UNIVERSIDADE FEDERAL DO PARANÁ

SETOR DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

GISELE TAM

GRR20102167

PROPOSTA DE TRABALHO: RECONHECIMENTO DE PLACA DE VEÍCULOS

Trabalho realizado para a disciplina de Processamento Digital de Sinais 2 para obtenção de nota parcial. Prof° Dr. Eduardo Parente Ribeiro.

CURITIBA

2015

1) INTRODUÇÃO

A placa do veículo é uma forma de identificá-lo corretamente, sem depender da cor ou formato externos. Geralmente são colocados na parte frontal e traseira de um veículo e são compostos por uma combinação de números e letras.

Com o aumento do número de câmeras nas ruas, um sistema de identificação automática pode ser conveniente para buscar um veículo específico sem o auxílio da presença humana. Isso é vantajoso tanto para questões de segurança, tais como furto, quanto para questões administrativas, como identificação de um veículo que ultrapassa o limite de velocidade num determinado ponto.

2) OBJETIVO

O objetivo deste trabalho é criar um sistema de reconhecimento automático de placa de veículos.

3) DESENVOLVIMENTO

O código foi baseado no diagrama de blocos ilustrado na Figura 1 e sua transcrição encontra-se no Apêndice A no final do documento.

FIGURA 1 - DIAGRAMA DE BLOCOS DO CÓDIGO

3.1) PRÉ-PROCESSAMENTO

O pré-processamento consiste em ler a figura desejada e deixá-la em escala cinza para a sua manipulação. As Figuras 2 (a) e (b) mostram a imagem original e em escala cinza, respectivamente.

FIGURA 2 - IMAGEM ORIGINAL (A) E EM ESCALA CINZA (B)

(a) (b)

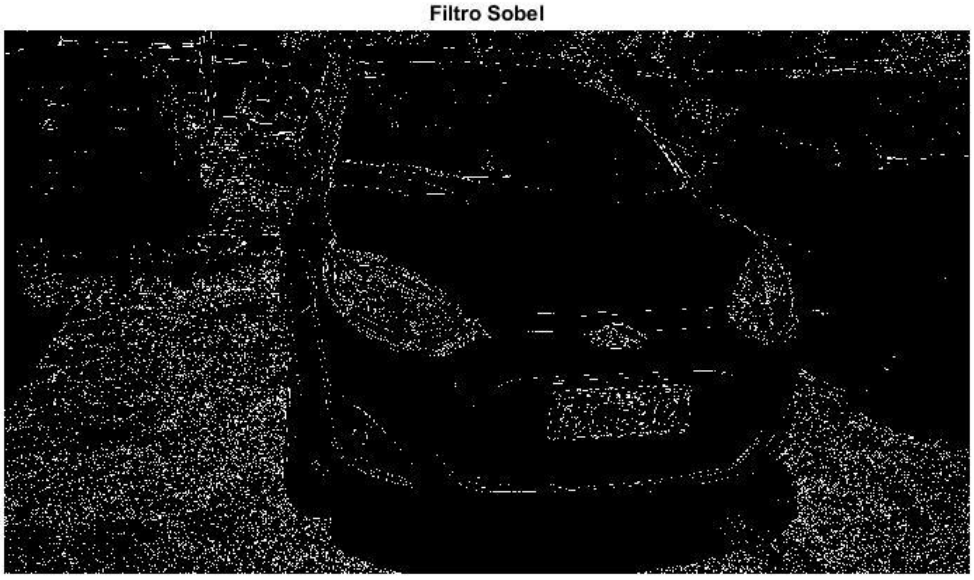
3.2) RECONHECIMENTO DE BORDAS

Esta etapa consiste na aplicação de seis passos para a separação de possíveis regiões em que a placa estará localizada.

i. Aplicação do Filtro Sobel

O filtro Sobel é um algorítmo para a detecção dos contornos horizontais e verticais de uma imagem. A Figura 3 mostra a imagem após aplicação do filtro Sobel.

FIGURA 3 - APLICAÇÃO DO FILTRO SOBEL



ii. Dilatação das Bordas

Após a aplicação do filtro Sobel foi feita a dilatação das linhas de contorno para que objetos fechados se sobressaíssem na imagem. A Figura 4 ilustra os contornos dilatados.

FIGURA 4 - CONTORNOS DILATADOS

iii. Erosão da Imagem

Este passo consiste em corroer os contornos da imagem de tal forma que seja possível separar as regiões de interesse. A Figura 5 demonstra o resultado.

FIGURA 5 - EROSÃO DOS CONTORNOS

iv. Limpeza dos cantos

Este passo consiste em remover as regões que não estão fechadas na imagem, ou seja, as regiões em branco que esão nas margens. A Figura 6 exemplica o resulatdo do processo.

FIGURA 6 - REMOÇÃO DOS CANTOS

v. Preenchimento de vazios

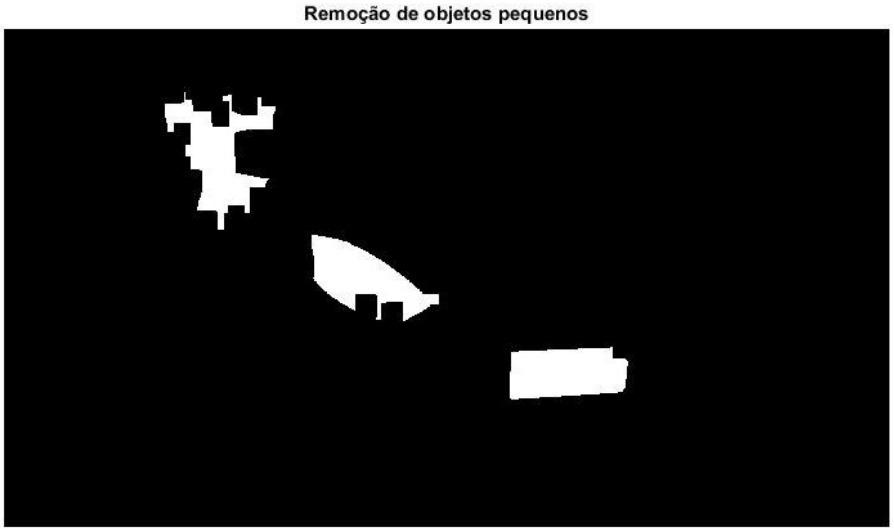
Após a remoção dos cantos da imagem, as áreas que continham vazios em seu interior foram preenchidas. A Figura 7 exibe o resultado.

FIGURA 7 - PREENCHIMENTO DE VAZIOS

vi. Remoção de objetos pequenos

Por fim, este passo remove as regiões menores que um valor determinado. A Figura 8 retrata o resultado.

FIGURA 8 - REMOÇÃO DE OBJETOS PEQUENOS



3.3) LOCALIZAÇÃO DA PLACA

Para a localização da placa foi utilizada a função nativa do Matlab *regionprops()* com parâmetro de saída '*table'* e parâmetro de propriedade *'BoundingBox'*. Esta função retorna um conjunto de valores para todos as regiões da imagem, que neste caso serão a posição do canto superior esquerdo e a altura e largura de cada região. Caso haja mais de uma região, o código escolhe as regiões em que a largura é maior que a altura e, por fim, o último critério é a escolha da região mais abaixo na imagem. A figura 9 ilustra a região escolhida para a placa após o processo e a Figura 10 representa a mesma região em preto e branco (binária).

FIGURA 9 - REGIÃO DA PLACA

FIGURA 10 - PLACA EM BINÁRIO

3.4) SEGMENTAÇÃO DOS CARACTERES

Para esta etapa, foi utilizada novamente a função *regionprops()* com parâmetro de propriedade *'BoundingBox'*. Contudo, desta vez o critério de seleção das regiões de interesse foi a relação entre a largura e a altura das regiões, assim como as regiões com maior altura da placa. Após a escolha das regiões, elas tiveram o tamanho alterado para uma matriz de 150 linhas por 150 colunas. A Figura 11 mostra o resultado obtido.

FIGURA 11 - CARACTERES DA PLACA

3.5) RECONHECIMENTO DOS CARACTERES

Esta última etapa do código é dividida em 3 passos: criação de um banco de amostras, trainamento dos aspectos HOG (*Histogram of Oriented Gradient)* e predição dos caracteres. Para os últimos dois passos foi utilizado o Computer Vision Toolbox do Matlab.

i. Criação do banco de amostras

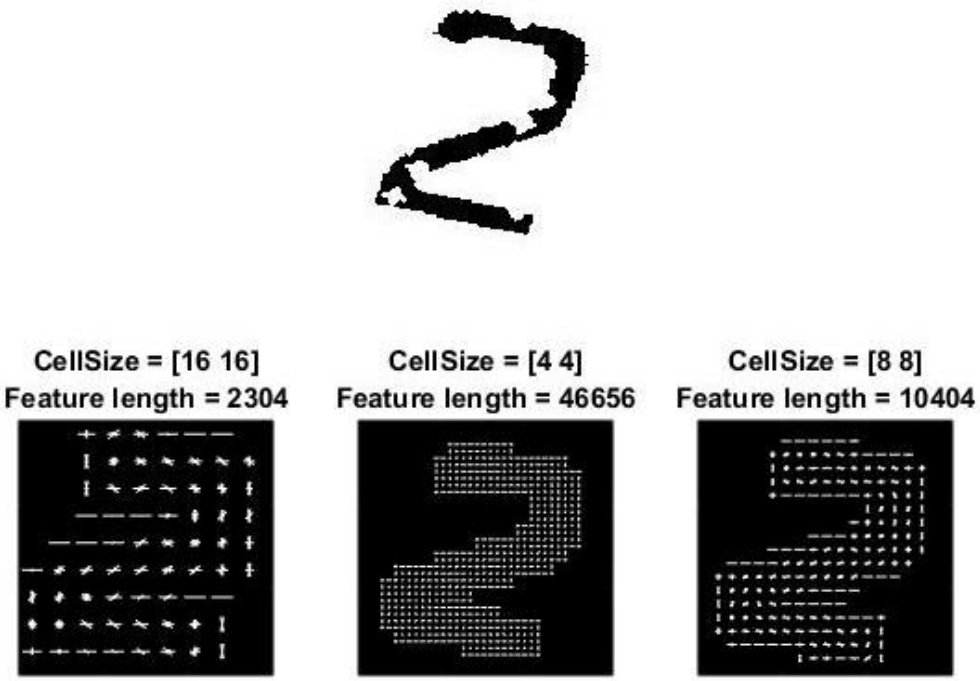
Foi necessário criar um banco de amostras para o treinamento dos aspectos HOG (*Histogram of Oriented* Gradients) de cada caractere. O código completo encontra-se no Apêndice B. Foram utilizadas imagens dos caracteres na fonte Mandatory e cada caractere teve um ruído gaussiano acrescentado e/ou foi rotacionado. A Figura 12 mostra o resultado após a aplicação do código (a primeira imagem do quadro é a original).

FIGURA 12 - CARACTERES PARA O BANCO DE AMOSTRAS

ii. Treinamento dos aspectos HOG

Os aspectos HOG (Histograma dos Gradientes Orientados, em português) é uma técnica da visão computacional para detecção de objetos. Ela é similar aos métodos de detecção de bordas, mas difere ao descrever o formato do objeto utilizando uma distribuição de direções de bordas (gradiente orientado). A imagem é dividida em várias células e em cada célula é calculado um gradiente de direção da intensidade. A Figura13 ilustra os gradientes com diferentes tamanhos de células.

FIGURA 13 - ASPECTOS HOG DO CARACTERE



Após a escolha do tamanho da célula, são extraídos os aspectos HOG de todos os caracteres do banco de amostras para então treinar a classificação dos números e letras. A transcrição completa do código é encontrada no Apêndice C.

iii. Predição dos caracteres

Por fim, após o treinamento dos aspectos é feita a predição dos caracteres. São extraídos os aspectos HOG dos caracteres obtidos na imagem original e eles são comparados com a classificação treinada com o auxílio da função nativa do Matlab *predict()*. A Figura 14 mostra o resultado final.

FIGURA 14 - IDENTIFICAÇÃO FINAL

Identificação da placa: AXL8912
>>

4) RESULTADOS E CONCLUSÃO

A tabela 1 mostra os resultados obtidos para 7 imagens testadas. Os caracteres em verde foram identificados corretamente e os em vermelho não.

TABELA 1 - RESULTADOS

Imagem Original	Placa	Identificação
		AXL8912
		AXL8912
		AXL8912
		AXL9912
		ARX9391
		IVS6368
		BLP7429

O programa funcionou corretamente em três das sete imagens testadas. Em outras três houve a identificação incorreta de um caractere. E na última houve dois erros de identificação. Uma possível solução para melhorar o resultado seria aumentar o banco de amostras e treinar melhor a classificação dos caracteres.

REFERÊNCIAS

How LPR works. Disponível em: http://www.licenseplatesrecognition.com/how-lpr-works.html [Acesso em 21 Outubro 2015].

Massoud, M., Sabee, M., Gergias, M. & Bakhit, R., 2013. **Automated new license plate recognition in Egyp**t. *Alexandria Engineering Journal.*

Wanniarachchi, W. K. I. L., Sonnadara, D. U. J. & Jayananda, M. K., 2007. ***License Plate Edentification Based on Image Processing Techniques.*** Sri Lanka, International Conference on Industrial and Information Systems.

MathWorks. **Detecting a Cell Using Image Segmentation**. http://www.mathworks.com/help/images/examples/detecting-a-cell-using-image-segmentation.html [Acesso em 16 Novembro 2015].

MathWorks. **Labeling and Measuring Objects in a Binary Image**. http://www.mathworks.com/help/images/labeling-and-measuring-objects-in-a-binary-image.html [Acesso em 16 Novembro 2015].

MathWorks. **Digit Classification Using HOG Features**. http://www.mathworks.com/help/vision/examples/digit-classification-using-hog-features.html [Acesso em 20 Novembro 2015].

Imagem dos caracteres disponível em: http://pt.ffonts.net/Mandatory.font [Acesso em 16 Novembro 2015].

APÊNDICE A - Código principal

```
%% Reconhecimento de Placa
% Trabalho 02 de Processamento Digital de Sinais 2
% Gisele Tam GRR20102167
%
% O código contém 5 etapas:
% * pré-processamento
% * reconhecimento de bordas
% * localização da placa
% * segmentação dos caracteres
% * reconhecimento dos caracteres
%
%% 1) Pré-processamento
clear all
clc
% Carregamento da imagem
arq = 'placa3.jpg';
im_orig = imread(arq);
figure, imshow(im_orig), title('Imagem Original')

% Imagem em escala cinza
im_gray = rgb2gray(im_orig);
figure, imshow(im_gray),title('Imagem em escala cinza')


%% 2) Reconhecimento de bordas

% Filtro Sobel
aux = im_gray;
[~,thres] = edge(aux,'sobel');
fator = 0.8;
im_sobel2 = edge(aux,'sobel',thres*fator);
figure
subplot(3,2,1), imshow(im_sobel2), title('Filtro Sobel')

% Dilatar a imagem
se = strel('rectangle',[50 20]);
im_dil = imdilate(im_sobel2,se);
subplot(3,2,2), imshow(im_dil), title('Bordas Dilatadas')

% Erosão da imagem
se = strel('rectangle',[100 75]);
im_er = imerode(im_dil,se);
subplot(3,2,3), imshow(im_er), title('Erosão da Imagem')

% Limpar as bordas
im_l = imclearborder(im_er,4);
subplot(3,2,4), imshow(im_l), title('Bordas retiradas')

% Preencher buracos
im_preenc = imfill(im_l,'holes');
subplot(3,2,5), imshow(im_preenc), title('Imagem Preenchida')

%Remover objetos pequenos
remover = 30000;
im_rem = bwareaopen(im_preenc, remover);
```

```
subplot(3,2,6), imshow(im_rem), title('Remoção de objetos pequenos')


%% 3) Localização da placa

stats = regionprops('table',im_rem,'BoundingBox');
% Cada linha do BoundingBox está no formato:
% [ul_x ul_y length_x length_y]; ul = upper left
%  As placas tem tamanho 40x13 cm; 13/70 = 0,325

regiao = 1; % se houver somente uma area

if height(stats)>1 % caso haja mais de uma área
    possivel_regiao = []; % armazenar as possíveis regiões
    criterio_alt = []; % critério de desempate: altura na imagem (não da
região)
    for i = 1:height(stats)
        if stats(i,1).BoundingBox(3) > 1.5*(stats(i,1).BoundingBox(4))
            % verificar a proporção entre os lados => não adianta com
            % placas inclinadas; fazer com comprimento > altura
%             prop = stats(i,1).BoundingBox(4)/stats(i,1).BoundingBox(3);
            possivel_regiao = [possivel_regiao i];
            criterio_alt = [criterio_alt stats(i,1).BoundingBox(2)];
        end
    end
    if length(possivel_regiao)==1
        regiao = possivel_regiao(1);
    else % existe mais de uma região que respeita a relação
        [i,ind_y] = max(criterio_alt); % recuperar o indice com maior
altura
        regiao = possivel_regiao(ind_y);
    end
else
    regiao = 1;
end

placa = imcrop(im_gray, stats(regiao,:).BoundingBox);
figure, imshow(placa), title('Placa do carro')


%% 4) Segmentação dos caracteres

% Imagem em binário
thresh = graythresh(placa);
placa_bw = im2bw(placa,thresh);
placa_bw = ~placa_bw;


%%
% Remover ruídos (imagens menor que o parâmetro)
area = 30;
pl_rem = bwareaopen(placa_bw, area);

% 2ª Erosão
seD = strel('diamond',2);
pl_erd = imerode(pl_rem,seD);


% Label(): conecta componentes em imagens binárias 2D
```

```
[label, nElem] = bwlabel(pl_erd);
% Propriedades da Imagem
statsPlaca = regionprops(label,'BoundingBox');
hold on

figure, imshow(pl_erd), title('Placa em binário')


%%
coord = [] ;
c = 1;
for ind=1:nElem
    aux = statsPlaca(ind,1).BoundingBox;
    prop = aux(3)/aux(4);  % base/altura; I = 10/63 e A = 54/63
    if (prop >= 0.08) & (prop <= 1.0)
        coord(c,:) = aux;
        c = c+1;
    end
end

% ordenar as linhas de acordo com as colunas
coord2 = sortrows(coord,4);
coordY = coord2(end-6:end,:); % retirar os últimos (mais altos)
coordX = sortrows(coordY,3); % ordenar de acordo com o comprimento
coordX2 = coordX(1:end,:); % retirar os mais largos
caract = sortrows(coordX2,1); % para que fique na ordem

figure
letra = [];
for ind=1:length(caract)
    x = floor(caract(ind,1));
    y = floor(caract(ind,2));
    dx = caract(ind,3);
    dy = caract(ind,4);
    n1 = pl_erd(y+1:y+dy-1,x+1:x+dx-1);
    n1 = imresize(n1, [150 150]);
    letra{ind} = ~n1;
    imshow(letra{ind});
    pause(0.5)
end

%% 5) Reconhecimento dos caracteres - Aspectos HOG
% Baseado em "Digit classification using HOG Features", do MathWorks
% http://www.mathworks.com/help/vision/examples/digit-classification-using-
hog-features.html

%%
% Treinamento

cellSize = [8 8];
load('trainHOG4.mat');

%%
% Avaliação dos caracteres da placa

% Letras
testFeaturesLetras = [];

for i=1:3
```

```
    img = letra{i};
    testFeaturesLetras(i,:) = extractHOGFeatures(img,'CellSize',cellSize);
end

% Números
testFeaturesNum = [];

for i=4:7
    img = letra{i};
    testFeaturesNum(i-3,:) = extractHOGFeatures(img,'CellSize',cellSize);
end

%%
% Predição dos caracteres

predictedLetras = predict(classLetras, testFeaturesLetras);
predictedNum = predict(classNum, testFeaturesNum);


%% Resultados
fprintf('Identificação da placa: %s%s \n', predictedLetras,predictedNum)
```

APÊNDICE B – Programa auxiliar para formação do banco de amostras dos caracteres

```
%% Criar as amostras das imagens

%% Manuseio das pastas

%Selecionar o caractere a ser modificado
caractere = 'Z';
numOUletra = 'letras';
% Pasta atual
pastaAtual = pwd;

% Adicionar as subpastas para procurar pelo caractere original
addpath(genpath(pwd));

% Pasta na qual as imagens serão salvas
pastaSalvar =
sprintf('%s\\Caracteres\\%s\\%c',pastaAtual,numOUletra,caractere);

%% O que mudar
% Rotacionar
girar = 35; % em graus

% Corroer bordas
corroer = 4;
se = strel('diamond',corroer);

%% Ler a imagem original da pasta
% Ler o caractere
a = imread(caractere,'jpg');
a1 = im2bw(a);
[m1 n1] = size(a1);
% figure
subplot(4,3,1), imshow(a1);

% Célula para armazenamento
cImagens{1} = a1;

%% Modificar original

a2 = imrotate(~a1,girar);
[m2 n2] = size(a2);
difm = abs(ceil((m2-m1)/2))+1;
difn = abs(ceil((n2-n1)/2))+1;
a2 = ~a2(difm:end-difm,difn:end-difn);
% a2 = ~imclearborder(a2);
subplot(4,3,2), imshow(a2);

cImagens{2} = a2;

a3 = imrotate(~a1,-girar);
% a3 = ~imclearborder(a3);
[m3 n3] = size(a3);
difm = abs(ceil((m3-m1)/2))+1;
difn = abs(ceil((n3-n1)/2))+1;
a3 = ~a3(difm:end-difm,difn:end-difn);
```

```
subplot(4,3,3), imshow(a3);
cImagens{3} = a3;

%%
% Acrescentar ruído
a4 = imnoise(a,'gaussian', 0.32);
a4 = im2bw(a4);

subplot(4,3,4), imshow(a4)
cImagens{4} = a4;

% Rotacionar
a5 = imrotate(~a4,girar);
% a5 = ~imclearborder(a5);
[m5 n5] = size(a5);
difm = abs(ceil((m5-m1)/2))+1;
difn = abs(ceil((n5-n1)/2))+1;
a5 = ~a5(difm:end-difm,difn:end-difn);

subplot(4,3,5), imshow(a5);
cImagens{5} = a5;

a6 = imrotate(~a4,-girar);
% a6 = ~imclearborder(a6);
[m6 n6] = size(a6);
difm = abs(ceil((m6-m1)/2))+1;
difn = abs(ceil((n6-n1)/2))+1;
a6 = ~a6(difm:end-difm,difn:end-difn);

subplot(4,3,6), imshow(a6);
cImagens{6} = a6;

%%
% Corroer as bordas
a7 = imdilate(a1,se);

subplot(4,3,7), imshow(a7)
cImagens{7} = a7;

% Rotacionar
a8 = imrotate(~a7,girar);
% a8 = ~imclearborder(a8);
[m8 n8] = size(a8);
difm = abs(ceil((m8-m1)/2))+1;
difn = abs(ceil((n8-n1)/2))+1;
a8 = ~a8(difm:end-difm,difn:end-difn);

subplot(4,3,8), imshow(a8);
cImagens{8} = a8;

a9 = imrotate(~a7,-girar);
% a9 = ~imclearborder(a9);
[m9 n9] = size(a9);
difm = abs(ceil((m9-m1)/2))+1;
difn = abs(ceil((n9-n1)/2))+1;
a9 = ~a9(difm:end-difm,difn:end-difn);

subplot(4,3,9), imshow(a9);
```

```
cImagens{9} = a9;

%%
% Acrescentar ruído e corroer bordas
a10 = imnoise(a,'gaussian', 0.3);
a10 = im2bw(a10);
se2 = strel('diamond',corroer);
a10 = imdilate(a10,se2);
[m10 n10] = size(a10);

subplot(4,3,10), imshow(a10)
cImagens{10} = a10;

% Rotacionar
a11 = imrotate(~a10,girar);
% a11 = ~imclearborder(a11);
[m11 n11] = size(a11);
difm = abs(ceil((m11-m10)/2))+1;
difn = abs(ceil((n11-n10)/2))+1;
a11 = ~a11(difm:end-difm,difn:end-difn);

subplot(4,3,11), imshow(a11);
cImagens{11} = a11;

a12 = imrotate(~a10,-girar);
% a12 = ~imclearborder(a12);
[m12 n12] = size(a12);
difm = abs(ceil((m12-m10)/2))+1;
difn = abs(ceil((n12-n10)/2))+1;
a12 = ~a12(difm:end-difm,difn:end-difn);

subplot(4,3,12), imshow(a12);
cImagens{12} = a12;


%% Salvar na pasta adequada

for i = 1:length(cImagens)
    ind = i+36;
    filename = sprintf('%s\\%c_%d.jpg',pastaSalvar,caractere,ind);
    im = cImagens{i};
    im = imresize(im, [150 150]);
    imwrite(im, filename);
end
```

APÊNDICE C – Treinamento dos aspectos HOG (Histogram of Oriented Gradients)

```
%% Treinamento das feições HOG (Histogram of Oriented Gradients)

% Baseado em "Digit classification using HOG Features", do MathWorks
% http://www.mathworks.com/help/vision/examples/digit-classification-using-
hog-features.html
clear
clc

% Setar pasta com as amostras
pathLetras = fullfile(pwd, 'Caracteres', 'letras');
pathNum = fullfile(pwd, 'Caracteres', 'num');

% Escanear recursivamente as imagens para o treinamento
trainSetLetras = imageSet(pathLetras, 'recursive');
trainSetNum = imageSet(pathNum, 'recursive');

%% Testar tamanho do histograma
%
% img = read(trainNum(3), 4);
%
% % Extract HOG features and HOG visualization
% [hog_16x16, vis2x2] = extractHOGFeatures(img,'CellSize',[16 16]);
% [hog_4x4, vis4x4] = extractHOGFeatures(img,'CellSize',[4 4]);
% [hog_8x8, vis8x8] = extractHOGFeatures(img,'CellSize',[8 8]);
%
% % Show the original image
% figure
% subplot(2,3,1:3), imshow(img);
%
% % Visualize the HOG features
% subplot(2,3,4), plot(vis2x2)
% title({'CellSize = [16 16]'; ['Feature length = '
num2str(length(hog_16x16))]});
%
% subplot(2,3,5), plot(vis4x4)
% title({'CellSize = [4 4]'; ['Feature length = '
num2str(length(hog_4x4))]});
%
% subplot(2,3,6), plot(vis8x8)
% title({'CellSize = [8 8]'; ['Feature length = '
num2str(length(hog_8x8))]});

%% Escolher com base no gráfico anterior

cellSize = [8 8];


%% Treinar a classificação das letras

img = read(trainSetLetras(3), 4);
[hog_8x8, vis8x8] = extractHOGFeatures(img,'CellSize',cellSize);
hogFeatureSize = length(hog_8x8);

trainFeaturesLetras = [];
```

```
trainLabelLetras = [];

for digit = 1:numel(trainSetLetras)

    numImagem = trainSetLetras(digit).Count;
    features = zeros(numImagem,hogFeatureSize,'single');

    for i = 1:numImagem
        img = read(trainSetLetras(digit),i);

        % pre-processamento da amostra
        lThresh = graythresh(img);
        img = im2bw(img,lThresh);

        features(i,:) = extractHOGFeatures(img,'CellSize',cellSize);
    end

    % Usar o nome da pasta como rótulo do treinamento
    label = repmat(trainSetLetras(digit).Description,numImagem,1);

    trainFeaturesLetras = [trainFeaturesLetras; features];
    trainLabelLetras = [trainLabelLetras; label];
end


%% Treinar a classificação dos números

img = read(trainSetNum(3), 4);
[hog_8x8, vis8x8] = extractHOGFeatures(img,'CellSize',cellSize);
hogFeatureSize = length(hog_8x8);

trainFeaturesNum = [];
trainLabelNum = [];

for digit = 1:numel(trainSetNum)

    numImagem = trainSetNum(digit).Count;
    features = zeros(numImagem,hogFeatureSize,'single');

    for i = 1:numImagem
        img = read(trainSetNum(digit),i);

        % pre-processamento da amostra
        lThresh = graythresh(img);
        img = im2bw(img,lThresh);

        features(i,:) = extractHOGFeatures(img,'CellSize',cellSize);
    end

    % Usar o nome da pasta como rótulo do treinamento
    label = repmat(trainSetNum(digit).Description,numImagem,1);

    trainFeaturesNum = [trainFeaturesNum; features];
    trainLabelNum = [trainLabelNum; label];
end
```

```
%% Treino do classificador

classLetras = fitcecoc(trainFeaturesLetras, trainLabelLetras);
classNum = fitcecoc(trainFeaturesNum, trainLabelNum);

%% Salvar as variáveis
save('trainHOG4.mat')
```