# Cân bằng xám ảnh màu (hàm eq\_channel)

clc;

clear;

close all;

% Đọc ảnh màu

I = imread('cola2.png');

I = double(I); % Chuyển sang kiểu double để xử lý

% Tách kênh màu R, G, B

R = I(:,:,1);

G = I(:,:,2);

B = I(:,:,3);

% Cân bằng histogram cho từng kênh màu

R\_eq = equalize\_channel(R);

G\_eq = equalize\_channel(G);

B\_eq = equalize\_channel(B);

% Ghép lại thành ảnh màu sau khi cân bằng

I\_eq = cat(3, uint8(R\_eq), uint8(G\_eq), uint8(B\_eq));

% Hiển thị ảnh gốc và ảnh đã cân bằng

figure;

subplot(2,2,1);

imshow(uint8(I));

title('Ảnh màu gốc');

subplot(2,2,2);

imshow(I\_eq);

title('Ảnh màu sau khi cân bằng');

% Hiển thị histogram trước và sau cân bằng của kênh đỏ (R)

subplot(2,2,3);

histogram(uint8(R(:)), 256, 'FaceColor', 'r');

title('Histogram R gốc');

subplot(2,2,4);

histogram(uint8(R\_eq(:)), 256, 'FaceColor', 'r');

title('Histogram R sau cân bằng');

% Định nghĩa hàm nằm cuối file

function eq\_channel = equalize\_channel(channel)

[rows, cols] = size(channel);

hist\_vals = zeros(1, 256);

% Tính histogram

for x = 1:rows

for y = 1:cols

pixel\_value = channel(x, y) + 1;

hist\_vals(pixel\_value) = hist\_vals(pixel\_value) + 1;

end

end

% Tính CDF (Hàm phân phối tích lũy)

cdf = cumsum(hist\_vals) / sum(hist\_vals);

% Ánh xạ mức xám mới

eq\_channel = zeros(size(channel));

for x = 1:rows

for y = 1:cols

eq\_channel(x, y) = round(cdf(channel(x, y) + 1) \* 255);

end

end

end

* **Lưu ý nếu lỗi: Index in position 3 exceeds array bounds (must not exceed 1). Thì là do ảnh đầu vào không phải ảnh màu nêu không có kênh màu thứ 2**
* **MATLAB yêu cầu mọi function trong script phải nằm cuối file, hoặc bạn phải chuyển toàn bộ thành một function lớn.**

# Phát hiện biên Hàm SOBEL

clc;

clear;

close all;

% Đọc ảnh màu

A = imread('cola1.png');

% Gọi hàm Sobel

G = sobel\_edge\_color(A);

% Hiển thị kết quả

figure;

subplot(1,2,1), imshow(A); title('Ảnh gốc');

subplot(1,2,2), imshow(G, []); title('Biên ảnh (Sobel)');

% ======= Hàm Sobel cho Ảnh Màu =======

function G = sobel\_edge\_color(A)

% Chuyển ảnh sang kiểu double để tính toán chính xác

A = double(A);

% Tách kênh màu R, G, B

R = A(:,:,1);

G = A(:,:,2);

B = A(:,:,3);

% Áp dụng Sobel cho từng kênh màu

edge\_R = sobel\_edge(R);

edge\_G = sobel\_edge(G);

edge\_B = sobel\_edge(B);

% Tổng hợp biên bằng cách lấy giá trị lớn nhất tại mỗi pixel

G = max(cat(3, edge\_R, edge\_G, edge\_B), [], 3);

end

% ======= Hàm Sobel cho Ảnh Xám =======

function G = sobel\_edge(A)

% Chuyển ảnh sang kiểu double

A = double(A);

% Định nghĩa mặt nạ Sobel

sobel\_x = [-1 0 1; -2 0 2; -1 0 1]; % Phát hiện biên dọc

sobel\_y = [-1 -2 -1; 0 0 0; 1 2 1]; % Phát hiện biên ngang

% Kích thước ảnh

[rows, cols] = size(A);

% Khởi tạo ảnh biên

Gx = zeros(rows, cols);

Gy = zeros(rows, cols);

G = zeros(rows, cols);

% Tính toán tích chập thủ công (bỏ qua viền ngoài)

for i = 2:rows-1

for j = 2:cols-1

% Cửa sổ 3x3 quanh pixel (i, j)

window = A(i-1:i+1, j-1:j+1);

% Áp dụng mặt nạ Sobel theo X và Y

Gx(i, j) = sum(sum(window .\* sobel\_x));

Gy(i, j) = sum(sum(window .\* sobel\_y));

% Tổng hợp biên

G(i, j) = sqrt(Gx(i, j)^2 + Gy(i, j)^2);

end

end

end

# Phát hiện biên Hàm PREWITT

clc;

clear;

close all;

% Đọc ảnh màu

A = imread('cola1.png');

% Gọi hàm Prewitt

G = prewitt\_edge\_color(A);

% Hiển thị kết quả

figure;

subplot(1,2,1), imshow(A); title('Ảnh gốc');

subplot(1,2,2), imshow(G, []); title('Biên ảnh (Prewitt)');

% ======= Hàm Prewitt cho Ảnh Màu =======

function G = prewitt\_edge\_color(A)

% Chuyển ảnh sang kiểu double để tính toán chính xác

A = double(A);

% Tách kênh màu R, G, B

R = A(:,:,1);

G = A(:,:,2);

B = A(:,:,3);

% Áp dụng Prewitt cho từng kênh màu

edge\_R = prewitt\_edge(R);

edge\_G = prewitt\_edge(G);

edge\_B = prewitt\_edge(B);

% Tổng hợp biên bằng cách lấy giá trị lớn nhất tại mỗi pixel

G = max(cat(3, edge\_R, edge\_G, edge\_B), [], 3);

end

% ======= Hàm Prewitt cho Ảnh Xám =======

function G = prewitt\_edge(A)

% Chuyển ảnh sang kiểu double

A = double(A);

% Định nghĩa mặt nạ Prewitt

prewitt\_x = [-1 0 1; -1 0 1; -1 0 1]; % Phát hiện biên dọc

prewitt\_y = [-1 -1 -1; 0 0 0; 1 1 1]; % Phát hiện biên ngang

% Kích thước ảnh

[rows, cols] = size(A);

% Khởi tạo ảnh biên

Gx = zeros(rows, cols);

Gy = zeros(rows, cols);

G = zeros(rows, cols);

% Tính toán tích chập thủ công (bỏ qua viền ngoài)

for i = 2:rows-1

for j = 2:cols-1

% Cửa sổ 3x3 quanh pixel (i, j)

window = A(i-1:i+1, j-1:j+1);

% Áp dụng mặt nạ Prewitt theo X và Y

Gx(i, j) = sum(sum(window .\* prewitt\_x));

Gy(i, j) = sum(sum(window .\* prewitt\_y));

% Tổng hợp biên

G(i, j) = sqrt(Gx(i, j)^2 + Gy(i, j)^2);

end

end

end

# Phát hiện biên Hàm ROBERTS

clc;

clear;

close all;

% Đọc ảnh màu

A = imread('cola1.png');

% Gọi hàm Roberts

G = roberts\_edge\_color(A);

% Hiển thị kết quả

figure;

subplot(1,2,1), imshow(A); title('Ảnh gốc');

subplot(1,2,2), imshow(G, []); title('Biên ảnh (Roberts)');

% ======= Hàm Roberts cho Ảnh Màu =======

function G = roberts\_edge\_color(A)

% Chuyển ảnh sang kiểu double để tính toán chính xác

A = double(A);

% Tách kênh màu R, G, B

R = A(:,:,1);

G = A(:,:,2);

B = A(:,:,3);

% Áp dụng Roberts cho từng kênh màu

edge\_R = roberts\_edge(R);

edge\_G = roberts\_edge(G);

edge\_B = roberts\_edge(B);

% Tổng hợp biên bằng cách lấy giá trị lớn nhất tại mỗi pixel

G = max(cat(3, edge\_R, edge\_G, edge\_B), [], 3);

end

% ======= Hàm Roberts cho Ảnh Xám =======

function G = roberts\_edge(A)

% Chuyển ảnh sang kiểu double

A = double(A);

% Định nghĩa mặt nạ Roberts

roberts\_x = [1 0; 0 -1]; % Phát hiện biên theo đường chéo chính

roberts\_y = [0 1; -1 0]; % Phát hiện biên theo đường chéo phụ

% Kích thước ảnh

[rows, cols] = size(A);

% Khởi tạo ảnh biên

Gx = zeros(rows, cols);

Gy = zeros(rows, cols);

G = zeros(rows, cols);

% Tính toán tích chập thủ công (bỏ qua viền ngoài)

for i = 1:rows-1

for j = 1:cols-1

% Cửa sổ 2x2 quanh pixel (i, j)

window = A(i:i+1, j:j+1);

% Áp dụng mặt nạ Roberts theo X và Y

Gx(i, j) = sum(sum(window .\* roberts\_x));

Gy(i, j) = sum(sum(window .\* roberts\_y));

% Tổng hợp biên

G(i, j) = sqrt(Gx(i, j)^2 + Gy(i, j)^2);

end

end

end