DADOS DA LISTA: Extração de características Surf e Hog

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

Alunos:

Higor Alves Ferreira

Lucas Macedo da Silva

Vitor de Almeida Silva

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):

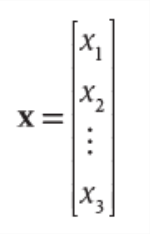
- **Características de locais específicos da imagem:** Também conhecidas como *keypoint features,* essas características são encontradas em locais específicos da imagem como cantos ou picos de montanhas;
- **Características que podem ser combinadas**: São características que podem ser combinadas por 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 Nx1 (vetor coluna), que são gerados por meio da extração de características de imagens associando n valores (descritores) a cada índice Xi de linha do vetor. A Figura 1 mostra o formato do vetor.

Figura 1: vetor de características genérico



Fonte: (GONZALEZ, WOODS, 2010)

- **Matriz de característica:** São estrutura vetoriais nxm, que contém valores de descritores de dados em cada ponto Mn,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

$$M_{n,m} = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} & \dots & M_{1,m} \\ M_{2,1} & . & . & & . \\ M_{3,1} & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ M_{nx1} & M_{n,2} & M_{n,3} & \dots & M_{n,m} \end{bmatrix}$$

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).

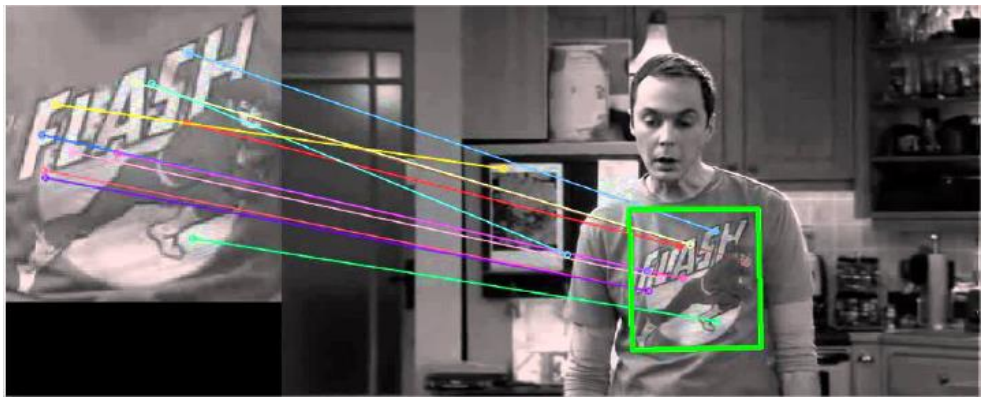
Com isso, o surfe é composto por 2 passos básicos segundo Tyagi (2019):

- **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

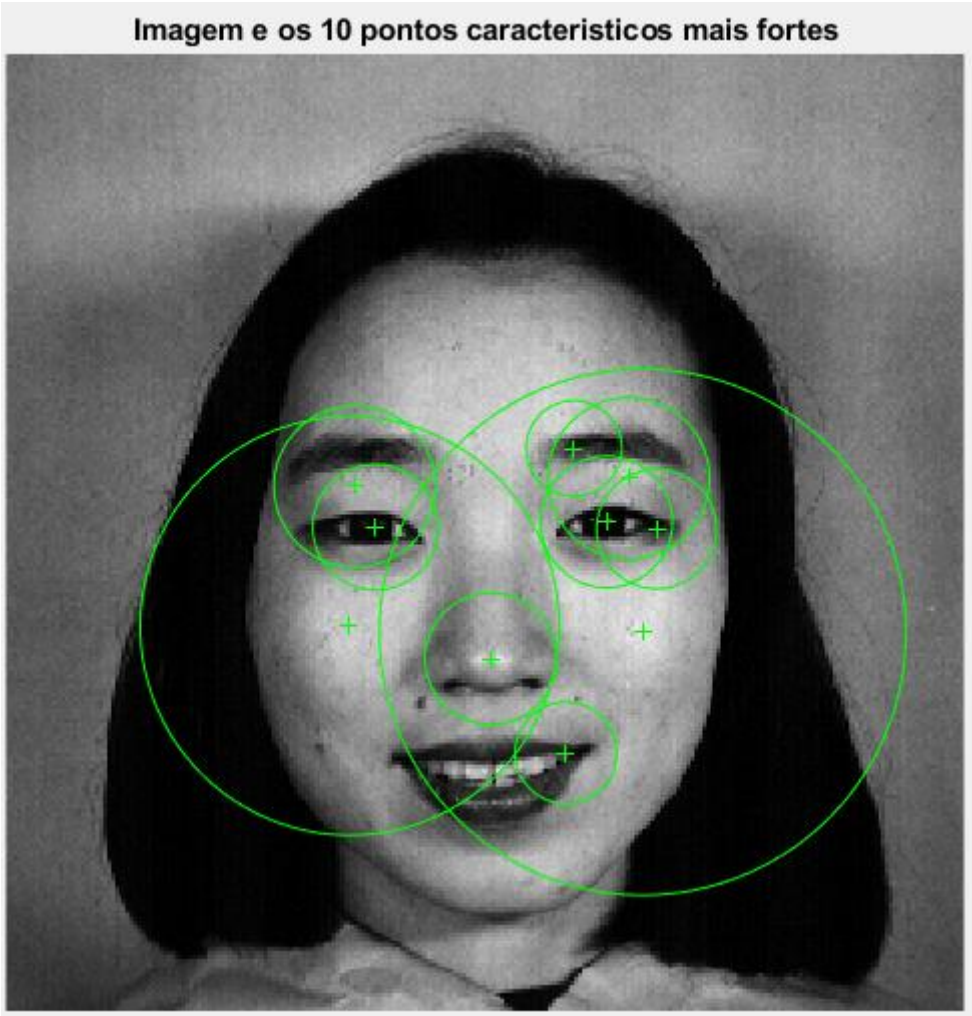


Fonte: (TYAGI, 2019)

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 N 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 caracteristicos mais fortes nessa imagem.

Figura 4: Exemplo de aplicação da técnica SURF para detectar caracteristicas 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:

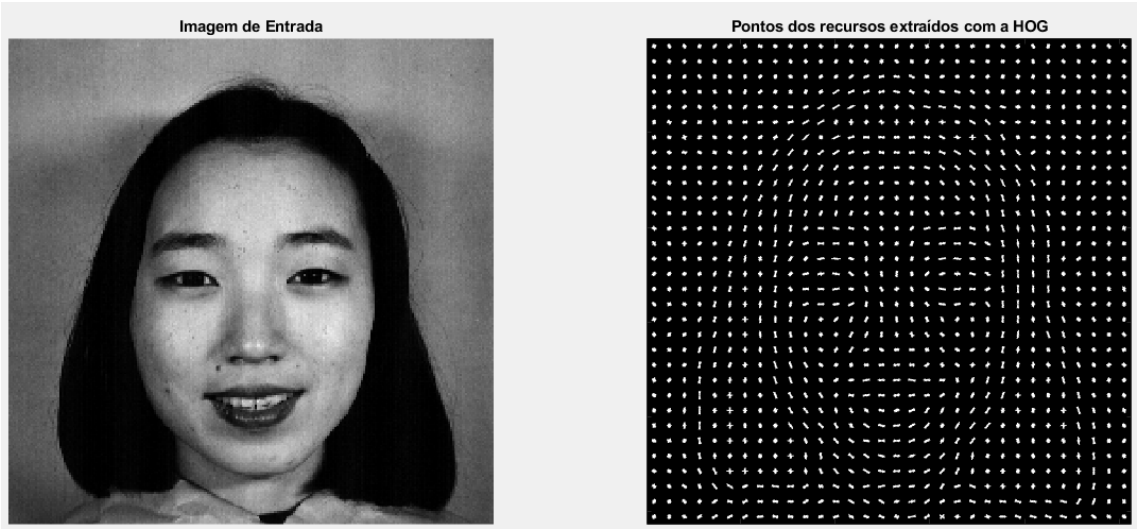
1. Pré-processamento: Aplicação de alguma operação na imagem, como por exemplo, o redimensionamento a conversão para a escala de cinza.
2. Cálculo dos gradientes das imagens: São calculados os gradientes horizontais e verticais da imagem. Para realizar esse procedimento, pode-se utilizar filtros de detecção 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, 0º, 20º, …, 180º
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 1xn 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));

    % Percore 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
```

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 raíz 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 = 50;
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;

    [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 contá a
```

```
    % quantidade de erros e acertos
    label_treino
    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;
        else
            erros(indice_amostra) = erros(indice_amostra) + 1;

        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]/(tam/qtd_classes)*100);
    legend('Acertos', 'Erros');
    title(strcat('Porcentagem de Acertos 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);

    % 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
```

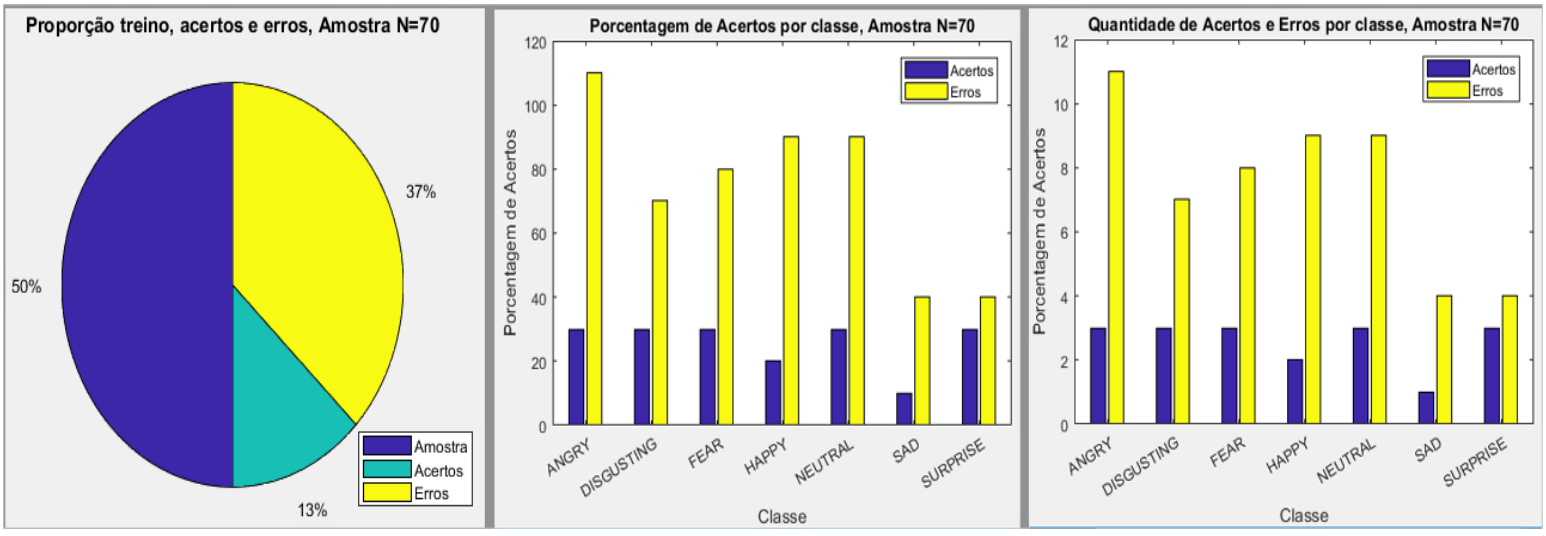
As variáveis porcentagem, path, extensão e tipo no início da função definem a porcentagem de treino e teste, o caminho em que a pasta raíz está localizada, a extensão das imagens e o tipo de função a ser aplicada nas imagens, respectivamente. Dessa forma para cada questão essas variáveis são configuradas.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

DESENVOLVIMENTO

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

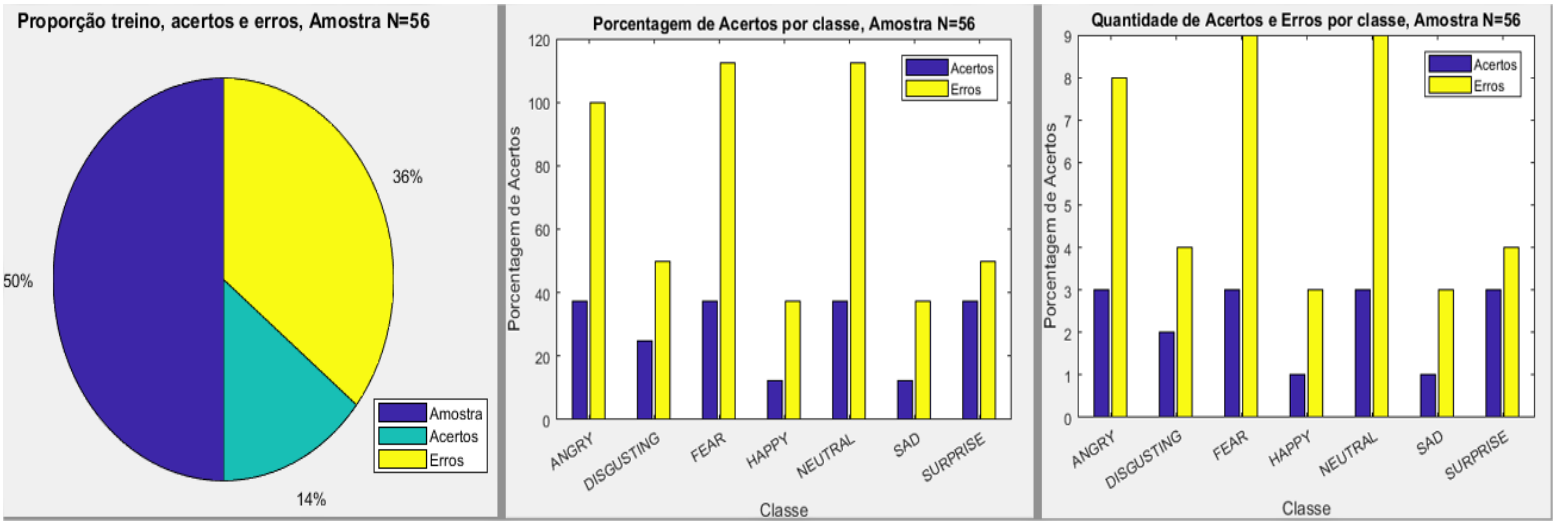
Resposta:

Figura 6: Resultados de acertos e erros, [treino, teste] = 50/50, N=70 (teste)



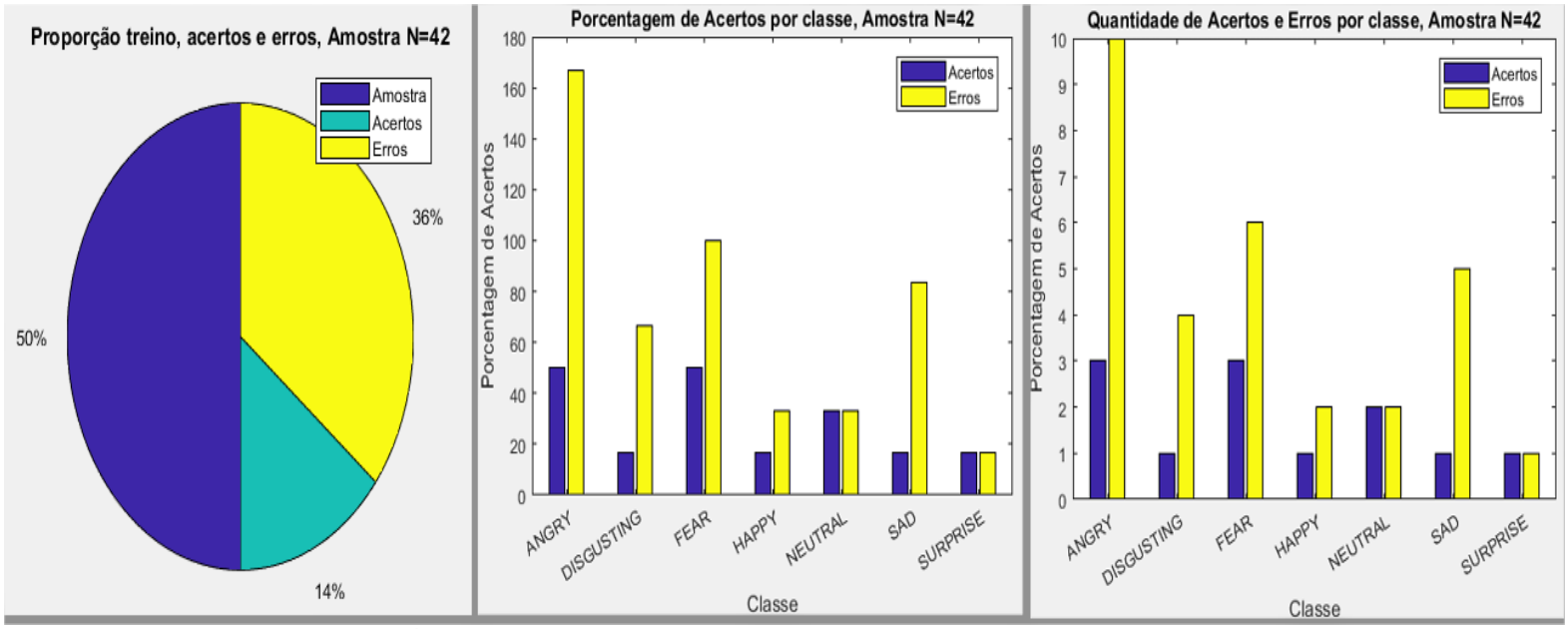
Fonte: Autoral

Figura 7: Resultados de acertos e erros, [treino, teste] = 60/40, N=56 (teste)



Fonte: Autoral

Figura 8: Resultados de acertos e erros, [treino, teste] = 70/30, N=56 (teste)



Fonte: Autoral

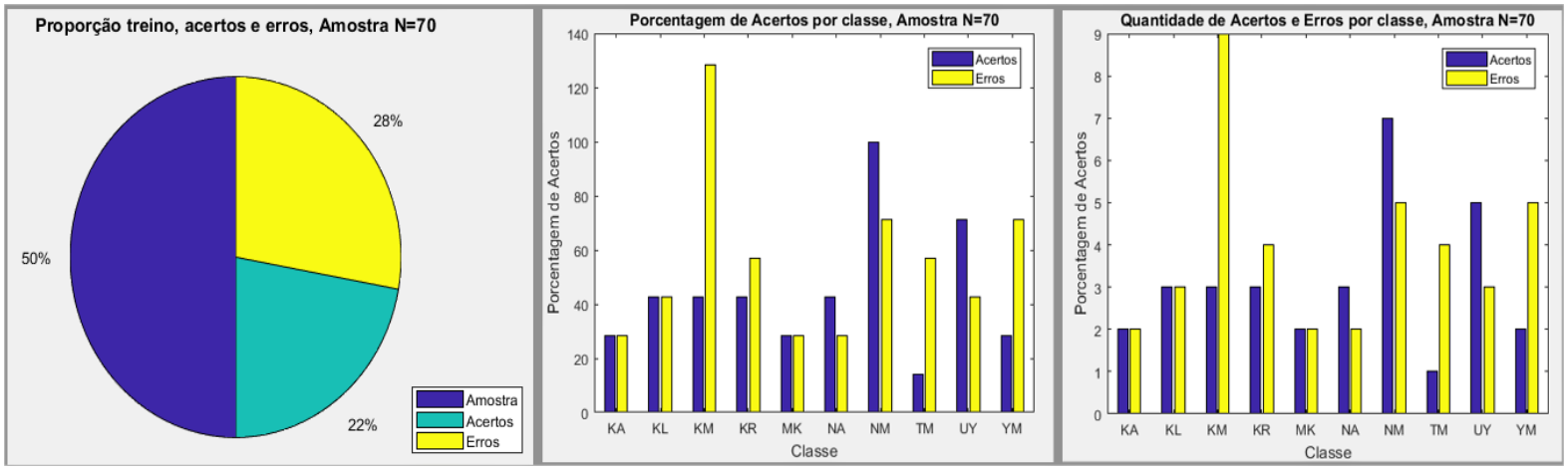
Por meio da análise das Figuras 6, 7 e 8, é possível notar que, a quantidade de erros foi maior que a de acertos. Desse modo, tem-se que para os valores de 60/40 e 70/40 a porcentagem permaneceu a mesma. Talvez a aplicação de alguma técnica de processamento nas imagens poderia melhorar de alguma forma os resultados.

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

Resposta:

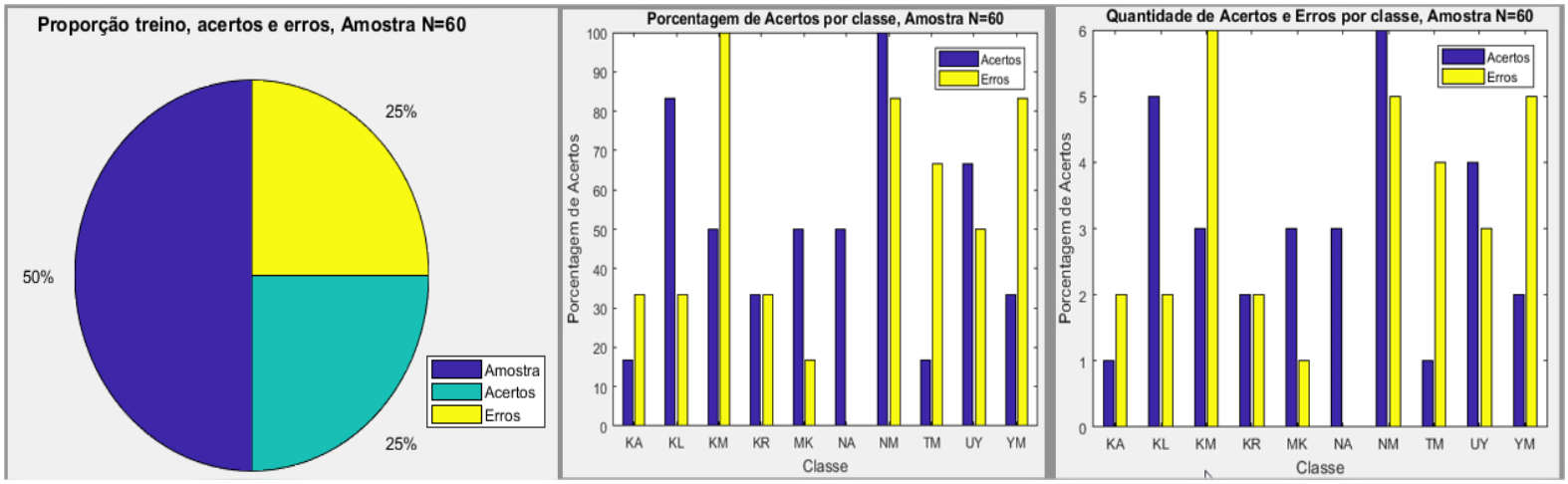
Paralelamente a questão 1, foi utilizado o método de SURF para extração das características e depois disso foi utilizado a PCA como classificador para classificar as pessoas. As Figuras 7, 8 e 9, mostram os resultados para os conjuntos de [treino, teste] = 50%/50%, [treino, teste] = 60%/40%, [treino,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

Figura 9: Resultados de acertos e erros, [treino, teste] = 50/50, N=70 (teste)



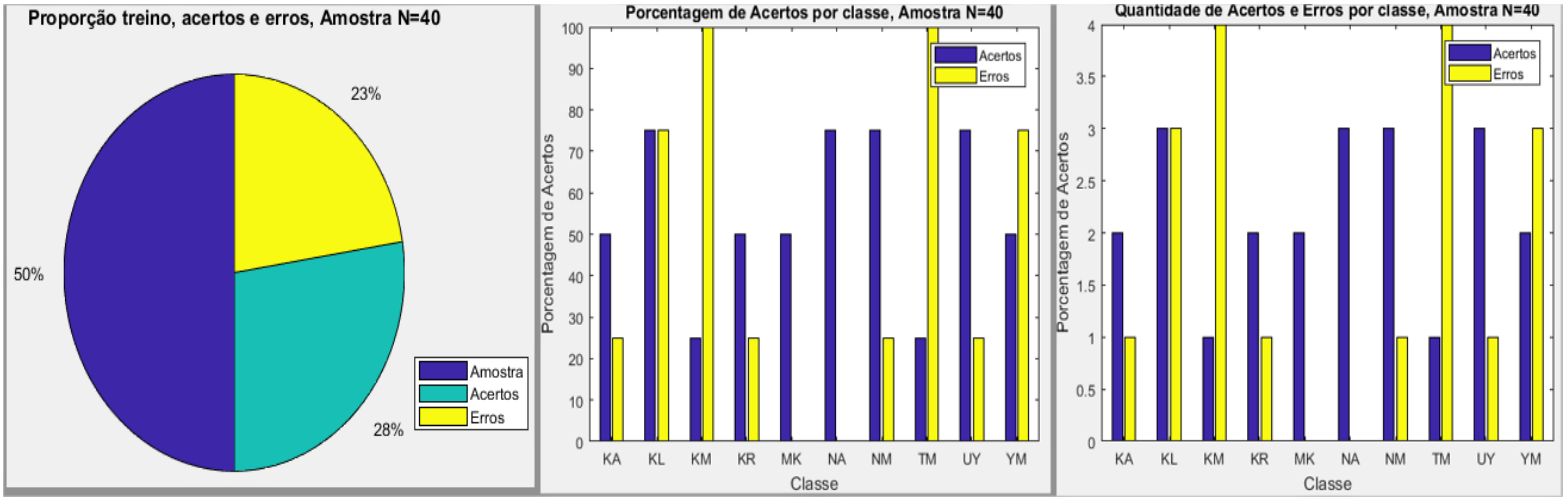
Fonte: Autoral

Figura 10: Resultados de acertos e erros, [treino, teste] = 60/40, N=60 (teste)



Fonte: Autoral

Figura 11: Resultados de acertos e erros, [treino, teste] = 70/30, N=40 (teste)



Fonte: Autoral

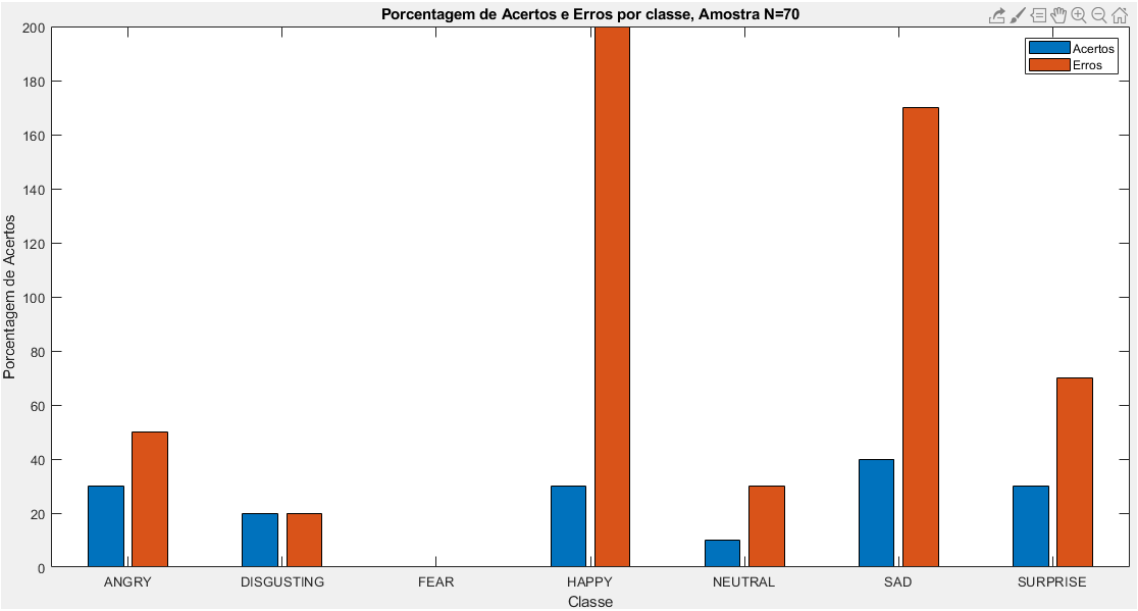
Analisando-se as Figuras 9, 10 e 11, nota-se que, em comparação com análise das emoções houve uma melhora considerável nos acertos. Essa molheira pode ser devido a mudança mais marcantes dos rostos de pessoa para pessoa. Da mesma forma, nesse caso, houve melhora na porcentagem de acertos com o aumento da porcentagem do conjunto de treino. A porcentagem de acertos começou em 25% para [treino, teste] = 50/50, N=70 (teste) e terminou em 28% para [treino, teste] = 70/30, N=40 (teste).

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 divido em treino e teste nas configurações 50%/50%, 60%/40% e 70%/30%. Os resultados obtidos foram os seguintes.

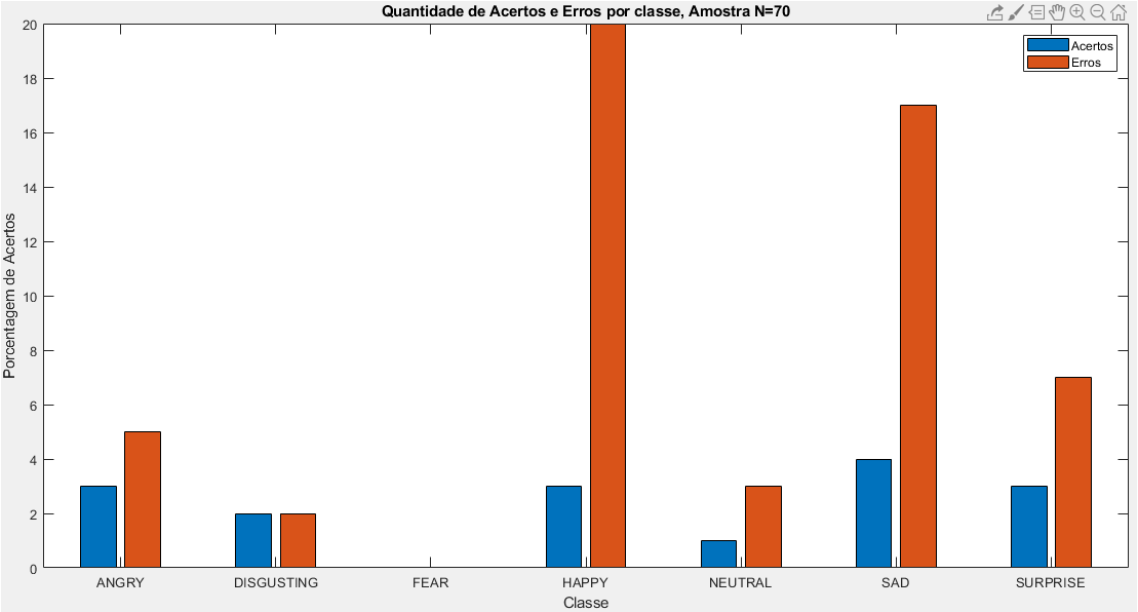
Configuração 50%/50% Amostra de 70 imagens para teste

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



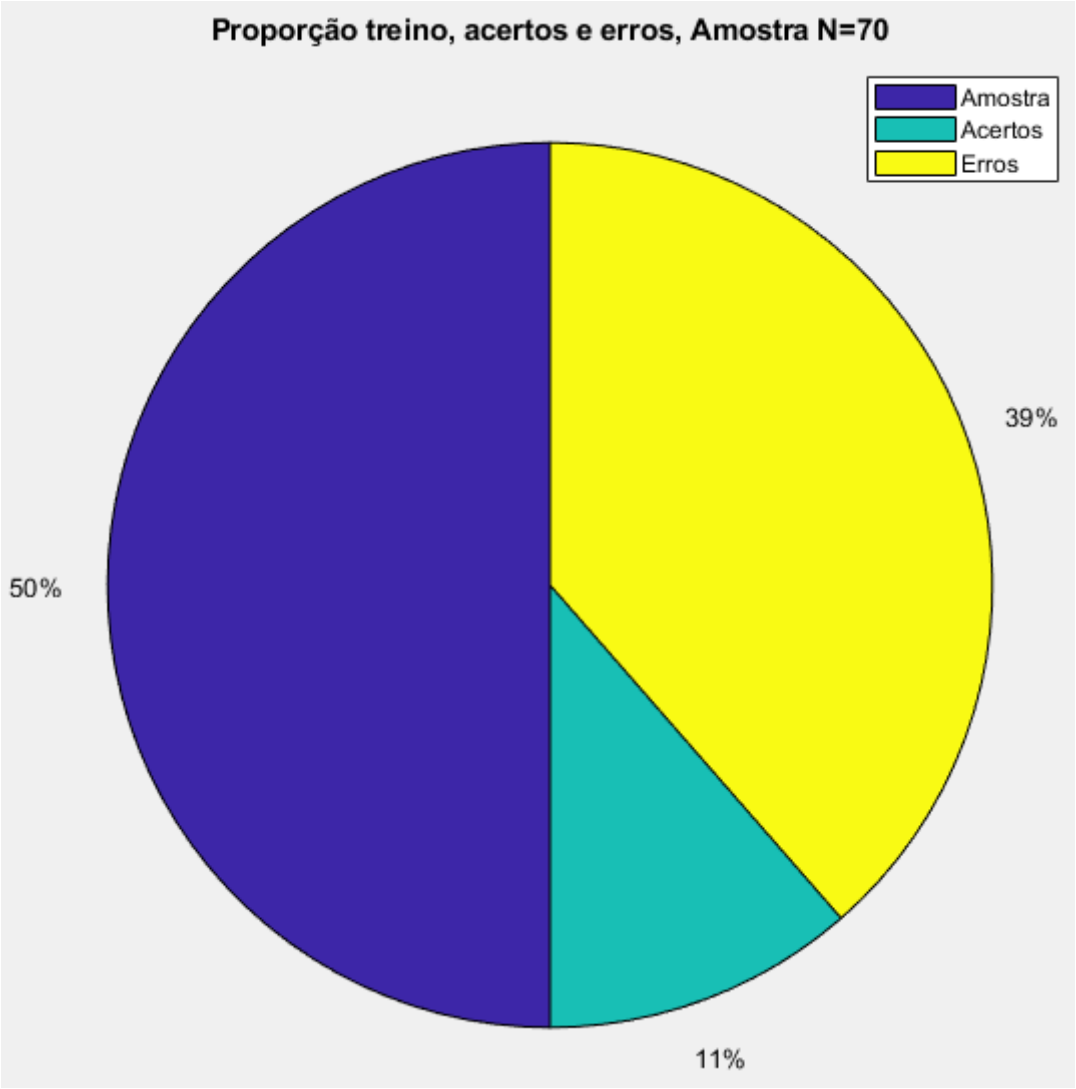
Fonte: Autoral

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



Fonte: Autoral

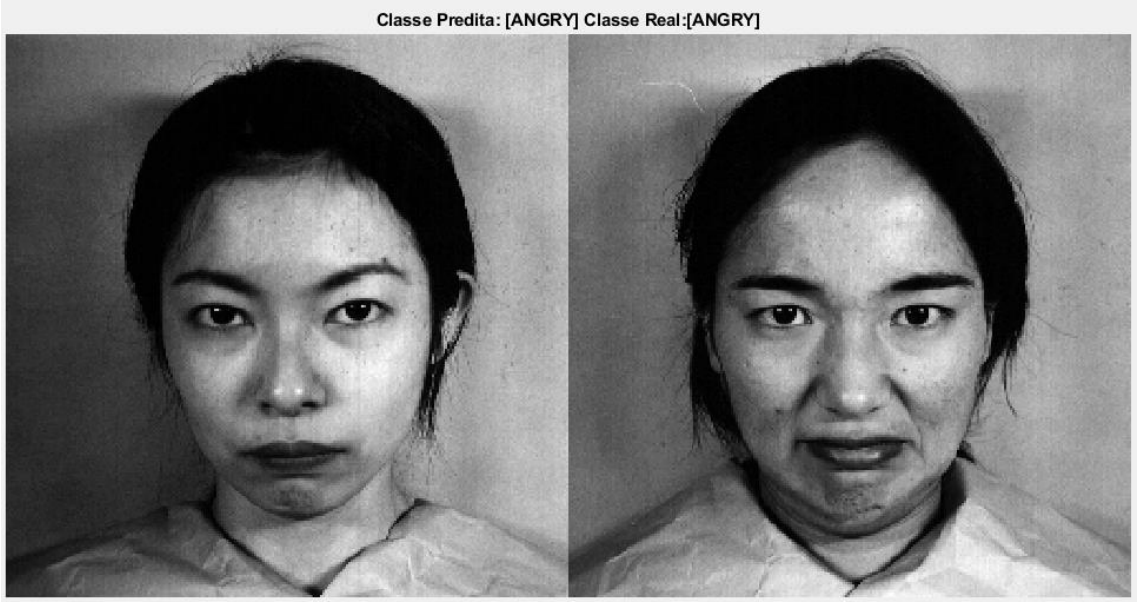
Figura N: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 70 para classificação de emoções utilizando a técnica HOG como pré-processamento



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 N: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

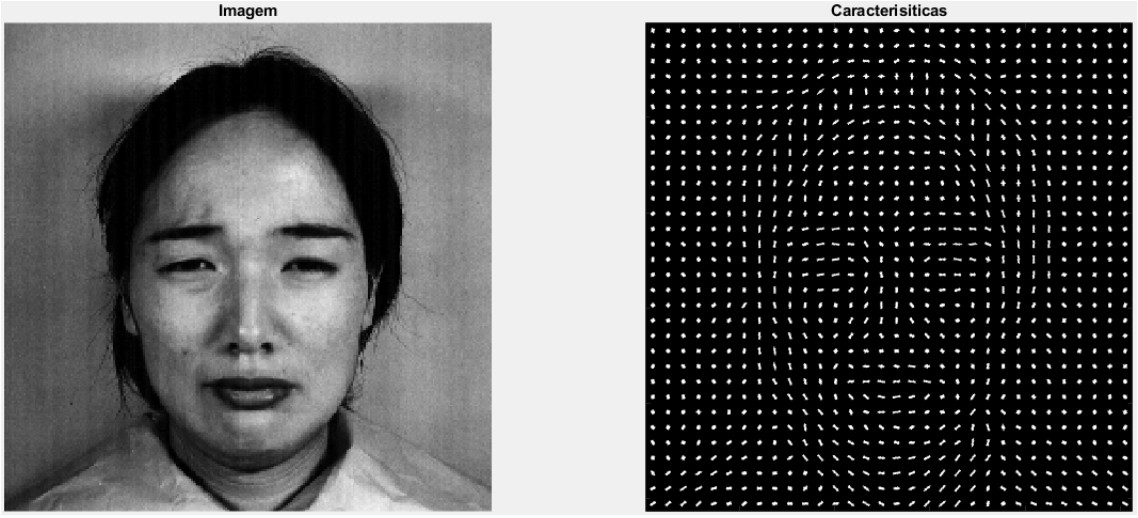
A Figura a seguir mostra um exemplo de classificação incorreta realizada pelo classificador. Em uma inspeção visual na imagem classificada incorretamente é possível verificar que as imagens são bastante similares. Os olhos e a boca estão dispostos de forma bastante semelhante. Para humanos classificarem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

Figura N: Exemplo de classificação incorreta realizada pelo algoritmo

Fonte: Autoral

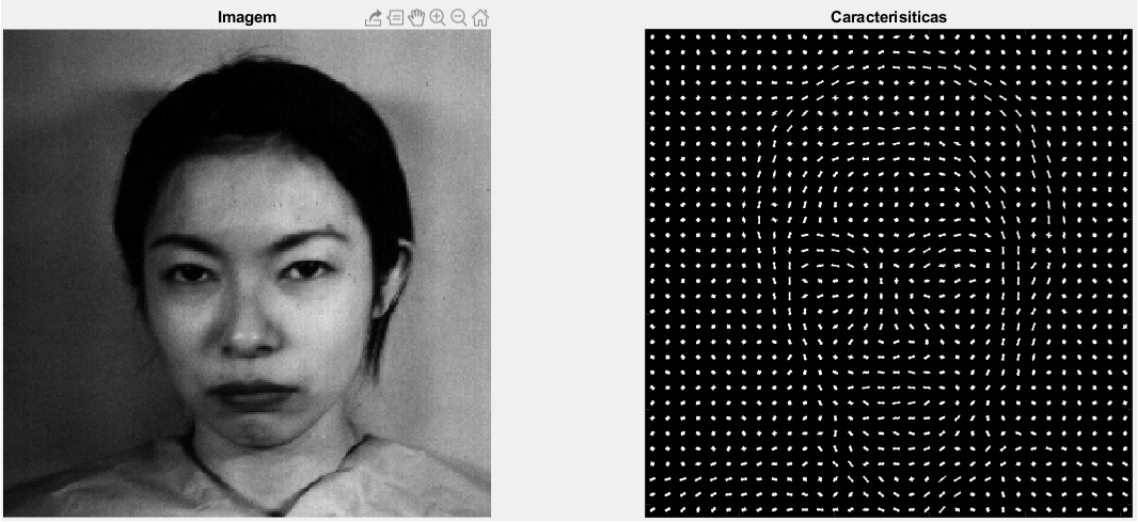
As Figuras a seguir ilustram as características extraídas das imagens classificadas incorretamente utilizando a técnica HOG como pré-processamento. Nas figuras à direita é o plot das características. Ao comparar ambos os plots é possível verificar que as características extraídas são muito similares,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

Figura N: Classe Real



Fonte: Autoral

Figura N: Classe Predita



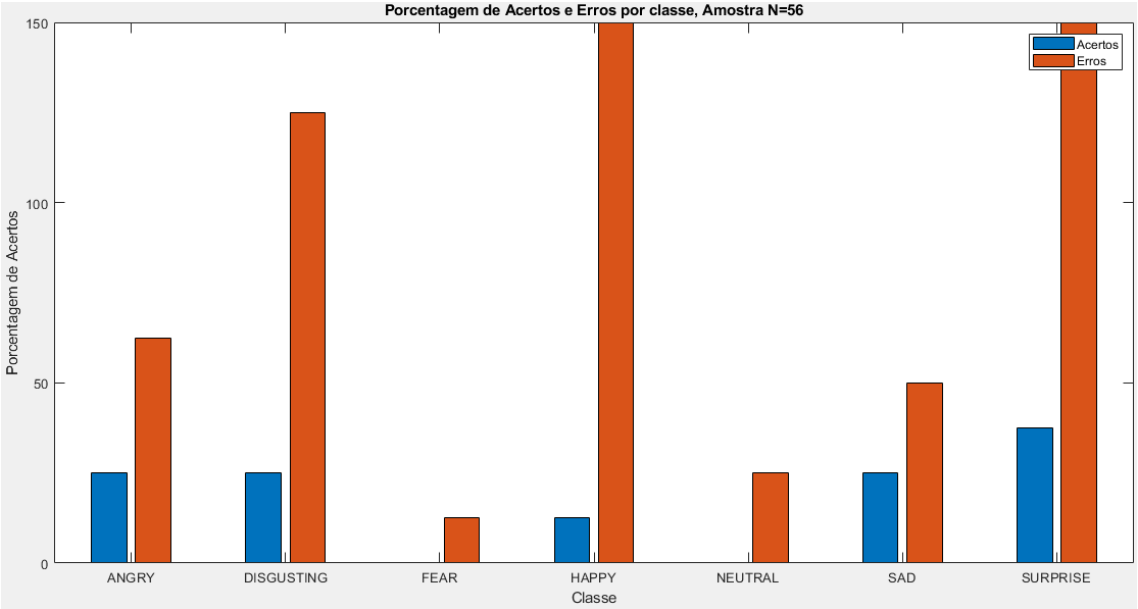
Fonte: Autoral

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

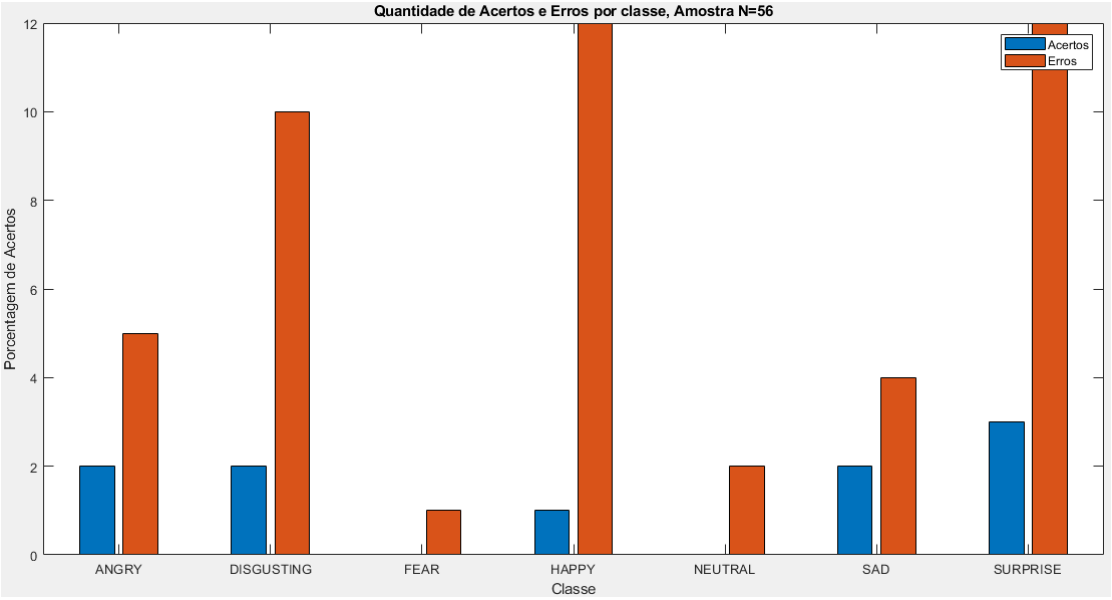
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

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



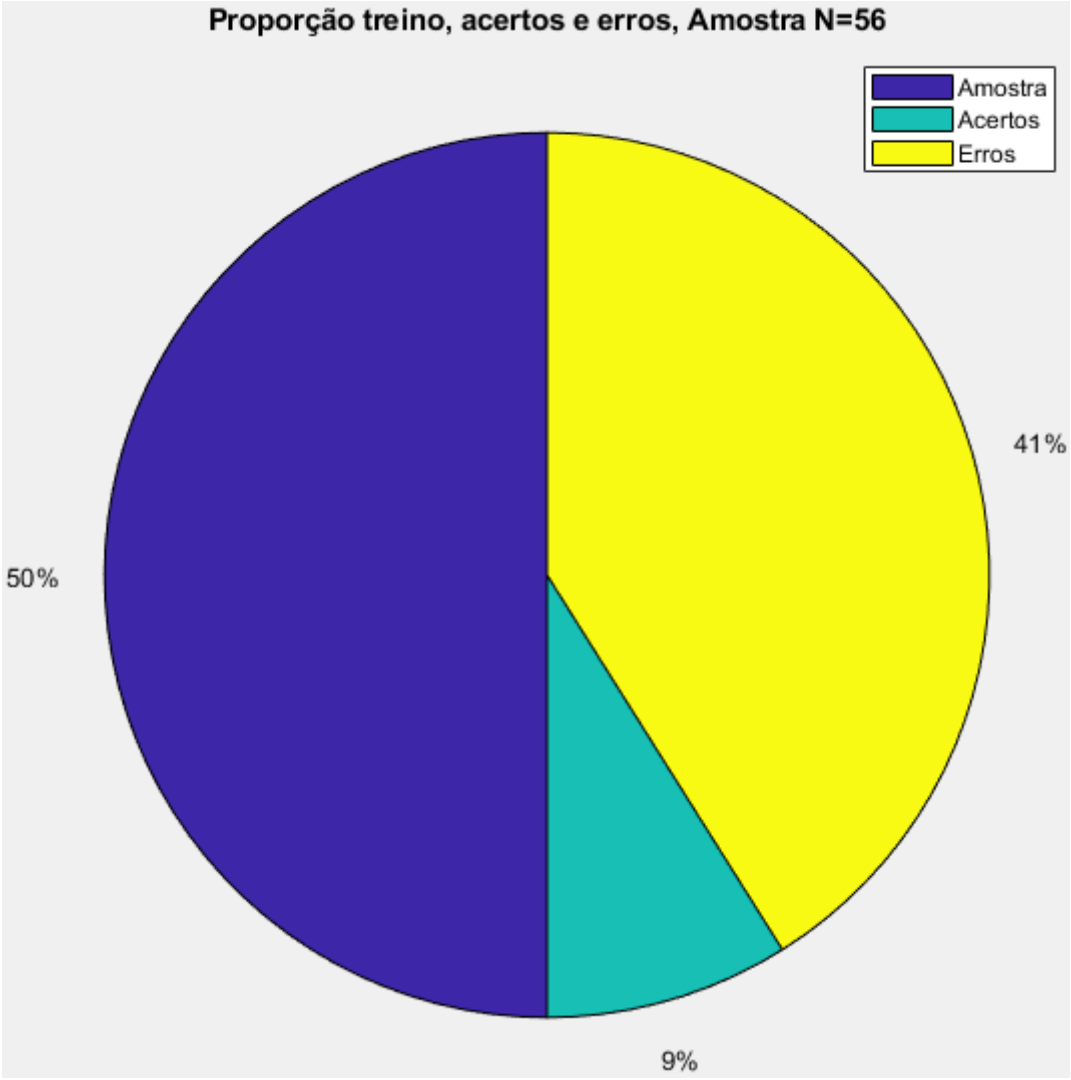
Fonte: Autoral

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



Fonte: Autoral

Figura N: 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 N: Exemplo de classificação correta realizada pelo algoritmo

Fonte: Autoral

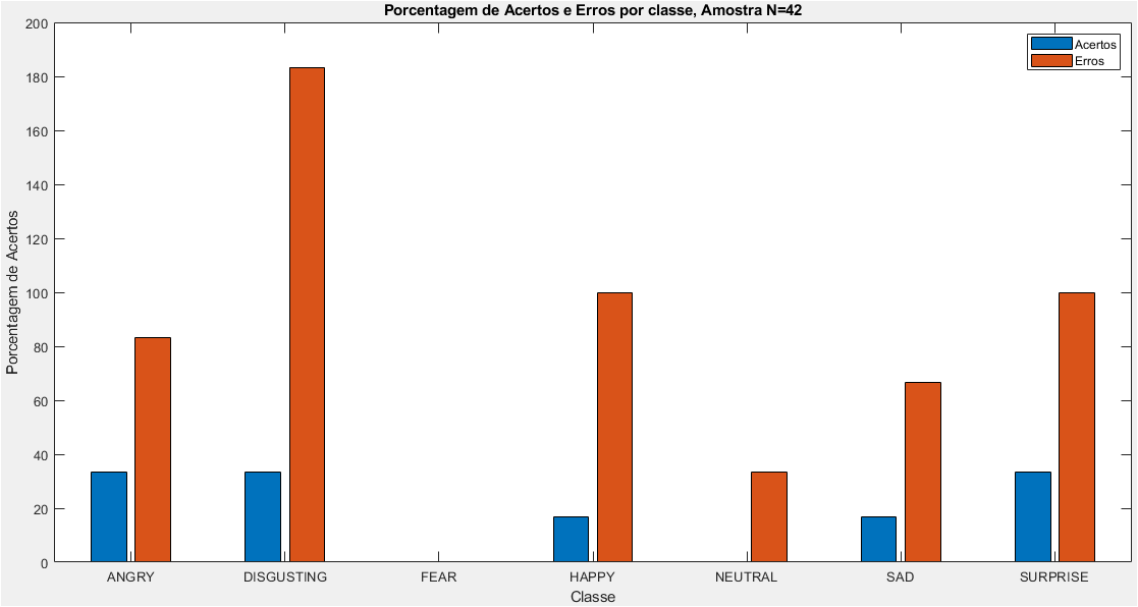
Figura N: Exemplo de classificação incorreta realizada pelo algoritmo

Fonte: Autoral

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

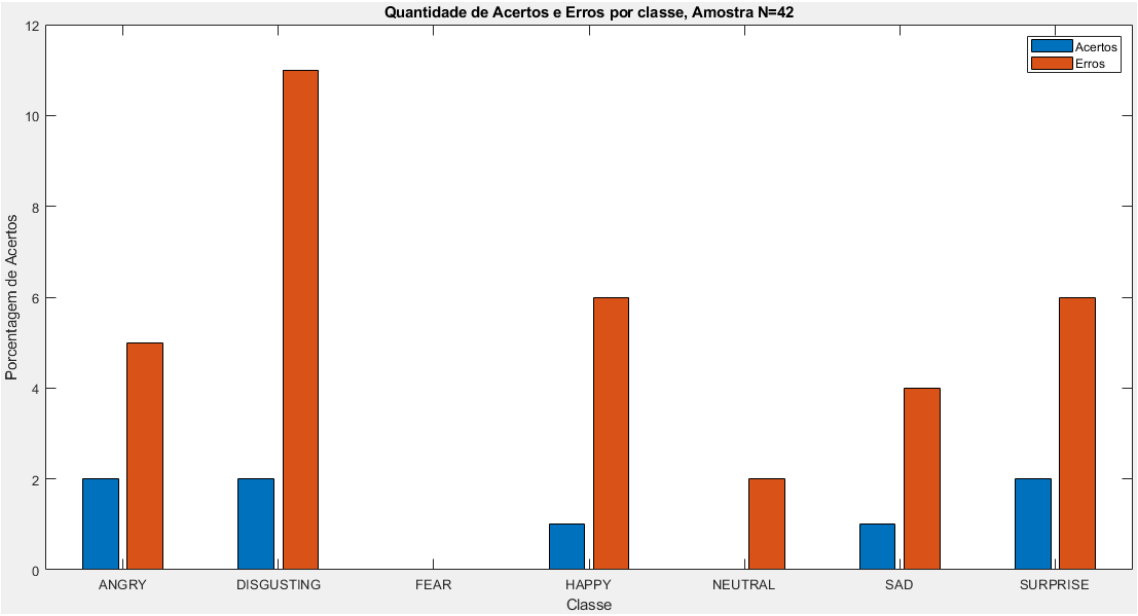
As figuras a seguir trazem os plots para as classes do banco de dados.

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



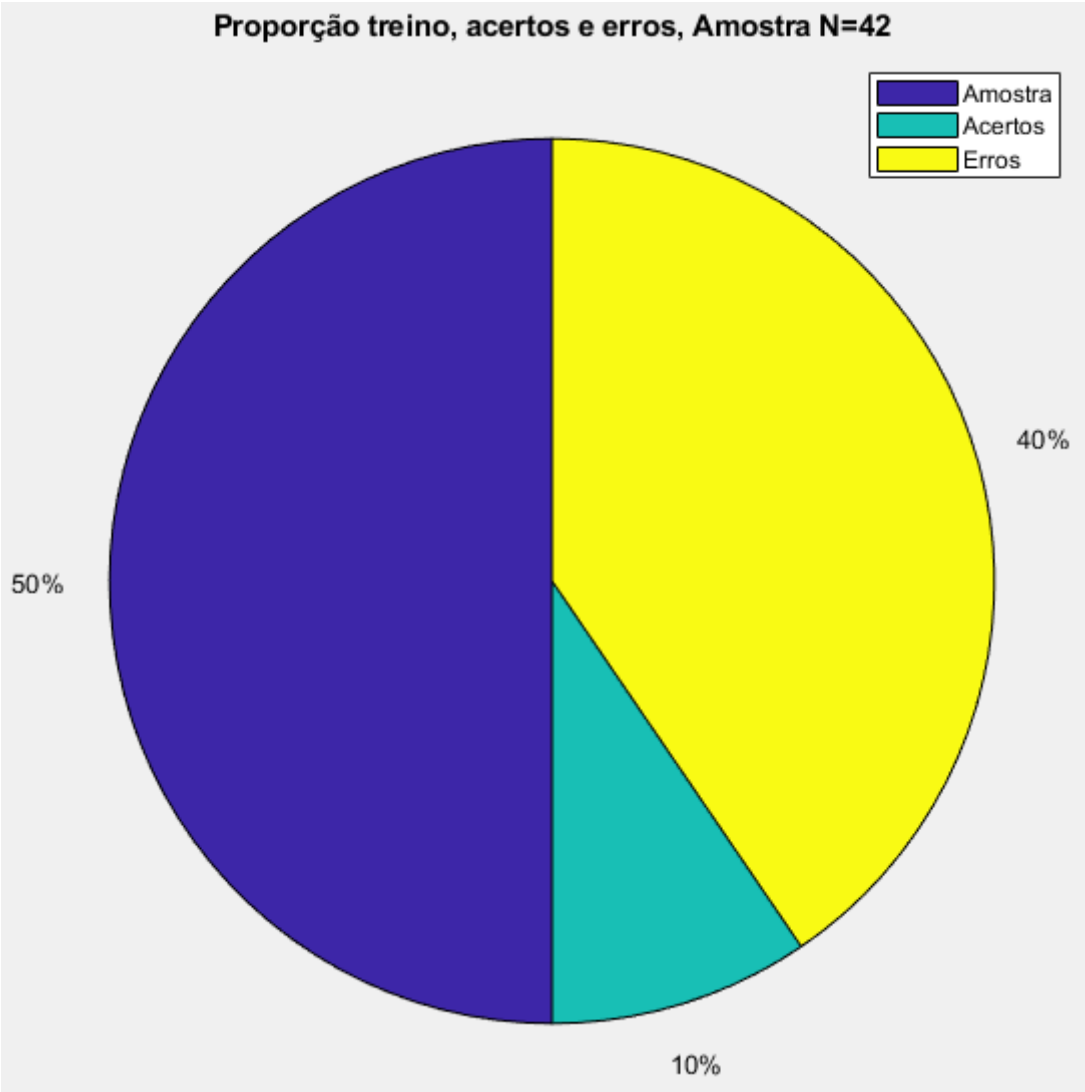
Fonte: Autoral

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



Fonte: Autoral

Figura N: 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 N: Exemplo de classificação correta realizada pelo algoritmo

Fonte: Autoral

Figura N: 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 N – Resumo das métricas de avaliação para as diferentes proporções de treino/teste

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

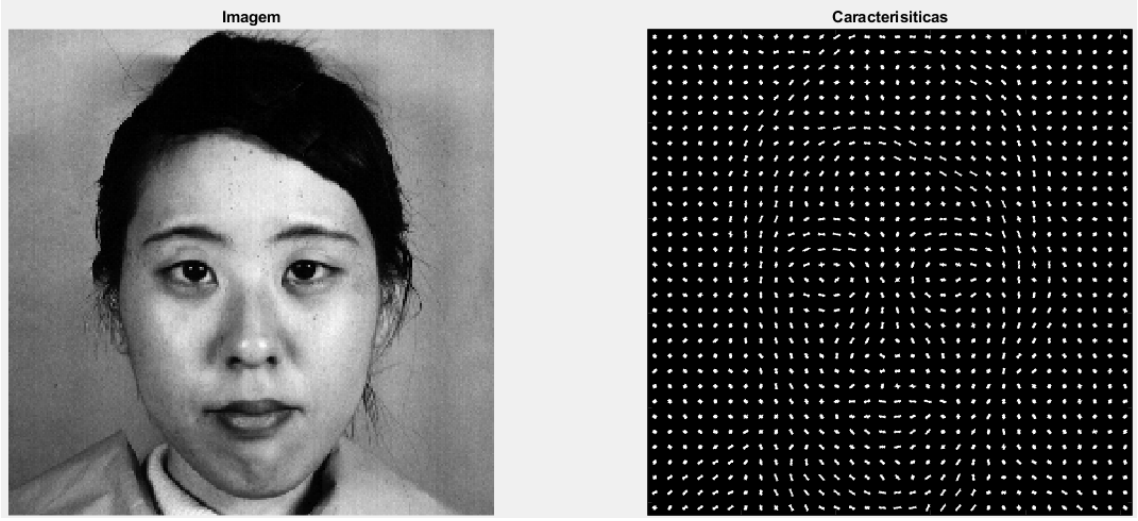
Fonte: Autoral

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.

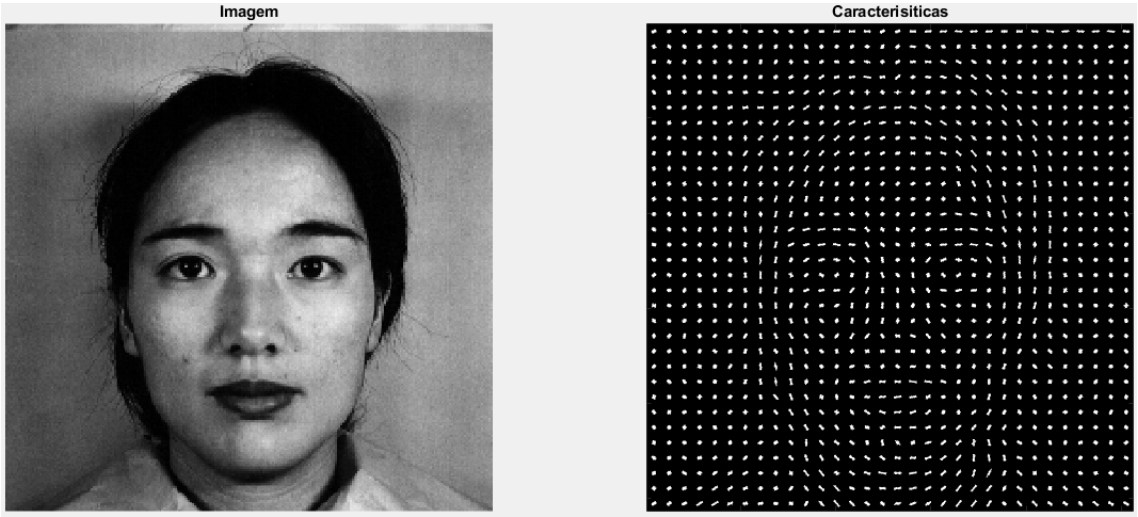
A Figura a seguir ilustra esse fato. A mulher da primeira figura pertence a classe ANGRY (Raiva) porém foi classificada como NEUTRAL (Neutro). Até visualmente é difícil afirmar que a mulher está expressando raiva. Além disso, o plot de seu HOG em comparação com o da mulher que ela foi classificada são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

Figura N: Exemplo de mulher classificada na classe incorreta



Fonte: Autoral

Figura N: 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 se 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

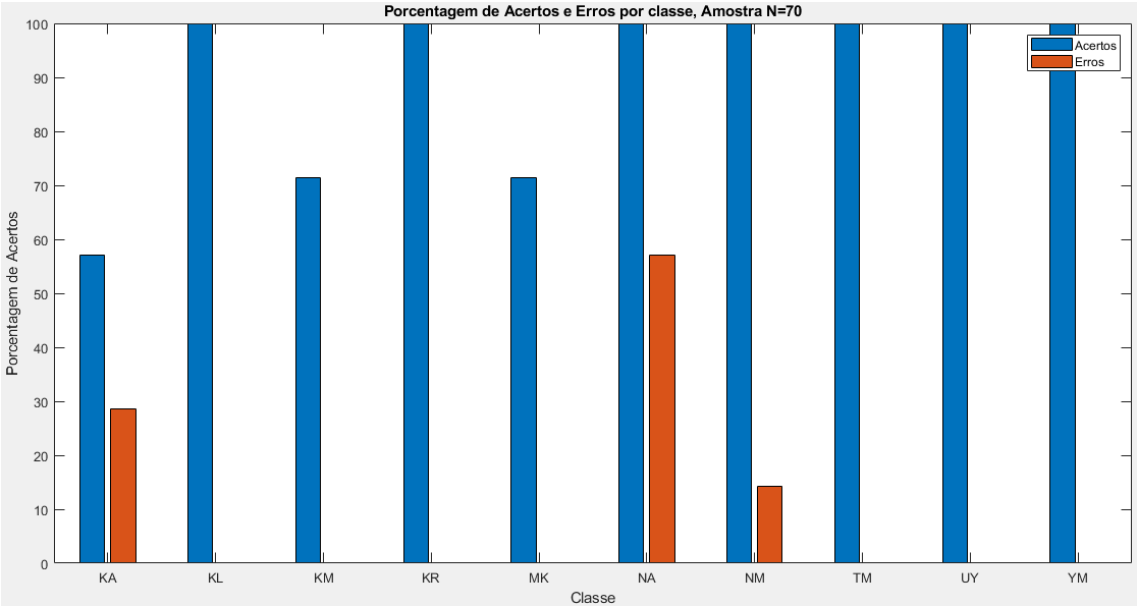
o resultado. Já que com isso, serão utilizadas mais imagens com pessoas diferentes para o treino e teste.

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 divido em treino e teste nas configurações 50%/50%, 60%/40% e 70%/30%. Os resultados obtidos foram os seguintes.

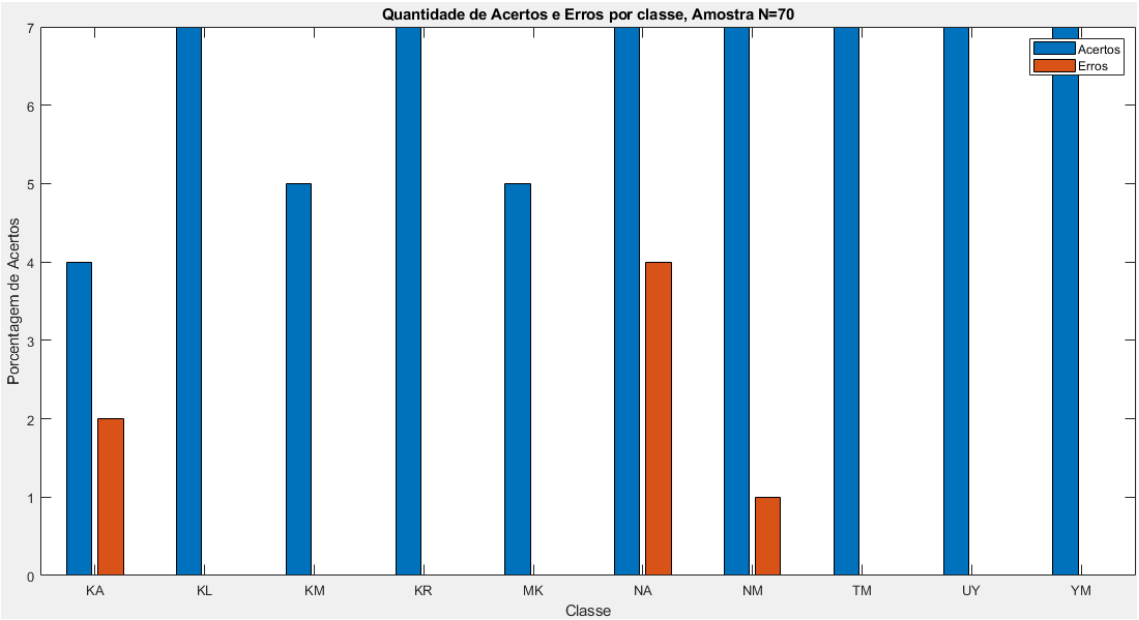
Configuração 50%/50% Amostra de 70 imagens para teste

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



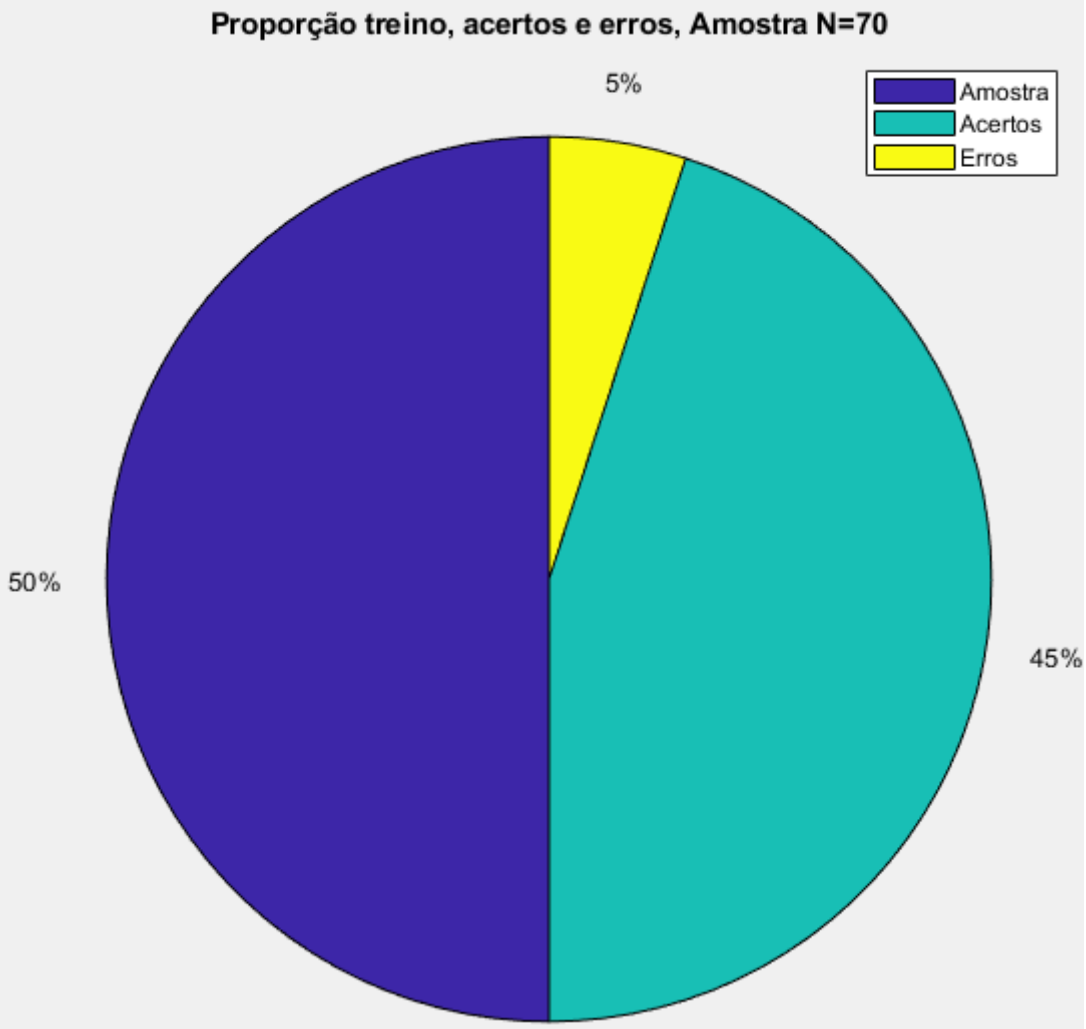
Fonte: Autoral

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



Fonte: Autoral

Figura N: 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



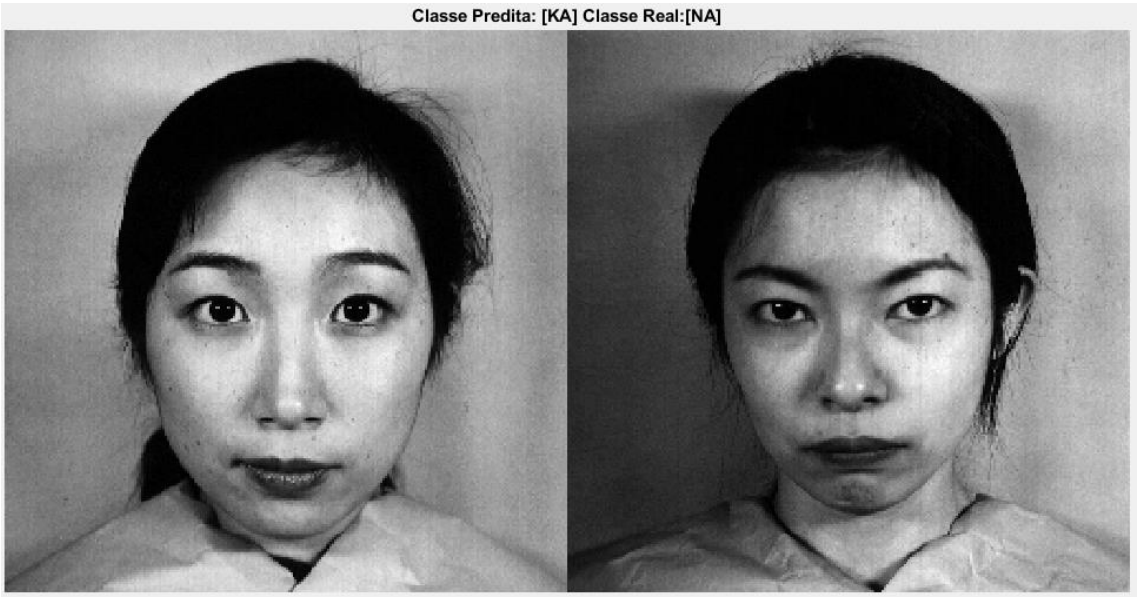
Fonte: Autoral

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 N: Exemplo de classificação correta realizada pelo algoritmo

Fonte: Autoral

Figura N: Exemplo de classificação incorreta realizada pelo algoritmo

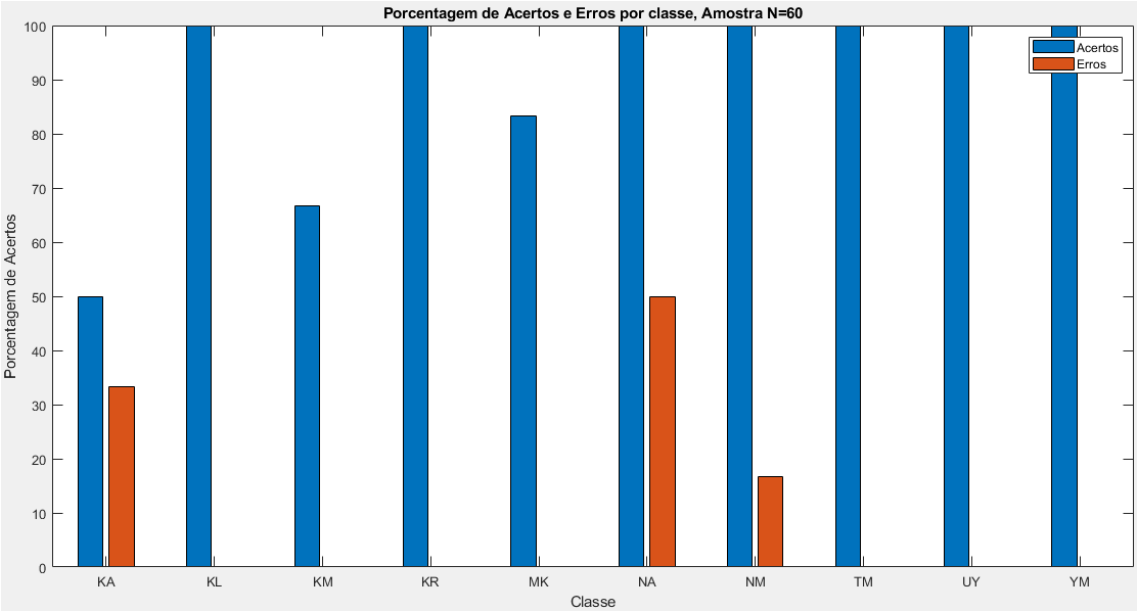


Fonte: Autoral

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

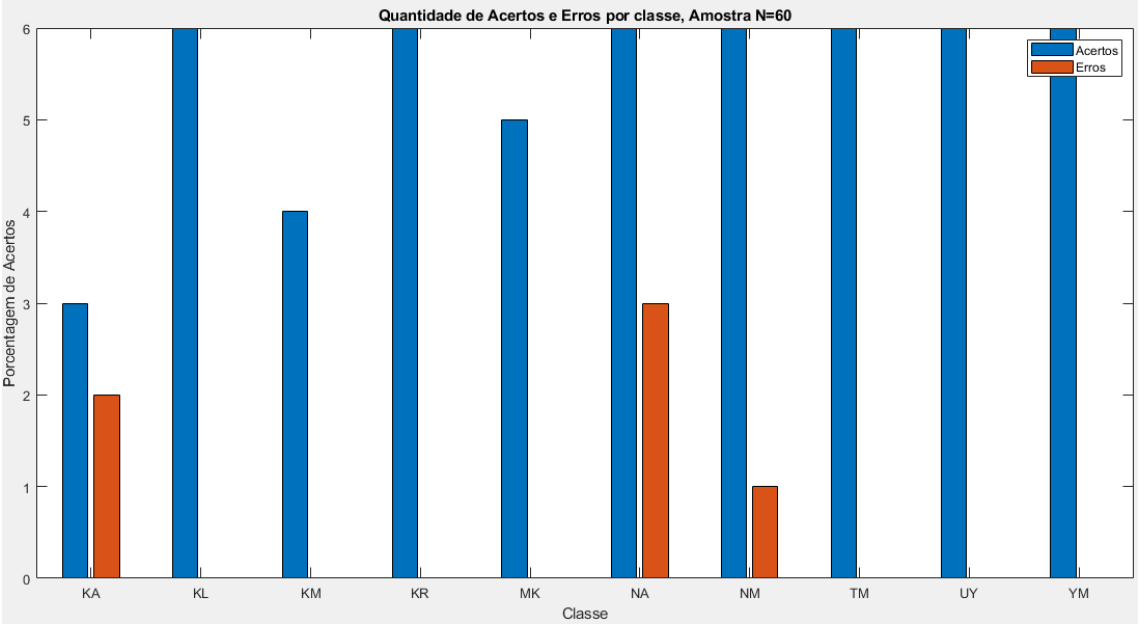
As figuras a seguir ilustram os plots com os resultados para a configuração de treino e teste.

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



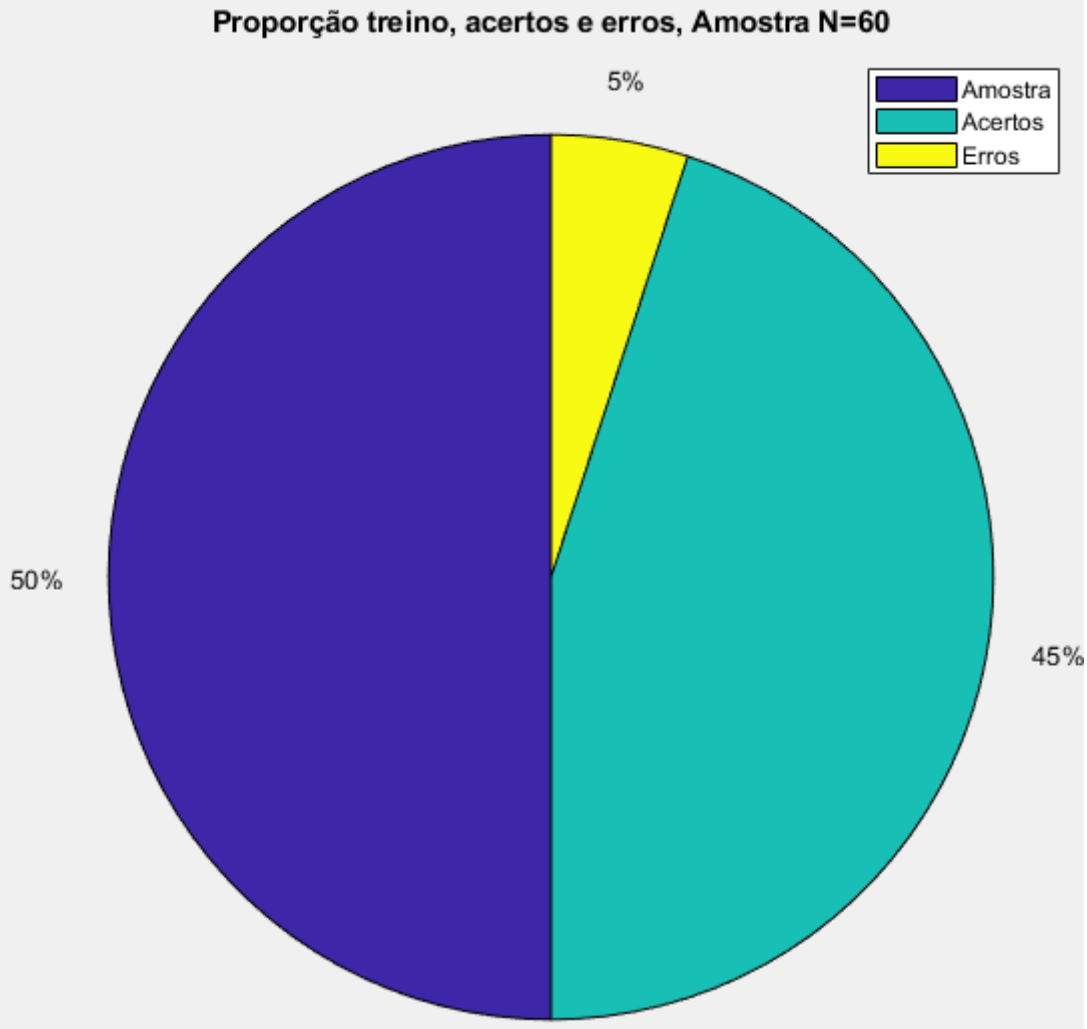
Fonte: Autoral

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



Fonte: Autoral

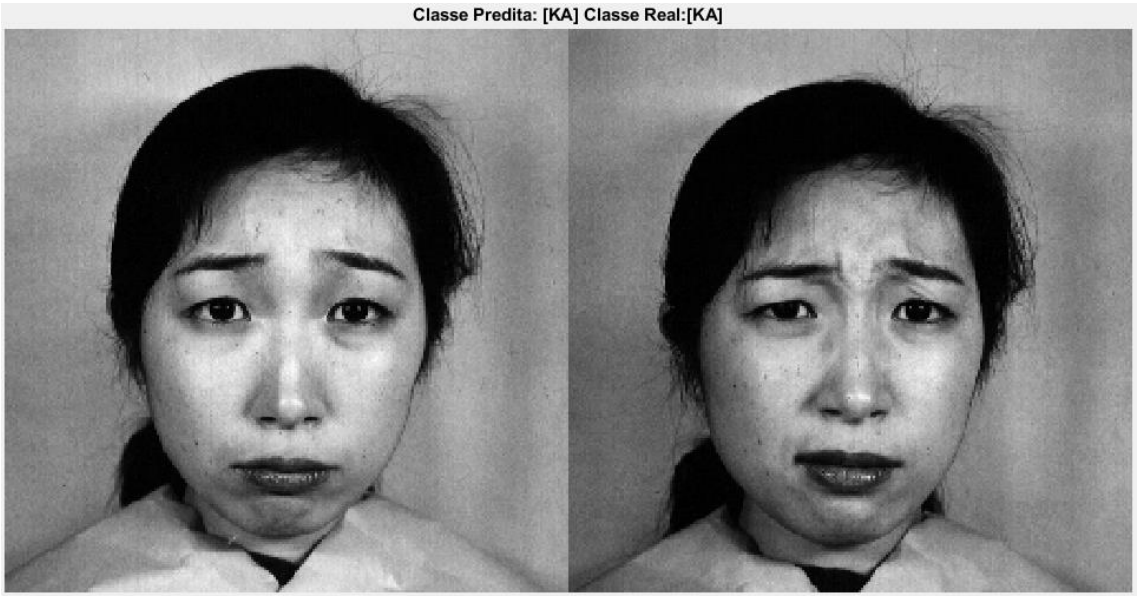
Figura N: Proporção do tamanho da amostra da quantidade de Acertos e Erros por classe, Amostra N= 60 para classificação de pessoas utilizando a técnica HOG como pré-processamento



Fonte: Autoral

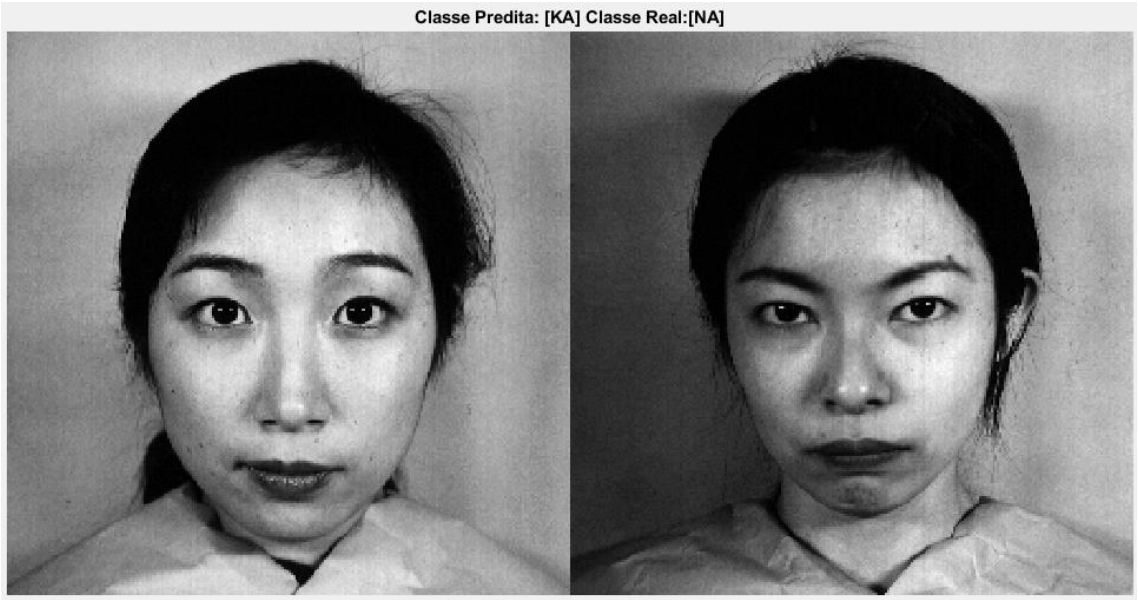
A média de acertos foi de 90% e a média de erros foi de 10%. Para essa configuração a PCA conseguiu classificar bem os dados. De 60 amostras para teste, o algoritmo classificou corretamente 54. As Figuras a seguir ilustram uma classificação correta e uma classificação incorreta realizada pelo algoritmo.

Figura N: Exemplo de classificação correta realizada pelo algoritmo



Fonte: Autoral

Figura N: Exemplo de classificação incorreta realizada pelo algoritmo

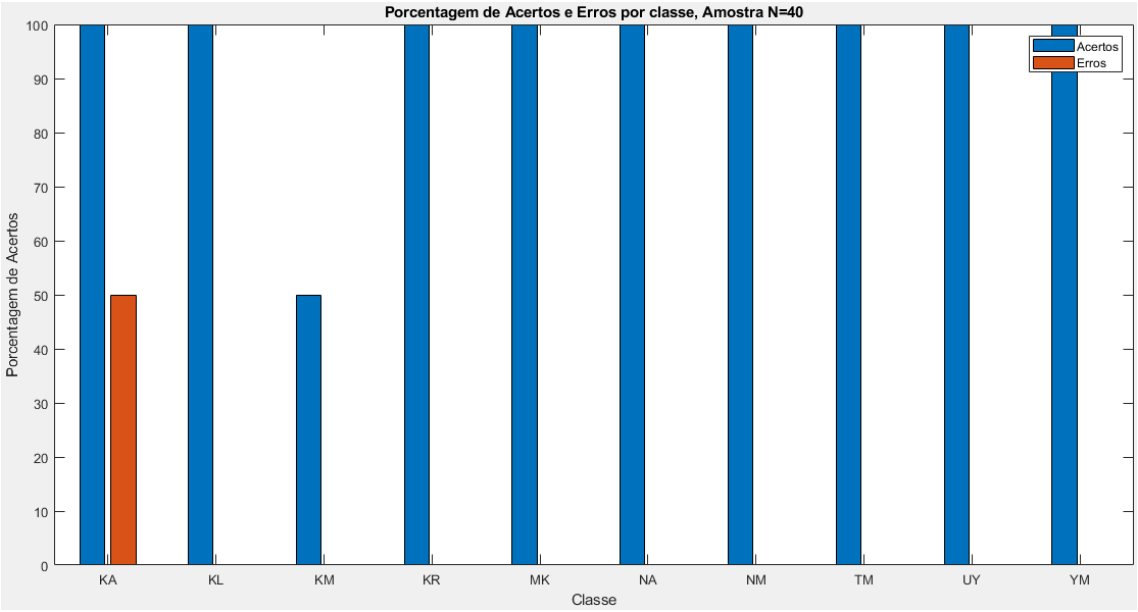


Fonte: Autoral

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

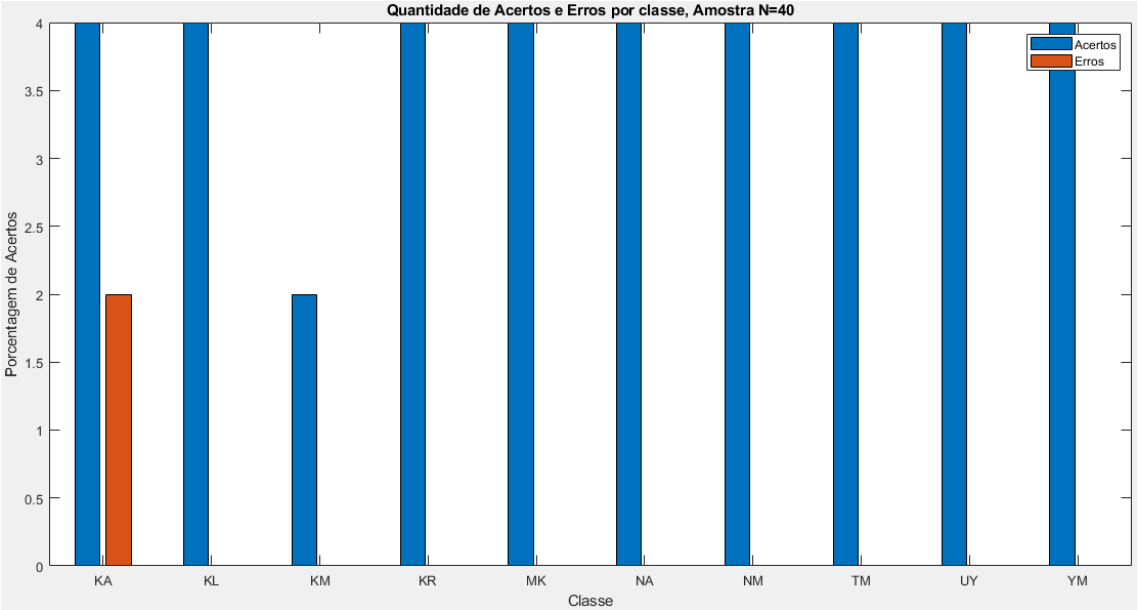
As figuras a seguir trazem os plots para as classes do banco de dados.

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



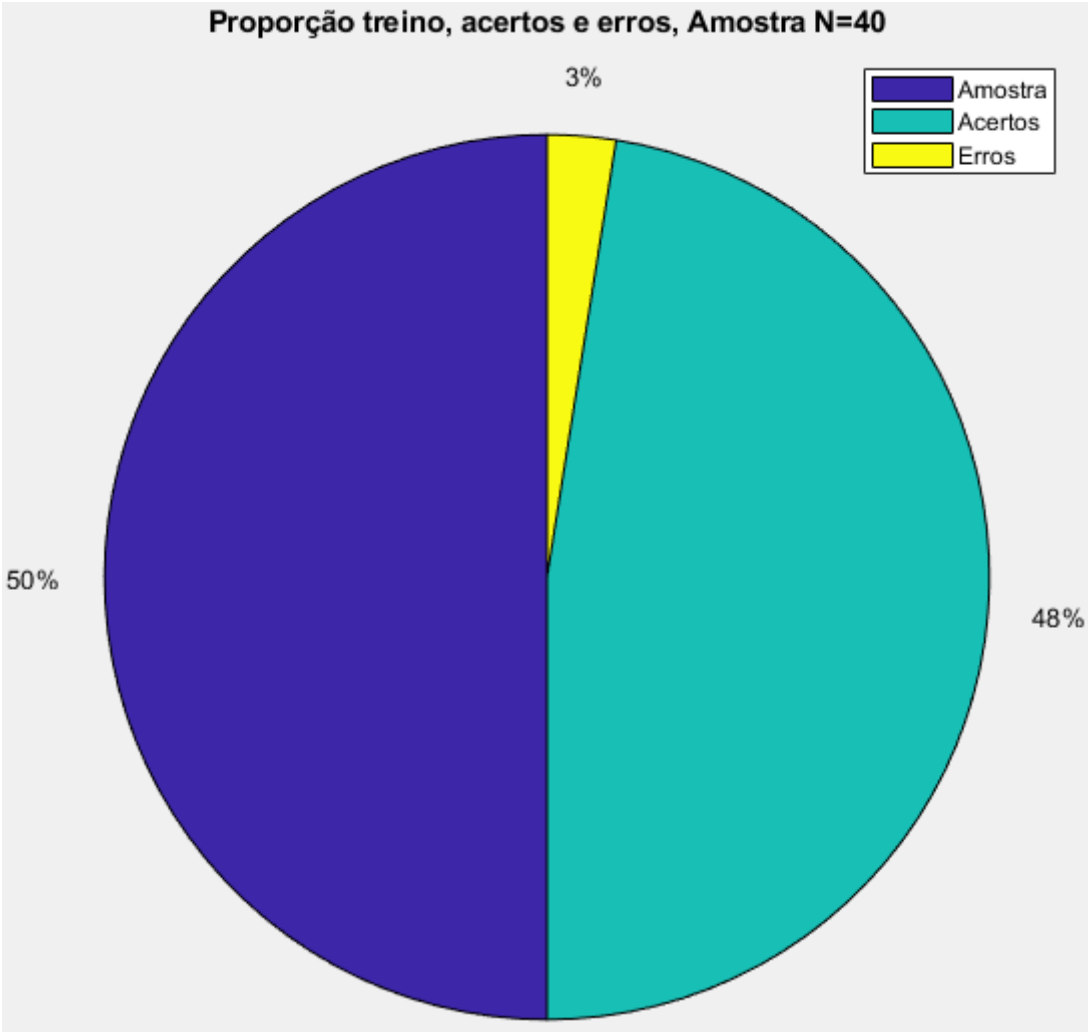
Fonte: Autoral

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



Fonte: Autoral

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



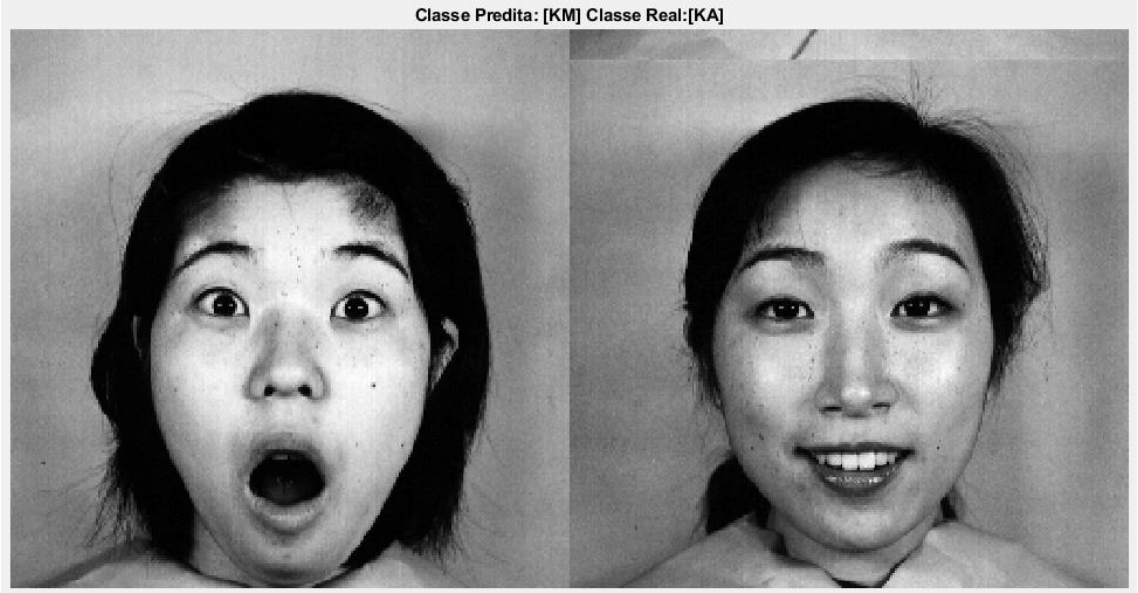
Fonte: Autoral

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 N: Exemplo de classificação correta realizada pelo algoritmo

Fonte: Autoral

Figura N: 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 N – Resumo das métricas de avaliação para as diferentes proporções de treino/teste

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

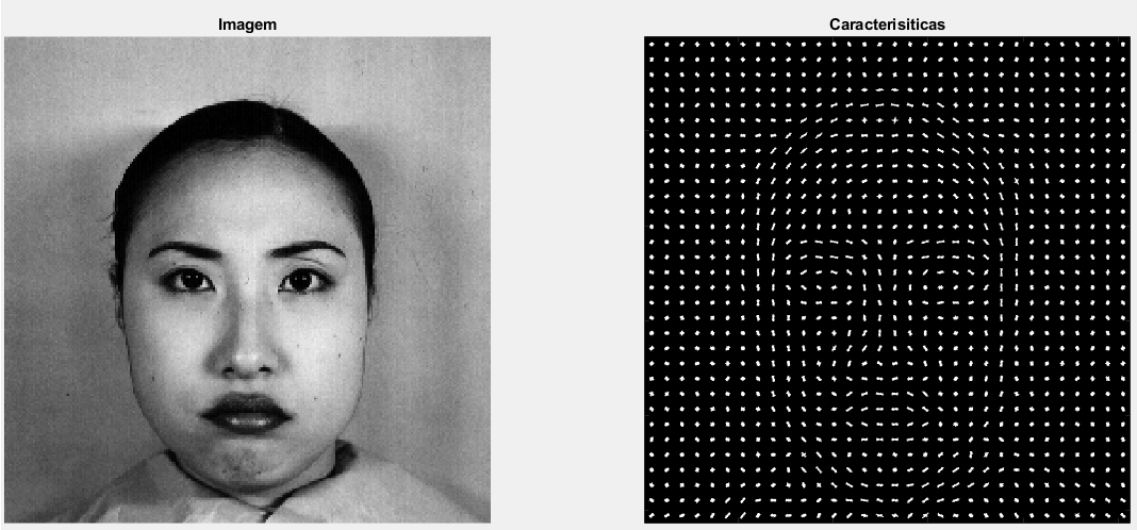
Fonte: Autoral

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.

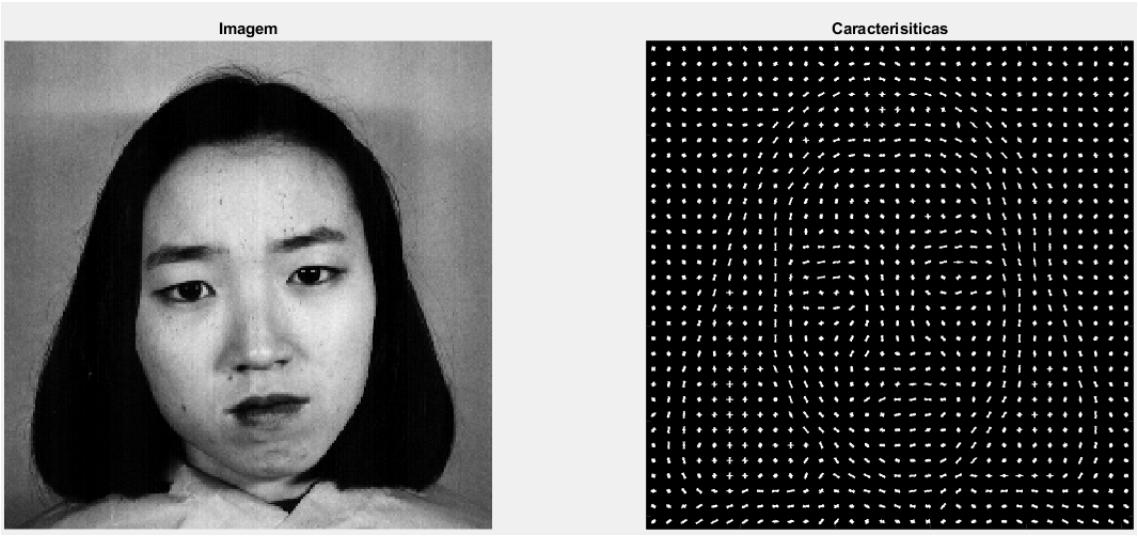
A Figura a seguir ilustra esse fato. A primeira imagem mostra uma mulher com o cabelo curto, o plot das características extraídas mostra que a região mais próxima da cabeça apresenta os pontos verticais, enquanto na outra imagem os pontos mais próximos da cabeça estão levemente inclinados devido a presença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

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



Fonte: Autoral

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



Fonte: Autoral

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 Digtal de Imagem.** São Paulo: Sv, 2010.

Lyons, Michael; Kamachi, Miyuki; Gyoba, Jiro. **The Japanese Female Facial Expression (JAFFE) 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.