**THUẬT TOÁN CÂN BẰNG MỨC XÁM**

% Đọc ảnh từ file anhXam.tif

grayImg = imread('anhXam.tif');

% Kiểm tra nếu ảnh không phải ảnh xám, chuyển về ảnh xám

if size(grayImg, 3) == 3

grayImg = rgb2gray(grayImg);

end

% Kích thước ảnh

[m, n] = size(grayImg);

L = 256; % Số mức xám

% Bước 1: Tính Histogram

histogram = zeros(1, L); % Khởi tạo histogram

for i = 1:m

for j = 1:n

histogram(grayImg(i, j) + 1) = histogram(grayImg(i, j) + 1) + 1;

end

end

% Bước 2: Tính xác suất xuất hiện của mỗi mức xám

pdf = histogram / (m \* n);

% Bước 3: Tính lược đồ tích lũy (CDF)

cdf = cumsum(pdf);

% Bước 4: Chuẩn hóa mức xám mới

newLevels = round((L - 1) \* cdf);

% Bước 5: Ánh xạ mức xám mới cho từng điểm ảnh

equalizedImg = uint8(zeros(m, n));

for i = 1:m

for j = 1:n

equalizedImg(i, j) = newLevels(grayImg(i, j) + 1);

end

end

% Hiển thị ảnh gốc và ảnh cân bằng lược đồ

figure;

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

subplot(1, 2, 2); imshow(equalizedImg); title('Ảnh cân bằng lược đồ');

**1. Triển khai thuật toán cân bằng lược đồ xám thủ công**

**Bài tập**

* Tính toán từng bước:
  1. Histogram.
  2. PDF (Probability Density Function).
  3. CDF (Cumulative Distribution Function).
  4. Ánh xạ lại giá trị pixel.

**Mã MATLAB**

function outputImg = histogramEqualization(inputImg)

% Kiểm tra ảnh đầu vào

if size(inputImg, 3) > 1

error('Ảnh đầu vào phải là ảnh mức xám.');

end

[rows, cols] = size(inputImg);

% Tính histogram

hist = zeros(1, 256);

for r = 1:rows

for c = 1:cols

intensity = inputImg(r, c);

hist(intensity + 1) = hist(intensity + 1) + 1;

end

end

% Tính PDF

pdf = hist / (rows \* cols);

% Tính CDF

cdf = zeros(1, 256);

cdf(1) = pdf(1);

for i = 2:256

cdf(i) = cdf(i - 1) + pdf(i);

end

% Chuẩn hóa CDF về dải [0, 255]

cdfNorm = round(cdf \* 255);

% Ánh xạ giá trị pixel

outputImg = zeros(rows, cols, 'uint8');

for r = 1:rows

for c = 1:cols

intensity = inputImg(r, c);

outputImg(r, c) = cdfNorm(intensity + 1);

end

end

end

% Sử dụng

img = imread('anhXam.tif');

equalizedImg = histogramEqualization(img);

imshow(equalizedImg); title('Ảnh sau cân bằng');

**Câu hỏi lý thuyết**

1. **Tại sao cần tính CDF trong cân bằng lược đồ xám?**  
   **Trả lời**: CDF cho phép ánh xạ các giá trị mức xám cũ sang dải giá trị mới, làm cho histogram phân bố đều hơn.
2. **Lợi ích của cân bằng lược đồ xám là gì?**  
   **Trả lời**: Tăng độ tương phản của ảnh, làm rõ chi tiết trong các vùng sáng hoặc tối.

**2. So sánh ảnh trước và sau cân bằng**

**Bài tập**

* Hiển thị ảnh gốc và ảnh sau cân bằng.
* Vẽ histogram để so sánh.

**Mã MATLAB**

**--------------------------------------- THÊM HÀM histogramEqualization Ở CÂU TRÊN VÀI ---------------------------**

function compareImages(inputImg)

% Kiểm tra ảnh đầu vào

if size(inputImg, 3) > 1

error('Ảnh đầu vào phải là ảnh mức xám.');

end

% Cân bằng lược đồ thủ công

outputImg = histogramEqualization(inputImg);

% Tính histogram thủ công cho ảnh gốc

histOriginal = zeros(1, 256);

[rows, cols] = size(inputImg);

for r = 1:rows

for c = 1:cols

intensity = inputImg(r, c);

histOriginal(intensity + 1) = histOriginal(intensity + 1) + 1;

end

end

% Tính histogram thủ công cho ảnh sau cân bằng

histEqualized = zeros(1, 256);

for r = 1:rows

for c = 1:cols

intensity = outputImg(r, c);

histEqualized(intensity + 1) = histEqualized(intensity + 1) + 1;

end

end

% Hiển thị ảnh và histogram

figure;

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

subplot(2, 2, 2); imshow(outputImg); title('Ảnh sau cân bằng');

subplot(2, 2, 3); bar(0:255, histOriginal); title('Histogram gốc');

subplot(2, 2, 4); bar(0:255, histEqualized); title('Histogram sau cân bằng');

end

% Sử dụng

img = imread('anhXam.tif');

compareImages(img);

**Câu hỏi lý thuyết**

1. **Histogram sau cân bằng khác gì histogram gốc?**  
   **Trả lời**: Histogram sau cân bằng phân bố đều hơn trên toàn dải [0,255][0, 255].
2. **Tại sao ảnh sau cân bằng có độ tương phản tốt hơn?**  
   **Trả lời**: Các mức xám được tái phân bố, làm rõ chi tiết trong các vùng sáng và tối.

**3. Tính thống kê trước và sau cân bằng**

**Bài tập**

* Tính trung bình (mean), độ lệch chuẩn (std) của ảnh trước và sau cân bằng.

**Mã MATLAB**

**--------------------------------------- THÊM HÀM histogramEqualization Ở CÂU TRÊN VÀI ---------------------------**

function calculateStatistics(inputImg)

% Kiểm tra ảnh đầu vào

if size(inputImg, 3) > 1

error('Ảnh đầu vào phải là ảnh mức xám.');

end

% Cân bằng lược đồ thủ công

outputImg = histogramEqualization(inputImg);

% Tính giá trị trung bình thủ công

[rows, cols] = size(inputImg);

totalPixels = rows \* cols;

meanBefore = sum(double(inputImg(:))) / totalPixels;

meanAfter = sum(double(outputImg(:))) / totalPixels;

% Tính độ lệch chuẩn thủ công

stdBefore = sqrt(sum((double(inputImg(:)) - meanBefore).^2) / totalPixels);

stdAfter = sqrt(sum((double(outputImg(:)) - meanAfter).^2) / totalPixels);

% Hiển thị kết quả

fprintf('Trước cân bằng: Mean = %.2f, Std = %.2f\n', meanBefore, stdBefore);

fprintf('Sau cân bằng: Mean = %.2f, Std = %.2f\n', meanAfter, stdAfter);

end

% Sử dụng

img = imread('anhXam.tif');

calculateStatistics(img);

**Câu hỏi lý thuyết**

1. **Độ lệch chuẩn thay đổi như thế nào sau cân bằng?**  
   **Trả lời**: Độ lệch chuẩn thường tăng, thể hiện mức độ phân tán cao hơn và tăng độ tương phản.
2. **Giá trị trung bình thay đổi như thế nào?**  
   **Trả lời**: Giá trị trung bình có thể thay đổi nhẹ, phụ thuộc vào phân bố cường độ sáng.

**4. Cân bằng lược đồ cục bộ**

**Bài tập**

* Chia ảnh thành các block nhỏ và thực hiện cân bằng riêng cho từng block.

**Mã MATLAB**

**--------------------------------------- THÊM HÀM histogramEqualization Ở CÂU TRÊN VÀI ---------------------------**

function localEqualization(inputImg, blockSize)

% Kiểm tra ảnh đầu vào

if size(inputImg, 3) > 1

error('Ảnh đầu vào phải là ảnh mức xám.');

end

[rows, cols] = size(inputImg);

outputImg = inputImg;

% Duyệt qua từng block

for r = 1:blockSize:rows

for c = 1:blockSize:cols

% Lấy block hiện tại

block = inputImg(r:min(r+blockSize-1, rows), c:min(c+blockSize-1, cols));

% Cân bằng lược đồ thủ công cho block

equalizedBlock = histogramEqualization(block);

% Gán lại vào ảnh

outputImg(r:min(r+blockSize-1, rows), c:min(c+blockSize-1, cols)) = equalizedBlock;

end

end

% Hiển thị kết quả

figure;

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

subplot(1, 2, 2); imshow(outputImg); title('Ảnh sau cân bằng cục bộ');

end

% Sử dụng

img = imread('anhXam.tif');

localEqualization(img, 50); % Cân bằng cục bộ với block 50x50

**Câu hỏi lý thuyết**

1. **Khi nào nên sử dụng cân bằng lược đồ cục bộ?**  
   **Trả lời**: Khi ảnh có vùng sáng và vùng tối rõ rệt, cân bằng cục bộ giữ chi tiết tốt hơn.
2. **Hạn chế của cân bằng lược đồ cục bộ là gì?**  
   **Trả lời**: Có thể gây nhiễu hoặc hiện tượng "gồ ghề" nếu kích thước block quá nhỏ.

**5. Cân bằng lược đồ RGB so với ảnh xám**

**Bài tập**

* Cân bằng lược đồ trên từng kênh màu (R, G, B) của ảnh RGB.

**Mã MATLAB**

**--------------------------------------- THÊM HÀM histogramEqualization Ở CÂU TRÊN VÀI ---------------------------**

function rgbEqualization(inputImg)

% Kiểm tra ảnh đầu vào

if size(inputImg, 3) ~= 3

error('Ảnh đầu vào phải là ảnh RGB.');

end

% Lấy từng kênh màu

imgR = inputImg(:, :, 1);

imgG = inputImg(:, :, 2);

imgB = inputImg(:, :, 3);

% Cân bằng lược đồ cho từng kênh

equalizedR = histogramEqualization(imgR);

equalizedG = histogramEqualization(imgG);

equalizedB = histogramEqualization(imgB);

% Kết hợp lại thành ảnh RGB

outputImg = cat(3, equalizedR, equalizedG, equalizedB);

% Hiển thị kết quả

figure;

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

subplot(1, 2, 2); imshow(outputImg); title('Ảnh sau cân bằng RGB');

end

% Sử dụng

img = imread('anhMau.jpg');

rgbEqualization(img);

**Câu hỏi lý thuyết**

1. **Tại sao không áp dụng trực tiếp cân bằng lược đồ xám cho ảnh RGB?**  
   **Trả lời**: Làm như vậy sẽ mất thông tin màu sắc, gây sai lệch hình ảnh.
2. **Kết quả cân bằng RGB có thể thay đổi màu sắc không?**  
   **Trả lời**: Có, vì cân bằng riêng từng kênh có thể thay đổi mối quan hệ giữa các kênh màu.

**6. Cân bằng lược đồ theo mục tiêu**

**Bài tập**

* Điều chỉnh histogram của ảnh để khớp với một histogram mục tiêu.

**Mã MATLAB**

function outputImg = matchHistogram(inputImg, targetHist)

histInput = zeros(1, 256);

for i = 1:numel(inputImg)

histInput(inputImg(i) + 1) = histInput(inputImg(i) + 1) + 1;

end

cdfInput = cumsum(histInput) / numel(inputImg);

cdfTarget = cumsum(targetHist);

mapping = zeros(1, 256, 'uint8');

for i = 1:256

[~, idx] = min(abs(cdfInput(i) - cdfTarget));

mapping(i) = idx - 1;

end

outputImg = mapping(inputImg + 1);

end

% Sử dụng

targetHist = [1:128, 128:-1:1];

targetHist = targetHist / sum(targetHist);

matchedImg = matchHistogram(img, targetHist);

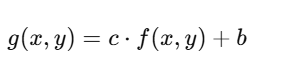
imshow(matchedImg); title('Ảnh sau ánh xạ histogram');

**Câu hỏi lý thuyết**

1. **Histogram mục tiêu là gì?**  
   **Trả lời**: Là histogram mà bạn muốn ảnh sau xử lý khớp với nó (ví dụ: tam giác, hình chuông).
2. **Ứng dụng của cân bằng theo mục tiêu là gì?**  
   **Trả lời**: Dùng trong xử lý ảnh y tế hoặc tái tạo hình ảnh theo tiêu chuẩn.

**BIẾN ĐỔI TUYẾN TÍNH**

Công thức:

****

* ccc: Hệ số tỷ lệ.
* bbb: Hệ số dịch.

**Triển khai trong MATLAB**

function outputImg = linearTransform(inputImg, c, b)

% Biến đổi tuyến tính g(x, y) = c \* f(x, y) + b

% inputImg: Ảnh xám đầu vào

% c: Hệ số tỷ lệ

% b: Hệ số dịch

% outputImg: Ảnh sau biến đổi

% Chuyển đổi sang kiểu double để tránh lỗi tràn

inputImg = double(inputImg);

% Áp dụng công thức

outputImg = c \* inputImg + b;

% Đưa về khoảng giá trị hợp lệ [0, 255]

outputImg(outputImg > 255) = 255;

outputImg(outputImg < 0) = 0;

% Chuyển về kiểu uint8 để hiển thị

outputImg = uint8(outputImg);

end

% Sử dụng hàm

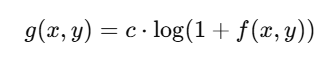
img = imread('anhXam.tif');

linearImg = linearTransform(img, 1.5, 20); % Tăng độ sáng

imshow(linearImg); title('Biến đổi tuyến tính');

**BIẾN ĐỔI LOGARIT**

Công thức:



* ccc: Hằng số để điều chỉnh phạm vi giá trị.

**Triển khai trong MATLAB**

function outputImg = logTransform(inputImg, c)

% Biến đổi logarit g(x, y) = c \* log(1 + f(x, y))

% inputImg: Ảnh xám đầu vào

% c: Hệ số điều chỉnh

% outputImg: Ảnh sau biến đổi

% Chuyển đổi sang kiểu double

inputImg = double(inputImg);

% Áp dụng công thức log

outputImg = c \* log(1 + inputImg);

% Chuẩn hóa về khoảng [0, 255]

outputImg = outputImg / max(outputImg(:)) \* 255;

% Chuyển về kiểu uint8 để hiển thị

outputImg = uint8(outputImg);

end

% Sử dụng hàm

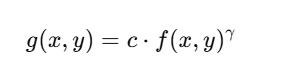
img = imread('anhXam.tif');

logImg = logTransform(img, 40); % Điều chỉnh với c = 40

imshow(logImg); title('Biến đổi logarit');

**BIẾN ĐỔI MŨ (POWER-LAW)**

Công thức:



* ccc: Hệ số điều chỉnh.
* γ\gammaγ: Thông số điều chỉnh độ sáng (gamma < 1 để làm sáng, gamma > 1 để làm tối).

**Triển khai trong MATLAB**

function outputImg = powerLawTransform(inputImg, c, gamma)

% Biến đổi mũ g(x, y) = c \* f(x, y)^gamma

% inputImg: Ảnh xám đầu vào

% c: Hệ số điều chỉnh

% gamma: Tham số gamma

% outputImg: Ảnh sau biến đổi

% Chuyển đổi sang kiểu double

inputImg = double(inputImg);

% Áp dụng công thức power-law

outputImg = c \* (inputImg .^ gamma);

% Chuẩn hóa về khoảng [0, 255]

outputImg = outputImg / max(outputImg(:)) \* 255;

% Chuyển về kiểu uint8 để hiển thị

outputImg = uint8(outputImg);

end

% Sử dụng hàm

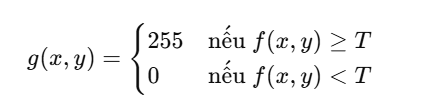
img = imread('anhXam.tif');

powerLawImg = powerLawTransform(img, 1, 0.5); % Làm sáng với gamma = 0.5

imshow(powerLawImg); title('Biến đổi mũ (Power-law)');

**CẮT NGƯỠNG (THRESHOLDING)**

Công thức:

* 
* TTT: Ngưỡng (Threshold).

**Triển khai trong MATLAB**

function outputImg = thresholdTransform(inputImg, T)

% Biến đổi ngưỡng g(x, y) = 255 nếu f(x, y) >= T, ngược lại = 0

% inputImg: Ảnh xám đầu vào

% T: Ngưỡng

% outputImg: Ảnh nhị phân sau biến đổi

% Áp dụng ngưỡng

outputImg = inputImg >= T;

% Nhân với 255 để chuyển thành ảnh nhị phân

outputImg = uint8(outputImg \* 255);

end

% Sử dụng hàm

img = imread('anhXam.tif');

thresholdImg = thresholdTransform(img, 100); % Ngưỡng T = 100

imshow(thresholdImg); title('Cắt ngưỡng');

**BÀI TẬP XỬ LÝ ẢNH**

**0. Viết chương trình lọc trung bình tuyến tính cho ảnh nhiễu với bộ lọc:**

**; *;***

***Yêu cầu bổ sung***: cho phép người sử dụng tự định nghĩa mặt nạ của bộ lọc Lowpass

function main()

% Đọc ảnh đầu vào

img = imread('anhXam.tif');

if size(img, 3) == 3

img = rgb2gray(img); % Chuyển ảnh về mức xám nếu là ảnh màu

end

% Hiển thị ảnh gốc

figure;

imshow(img);

title('Ảnh gốc');

% Các bộ lọc Lowpass được định nghĩa sẵn

filters = {

(1/9) \* [1 1 1; 1 1 1; 1 1 1], ...

(1/5) \* [0 1 0; 1 1 1; 0 1 0], ...

(1/32) \* [1 3 1; 3 16 3; 1 3 1], ...

(1/8) \* [1 1 1; 1 4 1; 1 1 1]

};

% Lọc ảnh với từng bộ lọc

for i = 1:length(filters)

filteredImg = averageFilter(img, filters{i});

figure;

imshow(filteredImg);

title(['Ảnh lọc với bộ lọc ', num2str(i)]);

end

% Người dùng tự định nghĩa mặt nạ Lowpass

disp('Bạn muốn tự định nghĩa mặt nạ của bộ lọc Lowpass.');

rows = input('Nhập số hàng của mặt nạ: ');

cols = input('Nhập số cột của mặt nạ: ');

disp('Nhập các giá trị của mặt nạ theo ma trận:');

customMask = zeros(rows, cols);

for r = 1:rows

for c = 1:cols

customMask(r, c) = input(['Nhập giá trị tại vị trí (', num2str(r), ',', num2str(c), '): ']);

end

end

scale = input('Nhập hệ số chia (ví dụ 1/9): ');

customMask = scale \* customMask;

% Lọc với mặt nạ tùy chỉnh

customFilteredImg = averageFilter(img, customMask);

figure;

imshow(customFilteredImg);

title('Ảnh lọc với mặt nạ tùy chỉnh');

end

function filteredImg = averageFilter(img, filterMask)

% Lọc trung bình tuyến tính với bộ lọc

% img: Ảnh đầu vào

% filterMask: Mặt nạ của bộ lọc

% Đảm bảo dữ liệu ảnh ở dạng double

img = double(img);

filterMask = double(filterMask);

% Kích thước mặt nạ

[kH, kW] = size(filterMask);

padH = floor(kH / 2);

padW = floor(kW / 2);

% Padding ảnh

paddedImg = zeros(size(img) + [2\*padH, 2\*padW]);

paddedImg(padH+1:end-padH, padW+1:end-padW) = img;

% Lấy giá trị tại biên bằng cách lặp lại pixel biên

for i = 1:padH

paddedImg(i, :) = paddedImg(padH+1, :);

paddedImg(end-i+1, :) = paddedImg(end-padH, :);

end

for j = 1:padW

paddedImg(:, j) = paddedImg(:, padW+1);

paddedImg(:, end-j+1) = paddedImg(:, end-padW);

end

% Kích thước ảnh

[m, n] = size(img);

filteredImg = zeros(m, n);

% Tích chập (convolution)

for i = 1:m

for j = 1:n

region = paddedImg(i:i + kH - 1, j:j + kW - 1);

filteredImg(i, j) = sum(sum(region .\* filterMask));

end

end

% Chuyển kết quả về dạng uint8

filteredImg = uint8(filteredImg);

end

**2.** Áp dụng bộ lọc để xác định đường biên trong ảnh mức xám với mặt nạ Sobel, Prewitt, Robert.

function edgeDetection()

% Đọc ảnh đầu vào

img = imread('anhXam.tif');

if size(img, 3) == 3

img = rgb2gray(img); % Chuyển ảnh màu về ảnh mức xám

end

% Hiển thị ảnh gốc

figure;

imshow(img);

title('Ảnh gốc');

% Áp dụng Sobel

sobelImg = applySobel(img);

figure;

imshow(sobelImg, []);

title('Đường biên - Bộ lọc Sobel');

% Áp dụng Prewitt

prewittImg = applyPrewitt(img);

figure;

imshow(prewittImg, []);

title('Đường biên - Bộ lọc Prewitt');

% Áp dụng Robert

robertImg = applyRobert(img);

figure;

imshow(robertImg, []);

title('Đường biên - Bộ lọc Robert');

end

function edgeImg = applySobel(img)

% Bộ lọc Sobel

Gx = [-1 0 1; -2 0 2; -1 0 1];

Gy = [-1 -2 -1; 0 0 0; 1 2 1];

% Tích chập với Sobel mask

edgeX = conv2(double(img), Gx, 'same');

edgeY = conv2(double(img), Gy, 'same');

% Tổng hợp gradient

edgeImg = sqrt(edgeX.^2 + edgeY.^2);

% Chuẩn hóa ảnh về khoảng [0, 255] để hiển thị đúng

edgeImg = uint8(255 \* mat2gray(edgeImg));

end

function edgeImg = applyPrewitt(img)

% Bộ lọc Prewitt

Gx = [-1 0 1; -1 0 1; -1 0 1];

Gy = [-1 -1 -1; 0 0 0; 1 1 1];

% Tích chập với Prewitt mask

edgeX = conv2(double(img), Gx, 'same');

edgeY = conv2(double(img), Gy, 'same');

% Tổng hợp gradient

edgeImg = sqrt(edgeX.^2 + edgeY.^2);

% Chuẩn hóa ảnh về khoảng [0, 255] để hiển thị đúng

edgeImg = uint8(255 \* mat2gray(edgeImg));

end

function edgeImg = applyRobert(img)

% Bộ lọc Robert

Gx = [1 0; 0 -1];

Gy = [0 1; -1 0];

% Tích chập với Robert mask

edgeX = conv2(double(img), Gx, 'same');

edgeY = conv2(double(img), Gy, 'same');

% Tổng hợp gradient

edgeImg = sqrt(edgeX.^2 + edgeY.^2);

% Chuẩn hóa ảnh về khoảng [0, 255] để hiển thị đúng

edgeImg = uint8(255 \* mat2gray(edgeImg));

end

**3. Viết hàm:**

* nhận 2 tham số đầu vào ***img*** và ***r***. Trong đó ***img*** là ảnh đầu vào, ***r*** là chỉ số hàng.
* tìm giá trị nhỏ nhất trên hàng thứ ***r*** của ảnh ***img*** (sử dụng vòng lặp ***for***)
* hàm trả về giá trị nhỏ nhất và chỉ số cột tương ứng

*hướng dẫn*:

**funtion [minValue, minColumn]=B1\_ex(img,r)**

-khởi tạo giá trị nhỏ nhất (minVal)

-cho vòng lặp chạy theo chỉ số cột size(img,2)

+kiểm tra nếu p.tử hiện tại < minVal

\*cập nhật minVal

\*cập nhật chỉ số cột (minColumn)

function [minValue, minColumn] = B1\_ex(img, r)

% B1\_ex: Tìm giá trị nhỏ nhất trên hàng thứ r và chỉ số cột tương ứng

% Input:

% img: Ảnh đầu vào (ma trận ảnh mức xám)

% r: Chỉ số hàng cần tìm giá trị nhỏ nhất

% Output:

% minValue: Giá trị nhỏ nhất trên hàng thứ r

% minColumn: Chỉ số cột của giá trị nhỏ nhất

% Lấy kích thước của ảnh

[rows, cols] = size(img);

% Kiểm tra nếu chỉ số hàng vượt quá kích thước ảnh

if r > rows || r < 1

error('Chỉ số hàng vượt quá kích thước của ảnh hoặc không hợp lệ');

end

% Khởi tạo giá trị nhỏ nhất và chỉ số cột

minValue = img(r, 1); % Giá trị đầu tiên trên hàng r

minColumn = 1; % Cột đầu tiên

% Duyệt qua từng cột trong hàng r

for c = 2:cols

if img(r, c) < minValue

% Cập nhật giá trị nhỏ nhất và chỉ số cột tương ứng

minValue = img(r, c);

minColumn = c;

end

end

end

img = imread('anhXam.tif'); % Đọc ảnh đầu vào

r = 5; % Chỉ số hàng cần tìm

[minValue, minColumn] = B1\_ex(img, r); % Gọi hàm tìm giá trị nhỏ nhất

fprintf('Giá trị nhỏ nhất trên hàng %d: %d tại cột %d\n', r, minValue, minColumn);

**4. Viết hàm:**

* nhận tham số đầu vào ***img*** –là ảnh đa cấp xám
* tìm giá trị trung bình cộng của ảnh img (sử dụng vòng lặp *for*)
* hàm trả về giá trị trung bình cộng và thời gian tính toán (dùng tic – bấm thời gian và toc – lấy thời gian )

*hướng dẫn*:

**function [averageVal, computationTime]=B2\_ex(img)**

-khởi tạo hàm đo thời gian (t=tic);

-quét toàn ảnh img tính tổng giá trị các pixel

-tính trung bình averageVal

-dừng đo thời gian computationTime=toc(t);

function [averageVal, computationTime] = B2\_ex(img)

% B2\_ex: Tính giá trị trung bình cộng của ảnh mức xám và thời gian tính toán

% Input:

% img: Ảnh đầu vào (ma trận ảnh mức xám, kiểu uint8)

% Output:

% averageVal: Giá trị trung bình cộng của ảnh

% computationTime: Thời gian tính toán (sử dụng tic và toc)

% Kiểm tra loại dữ liệu của ảnh

if ~isa(img, 'uint8')

error('Ảnh phải là kiểu uint8.');

end

% Bắt đầu đo thời gian

t = tic;

% Lấy kích thước ảnh

[rows, cols] = size(img);

% Khởi tạo tổng giá trị pixel

total = 0;

% Vòng lặp qua từng pixel để tính tổng giá trị

for r = 1:rows

for c = 1:cols

total = total + double(img(r, c)); % Chuyển về kiểu double để tránh lỗi làm tròn

end

end

% Tính giá trị trung bình cộng

averageVal = total / (rows \* cols);

% Dừng đo thời gian

computationTime = toc(t);

end

img = imread('anhXam.tif'); % Đọc ảnh đầu vào

[averageVal, computationTime] = B2\_ex(img); % Gọi hàm tính toán

fprintf('Giá trị trung bình cộng: %.5f\n', averageVal);

fprintf('Thời gian tính toán: %.5f giây\n', computationTime);

* **Tại sao cần cân bằng lược đồ?**
  + Trả lời: Để cải thiện độ tương phản của ảnh, làm nổi bật các chi tiết và phân phối mức xám đều hơn trên toàn bộ dải giá trị.
* **Lược đồ tích lũy (CDF) có vai trò gì?**
  + Trả lời: Lược đồ tích lũy giúp tính toán phân bố cường độ sáng tích lũy, từ đó ánh xạ các mức xám cũ sang mức xám mới.
* **Ưu điểm của cân bằng lược đồ?**
  + Trả lời: Cải thiện độ tương phản, tăng khả năng nhận diện chi tiết trong ảnh, đặc biệt hữu ích cho ảnh tối hoặc quá sáng.

 **Cân bằng lược đồ có phù hợp cho ảnh màu không?**

* Trả lời: Có, nhưng cần xử lý riêng từng kênh màu (R, G, B). Hoặc sử dụng không gian màu HSV, chỉ cân bằng lược đồ trên kênh V (Value).

 **So sánh cân bằng lược đồ toàn cục và cục bộ (CLAHE - Contrast Limited Adaptive Histogram Equalization)?**

* Trả lời: Cân bằng lược đồ toàn cục áp dụng trên toàn ảnh, trong khi CLAHE chia ảnh thành các ô nhỏ và áp dụng cân bằng lược đồ riêng từng ô để tránh over-enhancement.