCs** não foram alteradas. Já as demais funções foram alteradas para o uso no desenvolvimento do trabalho.

A função **lerimgs** foi alterada, obtendo-se a seguinte função representada a seguir.

```
function [label_treino, label_teste, dados_treino, dados_teste ....  
    img_treino, img_teste, qtd_classes] = lerImgs(porcentagem, path,  
extensao, tipo)  
    % Parametros  
    %   porcentagem -> Porcentagem de dados que serão da classe de  
treino  
    %   path -> Caminho da pasta que contém as subpastas que são as  
classes  
    %   extensao -> Extensão das imagens  
    %   Tipo -> Indica o tipo de operação que será empregada nas  
imagens  
    % Retornos
```



```

    % label_treino -> Vetor auxiliar com as labels (classes) da
parte da
    % parte do banco que será usada como treino
    % label_teste -> Vetor auxiliar com as labels (classes) da
parte da
    % parte do banco que será usada como teste
    % dados_treino -> Dados que serão usados como treino da PCA
    % dados_teste -> Dados que serão usados como teste da PCA
    % qtd_classes -> Quantidade de classes do banco de dados

dados_treino = [];
dados_teste = [];
label_treino = [];
label_teste = [];
img_treino = [];
img_teste = [];

% Acessa as pastas
S = dir(fullfile(path));

% Percorre cada pasta que contém as classes
qtd_classes = numel(S) - 2;
for pasta = 3:numel(S)
    % Concatena as strings para formar o nome de cada pasta
    pasta_classe = S(pasta).name;
    caminho = strcat(S(pasta).folder, '\', pasta_classe, '\');
    classe = dir(fullfile(caminho, strcat('*', extensao)));

    % Quantidade treino e teste
    qtd_dados = (size(classe, 1));

    % Separa a porcentagem de treino
    qtd_treino = round((qtd_dados * porcentagem) / 100);
    i = 1;
    for i = 1: qtd_treino

        w = fullfile(caminho, classe(i).name);
        x = imread(w);

        % Faz o Surf
        if (tipo == "SURF")
            y = SURF(x);

        end

        % Faz o HOG
        if (tipo == "HOG")
            y = HOG(x);
        end

        if (tipo == "SIMPLES")
            y = SIMPLES(x);
        end
    end
end

```

```

        %y = reshape(x, [size(x,1)*size(x,2),1]);
        dados_treino = [dados_treino, y];
        label_treino = [label_treino, string(pasta_classe)];
        img_treino = [img_treino, string(classe(i).name)];
    end
    % Separa a porcentagem de teste
    for j = i+1: qtd_dados
        w = fullfile(caminho, classe(j).name);
        x = imread(w);

        % Faz o Surf
        if (tipo == "SURF")
            y = SURF(x);
        end

        % Faz o HOG
        if (tipo == "HOG")
            y = HOG(x);
        end

        if (tipo == "SIMPLES")
            y = SIMPLES(x);
        end

        dados_teste = [dados_teste, y];
        label_teste = [label_teste, string(pasta_classe)];
        img_teste = [img_teste, string(classe(j).name)];
    end
end

% SURF
function [y] = SURF(x)
    points = detectSURFFeatures(x);
    teste2 = points.selectStrongest(5);
    teste2 = extractFeatures(x, teste2);
    y = gpuArray(reshape(teste2, [size(teste2,1)*size(teste2,2),1]));
end

% HOG
function [y] = HOG(x)
    [y, ~] = extractHOGFeatures(x);
    y=y';
end

% SIMPLES
function [y] = SIMPLES(x)
    % x = rgb2gray(x);
    % x = imresize(x, [100 100]);
    y = gpuArray(reshape(x, [size(x,1)*size(x,2),1]));
end

```

Ela recebe como parâmetros:

- **porcentagem:** Porcentagem de dados que serão da classe de treino. É um inteiro entre 0 e 100, onde 0 representa 0% e 100 representa 100%. Assim porcentagem igual a 70 indica que 70% dos dados serão para treino e 30% serão para teste;
- **path:** Pasta que contém as demais pastas que separam as classes.
- **extensao:** Refere-se a extensão das imagens;
- **tipo:** Indica que técnica será empregada nas imagens. Podem ser realizados três tipos de operações:
 - **Simples:** Parâmetro tipo igual a 'SIMPLES', nesse modo de operação será empregada apenas a classificação com PCA;
 - **Hog:** Parâmetro tipo igual a 'HOG', nesse modo de operação será empregada a técnica HOG;
 - **Surf:** Parâmetro tipo igual a SURF, nesse modo de operação será empregada a técnica SURF.

A função retorna:

- **label_treino:** Vetor auxiliar contendo a classe dos dados de treino;
- **label_teste:** Vetor auxiliar contendo a classe dos dados de teste;
- **dados_treino:** Estrutura de dados contendo os dados separados para treino;
- **dados_teste:** Estrutura de dados contendo os dados separados para teste;
- **img_treino:** Vetor contendo o nome das imagens utilizadas para o treino. Os índices da classe representada no vetor label_treino estão relacionadas com as imagens, ou seja, a classe i no vetor label_treino representa a classe da imagem i do vetor img_treino;
- **img_teste:** Vetor contendo o nome das imagens utilizadas para o treino. Os índices da classe representada no vetor label_teste estão relacionadas com as imagens, ou seja, a classe i no vetor label_teste representa a classe da imagem i do vetor img_teste;
- **qtd_classes:** Quantidade de classes encontradas na pasta raíz.

O método de separação é simples, as primeiras imagens $n\%$ são separadas para teste e as $(100 - n)\%$ são separadas para treino.

O primeiro passo da função é a inicialização das variáveis, depois é contabilizada a quantidade de classes baseadas na quantidade de pastas que a pasta raiz possui. É importante que, as classes estejam previamente separadas em pasta, no presente trabalho elas foram separadas manualmente. Depois, para cada pasta presente na pasta raiz são separados os dados, esses dados, são então, salvos e o nome de suas classes (o mesmo nome da pasta) são também armazenadas.

A função **ProjetarAmostra** teve apenas a linha que redimensiona os dados comentada. Essa etapa foi transferida para a função `lerImgs`. A função alterada se encontra definida a seguir.

```
%Projeta uma nova amostra no espaço vetorial P
%Entrada:
%   x = amostra a ser projetada
%   mn = média de cada coluna da matriz de dados
%   P = dados no novo espaço vetorial (autofaces no caso de imagens)
%Saída:
%   x = amostra no novo espaço vetorial
function x = ProjetarAmostra(x,mn,P)
    %x = reshape(x,[size(x,1)*size(x,2),1]);
    x = reshape(x,[size(x,1)*size(x,2),1]);
    x = double(x) - mn;
    x = P' * x;
end
```

A função **main** foi remodelada de forma a chamar a função `lerImgs` várias vezes, classificar os dados e contar os erros e acertos. Bem como apresentar as métricas de desempenho no formato de gráficos. As métricas utilizadas foram: porcentagem de acertos; quantidade de erros e acertos por classe; proporção de treino, acertos e erros; e por fim, a média de erros e acertos.

A função se encontra definida a seguir:

```
clear all;
close all;
clc;

% Parametros
% porcentagem -> Porcentagem de dados que serão da classe de treino
% path -> Caminho da pasta que contém as subpastas que são as classes
```

```

% extensao -> Extensão das imagens
% tipo, tipo de algoritmos para extração de características
%tipo = "SURF";
%tipo = "HOG";
porcentagem = 70;
path = 'D:\Estudos\Eng. de Computação\Processamento Digital de
Imagens\Listas\Lista 6\Pessoas';
extensao = '.tiff';
tipo = "HOG";

[label_treino, label_teste, dados_treino, dados_teste ....
img_treino, img_teste, qtd_classes] = funcao(path ,porcentagem,
extensao , tipo);

function [label_treino, label_teste, dados_treino, dados_teste ....
img_treino, img_teste, qtd_classes] = funcao(path, porcentagem,
extensao, tipo)
% Parametros
%   porcentagem -> Porcentagem de dados que serão da classe de
treino
%   path -> Caminho da pasta que contém as subpastas que são as
classes
%   extensao -> Extensão das imagens
% Retornos
%   label_treino -> Vetor auxiliar com as labels (classes) da
parte da
%   parte do banco que será usada como treino
%   label_teste -> Vetor auxiliar com as labels (classes) da
parte da
%   parte do banco que será usada como teste
%   dados_treino -> Dados que serão usados como treino da PCA
%   dados_teste -> Dados que serão usados como teste da PCA
%   qtd_classes -> Quantidade de classes do banco de dados

% Para mostrar uma imagem certa e uma imagem errada
aux = 0;
aux1 = 0;

img_a = [];
img_e = [];

[label_treino, label_teste, dados_treino, dados_teste ....
img_treino, img_teste, qtd_classes] = lerImgs(porcentagem,
path, extensao, tipo);

% Qunatidade de amostras de teste
tam = size(dados_teste, 2);
qtd_dados = tam;

% Gera a PCA
[P PC mn] = GerarPCs(dados_treino);

% Vetores que contém a quantidade de acertos e erros por cada
classe
acertos = zeros(qtd_classes,1)';
erros = zeros(qtd_classes,1)';

acertos = zeros(qtd_classes, 1);

```

```

erros = zeros(qtd_classes, 1);

% Converte as categorias para números para ser possível contar a
% quantidade de erros e acertos
c = grp2idx(categorical(label_treino));

for i = 1: tam

    % Gera a amostra
    amostra = ProjotarAmostra(dados_teste(:, i), mn, P);
    d = round(Classificar(PC, amostra));

    indice_amostra = c(d);

    if isequal(label_teste(i), label_treino(d))

        acertos(indice_amostra) = acertos(indice_amostra) + 1;
        if(aux == 0)
            figure;
            pred =
imread(strcat(path, '\', label_teste(i), '\', img_teste(i)));
            real = imread(strcat(path, '\', label_treino(d),
'\', img_treino(d)));
            imshowpair(real, pred, 'montage');
            title(strcat('Classe Predita: ', label_treino(d), ....
' ] Classe Real: ', [' ', label_teste(i), '']));
            aux = aux + 1;
        end
        img_a = [img_a, img_teste(i)];
    else
        erros(indice_amostra) = erros(indice_amostra) + 1;
        if(aux1 == 0)
            figure;

            pred =
imread(strcat(path, '\', label_teste(i), '\', img_teste(i)));
            real = imread(strcat(path, '\', label_treino(d),
'\', img_treino(d)));
            imshowpair(real, pred, 'montage');
            title(strcat('Classe Predita: ', label_treino(d), ....
' ] Classe Real: ', [' ', label_teste(i), '']));
            aux1 = aux1 + 1;
        end
        img_e = [img_e, img_teste(i)];

    end
end

qtd_acerto = sum(acertos);
qtd_erro = sum(erros);

media_acerto = (100 * qtd_acerto) / qtd_dados;
media_erro = (100 * qtd_erro) / qtd_dados;

% Converte as labels para categorias
labels = categorical(unique(label_treino));

% ---- Plots ----

```

```

% Grafico de barras de porcentagem de acertos e erros
figure;
bar(labels, ([acertos erros]./[acertos + erros])*100);
legend('Acertos', 'Erros');
title(strcat('Porcentagem de Acertos e Erros por classe, Amostra
N=', num2str(tam)));
xlabel('Classe');
ylabel('Porcentagem de Acertos');

% Grafico de barras de quantidade de acertos e erros
figure;
bar(labels, [acertos erros]);
legend('Acertos', 'Erros');
title(strcat('Quantidade de Acertos e Erros por classe, Amostra
N=', num2str(tam)));
xlabel('Classe');
ylabel('Porcentagem de Acertos');

% Grafico de piza de porcentagem da amostra de acertos e erros
figure;
labels = {'Amostra', 'Acertos', 'Erros'};
pie([tam, qtd_acerto, qtd_erro]);
title(strcat('Proporção treino, acertos e erros, Amostra
N=', num2str(tam)));
legend(labels);

% Grafico de piza de porcentagem da amostra de acertos e erros
figure;
labels = {'Acertos', 'Erros'};
pie([qtd_acerto, qtd_erro]);
title(strcat('Proporção acertos e erros, Amostra
N=', num2str(tam)));
legend(labels);

% Quantidade de acerto e erro geral e média de erro e acerto
geral
fprintf('\nQuantidade de Acertos: %i\nPorcentagem de Acertos:
%.2f\n', qtd_acerto, media_acerto);
fprintf('\nQuantidade de Erros: %i\nPorcentagem de Erros:
%.2f\n', qtd_erro, media_erro);
end

```

As variáveis porcentagem, path, extensão e tipo no início da função definem a porcentagem de treino e teste, o caminho em que a pasta raiz está localizada, a extensão das imagens e o tipo de função a ser aplicada nas imagens, respectivamente. Dessa forma para cada questão essas variáveis são configuradas. Para todas as questões as imagens foram separadas em treino e teste na proporção de 50/50, 60/40 e 70/30.

Vale ressaltar que, o código completo também se encontra em apêndice.

DESENVOLVIMENTO

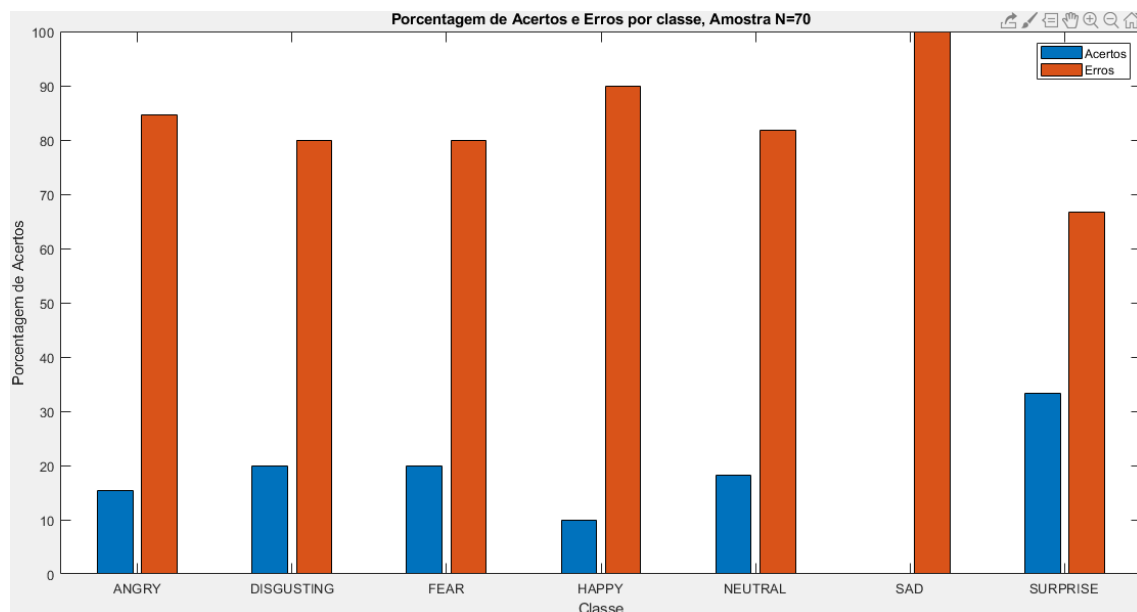
1. Utilize o algoritmo SURF para extrair características e treine a PCA para classificar as emoções.

Resposta:

- Configuração 50%/50% Amostra de 70 imagens para teste

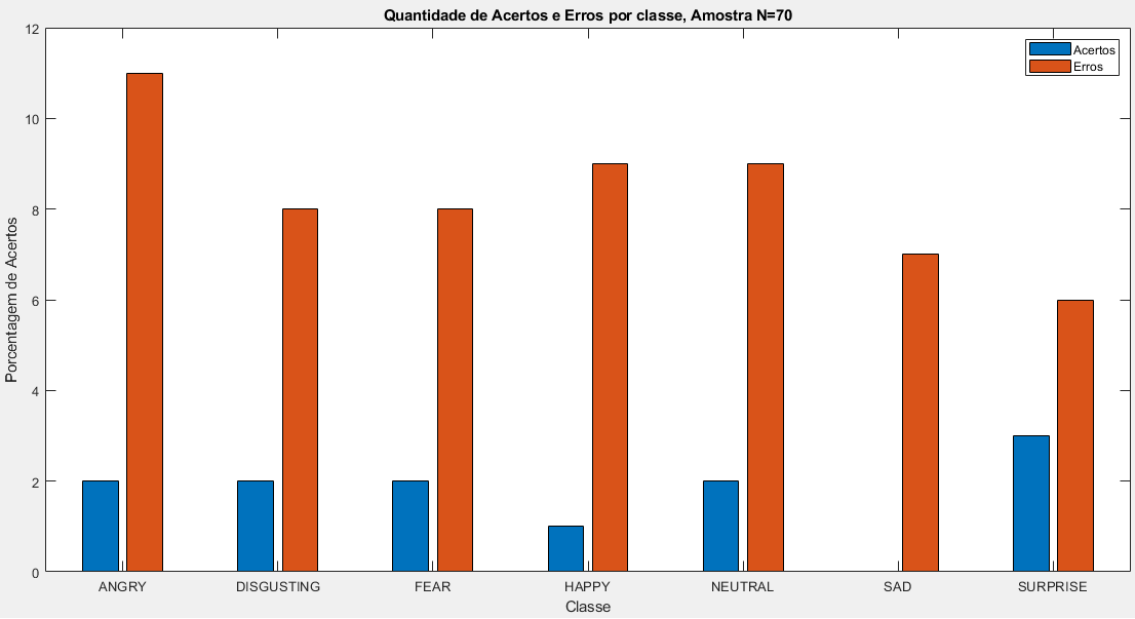
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 6: Porcentagem de Acertos e Erros por classe, Amostra N= 70 para classificação de emoções utilizando a técnica SURF como pré-processamento



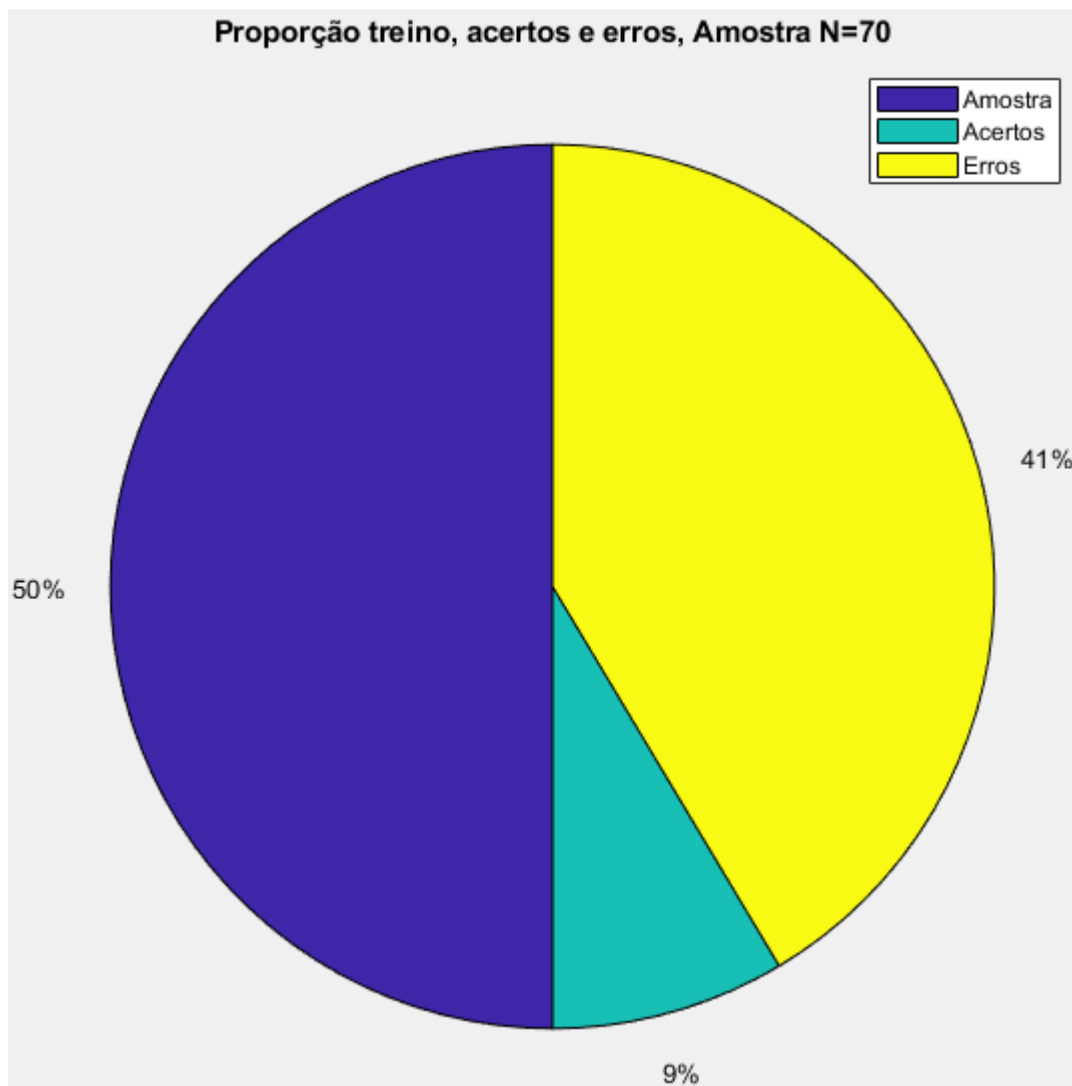
Fonte: Autoral

Figura 7: Quantidade de Acertos e Erros por classe, Amostra N= 70 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

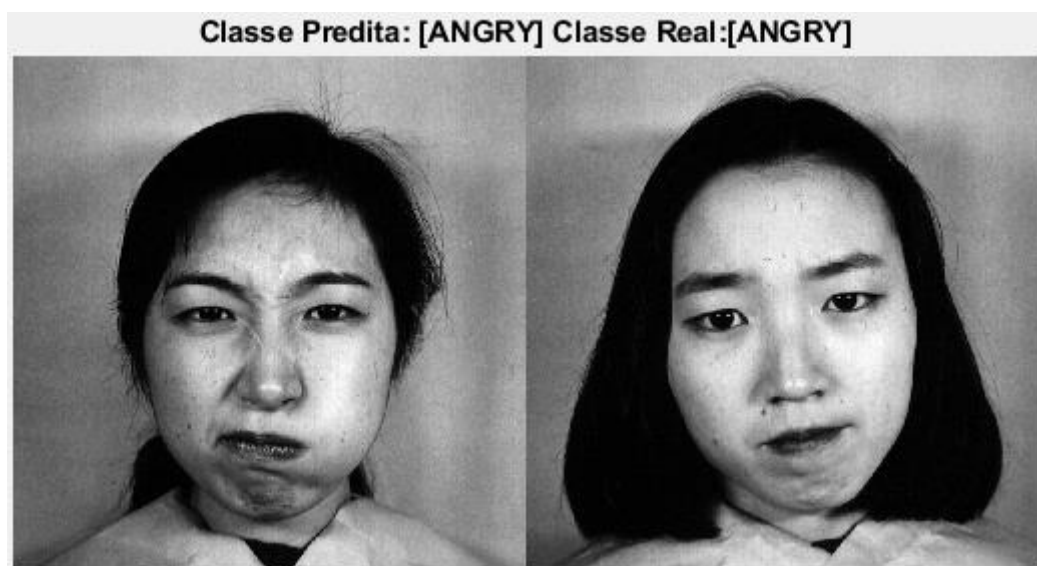
Figura 8: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 70 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

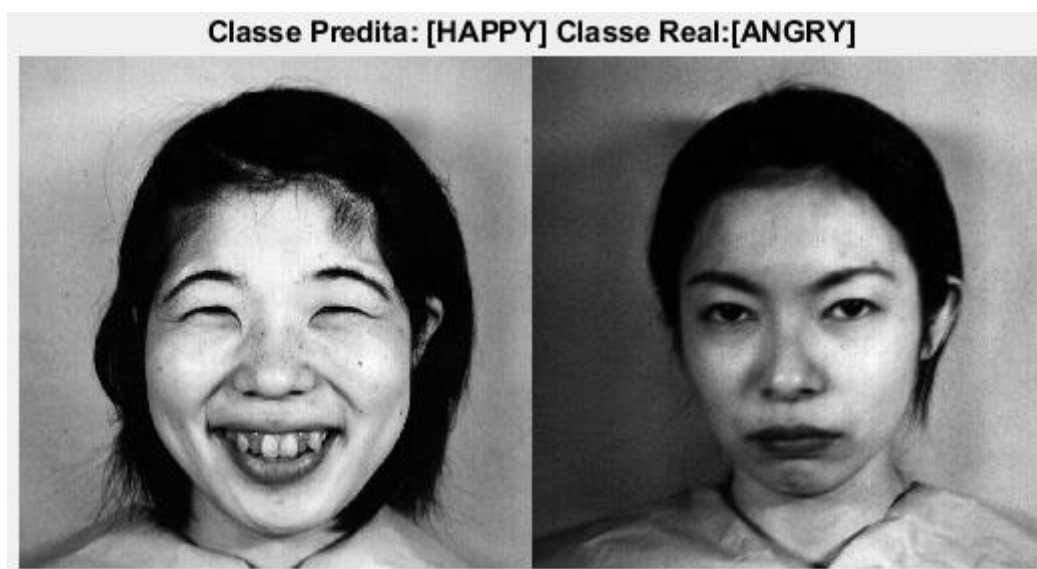
A porcentagem de acertos foi de 17.14% e a porcentagem de erros foi de 82.86%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 70 amostras para teste, o algoritmo classificou corretamente apenas 12. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 9: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 10: Exemplo de classificação incorreta realizada pelo algoritmo

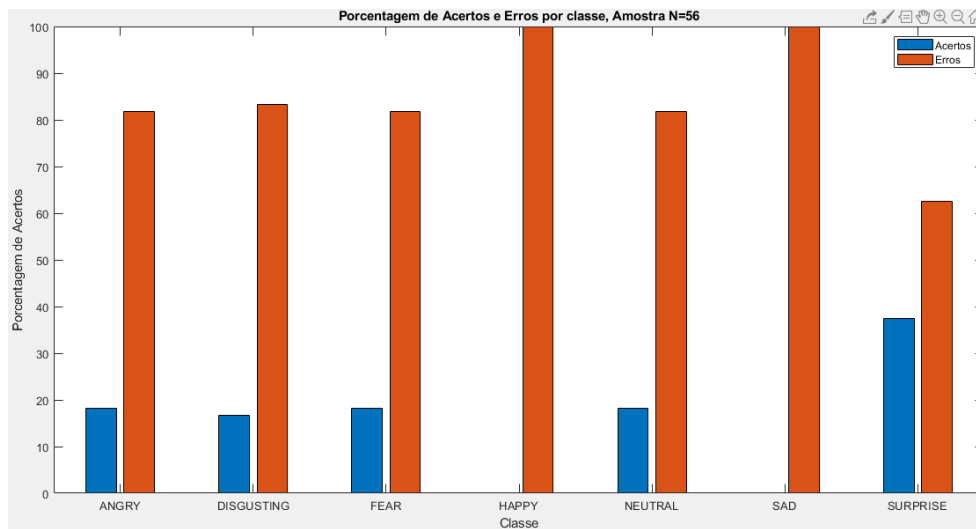


Fonte: Autoral

- Configuração 60%/40% Amostra de 56 imagens para teste

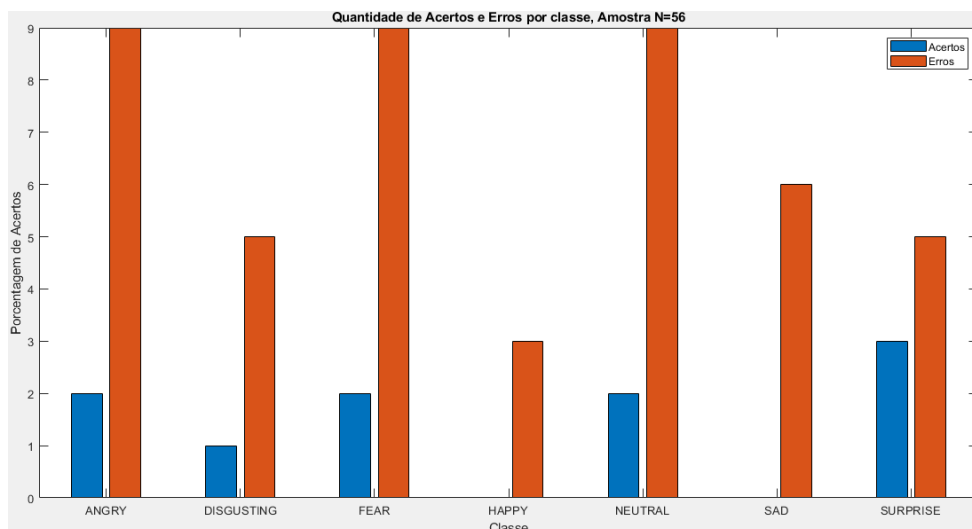
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 11: Porcentagem de Acertos e Erros por classe, Amostra N= 56 para classificação de emoções utilizando a técnica SURF como pré-processamento



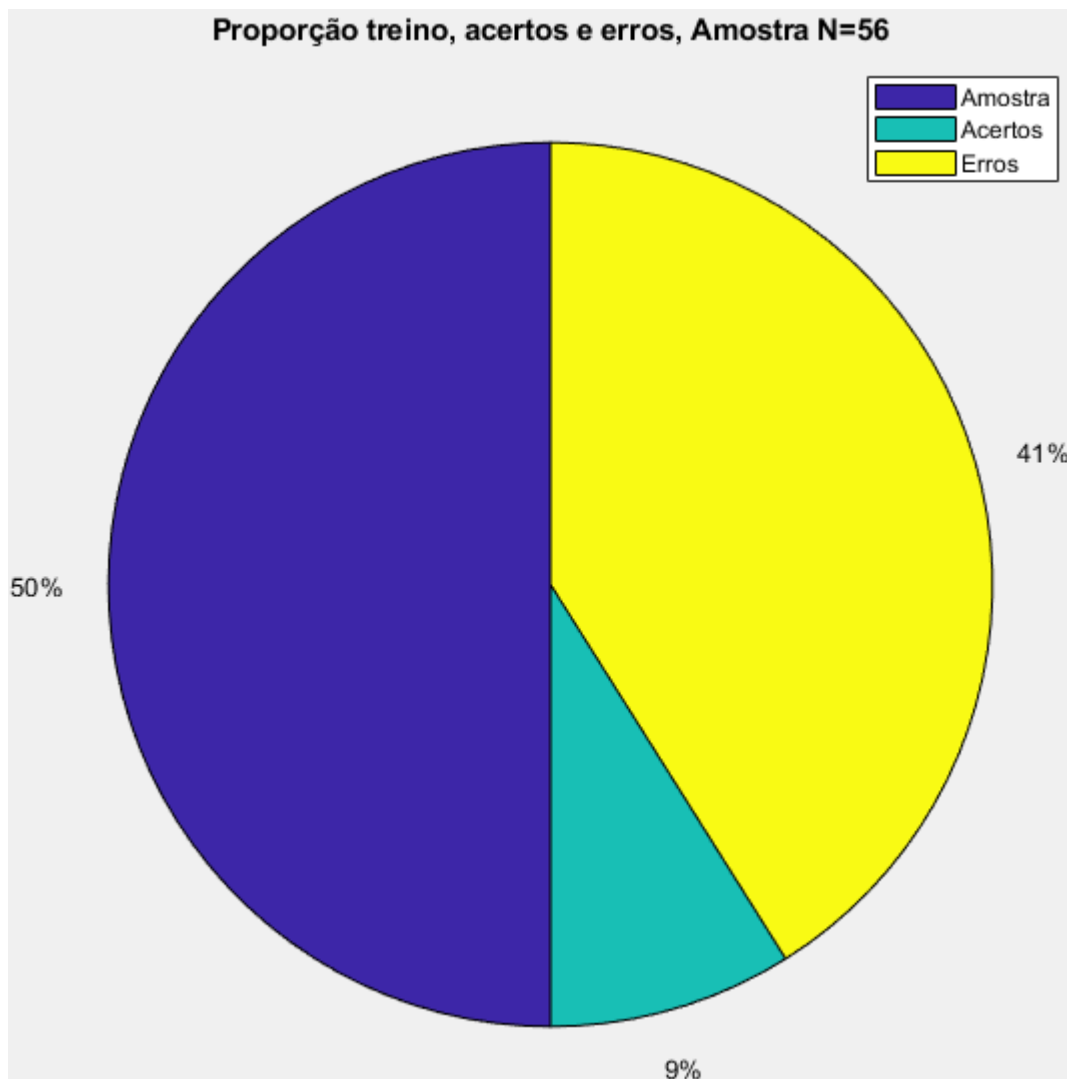
Fonte: Autoral

Figura 12: Quantidade de Acertos e Erros por classe, Amostra N= 56 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

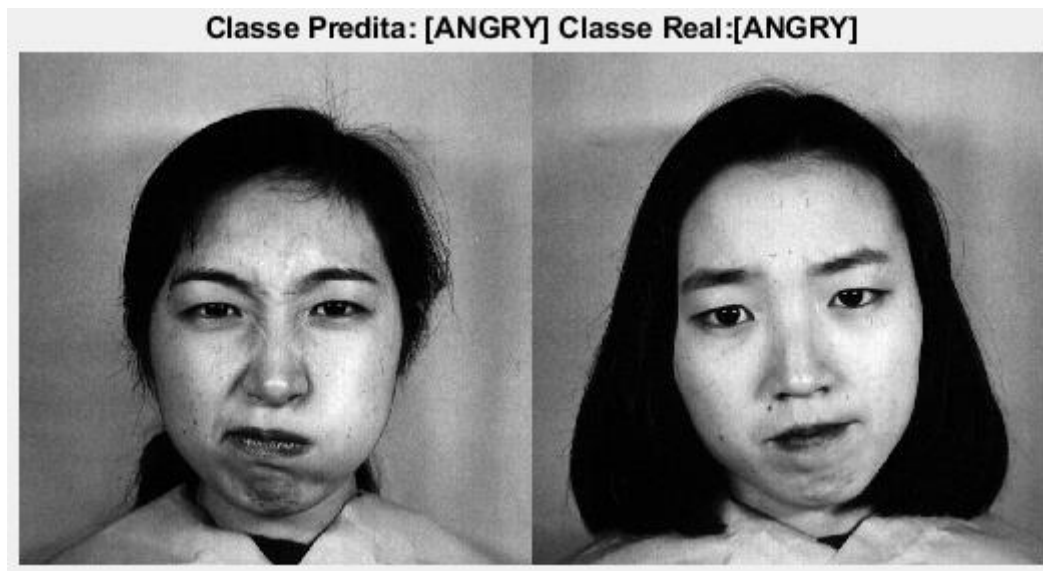
Figura 13: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 56 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

A porcentagem de acertos foi de 17.86% e a porcentagem de erros foi de 82.14%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 56 amostras para teste, o algoritmo classificou corretamente apenas 10. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 14: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 15: Exemplo de classificação incorreta realizada pelo algoritmo

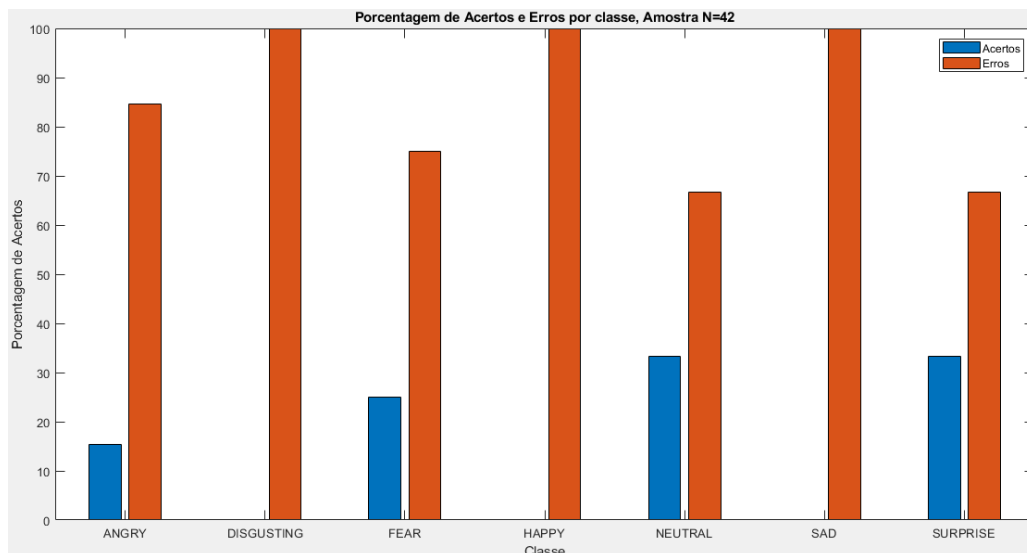


Fonte: Autoral

- Configuração 70%/30% Amostra de 42 imagens para teste

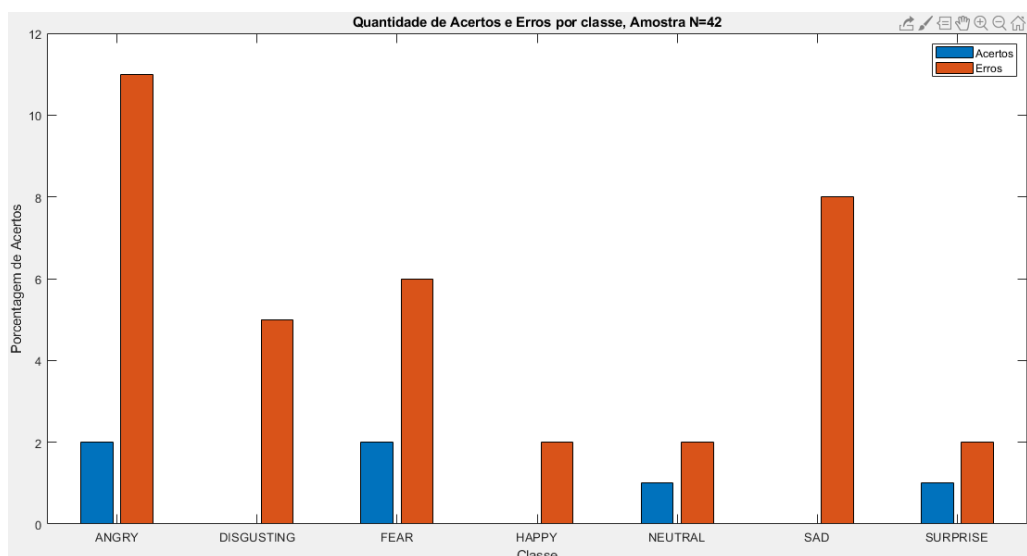
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 16: Porcentagem de Acertos e Erros por classe, Amostra N= 56 para classificação de emoções utilizando a técnica SURF como pré-processamento



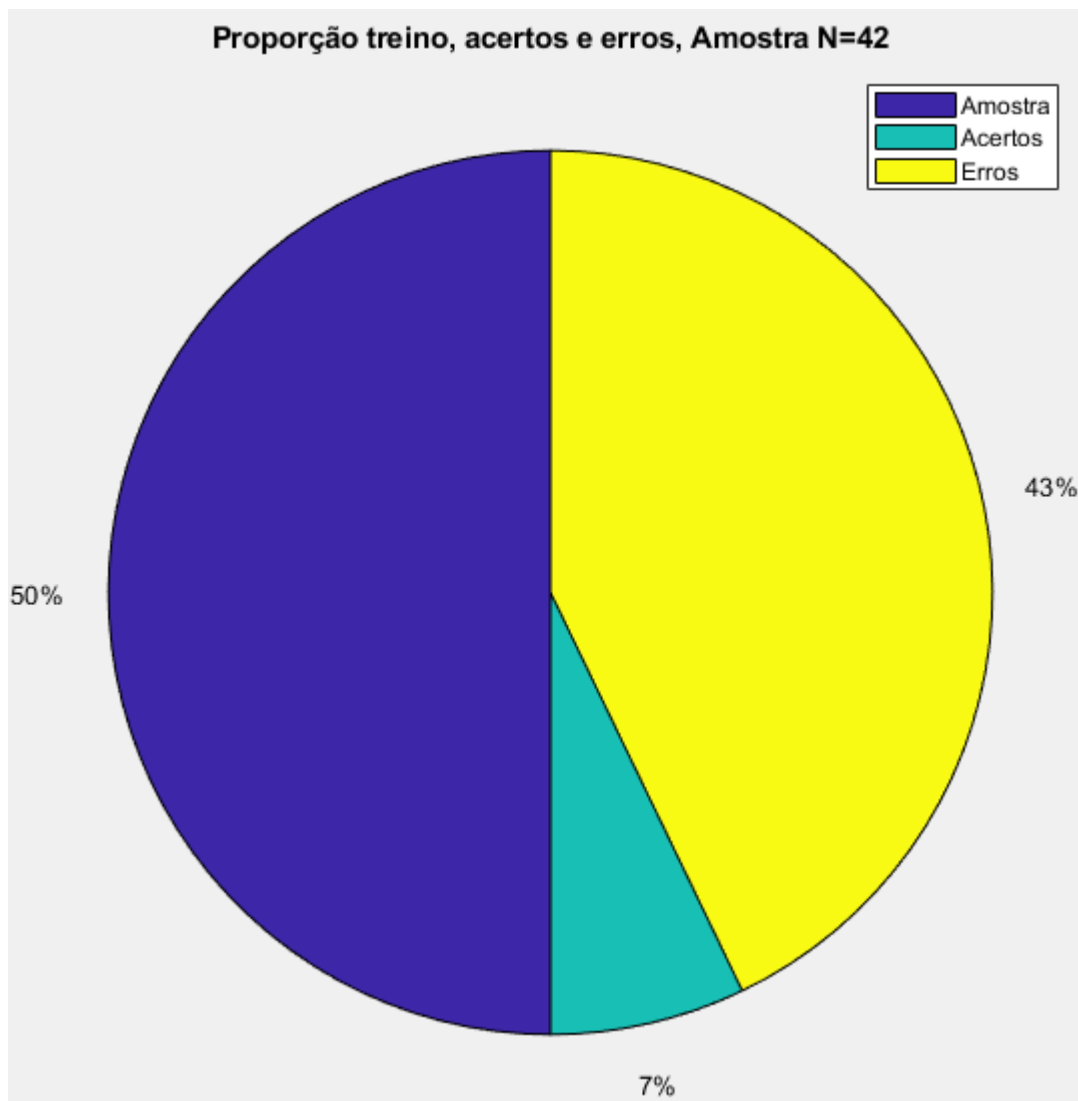
Fonte: Autoral

Figura 17: Quantidade de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

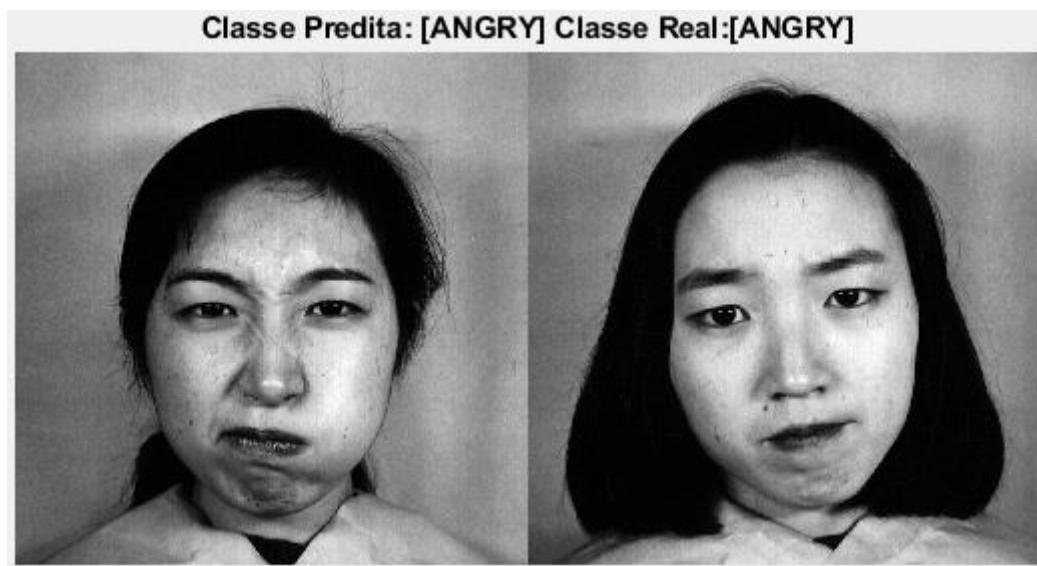
Figura 18: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

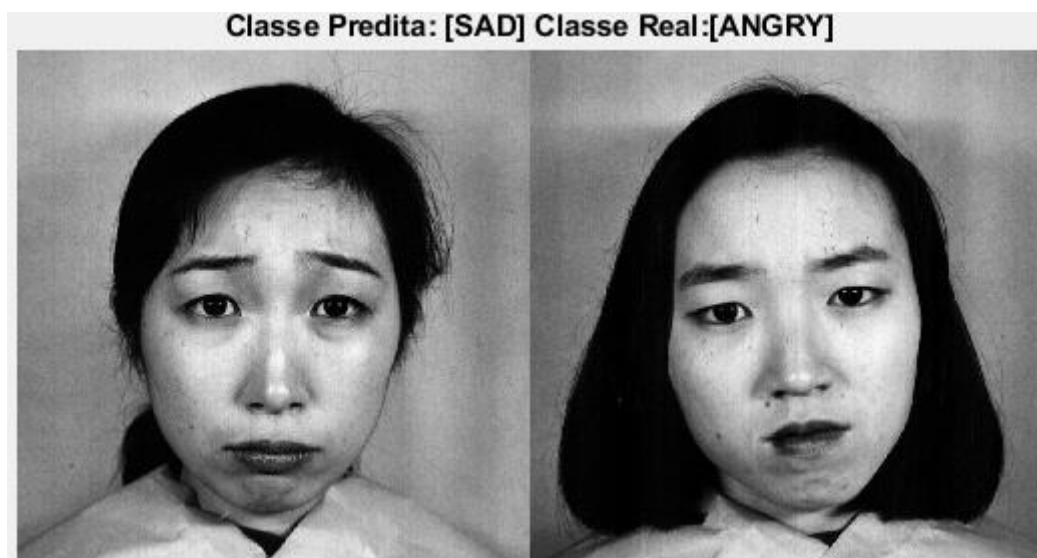
A porcentagem de acertos foi de 14.29% e a porcentagem de erros foi de 85,71%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 42 amostras para teste, o algoritmo classificou corretamente apenas 6. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 19: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 20: Exemplo de classificação incorreta realizada pelo algoritmo



Fonte: Autoral

A tabela a seguir resume os resultados obtidos com as diferentes proporções de treino e teste utilizando o algoritmo SURF para classificação de emoções.

Tabela 1 – Resumo das métricas obtidas ao classificar emoções com o algoritmo

Proporção Treino/Teste	Acerto		Erro	
	Quantidade	Porcentagem	Quantidade	Porcentagem
50/50	12	17,14%	58	82,86%
60/40	10	17,86%	46	82,14%
70/30	6	14,29%	36	85,71%

Fonte: Autoral

Como pode-se verificar a porcentagem de acertos foi baixa para todas as configurações. Alguns fatos que corroboram para essa performance podem estar relacionados à baixa quantidade de imagens para cada pessoa, representar uma emoção, existem apenas duas imagens de cada pessoa para uma mesma emoção. Além disso, a separação em treino e teste é feita de forma sequencial, onde as primeiras presentes na pasta são separadas para treino e as demais para teste.

A Figura a seguir mostra uma das imagens do banco de dados em conjunto com os seus 5 pontos característicos mais fortes usados para o processo de classificação. Como pode-se ver, eles estão próximos dos olhos e da boca. A classe correta dessa imagem é *Angry*.

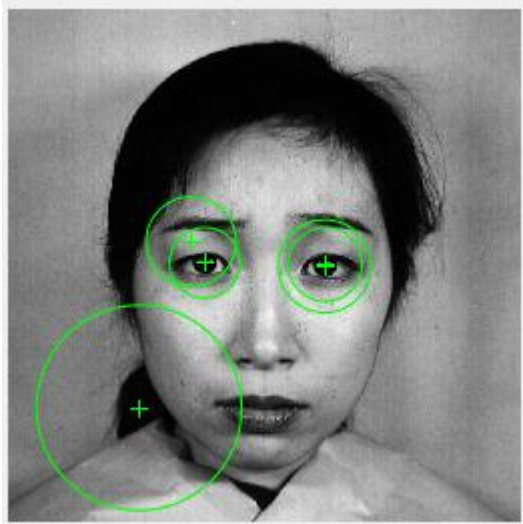
Figura 22 – Imagem classificada incorretamente e seus 5 pontos característicos mais fortes



Fonte: Autoral

A Figura a seguir, é a imagem da qual a imagem da Figura 11 foi classificada, na configuração de proporção 70/30. Nessa figura estão representados a imagem e os seus 5 pontos característicos mais fortes.

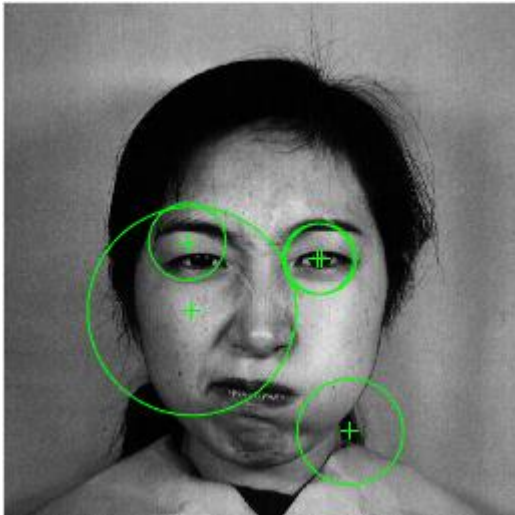
Figura 23 – Imagem à qual a Figura 22 foi atribuída como pertencente a classe



Fonte: Autoral

Como é possível verificar, os pontos característicos de ambas as imagens estão localizados relativamente próximos entre si. Já em relação aos pontos característicos de uma das imagens da classe *Angry* utilizada para teste se encontra ilustrada a seguir. Como pode-se ver, os pontos característicos em comparação com os pontos da imagem de teste (Figura 22) estão diferentes. Localizados próximos aos olhos, porém, um dos pontos se encontra próximo ao cabelo a direita, isso pode ter impactado na classificação incorreta.

Figura 24 – Uma das imagens utilizadas para treinar a classe Angry



Fonte: Autoral

Outro fato que pode ser explorado para melhorar a classificação, é aumentar o número de pontos característicos mais fortes, buscando assim, que os pontos capturem melhor microexpressões da pessoa.

2. Utilize o algoritmo SURF para extrair características e treine a PCA para classificar as pessoas.

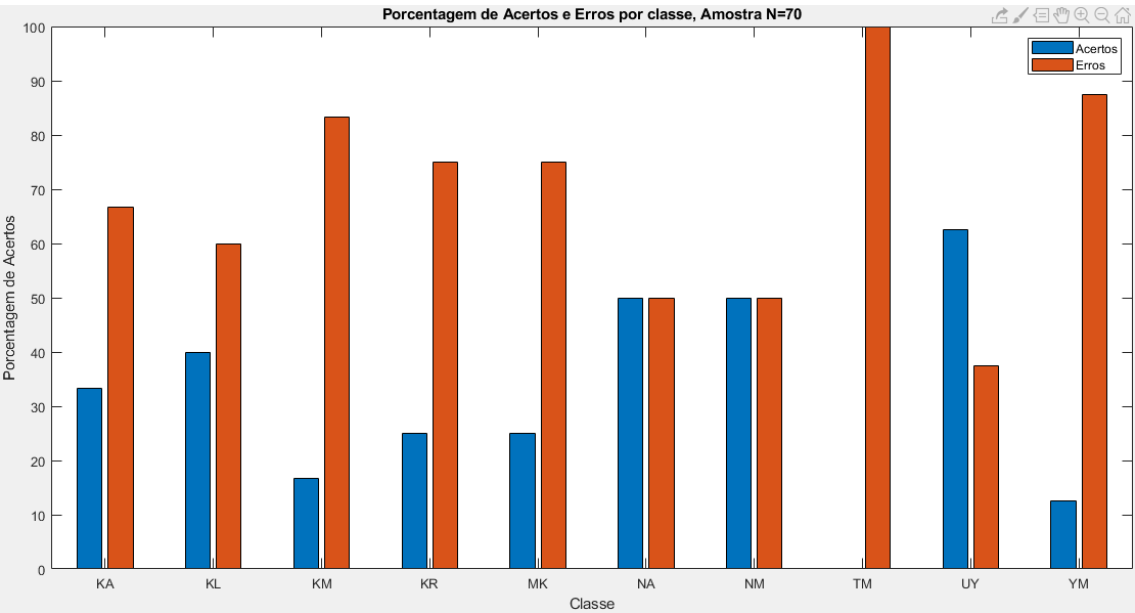
Resposta:

Paralelamente a questão 1, foi utilizado o método de SURF para extração das características e depois disso foi utilizado a PCA como classificador para classificar as pessoas. As Figuras 7, 8 e 9, mostram os resultados para os conjuntos de [treino, teste] = 50%/50%, [treino, teste] = 60%/40%, [treino, teste] = 70%/30% respectivamente.

Configuração 50%/50% Amostra de 70 imagens para teste

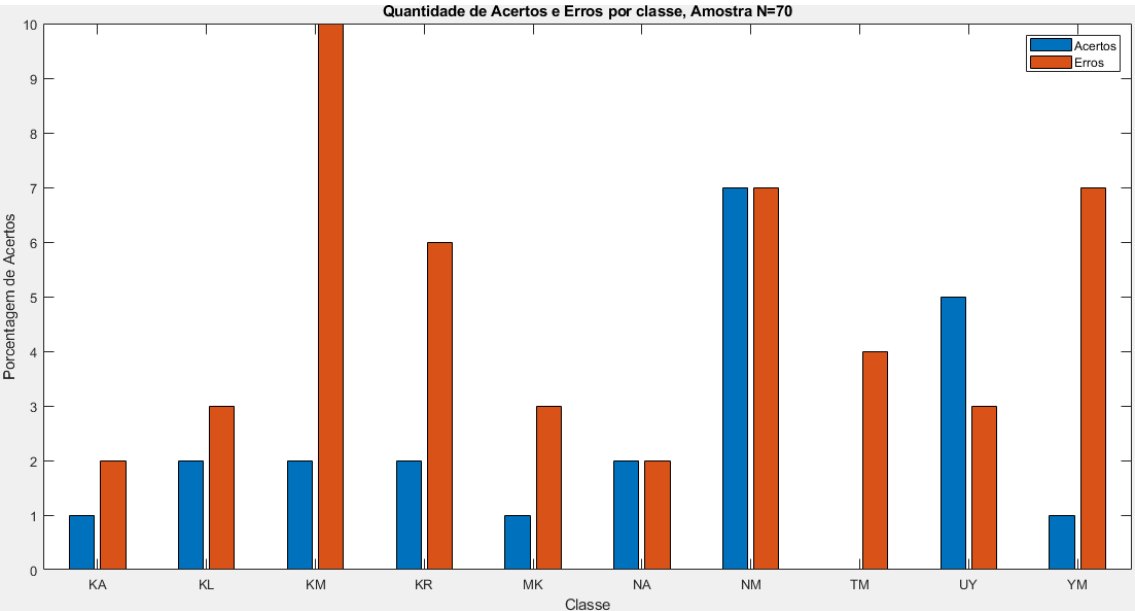
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 25: Porcentagem de Acertos e Erros por classe, Amostra N= 56 para classificação de emoções utilizando a técnica SURF como pré-processamento



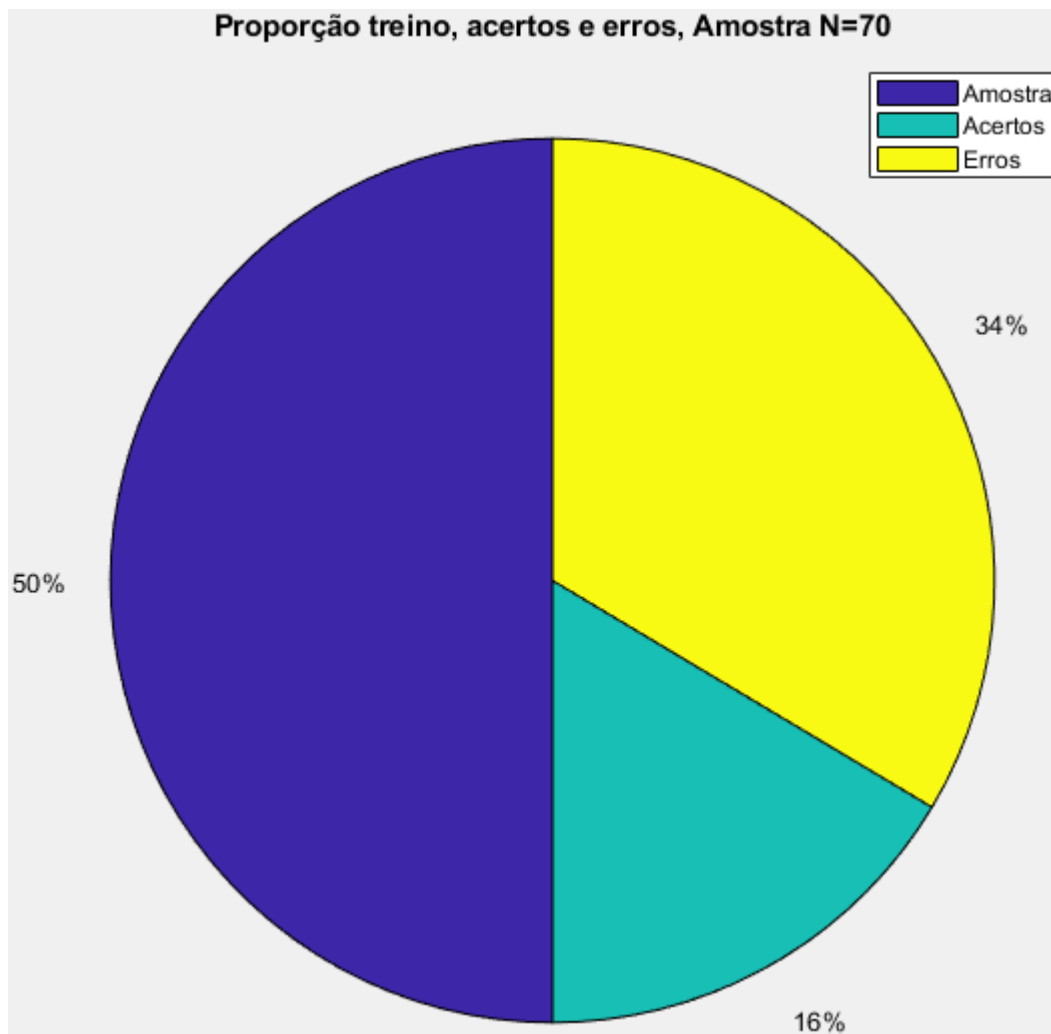
Fonte: Autoral

Figura 26: Quantidade de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

Figura 27: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

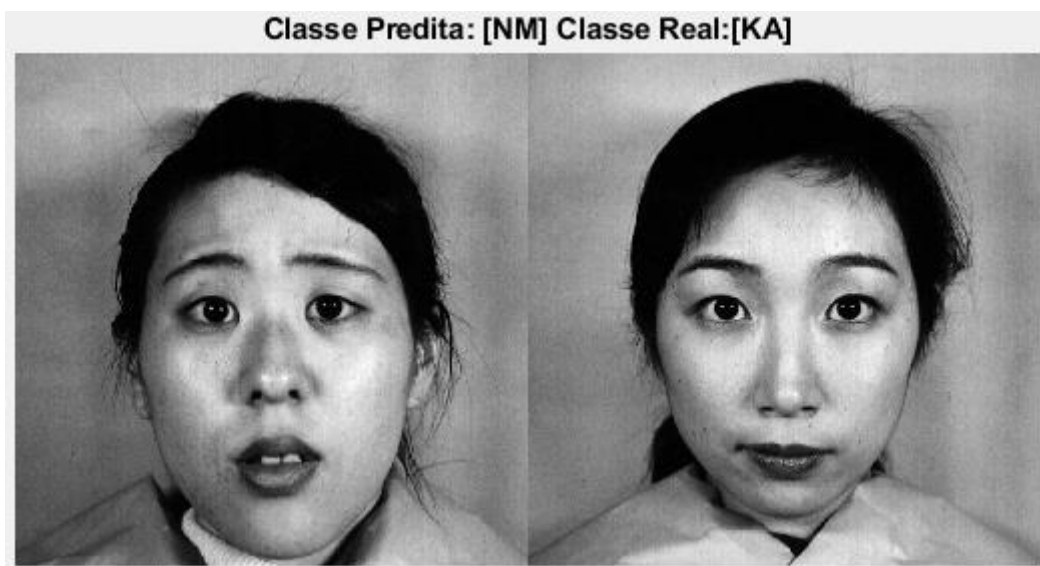
A porcentagem de acertos foi de 32,86% e a porcentagem de erros foi de 67,14%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 70 amostras para teste, o algoritmo classificou corretamente apenas 23. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 28: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 29: Exemplo de classificação incorreta realizada pelo algoritmo

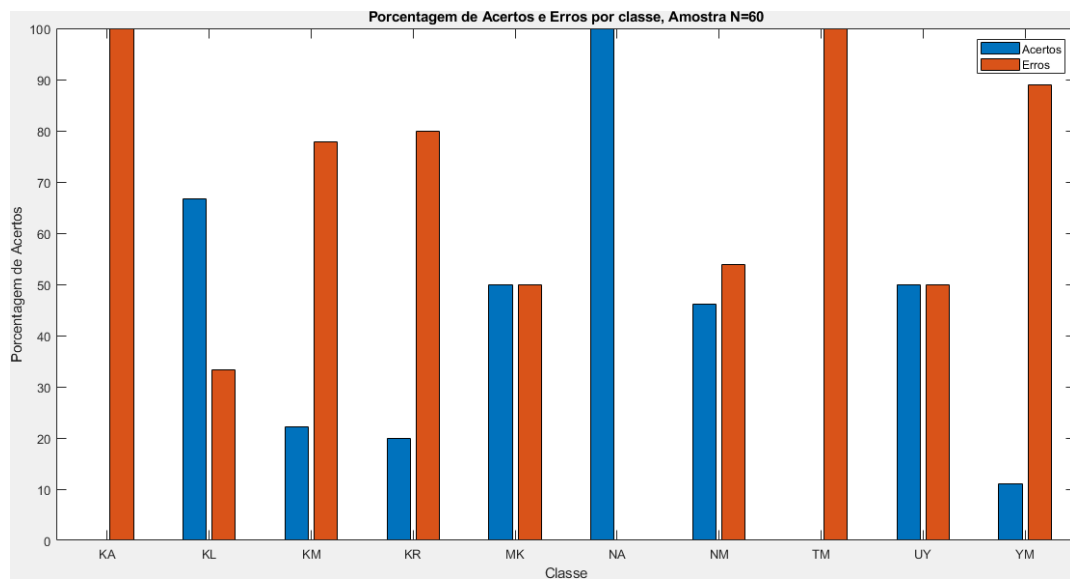


Fonte: Autoral

Configuração 60%/40% Amostra de 60 imagens para teste

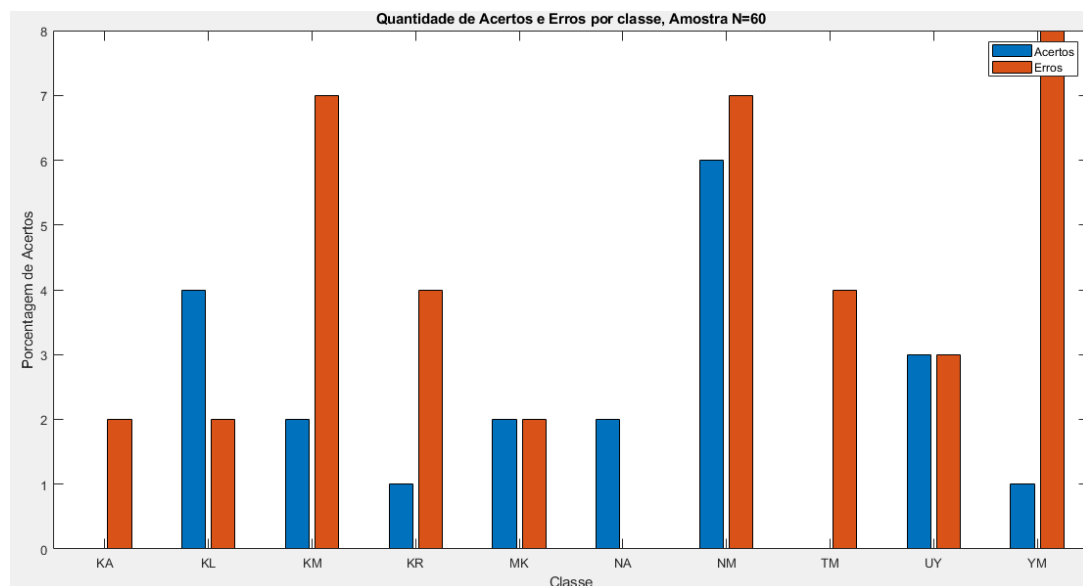
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 30: Porcentagem de Acertos e Erros por classe, Amostra N= 60 para classificação de emoções utilizando a técnica SURF como pré-processamento



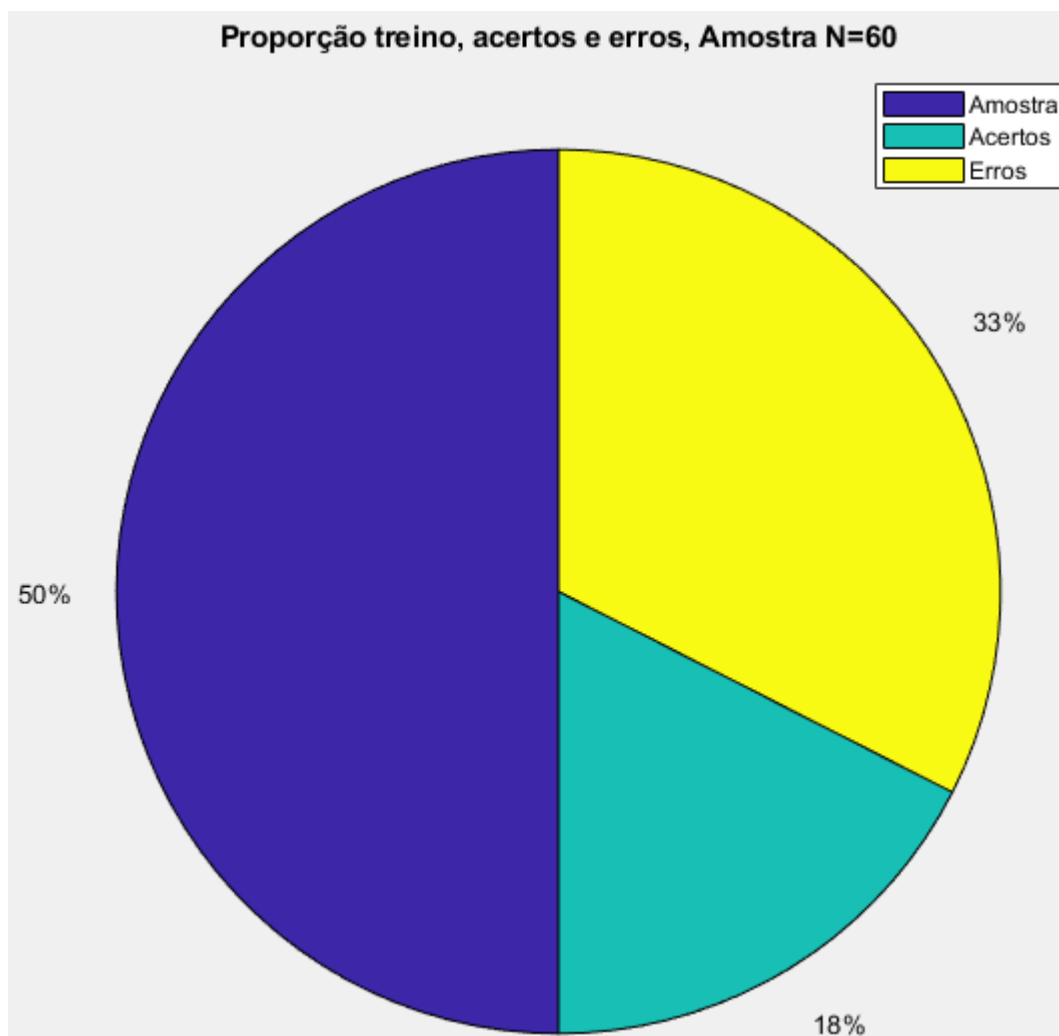
Fonte: Autoral

Figura 31: Quantidade de Acertos e Erros por classe, Amostra N= 60 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

Figura 32: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 60 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

A porcentagem de acertos foi de 35% e a porcentagem de erros foi de 65%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 60 amostras para teste, o algoritmo classificou corretamente apenas 21. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 33: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 34: Exemplo de classificação incorreta realizada pelo algoritmo

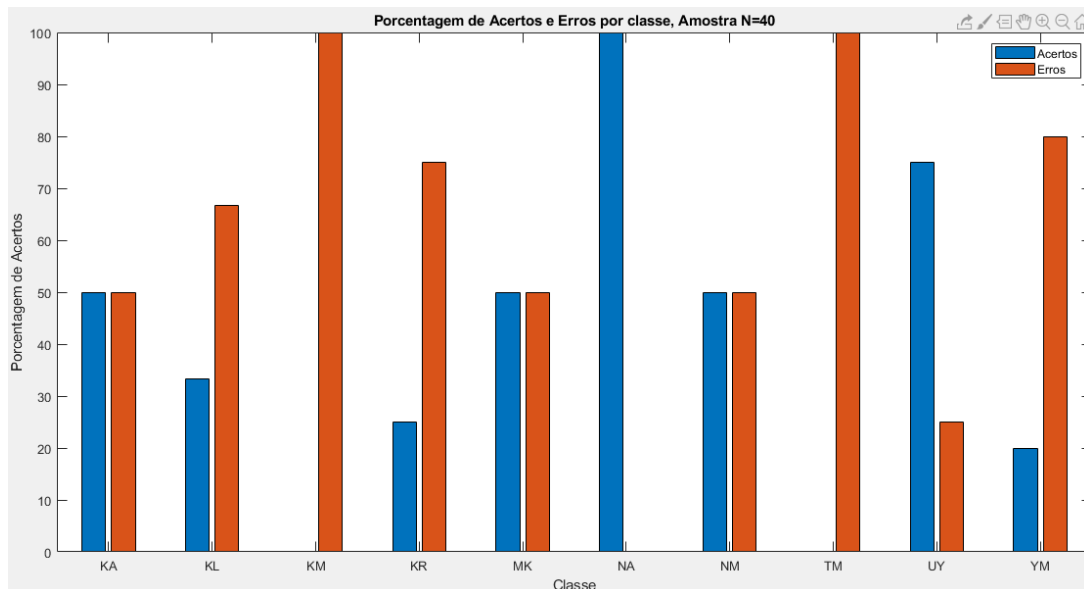


Fonte: Autoral

Configuração 70%/30% Amostra de 40 imagens para teste

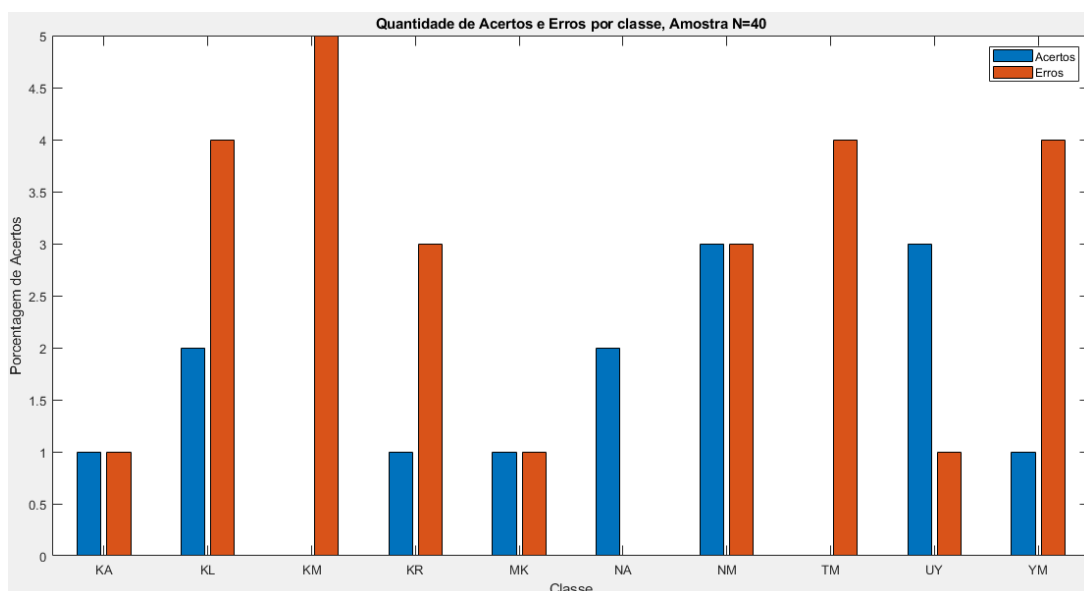
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 35: Porcentagem de Acertos e Erros por classe, Amostra N= 40 para classificação de emoções utilizando a técnica SURF como pré-processamento



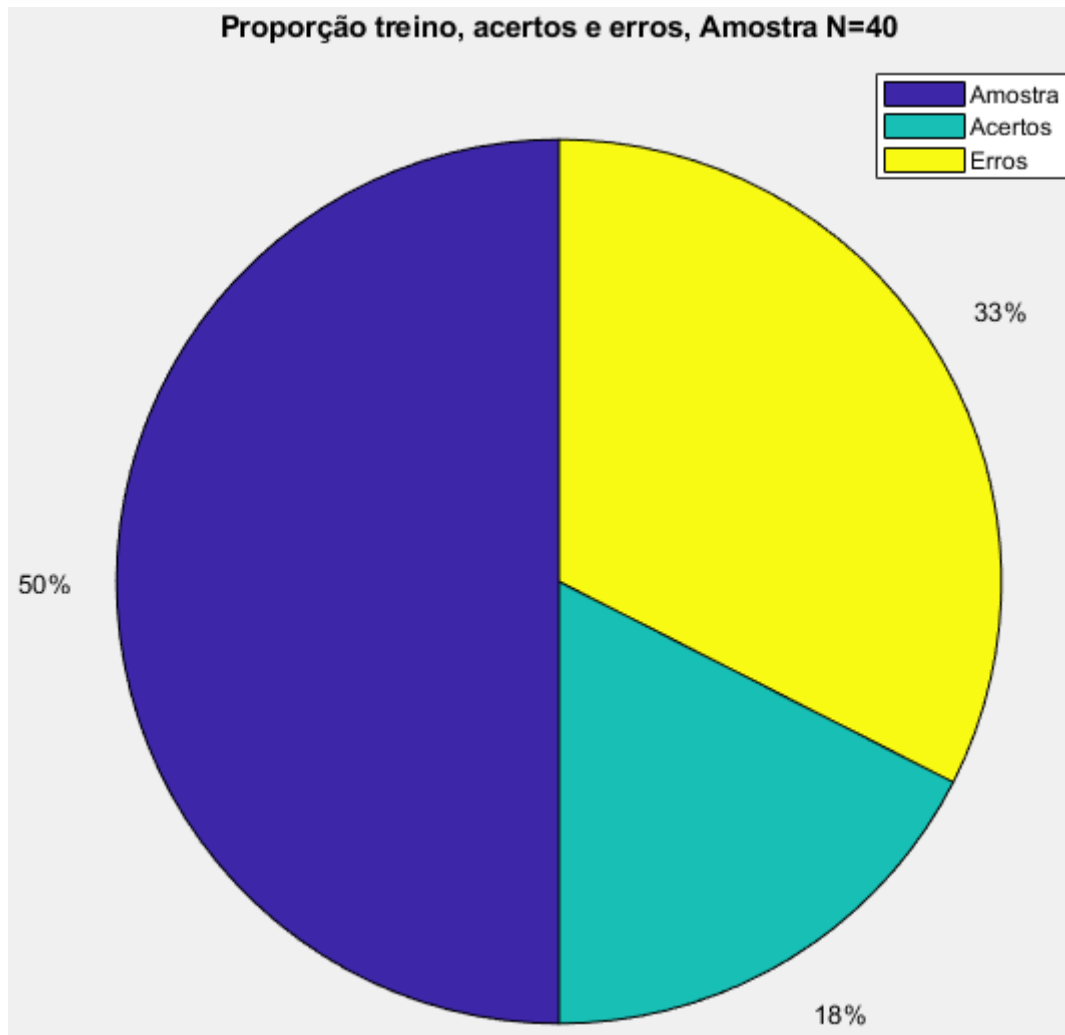
Fonte: Autoral

Figura 36: Quantidade de Acertos e Erros por classe, Amostra N= 40 para classificação de emoções utilizando a técnica SURF como pré-processamento



Fonte: Autoral

Figura 37: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 40 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

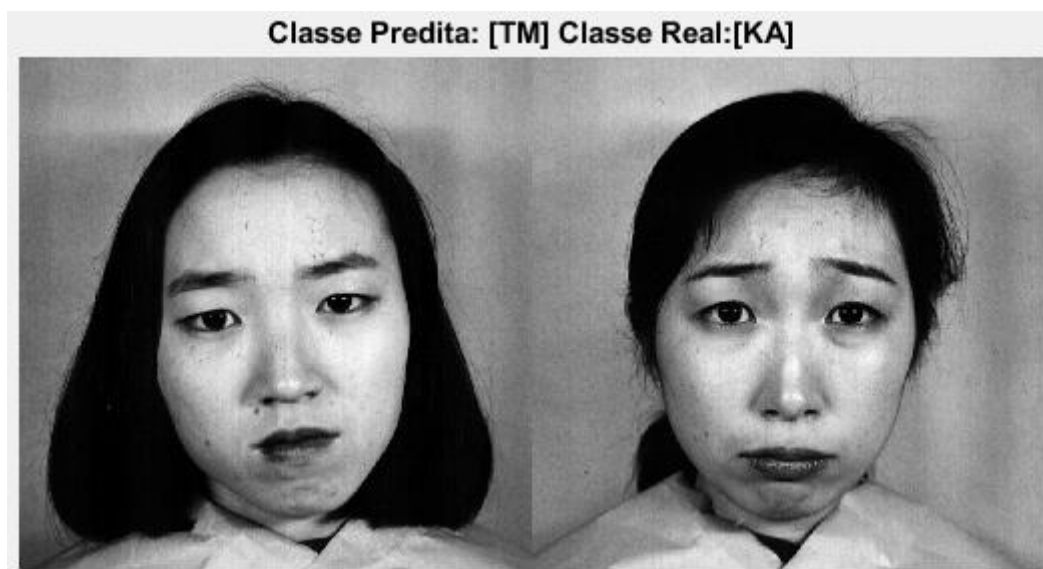
A porcentagem de acertos foi de 35% e a porcentagem de erros foi de 65%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 40 amostras para teste, o algoritmo classificou corretamente apenas 14. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 38: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 39: Exemplo de classificação incorreta realizada pelo algoritmo



Fonte: Autoral

A tabela a seguir resume os resultados obtidos com as diferentes proporções de treino e teste utilizando o algoritmo SURF para classificação de emoções.

Tabela 2 – Resumo das métricas obtidas ao classificar emoções com o algoritmo

Porporção Treino/Teste	Acerto		Erro	
	Quantidade	Porcentagem	Quantidade	Porcentagem
50/50	23	32,86%	47	67,14%
60/40	21	35,00%	39	65,00%
70/30	14	35,00%	26	65,00%

Fonte: Autoral

A melhor porcentagem de erro foi de 35% para as configurações 60/40 e 70/30. A porcentagem foi relativamente baixa. Mais uma vez, esse fato pode estar relacionado a quantidade de pontos característicos usados pelo classificador 5. A Figura 40 traz um exemplo de imagem usada para treino e teste com seus pontos característicos evidenciados.

Figura 40: A esquerda imagem usada para teste com 5 pontos característicos e a direita imagem usada para treino com 5 pontos característicos

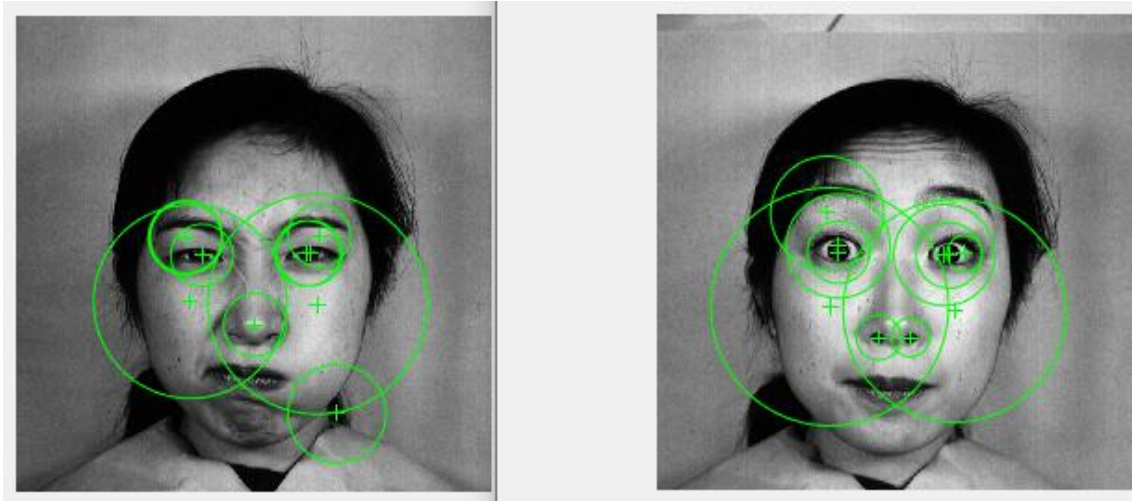


Fonte: Autoral

Como pode-se verificar a imagem usada para teste os pontos se concentram no rosto nas regiões dos olhos e do nariz, já na imagem usada para treino os pontos característicos se concentram nos olhos e um dos pontos se encontra no cabelo ao lado direito. Dessa forma, ao aumentar o número de pontos característicos espera-se aumentar a evidência de regiões similares nas imagens aumentando assim, a acurácia da classificação. A Figura 41 ilustra as mesmas imagens (Figura 40) com mais pontos característicos evidenciados.

Na Figura 41 é visível que os pontos estão mais semelhantes entre si do que em comparação com os pontos da Figura 40. Na imagem utilizada para treino na região do nariz é possível verificar que foram encontrados pontos característicos o que não ocorria na Figura 40.

Figura 41: A esquerda imagem usada para treino com 10 pontos característicos e a direita imagem usada para teste com 10 pontos característicos



Fonte: Autoral

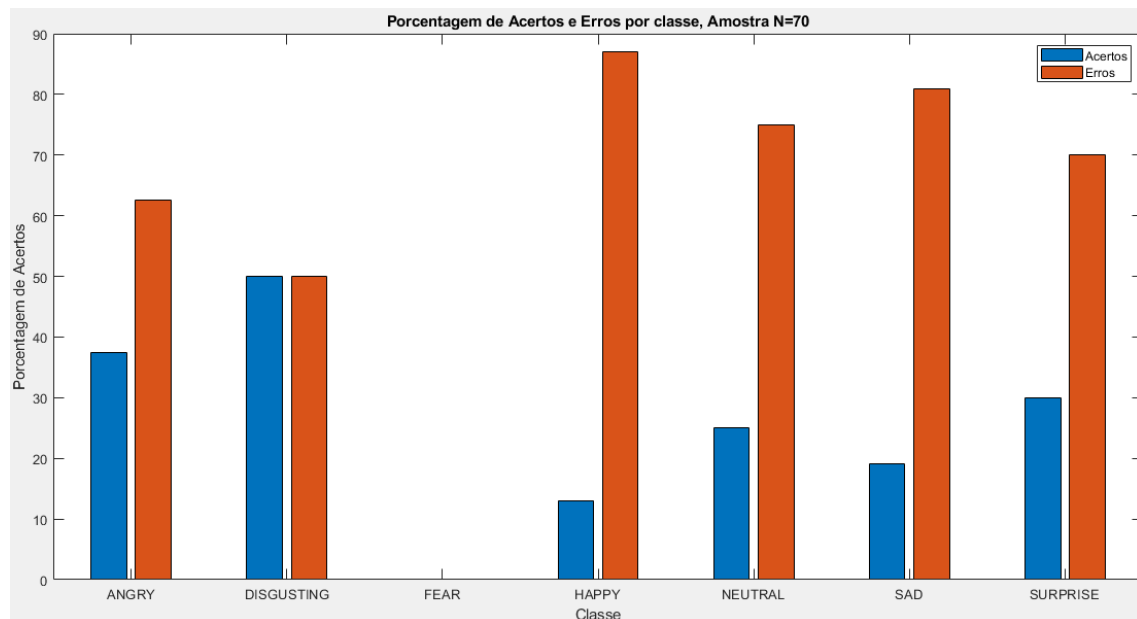
Dessa forma, uma maneira de melhorar a classificação pode ser o aumento do número de pontos característicos para melhorar a semelhança entre as imagens. Além disso, outras técnicas de PDI também podem ser exploradas para melhorar as imagens, como a segmentação, por exemplo.

3. Utilize o algoritmo HOG para extrair características e treine a PCA para classificar as emoções.

A função main foi configurada para ser possível classificar emoções no banco de dados. O banco de dados foi dividido em treino e teste nas configurações 50%/50%, 60%/40% e 70%/30%. Os resultados obtidos foram os seguintes.

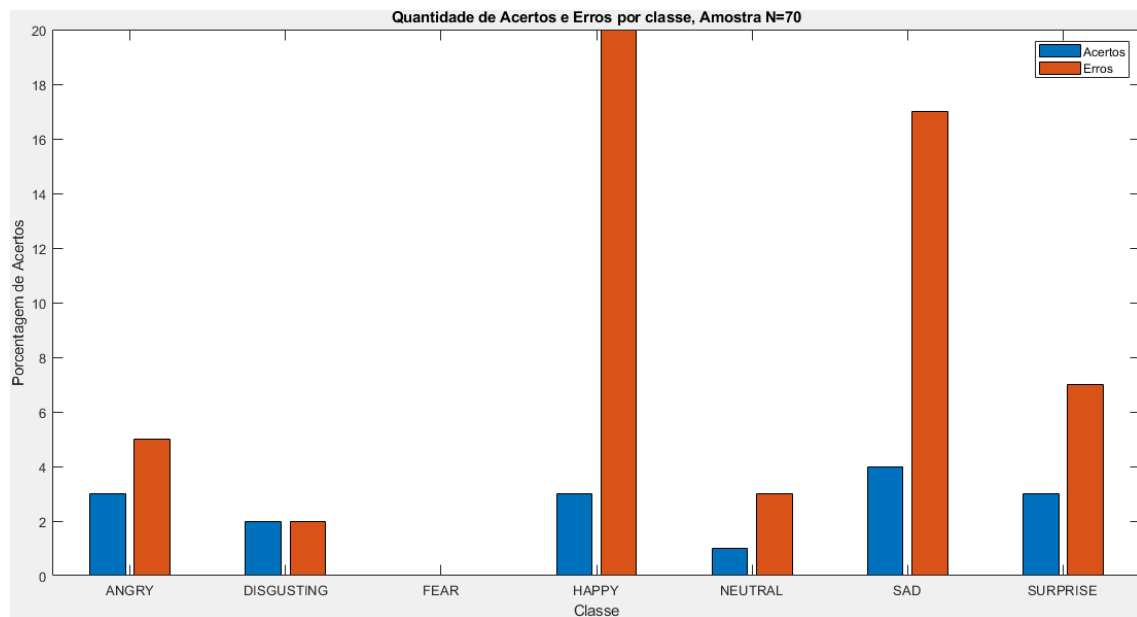
Configuração 50%/50% Amostra de 70 imagens para teste

Figura 42: Porcentagem de Acertos e Erros por classe, Amostra N= 70 para classificação de emoções utilizando a técnica HOG como pré-processamento



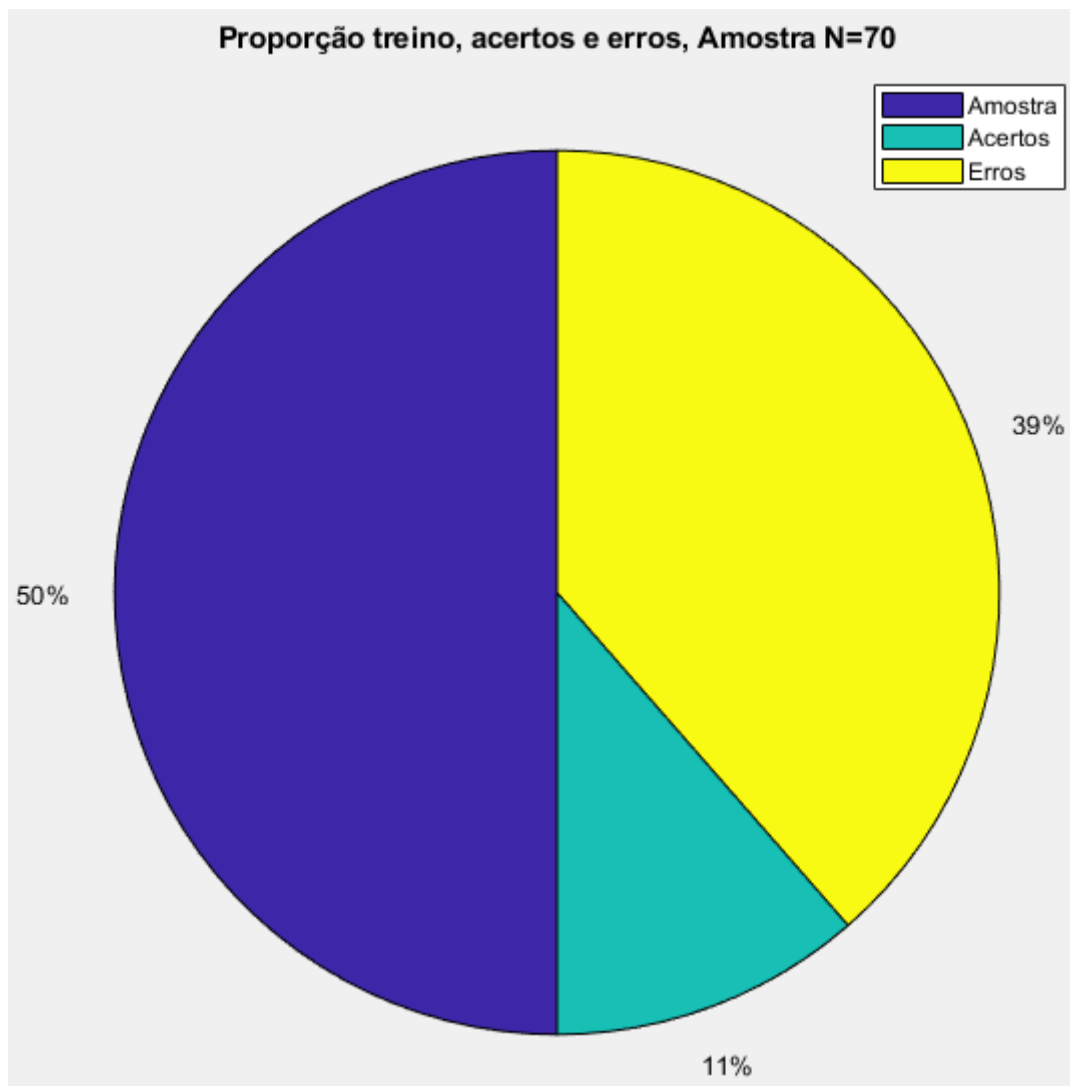
Fonte: Autoral

Figura 43: Quantidade de Acertos e Erros por classe, Amostra N= 70 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

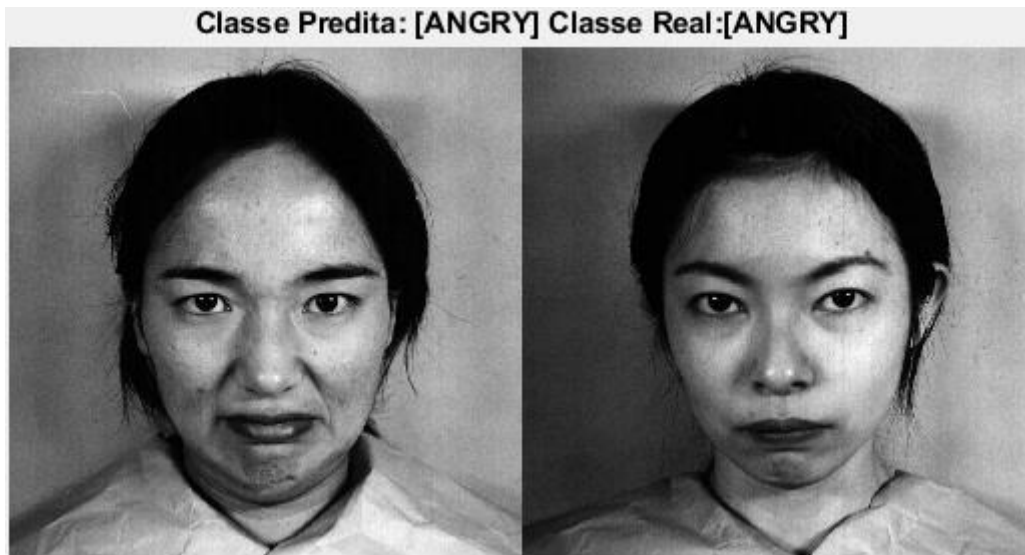
Figura 44: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 70 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

A média de acertos foi de 22.86% e a média de erros foi de 77.14%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 70 amostras para teste, o algoritmo classificou corretamente apenas 16. A Figura a seguir ilustra uma classificação correta realizada pelo algoritmo.

Figura 45: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

A Figura a seguir mostra um exemplo de classificação incorreta realizada pelo classificador. Em uma inspeção visual na imagem classificada incorretamente é possível verificar que as imagens são bastante similares. Os olhos e a boca estão dispostos de forma bastante semelhante. Para humanos classificarem as imagens é muito simples já que as microexpressões são mais visíveis já para o algoritmo é mais complexo.

Figura 46: Exemplo de classificação incorreta realizada pelo algoritmo



Fonte: Autoral

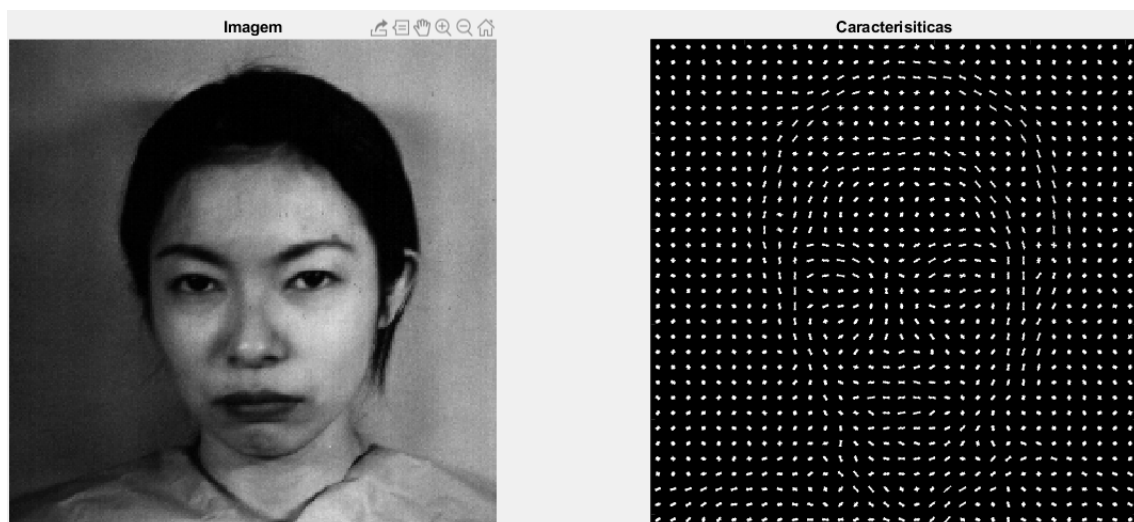
As Figuras a seguir ilustram as características extraídas das imagens classificadas incorretamente utilizando a técnica HOG como pré-processamento. Nas figuras à direita é o plot das características. Ao comparar ambos os plots é possível verificar que as características extraídas são muito similares, a diferença entre elas é muito pequena, sendo quase imperceptível para olhos humanos.

Figura 47: Classe predita



Fonte: Autoral

Figura 48: Classe real



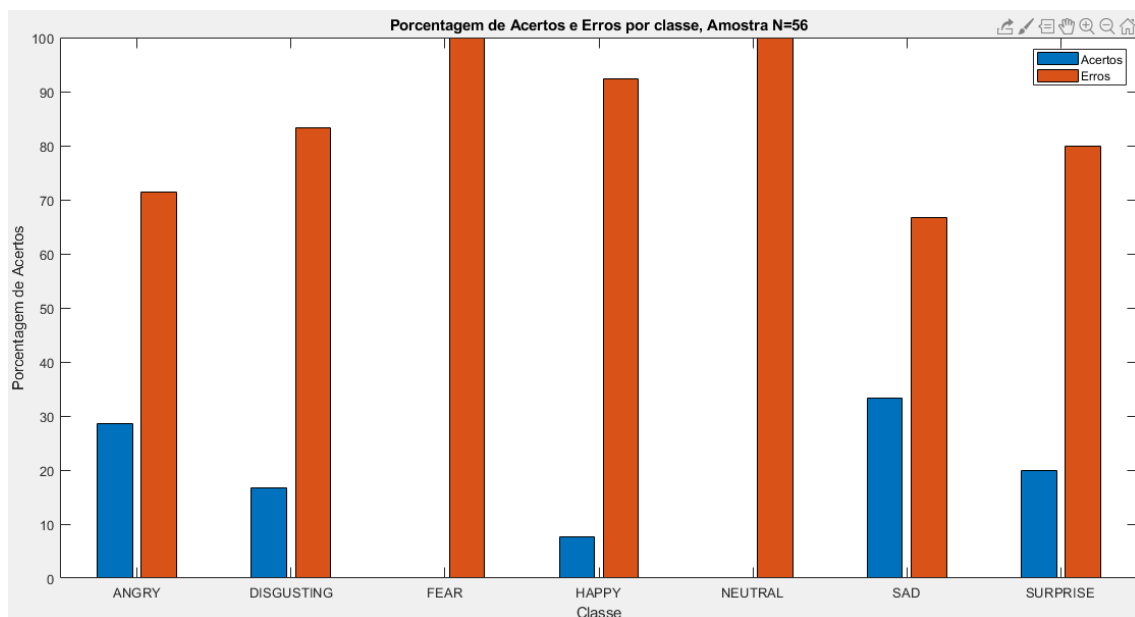
Fonte: Autoral

A similaridade entre as imagens é o que pode ter causado a alta taxa de erro do algoritmo. Além disso, algumas expressões diferenciam entre si a partir de microexpressões que podem não ficar tão evidentes com a técnica HOG.

Configuração 60%/40% Amostra de 56 imagens para teste

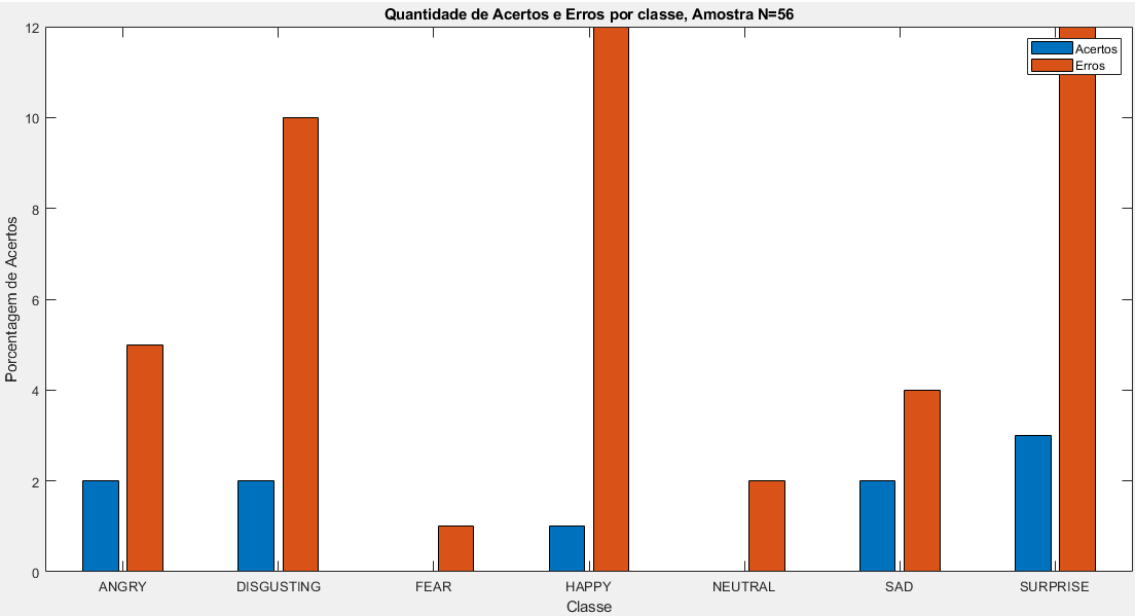
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 49: Porcentagem de Acertos e Erros por classe, Amostra N= 56 para classificação de emoções utilizando a técnica HOG como pré-processamento



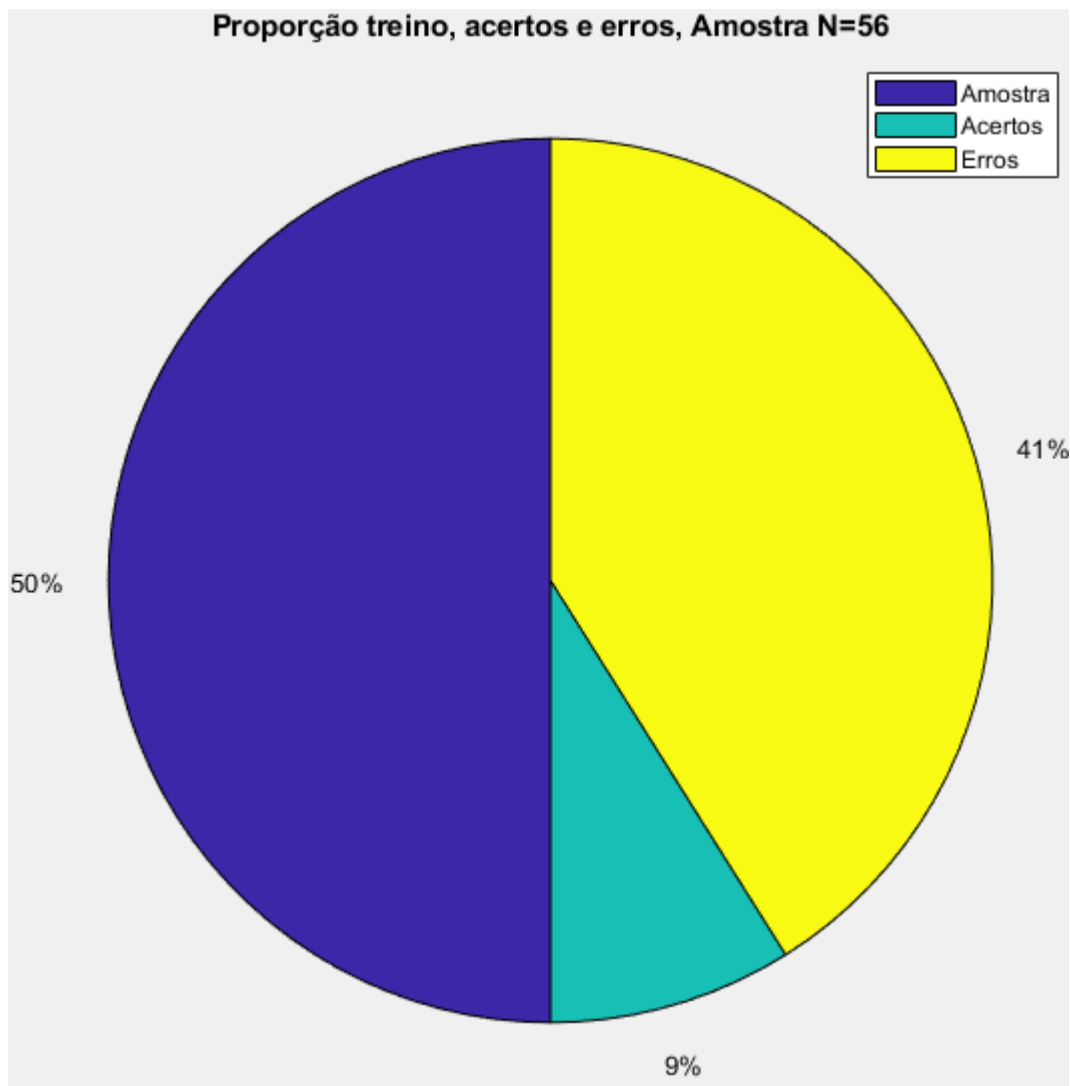
Fonte: Autoral

Figura 50: Quantidade de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

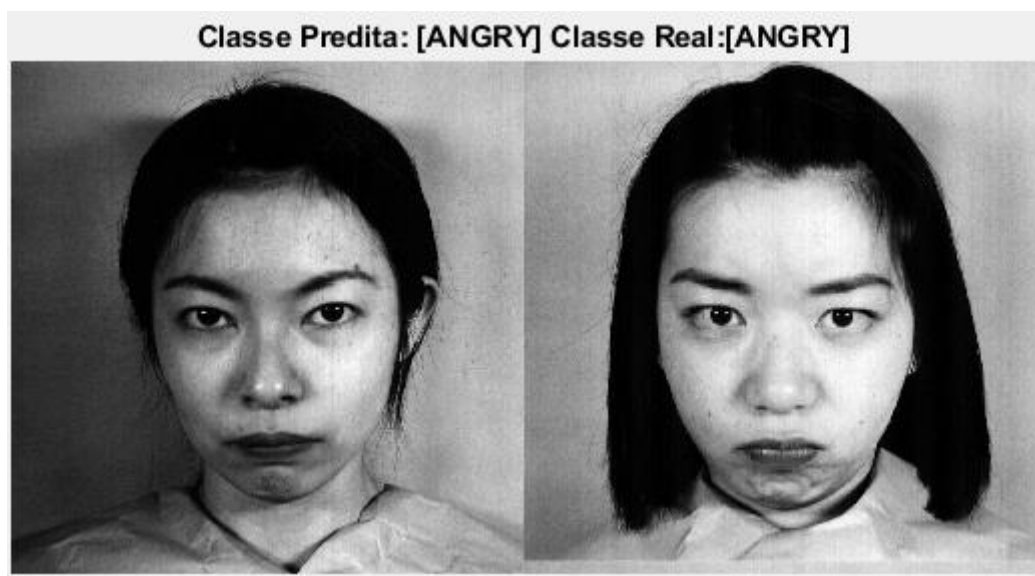
Figura 51: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 56 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

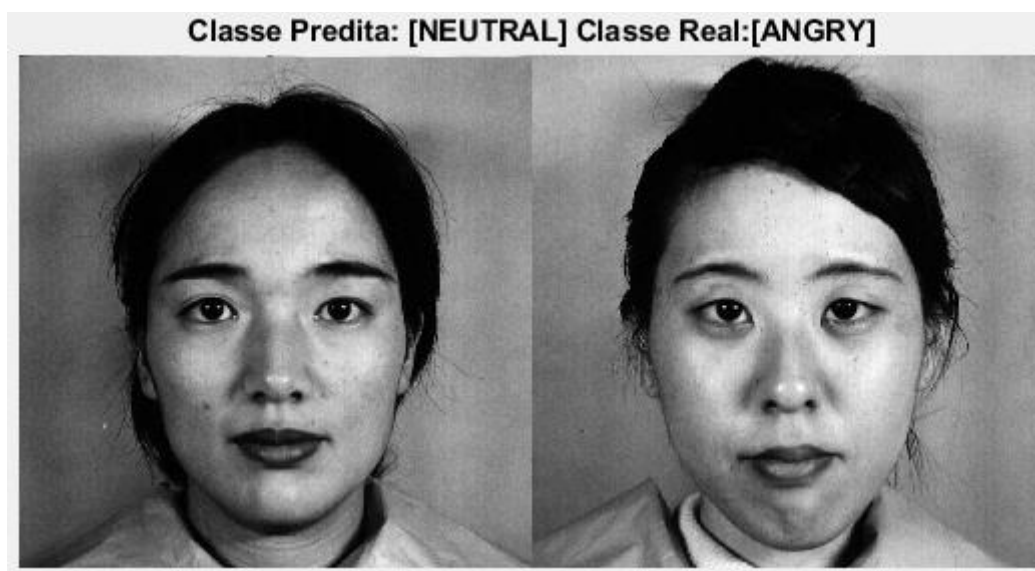
A média de acertos foi de 17.86% e a média de erros foi de 82.14%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 46 amostras para teste, o algoritmo classificou corretamente apenas 10. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 52: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 53: Exemplo de classificação incorreta realizada pelo algoritmo

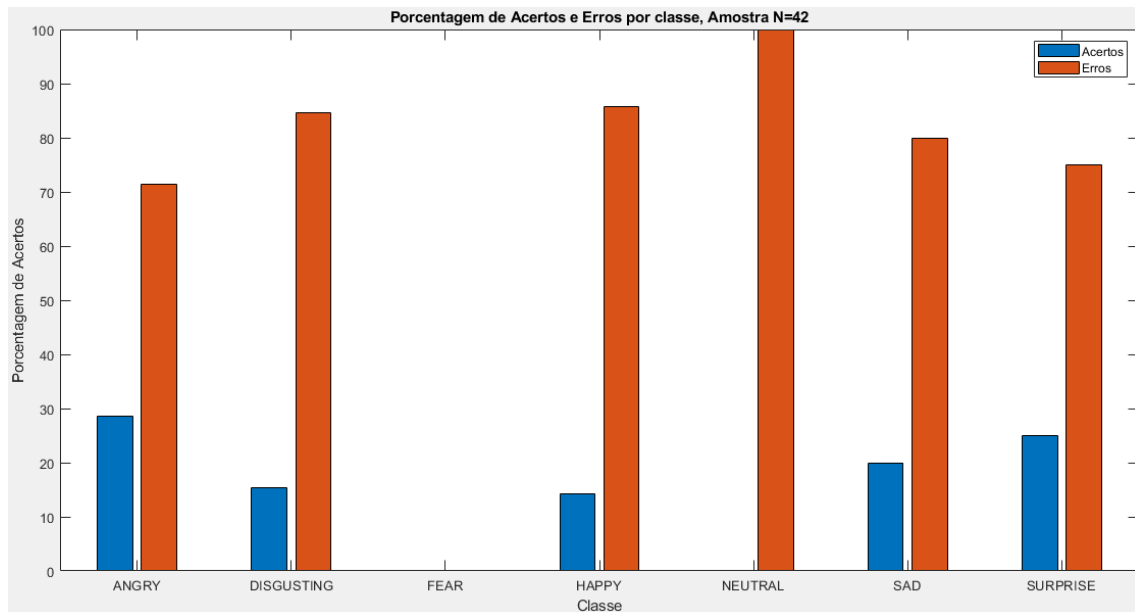


Fonte: Autoral

Configuração 70%/30% Amostra de 42 imagens para teste

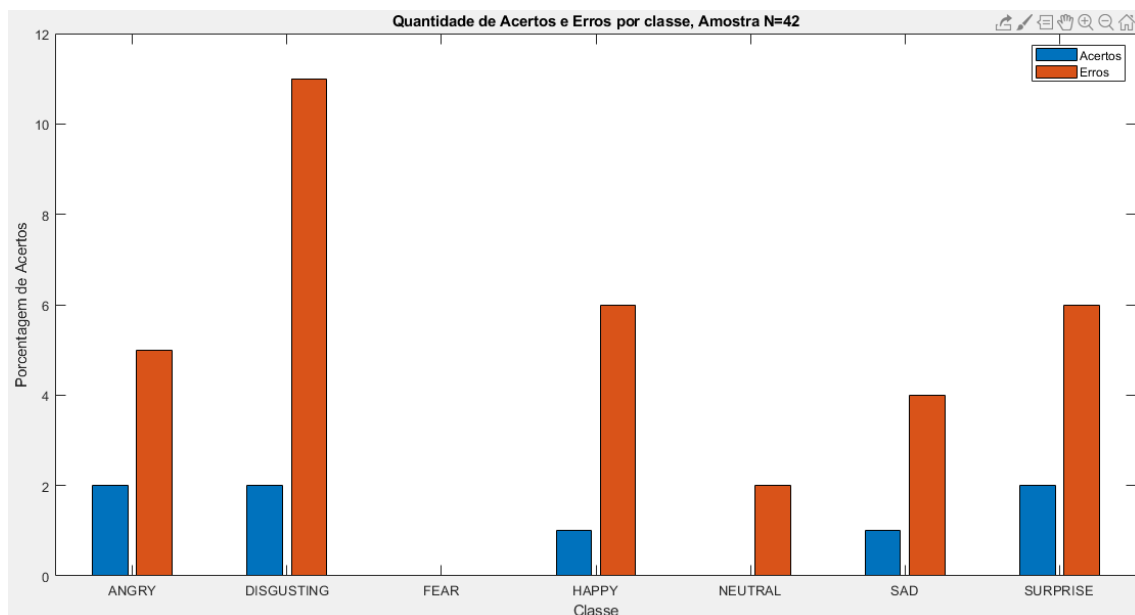
As figuras a seguir trazem os plots para as classes do banco de dados.

Figura 54: Porcentagem de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



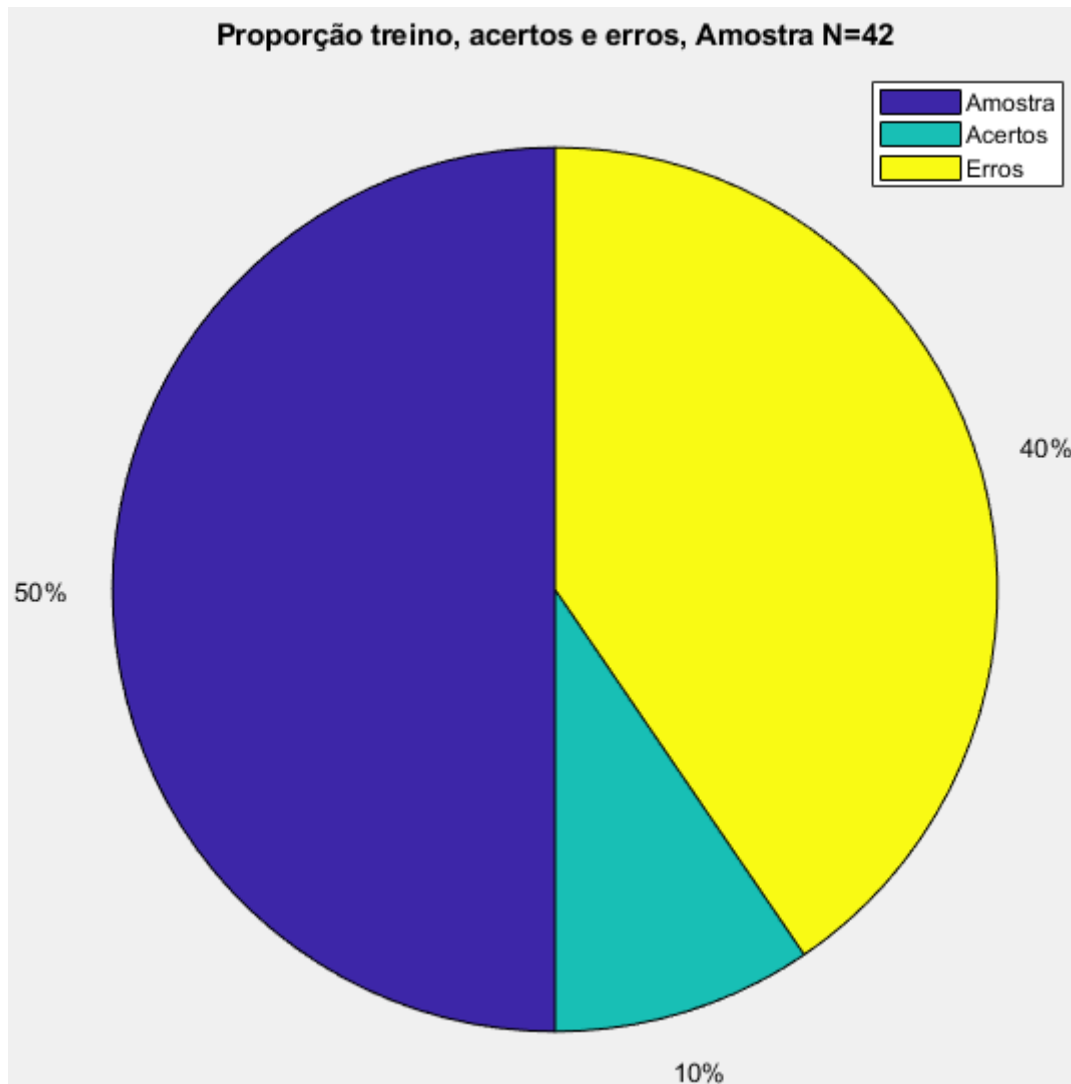
Fonte: Autoral

Figura 55: Porcentagem de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

Figura 56: Porcentagem de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

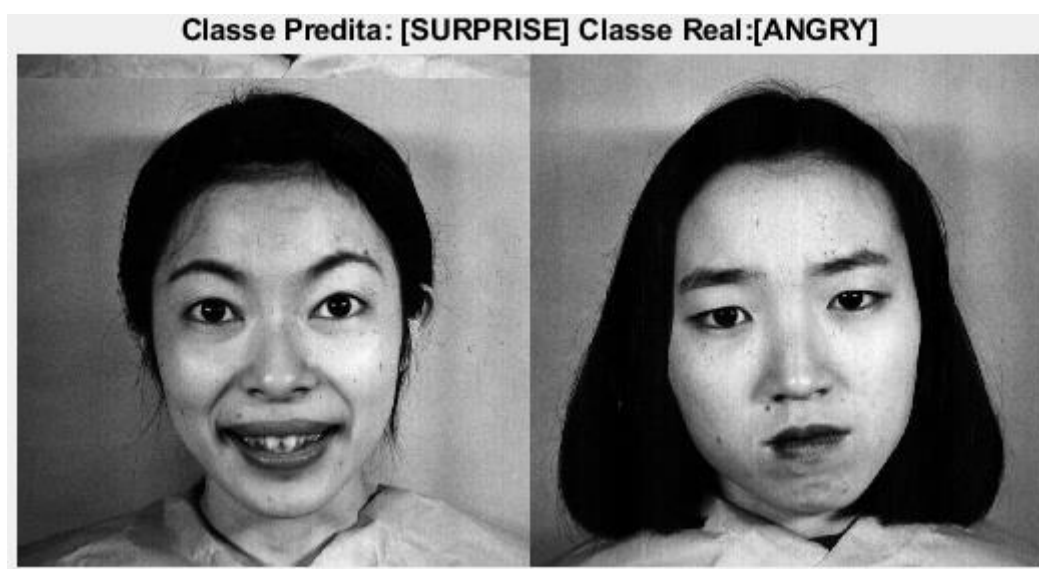
A média de acertos foi de 19.05% e a média de erros foi de 80.95%. Para essa configuração a PCA não conseguiu classificar bem os dados. De 42 amostras para teste, o algoritmo classificou corretamente apenas 8. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 57: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 58: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

A tabela a seguir resume os resultados obtidos com as diferentes proporções de treino/teste.

Tabela 3 – Resumo das métricas de avaliação para as diferentes proporções de treino/teste

Porporção Treino/Teste	Acerto		Erro	
	Quantidade	Porcentagem	Quantidade	Porcentagem
50/50	16	22,86%	54	77,14%
60/40	10	17.86%	46	82.14%
70/30	8	19.05%	34	80.95%

Fonte: Autoral

Como pode-se verificar na tabela, a proporção 50/50 foi a que obteve a melhor média de acertos. Isso se deve ao fato de que existiam mais imagens para o treino e consequentemente mais características das diferentes imagens seriam reconhecidas pelo classificador.

Porém, em todas as proporções o erro foi muito alto. Isso pode estar relacionado ao fato de que existem em cada classe (cada emoção) duas pessoas transmitindo a mesma emoção e como as pessoas transmitem as emoções de forma diferente as características no rosto da pessoa também variam de pessoa para pessoa. E como foi discutido no tópico da proporção 50/50, o HOG exalta os contornos da imagem, mas algumas das imagens no banco se diferenciam em um nível mais alto.

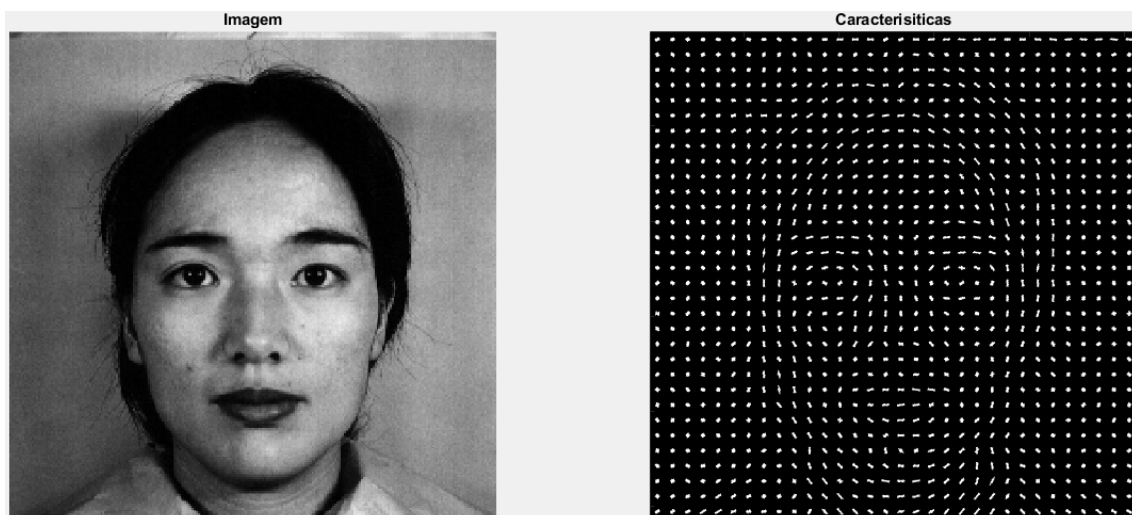
A Figura a seguir ilustra esse fato. A mulher da primeira figura pertence a classe ANGRY (Raiva) porém foi classificada como NEUTRAL (Neutro). Até visualmente é difícil afirmar que a mulher está expressando raiva. Além disso, o plot de seu HOG em comparação com o da mulher que ela foi classificada são muito similares diferindo entre si muito pouco.

Figura 59: Exemplo de mulher classificada na classe incorreta



Fonte: Autoral

Figura 60: Mulher ao qual a mulher representada na figura anterior foi classificada



Fonte: Autoral

Desta forma, para melhorar a performance do classificador para esse problema, pode ser necessário a aplicação de mais alguma técnica de processamento digital de imagens que torne mais visível para o algoritmo características mais implícitas das classes.

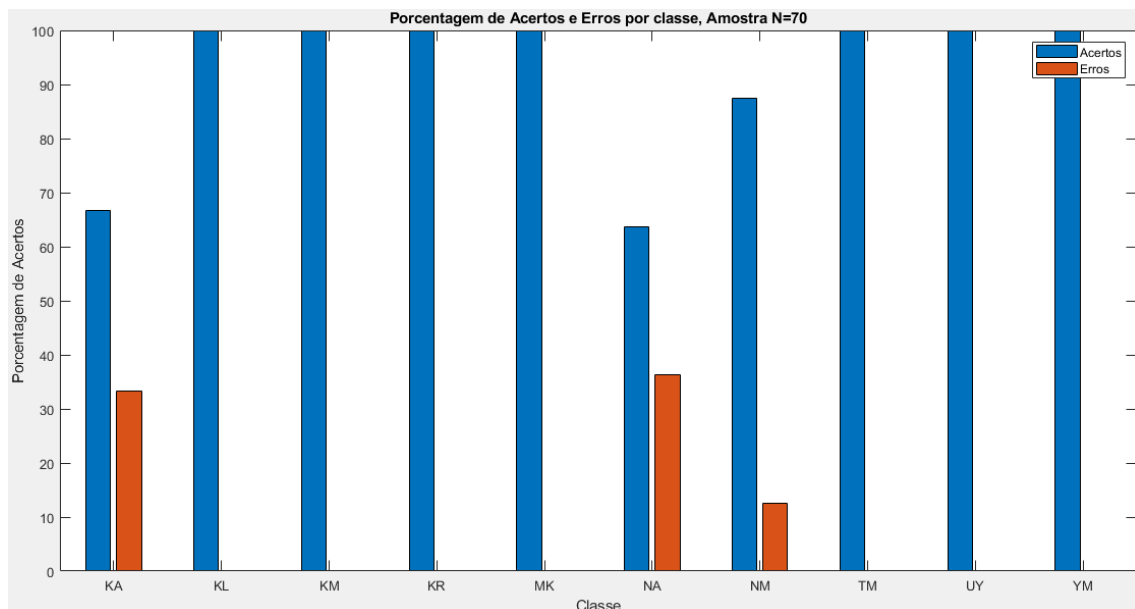
Outro fato que pode corroborar para esse problema é a separação dos dados que é feita de forma simples, pegando as primeiras imagens para treino e as demais para teste. A exploração da separação aleatória pode vir a melhorar o resultado. Já que com isso, serão utilizadas mais imagens com pessoas diferentes para o treino e teste.

4. Utilize o algoritmo HOG para extrair características e treine a PCA para classificar as pessoas.

A função main foi configurada para ser possível classificar pessoas no banco de dados. O banco de dados foi dividido em treino e teste nas configurações 50%/50%, 60%/40% e 70%/30%. Os resultados obtidos foram os seguintes.

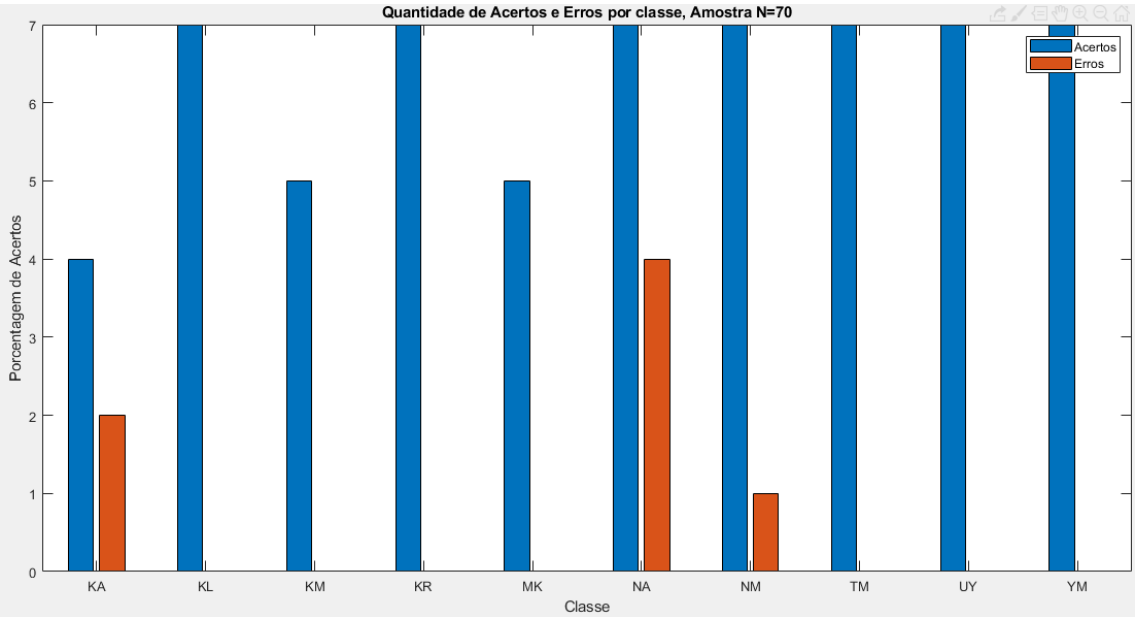
Configuração 50%/50% Amostra de 70 imagens para teste

Figura 61: Porcentagem de Acertos e Erros por classe, Amostra N= 70 para classificação de pessoas utilizando a técnica HOG como pré-processamento



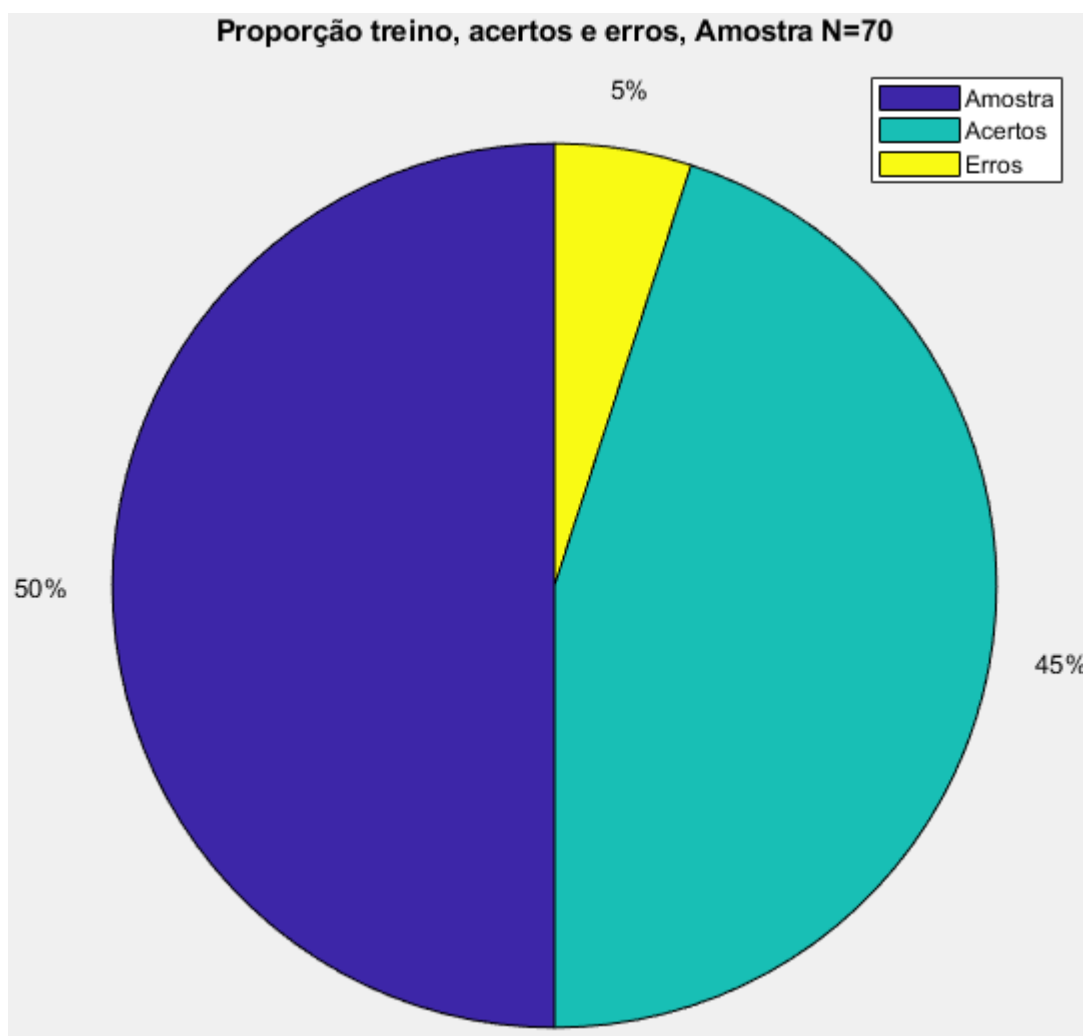
Fonte: Autoral

Figura 62: Quantidade de Acertos e Erros por classe, Amostra N= 70 para classificação de pessoas utilizando a técnica HOG como pré-processamento



Fonte: Autoral

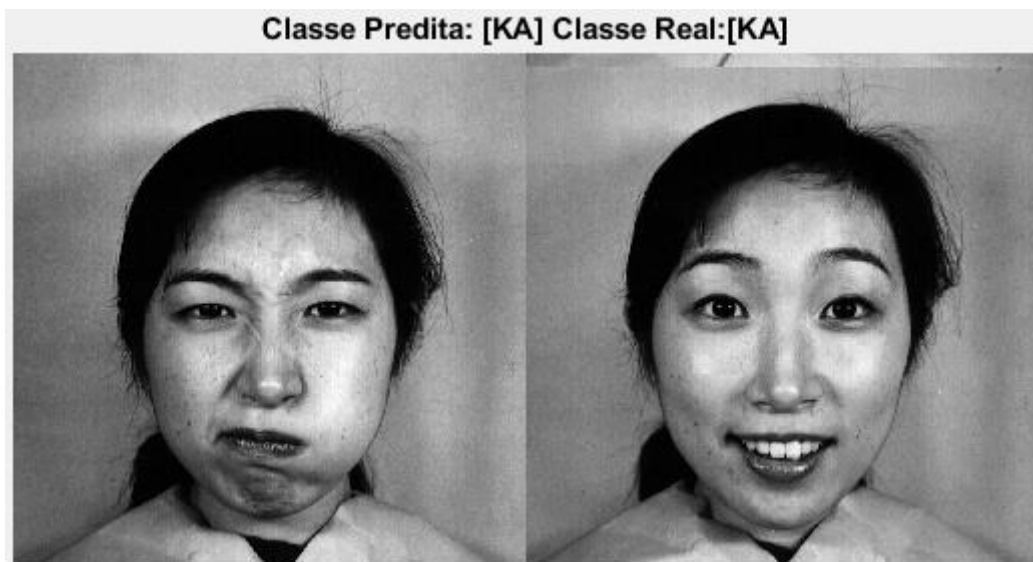
Figura 63: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 70 para classificação de pessoas utilizando a técnica HOG como pré-processamento



Fonte: Autoral

A média de acertos foi de 90% e a média de erros foi de 10%. Para essa configuração a PCA conseguiu classificar bem os dados. De 70 amostras para teste, o algoritmo classificou corretamente 63. A Figura a seguir ilustra uma classificação correta realizada pelo algoritmo.

Figura 64: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 65: Exemplo de classificação incorreta realizada pelo algoritmo

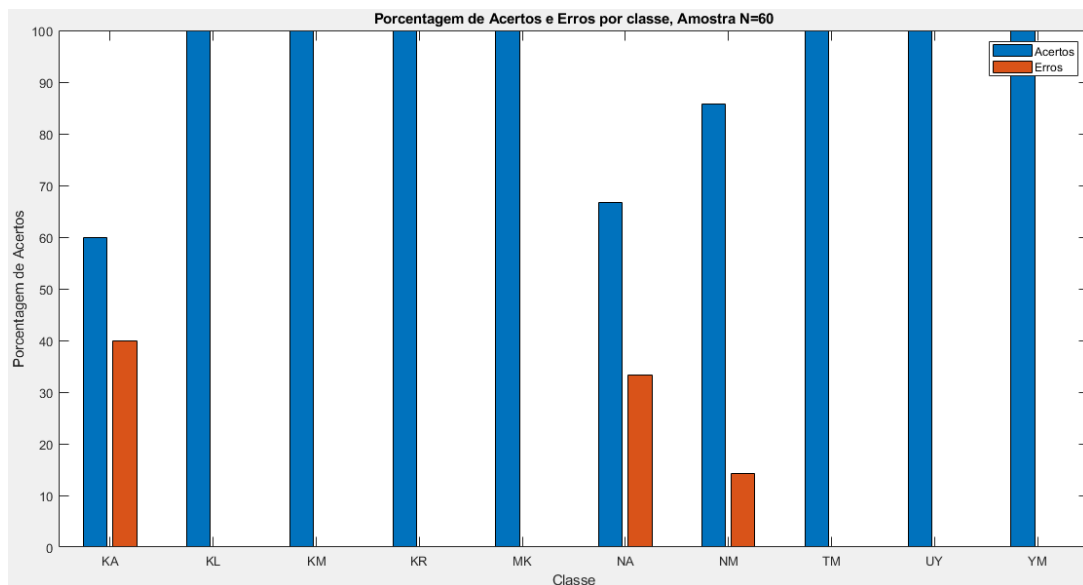


Fonte: Autoral

Configuração 60%/40% Amostra de 60 imagens para teste

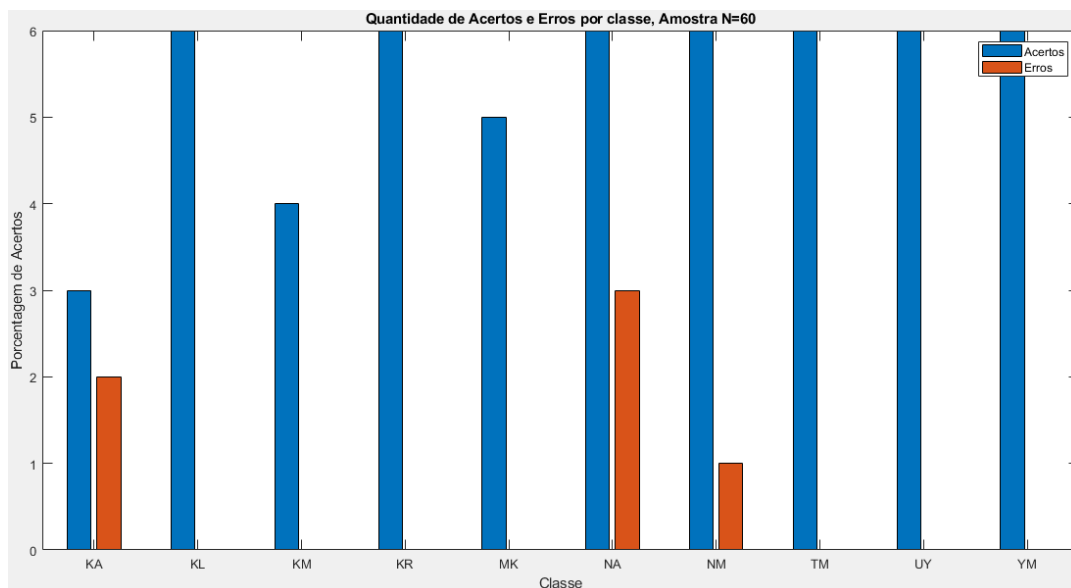
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

Figura 66: Porcentagem de Acertos e Erros por classe, Amostra N= 60 para classificação de pessoas utilizando a técnica HOG como pré-processamento



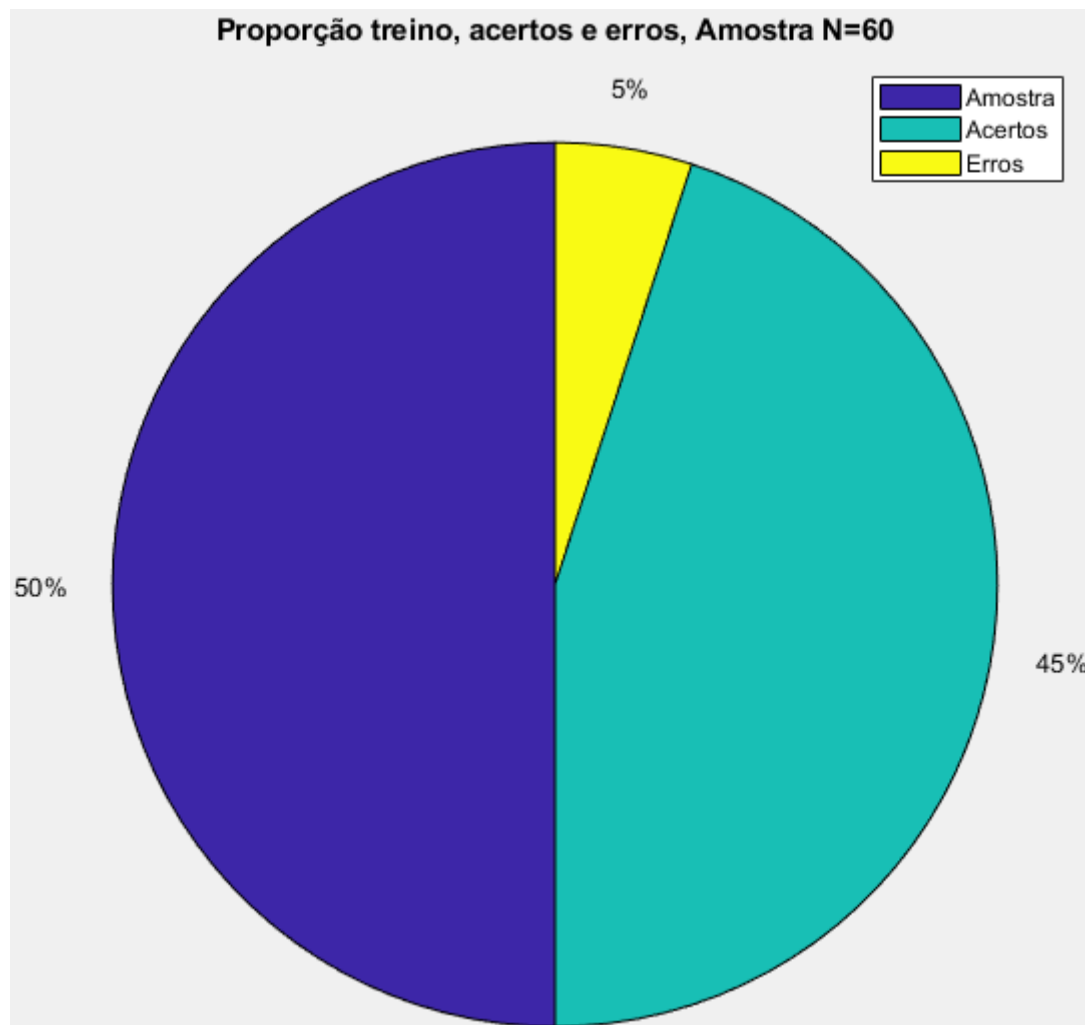
Fonte: Autoral

Figura 67: Quantidade de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

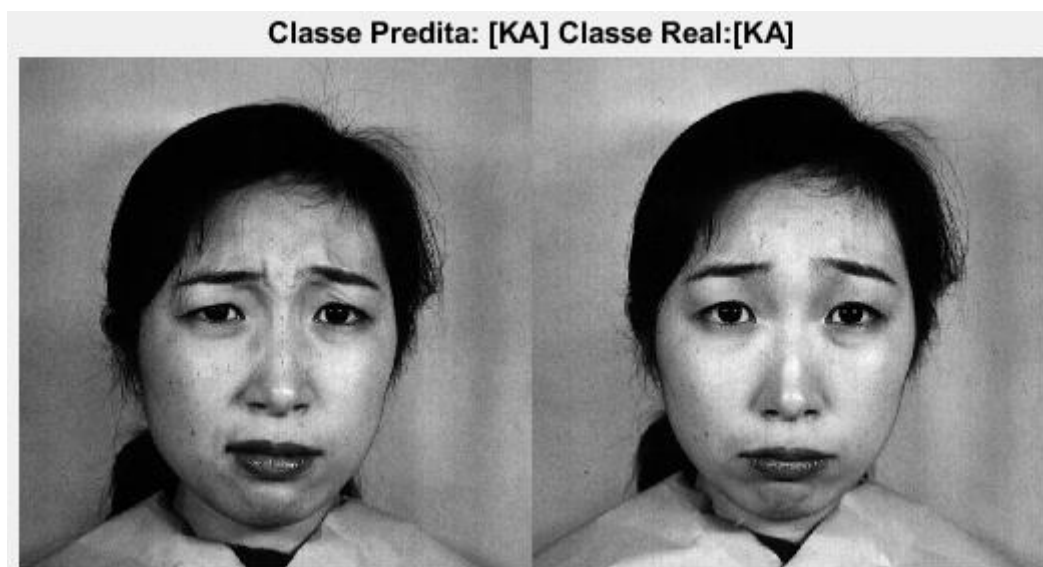
Figura 68: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 60 para classificação de pessoas utilizando a técnica HOG como pré-processamento



Fonte: Autoral

A média de acertos foi de 90% e a média de erros foi de 10%. Para essa configuração a PCA conseguiu classificar bem os dados. De 60 amostras para teste, o algoritmo classificou corretamente 54. As Figuras a seguir, ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 69: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 70: Exemplo de classificação incorreta realizada pelo algoritmo

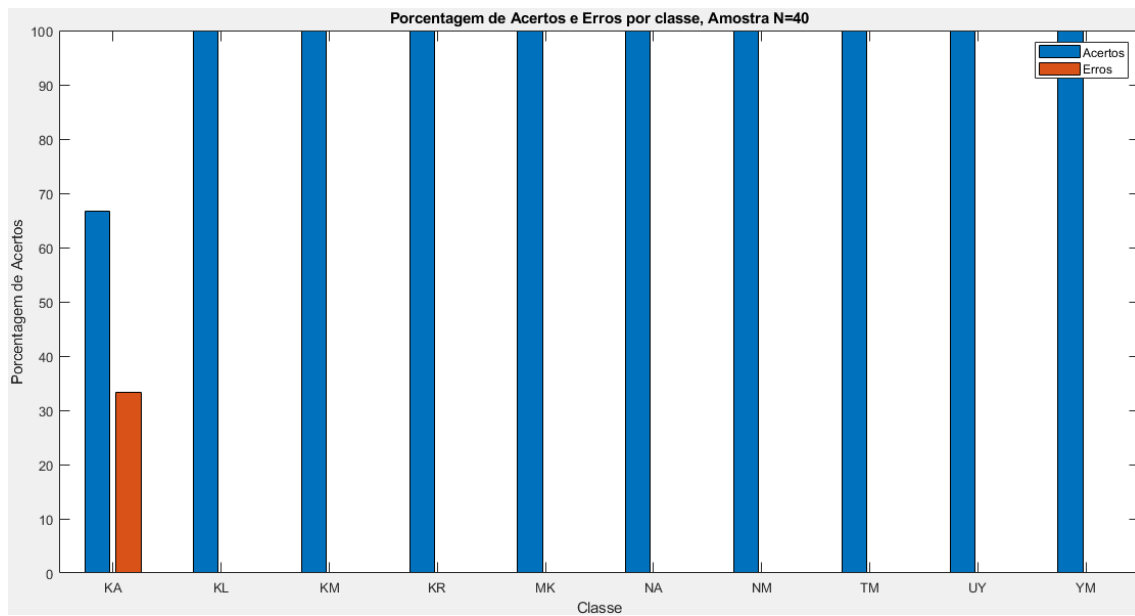


Fonte: Autoral

Configuração 70%/30% Amostra de 40 imagens para teste

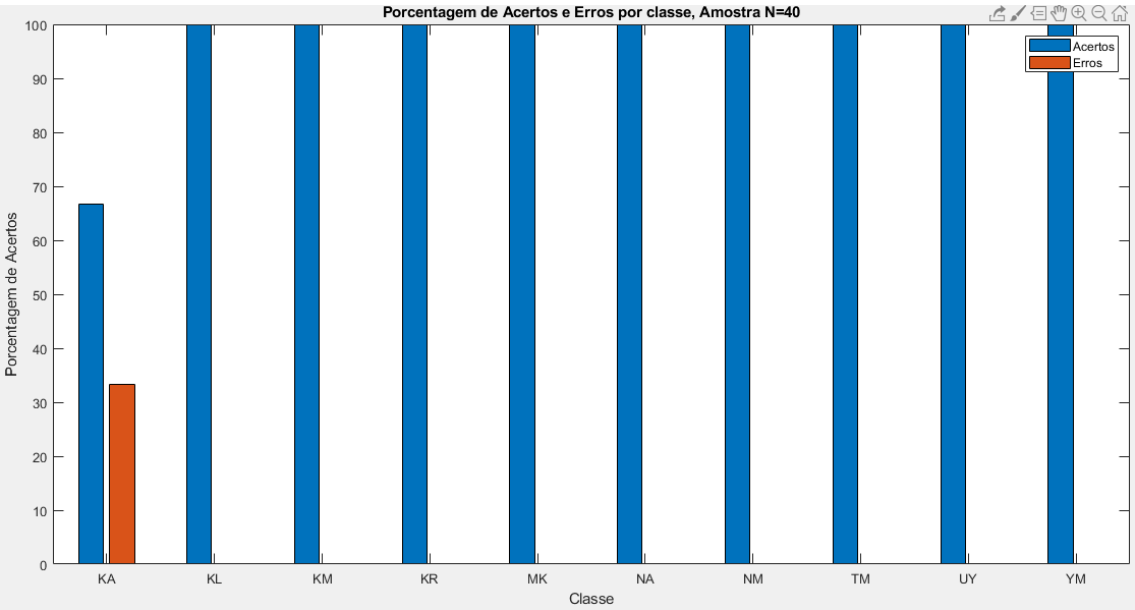
As figuras a seguir trazem os plots para as classes do banco de dados.

Figura 71: Porcentagem de Acertos e Erros por classe, Amostra N= 40 para classificação de pessoas utilizando a técnica HOG como pré-processamento



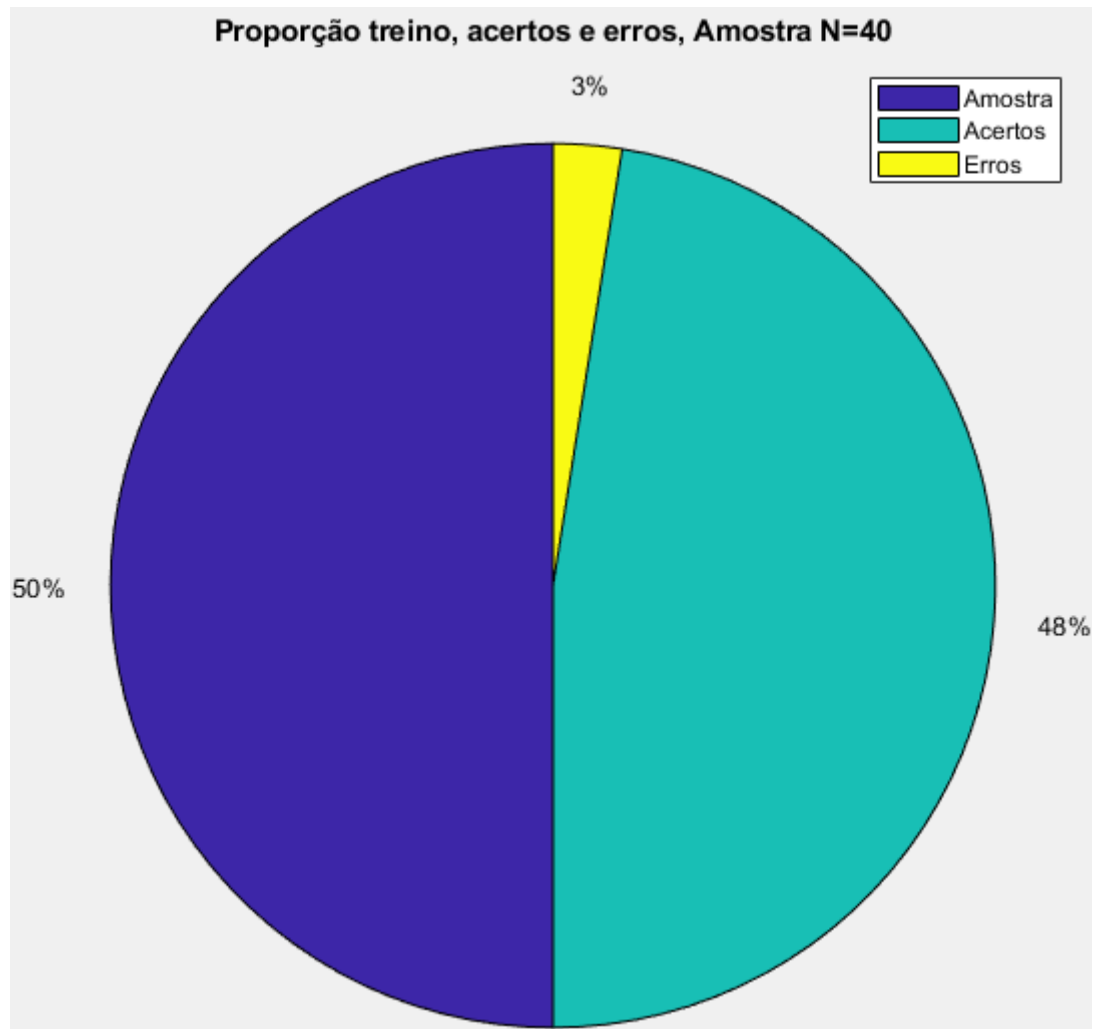
Fonte: Autoral

Figura 72: Porcentagem de Acertos e Erros por classe, Amostra N= 40 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

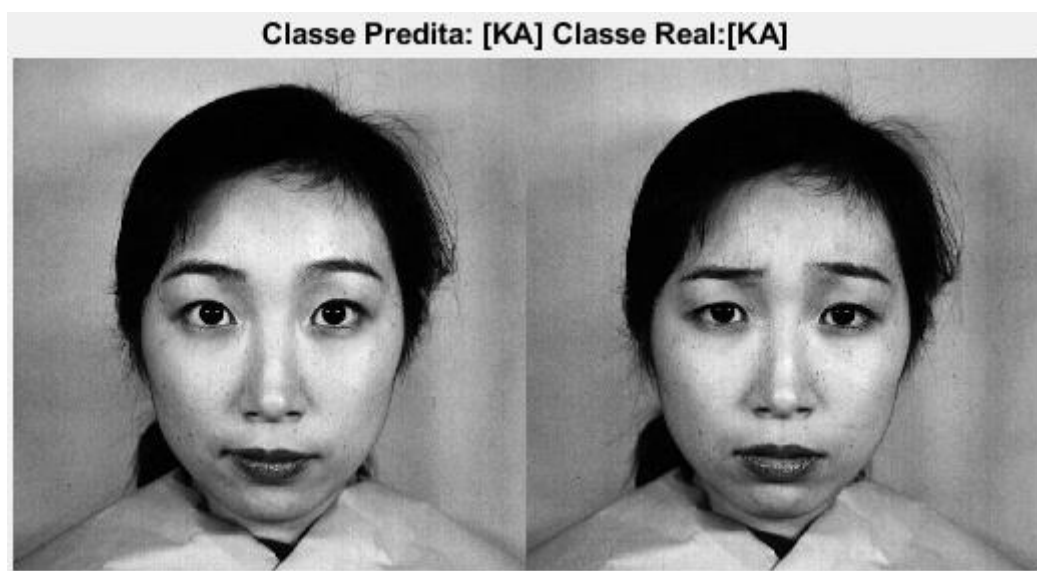
Figura 73: Porcentagem de Acertos e Erros por classe, Amostra N= 42 para classificação de emoções utilizando a técnica HOG como pré-processamento



Fonte: Autoral

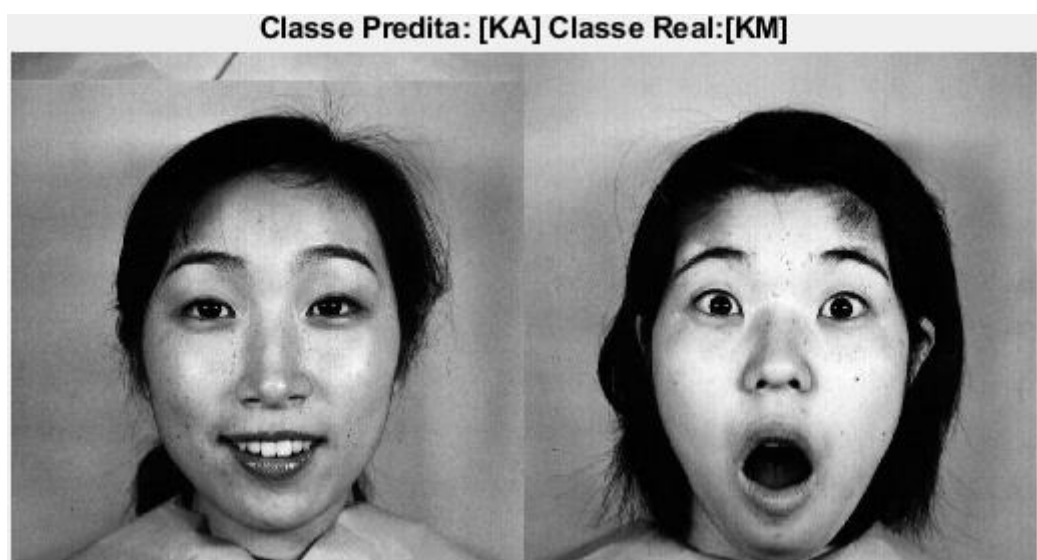
A média de acertos foi de 95% e a média de erros foi de 5%. Para essa configuração a PCA conseguiu classificar bem os dados. De 40 amostras para teste, o algoritmo classificou corretamente 38. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura 74: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura 75: Exemplo de classificação incorreta realizada pelo algoritmo



Fonte: Autoral

A tabela a seguir resume os resultados obtidos com as diferentes proporções de treino/teste.

Tabela 4 – Resumo das métricas de avaliação para as diferentes proporções de treino/teste

Proporção Treino/Teste	Acerto		Erro	
	Quantidade	Porcentagem	Quantidade	Porcentagem
50/50	63	90,00%	7	10,00%
60/40	54	90,00%	6	10,00%
70/30	38	95,00%	2	5%

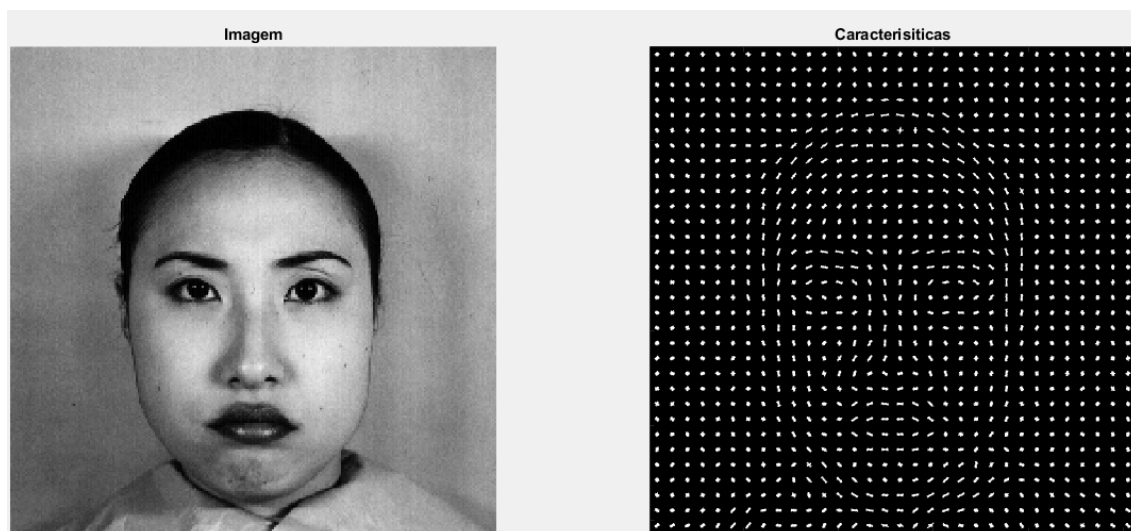
Fonte: Autoral

A partir da tabela acima é possível verificar que quanto maior a amostra de treino melhor o algoritmo classifica os dados. Isso pode estar relacionado ao fato de que se aumenta o número de dados para treino e consequentemente obtém-se uma melhora na performance do algoritmo.

Dessa forma o algoritmo se mostrou eficaz na classificação das pessoas. Como as características das pessoas é mais fácil de distinguir de uma pessoa para outra o classificador conseguiu obter bons resultados. Já em comparação com a classificação de emoções, esse tipo de classificação pode levar em conta características mais distinguíveis de uma pessoa a outra como por exemplo o cabelo da pessoa, a distância entre os olhos etc.

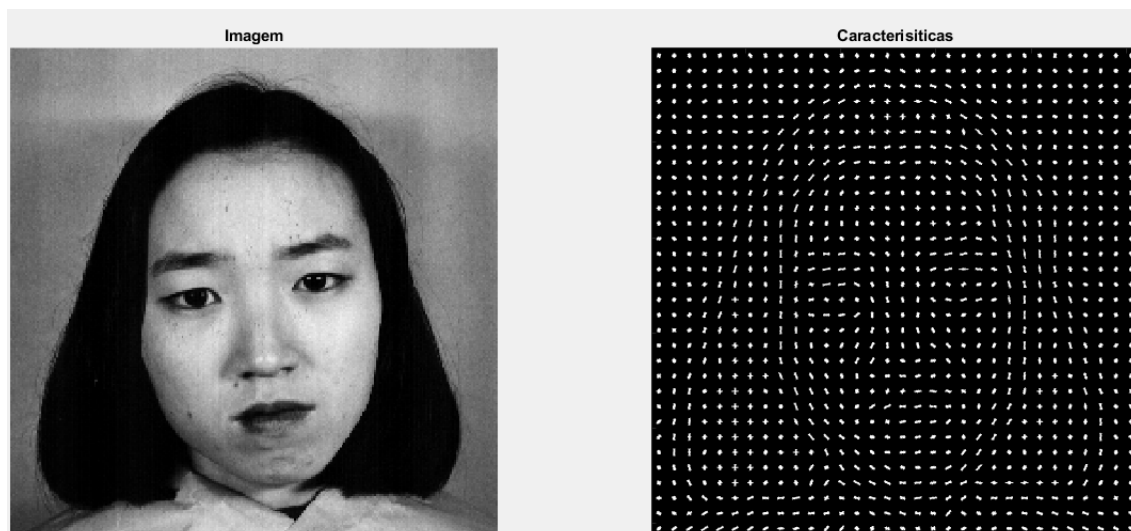
A Figura a seguir ilustra esse fato. A primeira imagem mostra uma mulher com o cabelo curto, o plot das características extraídas mostra que a região mais próxima da cabeça apresenta os pontos verticais, enquanto na outra imagem os pontos mais próximos da cabeça estão levemente inclinados devido a presença do cabelo. Com isso, o algoritmo conseguiu melhor aprender essas características.

Figura 76: Exemplo de mulher com cabelo curto com as características extraídas



Fonte: Autoral

Figura 77: Exemplo de mulher com cabelo longo com as características extraídas



Fonte: Autoral

CONCLUSÃO

A tabela a seguir resume a porcentagem de acerto para cada algoritmo no problema de classificação de emoções.

Tabela 5 – Comparação da porcentagem de acertos para os algoritmos no problema de classificação de emoções

Porporção Treino/Teste	Emoções		
	Algoritmo		
	PCA	SURF	HOG
50/50	20,00%	17,14%	22,86%
60/40	19,64%	17,86%	17,86%
70/30	14,29%	14,29%	19,05%

Fonte: Autoral

A partir da tabela, o algoritmo que melhor se saiu no problema foi o HOG com a proporção de treino e teste de 50/50 com 22,86% de acerto. Em comparação com o PCA puro, observa-se certa melhora na classificação.

As porcentagens de classificação baixa, podem estar relacionadas ao fato de que o banco usado para treino nesse problema é mais heterogêneo, isto é, as pessoas expressam as emoções de forma diferentes e em alguns casos as emoções de algumas pessoas se assemelham com outras emoções de outras pessoas. Por exemplo, a pessoa A expressa o medo e a pessoa B expressa a surpresa de forma semelhante. Assim, uma análise mais profunda das microexpressões pode vir a melhorar a acurácia do modelo, bem como a aplicação de técnicas de processamento digital de imagens também pode vir a melhorar o cenário.

Tabela 6 – Comparação da porcentagem de acertos para os algoritmos no problema de classificação de pessoas

Porporção Treino/Teste	Pessoas		
	Algoritmo		
	PCA	SURF	HOG
50/50	78,57%	32,86%	90,00%
60/40	86,67%	35,00%	90,00%
70/30	92,50%	35,00%	95,00%

Fonte: Autoral

A partir da tabela o algoritmo que melhor se saiu no problema foi o HOG com a proporção de treino e teste de 70/30 com 95% de acerto. Em comparação com o PCA puro, observa-se certa melhora na classificação.

Os resultados para esse problema foram promissores quando comparados com a classificação de emoções, de fato, nesse caso o modelo é treinado com dados mais homogêneos da mesma pessoa o que pode estar relacionado com a acurácia obtida. O algoritmo SURF foi o que se saiu pior, porém espera-se que ao usar um número maior de pontos característicos possa vir a melhorar a acurácia obtida.

REFERÊNCIAS

AHMAD, Rehan. **A Take on H.O.G Feature Descriptor**. Disponível em: <https://medium.com/analytics-vidhya/a-take-on-h-o-g-feature-descriptor-e839ebba1e52>. Acesso em: 20 maio 2020.

GONZALEZ, Rafael C., WOODS, Richard E. **Processamento Digital de Imagem**. São Paulo: Sv, 2010.

Lyons, Michael; Kamachi, Miyuki; Gyoba, Jiro. **The Japanese Female Facial Expression (JAFPE) Database**. Disponível em: <https://zenodo.org/record/3451524#.XsV39mhKjIU>. Acesso em: 20 maio 2020.

MALLICK, Satya. **Histogram of Oriented Gradients**. Disponível em: <https://www.learnopencv.com/histogram-of-oriented-gradients/>. Acesso em: 20 maio 2020.

MATHWORKS. **ExtractHOGFeatures**. Disponível em: <https://www.mathworks.com/help/vision/ref/extracthogfeatures.html>. Acesso em: 20 maio 2020.

MATHWORKS. **DetectSURFFeatures**. Disponível em: <https://www.mathworks.com/help/vision/ref/detectsurffeatures.html>. Acesso em: 20 maio 2020.

MATHWORKS. **SelectStrongest**. Disponível em: <https://www.mathworks.com/help/vision/ref/kazepoints.selectstrongest.html>. Acesso em: 20 maio 2020.

MATHWORKS. **ExtractFeatures**. Disponível em: <https://www.mathworks.com/help/vision/ref/kazepoints.selectstrongest.html>. Acesso em: 20 maio 2020.

SANTO, Rafael do Espírito. Principal Component Analysis applied to digital image compression. **Einstein (São Paulo)**, [s.l.], v. 10, n. 2, p. 135-139, jun. 2012. FapUNIFESP (SciELO). <http://dx.doi.org/10.1590/s1679-45082012000200004>.

TYAGI, Deepanshu. **Introduction To Feature Detection And Matching**. 2019. Disponível em: <https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d>. Acesso em: 18 maio 2020.

TYAGI, Deepanshu. **Introduction to SURF (Speeded-Up Robust Features)**. 2019. Disponível em: <https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e>. Acesso em: 18 maio 2020.

Apêndice A – Código main.m

```
clear all;
close all;
clc;

% Parametros
% porcentagem -> Porcentagem de dados que serão da classe de treino
% path -> Caminho da pasta que contém as subpastas que são as
classes
% extensao -> Extensão das imagens
% tipo, tipo de algoritmos para extração de características
%tipo = "SURF";
%tipo = "HOG";
porcentagem = 70;
path = 'D:\Estudos\Eng. de Computação\Processamento Digital de
Imagens\Listas\Lista 6\Pessoas';
extensao = '.tiff';
tipo = "SURF";

[label_treino, label_teste, dados_treino, dados_teste ....
img_treino, img_teste, qtd_classes] = funcao(path ,porcentagem,
extensao , tipo);

function [label_treino, label_teste, dados_treino, dados_teste ....
img_treino, img_teste, qtd_classes] = funcao(path, porcentagem,
extensao, tipo)
    % Parametros
    % porcentagem -> Porcentagem de dados que serão da classe de
treino
    % path -> Caminho da pasta que contém as subpastas que são as
classes
    % extensao -> Extensão das imagens
    % Retornos
    % label_treino -> Vetor auxiliar com as labels (classes) da
parte da
    % parte do banco que será usada como treino
    % label_teste -> Vetor auxiliar com as labels (classes) da
parte da
    % parte do banco que será usada como teste
    % dados_treino -> Dados que serão usados como treino da PCA
    % dados_teste -> Dados que serão usados como teste da PCA
    % qtd_classes -> Quantidade de classes do banco de dados

    % Para mostrar uma imagem certa e uma imagem errada
    aux = 0;
    aux1 = 0;

    img_a = [];
    img_e = [];

    [label_treino, label_teste, dados_treino, dados_teste ....
img_treino, img_teste, qtd_classes] = lerImgs(porcentagem,
path, extensao, tipo);

    % Qunatidade de amostras de teste
    tam = size(dados_teste, 2);
    qtd_dados = tam;
```

```

% Gera a PCA
[P PC mn] = GerarPCs(dados_treino);

% Vetores que contém a quantidade de acertos e erros por cada
classe
acertos = zeros(qtd_classes,1)';
erros = zeros(qtd_classes,1)';

acertos = zeros(qtd_classes, 1);
erros = zeros(qtd_classes, 1);

% Converte as categorias para números para ser possível contar a
% quantidade de erros e acertos
c = grp2idx(categorical(label_treino));

for i = 1: tam

    % Gera a amostra
    amostra = ProjetarAmostra(dados_teste(:, i),mn,P);
    d = round(Classificar(PC,amostra));

    indice_amostra = c(d);

    if isequal(label_teste(i), label_treino(d))

        acertos(indice_amostra) = acertos(indice_amostra) + 1;
        if(aux == 0)
            figure;
            pred =
imread(strcat(path, '\', label_teste(i), '\', img_teste(i)));
            real = imread(strcat(path, '\', label_treino(d),
'\', img_treino(d)));
            imshowpair(real, pred, 'montage');
            title(strcat('Classe Predita: ', label_treino(d),
.....
                        ']' Classe Real: ', '[' label_teste(i), '']));
            aux = aux + 1;
        end
        img_a = [img_a, img_teste(i)];
    else
        erros(indice_amostra) = erros(indice_amostra) + 1;
        if(aux1 == 0)
            figure;

            pred =
imread(strcat(path, '\', label_teste(i), '\', img_teste(i)));
            real = imread(strcat(path, '\', label_treino(d),
'\', img_treino(d)));
            imshowpair(real, pred, 'montage');
            title(strcat('Classe Predita: ', label_treino(d),
.....
                        ']' Classe Real: ', '[' label_teste(i), '']));
            aux1 = aux1 + 1;
        end
        img_e = [img_e, img_teste(i)];
    end
end
end

```

```

qtd_acerto = sum(acertos);
qtd_erro = sum(erros);

media_acerto = (100 * qtd_acerto) / qtd_dados;
media_erro = (100 * qtd_erro) / qtd_dados;

% Converte as labels para categorias
labels = categorical(unique(label_treino));

% ---- Plots ----
% Grafico de barras de porcentagem de acertos e erros
figure;
bar(labels, ([acertos erros]./[acertos + erros])*100);
legend('Acertos', 'Erros');
title(strcat('Porcentagem de Acertos e Erros por classe, Amostra
N=', num2str(tam)));
xlabel('Classe');
ylabel('Porcentagem de Acertos');

% Grafico de barras de quantidade de acertos e erros
figure;
bar(labels, [acertos erros]);
legend('Acertos', 'Erros');
title(strcat('Quantidade de Acertos e Erros por classe, Amostra
N=', num2str(tam)));
xlabel('Classe');
ylabel('Porcentagem de Acertos');

% Grafico de piza de porcentagem da amostra de acertos e erros
figure;
labels = {'Amostra', 'Acertos', 'Erros'};
pie([tam, qtd_acerto, qtd_erro]);
title(strcat('Proporção treino, acertos e erros, Amostra
N=', num2str(tam)));
legend(labels);

% Grafico de piza de porcentagem da amostra de acertos e erros
figure;
labels = {'Acertos', 'Erros'};
pie([qtd_acerto, qtd_erro]);
title(strcat('Proporção acertos e erros, Amostra
N=', num2str(tam)));
legend(labels);

% Quantidade de acerto e erro geral e média de erro e acerto
geral
fprintf('\nQuantidade de Acertos: %i\nPorcentagem de Acertos:
%.2f\n', qtd_acerto, media_acerto);
fprintf('\nQuantidade de Erros: %i\nPorcentagem de Erros:
%.2f\n', qtd_erro, media_erro);
end

```


Apêndice B – Classificar.m

```
%Classificador: Distância Euclidiana
%Entrada:
%   PC = Conjunto de treinamento
%   x = amostra a ser comparada
%Saída
%d = posição do item com a menor distância euclidiana entre a
amostra e as
%colunas do conjunto de treinamento
function d = Classificar(PC,x)
    minDist = norm(PC(:,1) - x);
    d = 1;
    for j = 2:size(PC)
        EucDist = norm(PC(:,j) - x);
        if minDist > EucDist
            d = j;
            minDist = EucDist;
        end
    end
end
```

Apêndice C – GerarPCs.m

```
%Gera a matriz de componentes principais
%Entrada:
%   data = matriz de dados. Cada amostra de medições é representada
em uma
%coluna
%saída:
%   P = dados no novo espaço vetorial (autofaces no caso de imagens)
%   PC = matriz componentes principais
%   mn = média de cada coluna da matriz de dados
function [P PC mn] = GerarPCs(data)
    [M,N] = size(data);
    mn = mean(data,2); %calcular a média
    data = double(data) - repmat(mn,1,N); %imagens - media
    covariancia = data' * data; %matriz de covariância
    [PC,V] = eig(covariancia); %autovetores e autovalores

    V = diag(V);

    clear covariancia, clear M, clear N;

    %ordenar autovetores e autovalores

    [junk,rindices] = sort(-1*V);
    V = V(rindices);
    V = diag(V);
    PC = PC(:,rindices);

    clear junk , clear rindices, clear V;

    %cálculo das autofaces

    P = data * PC;
    i = size(PC,2);

    clear PC;

    %projeção das imagens no espaço de faces
    PC = [];
    for j = 1:i
        temp = P' * data(:,j);
        PC = [PC temp];
    end

    clear i, clear j, clear temp;
end
```

Apêndice D – lerImgs

```
function [label_treino, label_teste, dados_treino, dados_teste ....
    img_treino, img_teste, qtd_classes] = lerImgs(porcentagem, path,
    extensao, tipo)
    % Parametros
    %   porcentagem -> Porcentagem de dados que serão da classe de
    treino
    %   path -> Caminho da pasta que contém as subpastas que são as
    classes
    %   extensao -> Extensão das imagens
    %   Tipo -> Indica o tipo de operação que será empregada nas
    imagens
    % Retornos
    %   label_treino -> Vetor auxiliar com as labels (classes) da
    parte da
    %   parte do banco que será usada como treino
    %   label_teste -> Vetor auxiliar com as labels (classes) da
    parte da
    %   parte do banco que será usada como teste
    %   dados_treino -> Dados que serão usados como treino da PCA
    %   dados_teste -> Dados que serão usados como teste da PCA
    %   qtd_classes -> Quantidade de classes do banco de dados

    dados_treino = [];
    dados_teste = [];
    label_treino = [];
    label_teste = [];
    img_treino = [];
    img_teste = [];

    % Acessa as pastas
    S = dir(fullfile(path));

    % Percorre cada pasta que contém as classes
    qtd_classes = numel(S) - 2;
    for pasta = 3:numel(S)
        % Concatena as strings para formar o nome de cada pasta
        pasta_classe = S(pasta).name;
        caminho = strcat(S(pasta).folder, '\\', pasta_classe, '\\');
        classe = dir(fullfile(caminho, strcat('*', extensao)));

        % Quantidade treino e teste
        qtd_dados = (size(classe, 1));

        % Separa a porcentagem de treino
        qtd_treino = round((qtd_dados * porcentagem) / 100);
        i = 1;
        for i = 1: qtd_treino

            w = fullfile(caminho, classe(i).name);
            x = imread(w);

            % Faz o Surf
            if (tipo == "SURF")
                y = SURF(x);
```

```

        end

%         % Faz o HOG
        if (tipo == "HOG")
            y = HOG(x);
        end

        if (tipo == "SIMPLES")
            y = SIMPLES(x);
        end

        %y = reshape(x, [size(x,1)*size(x,2),1]);
        dados_treino = [dados_treino, y];
        label_treino = [label_treino, string(pasta_classe)];
        img_treino = [img_treino, string(classe(i).name)];
    end
    % Separa a porcentagem de teste
    for j = i+1: qtd_dados
        w = fullfile(caminho, classe(j).name);
        x = imread(w);

        % Faz o Surf
        if (tipo == "SURF")
            y = SURF(x);
        end

%         % Faz o HOG
        if (tipo == "HOG")
            y = HOG(x);
        end

        if (tipo == "SIMPLES")
            y = SIMPLES(x);
        end

        dados_teste = [dados_teste, y];
        label_teste = [label_teste, string(pasta_classe)];
        img_teste = [img_teste, string(classe(j).name)];
    end
end
end

% SURF
function [y] = SURF(x)
    points = detectSURFFeatures(x);
    teste2 = points.selectStrongest(5);
    teste2 = extractFeatures(x, teste2);
    y = gpuArray(reshape(teste2, [size(teste2,1)*size(teste2,2),1]));
end

% HOG
function [y] = HOG(x)
    [y, ~] = extractHOGFeatures(x);
    y=y';
end

```

```
% SIMPLES
function [y] = SIMPLES(x)
    x = rgb2gray(x);
    x = imresize(x, [100 100]);
    y = gpuArray(reshape(x, [size(x,1)*size(x,2),1]));
end
```

Apêndice E – ProjotarAmostra.m

```
%Projeta uma nova amostra no espaço vetorial P
%Entrada:
%   x = amostra a ser projetada
%   mn = média de cada coluna da matriz de dados
%   P = dados no novo espaço vetorial (autofaces no caso de imagens)
%Saída:
%   x = amostra no novo espaço vetorial
function x = ProjotarAmostra(x,mn,P)
    %x = reshape(x, [size(x,1)*size(x,2),1]);
    x = reshape(x, [size(x,1)*size(x,2),1]);
    x = double(x) - mn;
    x = P' * x;
end
```