**CONCEPTS**

**A picture containing text, screenshot, diagram, plan

Description automatically generated**

A picture containing outdoor, black and white, screenshot, ground

Description automatically generated A collage of images of cars

Description automatically generated with low confidence

**MATLAB® Code**

clc; clearvars; close all;

img = imread('019.png');

img = rgb2gray(img);

%% Equalization

[x,y,z] = size(img);

img = reshape(img,x,y\*z);

BHE = histeq(img);

CLAHE = adapthisteq(img);

%% Calculation of BDIM-Based Weights

kernel = 3;

Condition = 0;

[w,~] = Weighting\_Map();

WDIMTE\_BHE = filter2(w, BHE);

WDIMTE\_BHE = NormalRange(WDIMTE\_BHE,0,255,1);

WDIMTE\_CLAHE = filter2(w, CLAHE);

WDIMTE\_CLAHE = NormalRange(WDIMTE\_CLAHE,0,255,1);

[BDIM\_BHE,~] = Thermal\_DIMTE(WDIMTE\_BHE,kernel,Condition);

[BDIM\_CLAHE,~] = Thermal\_DIMTE(WDIMTE\_CLAHE,kernel,Condition);

Adapt\_Weights = BDIM\_BHE + BDIM\_CLAHE;

W1 = BDIM\_BHE ./ Adapt\_Weights;

W2 = BDIM\_CLAHE ./ Adapt\_Weights;

%% Reshape

BHE = reshape(BHE,x,y,z);

CLAHE = reshape(CLAHE,x,y,z);

W1 = reshape(W1,x,y,z);

W2 = reshape(W2,x,y,z);

%% Adaptive Image Fusion

Fused = uint8(W1 .\* double(BHE) + W2 .\* double(CLAHE));

%% Recoloring

[Recolored\_img] = Colormap(img,0);

[Recolored\_BHE] = Colormap(BHE,0);

[Recolored\_CLAHE] = Colormap(CLAHE,0);

[Recolored\_Fused] = Colormap(Fused,0);

%% Display images

figure; subplot(2,2,1); imshow(img); title('Input');

subplot(2,2,2); imshow(Fused); title('Fused');

subplot(2,2,3); imshow(BHE); title('Enhanced 1');

subplot(2,2,4); imshow(CLAHE); title('Enhanced 2');

figure; subplot(2,2,1); imshow(Recolored\_img); title('Input');

subplot(2,2,2); imshow(Recolored\_Fused); title('Fused');

subplot(2,2,3); imshow(Recolored\_BHE); title('Enhanced 1');

subplot(2,2,4); imshow(Recolored\_CLAHE); title('Enhanced 2');

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

where and respectively denote the minimum probability density value and the maximum probability density value in each local tile , and represent the minimum intensity value and the maximum intensity value in each local tile , respectively, and refers to an offset value.

function [BDIM,DIMTE] = Thermal\_DIMTE(img,kernel,Condition)

% Condition = normalize : Rescale the thermal metric to [0,1]

% Kernel : Size of a tile

% k = 3;

% Condition = 'normalized';

k = kernel;

S = round(k/2) - 1;

img = padarray(img,[S S],'symmetric');

[~,~,z] = size(img);

if z == 3

img = rgb2gray(img);

end

I = GrayDouble(img);

Local\_Block = im2col(I,[k k]);

DIMTE = zeros(1,size(Local\_Block,2));

**parfor** i = 1:size(Local\_Block,2) **%% In this line, I enable the parallel**

**%% processing. If your processor doesn’t**

**%% support, you can change from parfor to for**

Current\_I = sort(Local\_Block(:,i));

Current\_P = Current\_I/sum(Current\_I(:));

Current\_C = cumsum(Current\_P);

Pmin = min(Current\_P(:));

Pmax = max(Current\_P(:));

Imin = min(Current\_I(:));

Imax = max(Current\_I(:));

P\_Dark = Current\_C(1:end-1);

P\_Bright = 1 - P\_Dark;

P\_Entropy = P\_Dark .\* log10(P\_Dark ./ P\_Bright);

T = find(P\_Entropy==min(P\_Entropy(:)), 1);

if isempty(T) == 0

DIMTE(i) = (Pmin(1) / (Pmax(1) + Pmin(1))) \* ((Imin(1) / (Imax(1) + Imax(1))) ^ 2);

end

end

DIMTE = reshape(DIMTE,size(img,1)-k+1,size(img,2)-k+1);

InvertDIMTE = 1 ./ (1 + DIMTE);

if Condition == 'normalized'

%% Normalization

DIMTE = NormalRange(DIMTE,0,1,0);

InvertDIMTE = NormalRange(InvertDIMTE,0,1,0);

end

BDIM = InvertDIME;

end

function newimg = NormalRange(img, Jmin, Jmax, frac)

if ~isa(img, 'double' )

img = double(img);

end

Imax = max(img(:));

Imin = min(img(:));

if Imin < 0

img = (img - Imin) / (abs(Imin) + Imax);

Imin = 0;

end

newimg = img - Imin;

newimg = newimg ./ max(newimg(:)) \* (Jmax - Jmin);

newimg = newimg + Jmin;

if frac == 1

newimg = uint8(newimg);

end

end

function newimg = GrayDouble(img)

if size(img,3) == 3

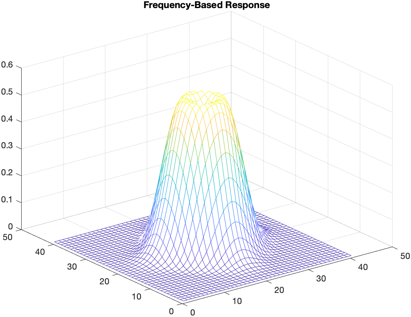
newimg = double(rgb2gray(img));

else

newimg = double(img);

end

end

function [w,map] = Weighting\_Map()

min\_f = -20;

max\_f = 20;

step\_f = 1;

u = min\_f:step\_f:max\_f;

U = u;

for i = 1:(length(u) - 1)

U = cat(1,U,u);

end

sigma = 2;

u = U;

v = U';

f = sqrt(u .\* u + v .\* v);

w = 2 .\* pi .\* f ./ 60;

Sw = 1.5 .\* exp(-sigma .^2 .\*w .^ 2 ./ 2)-exp(-2 .\* sigma .^ 2 .\* w .^2 ./ 2);

% Modification in High frequency

sita = atan(v./(u+eps));

bita = 8;

f0 = 11.13;

w0 = 2 .\* pi .\* f0 ./ 60;

Ow = (1 + exp(bita .\* (w - w0)) .\* (cos(2 .\* sita)) .^ 4) ./ (1 + exp(bita .\* (w - w0)));

% Compute final response

map = Sw .\* Ow;

w = fsamp2(map);

end