TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Trần Hoàng Quân

BÁO CÁO BÀI TẬP LAB1

THỊ GIÁC MÁY TÍNH
CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN
Thầy Phạm Minh Hoàng

Tp. Hồ Chí Minh, tháng 4/2024

Mục lục

Chương 1

Mô tả chung

1.1 Mô tả chung

Chương trình được viết bằng ngôn ngữ C/C++ trên hệ điều hành Window 11. Có sử dụng thư viện OpenCV (phiên bản 4.9.0) (hàm *imread*, *imwrite*, và *imshow*); thư viện string để thực hiện xử lý chuỗi kí tự.

Các hàm thủ tục được viết trong class *Func*. Sau khi nhập lệnh vào cmd sẽ tiến hành thực thi chương trình. Trong trường hợp nếu lệnh nhập vào sai sẽ không thực hiện chương trình.

Command line sẽ có dạng giống như đề bài yêu cầu:

<ExcutableFile> -[method] <InputFilePath> <OutPutFilePath>

Chương trình sẽ thực hiện theo thứ tự: đọc ảnh, xử lý ảnh theo lệnh đã nhận, ghi ảnh vào đường dẫn đã nhận và hiện ảnh kết quả và ảnh đã đọc lên màn hình.

1.2 Một số phương pháp đã thực hiện

1.2.1 Duyệt ảnh

Ảnh sẽ được lưu trong class Mat của thư viện OpenCV. Để duyệt qua ảnh màu rgb ta sử dụng 3 vòng for và lệnh

$$Mat.at<kieu\_du\_lieu>(y,\ x)[z]$$

Trong đó

- kieu_du_lieu là kiểu dữ liệu được lưu trong Mat.
- y là tọa độ theo hàng của điểm ảnh.

- x là tọa độ theo cột của điểm ảnh.
- z là kênh màu (ảnh rgb có 3 kênh 0, 1, 2 lần lượt là blue, red, green).

Trong trường hợp ảnh xám thì chỉ có 1 kênh màu nên để duyệt qua các điểm ảnh ta sử dụng lệnh

Mat.at<kieu_du_lieu>(y, x)

1.2.2 Duyệt window

Giả sử ta có một window có kích thước k*k (Trong bài chỉ chấp nhận k lẻ). Khi đó ta sẽ tính cận dưới của nửa kích thước window là n = int(k/2). Ta sẽ dùng 2 vòng for lồng nhau và duyệt từ -n tới n (có nhận giá trị n). Để truy xuất giá trị của window trên ảnh sẽ là

Mat.at<kieu_du_lieu>(y+i+n, x+i+n)

Ngoài ra để đảm bảo về miền giá trị thì khi duyệt ảnh, ta không duyệt từ 0 đến (rows - 1) và (cols - 1) mà duyệt từ n đến (rows - n - 1) và (cols - n - 1). Các điểm nằm ở biên ta xem như giữ nguyên giá trị.

Chương 2

Các hàm đã cài đặt

2.1 saturate_cast

Hàm này nhận vào một giá trị double và trả về giá trị trong khoảng của kiểu dữ liệu được đưa vào template. Ví dụ kiểu uchar nhận giá trị trong khoảng từ 0 đến 255 thì nếu giá trị đưa vào bé hơn 0 thì trả về 0 và nếu lớn hơn 255 thì trả về 255.

2.2 insertionSort

Hàm này nhận vào 1 mảng, tiếng hành sắp xếp và trả về mảng được sắp xếp theo thứ tự tăng dần.

2.3 median

Hàm này nhận vào 1 mảng và trả về giá trị trung vị của mảng (sau khi được sắp xếp).

2.4 rgb2gray

Hàm này dùng để chuyển ảnh màu rgb sang ảnh xám. Đầu vào là địa chỉ của 2 biến lưu ảnh nguồn và ảnh đích. sau khi thực hiện xong kết quả được lưu trong ảnh đích. Ta duyệt qua các điểm ảnh, kênh màu và cập nhập kết quả cho điểm anh ở ảnh đích theo công thức là

$$blue * 0.11 + green * 0.59 + red * 0.3$$

Trong đóc blue, green, red lần lượt là giá trị của điểm ảnh ở kênh màu xanh lam, xanh lục, đỏ. Ảnh đích lúc này là ảnh xám.

2.5 brightness

Hàm này dùng để điều chỉnh độ sáng của ảnh. Đầu vào là Đầu vào là địa chỉ của 2 biến lưu ảnh nguồn, ảnh đích và một số nguyên c là số độ sáng cần điều chỉnh. Ta thực hiện duyệt qua từng điểm ảnh, từng kênh ảnh và cộng thêm giá trị c vào mỗi điểm ảnh ở 3 kênh ảnh (có dùng saturate_cast). Ảnh đích lúc này vẫn là ảnh màu rgb.

Hình 2.1: Kết quả sau khi tăng độ sáng lên 20

2.6 constrast

Hàm này dùng để điều chỉnh độ tương phản của ảnh. Đầu vào là Đầu vào là địa chỉ của 2 biến lưu ảnh nguồn, ảnh đích và một số nguyên c là số độ tương phản cần điều chỉnh. Ta thực hiện duyệt qua từng điểm ảnh,

từng kênh ảnh và nhân giá trị c vào mỗi điểm ảnh ở 3 kênh ảnh (có dùng saturate_cast). Ảnh đích lúc này vẫn là ảnh màu rgb.

Hình 2.2: Kết quả sau khi tăng độ tương phản lên 2

2.7 avg

Hàm này dùng để áp dụng filter trung bình vào ảnh. Đầu vào là Đầu vào là địa chỉ của 2 biến lưu ảnh nguồn, ảnh đích và một số nguyên k là kích thước của kernel. Ta thực hiện duyệt cửa sổ có kích thước bằng với kernel trên ảnh (phương pháp đã nói ở trên). Tại mỗi điểm ảnh, giá trị mới sẽ bằng trung bình các giá trị bên trong cửa sổ ảnh. Ảnh đích lúc này vẫn là ảnh màu rgb.

Hình 2.3: Kết quả sau khi áp dụng filter trung bình với kernel size = 3

2.8 med

Hàm này dùng để áp dụng filter trung vị vào ảnh. Đầu vào là Đầu vào là địa chỉ của 2 biến lưu ảnh nguồn, ảnh đích và một số nguyên k là kích thước của kernel. Ta thực hiện duyệt cửa sổ có kích thước bằng với kernel trên ảnh (phương pháp đã nói ở trên). Tại mỗi điểm ảnh, giá trị mới sẽ bằng trung vị (sử dụng hàm median đã được viết). các giá trị bên trong cửa sổ ảnh. Ảnh đích lúc này vẫn là ảnh màu rgb.

Hình 2.4: Kết quả sau khi áp dụng filter trung bình với kernel size = 3

2.9 gau

Hàm này dùng để áp dụng filter gaussian vào ảnh. Đầu vào là Đầu vào là địa chỉ của 2 biến lưu ảnh nguồn, ảnh đích và một số nguyên k là kích thước của kernel.Ta thực hiện tạo kernel bằng công thức của hàm gaussian

$$value(y, x) = \frac{e^{\frac{-r}{s}}}{2 * \sigma * \pi}$$

với

$$r = \sqrt{x^2 + y^2}$$

$$s = 2 * \sigma^2$$

Sau đó thực hiện chuẩn hóa bằng cách chia các giá trị cho tổng của chúng.

Ta thực hiện duyệt cửa sổ có kích thước bằng với kernel trên ảnh (phương pháp đã nói ở trên). Tại mỗi điểm ảnh, giá trị mới sẽ bằng tích

chập của window trên ảnh và gaussian kernel. Ảnh đích lúc này vẫn là ảnh màu rgb.

Hình 2.5: Kết quả sau khi áp dụng filter gaussian với kernel size = 3

2.10 sobel

Hàm này dùng để xác định cạnh bằng Sobel kernel 3 x 3. Đấu tiên ta phải chuyển ảnh rgb sang ảnh xám. Ta có 2 Sobel kernel là xSobel [1] $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ và ySobel [1] $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$.

Ta lần lượt duyệt cửa sổ và lấy tích chập với xSobel và ySobel sẽ cho được 2 ảnh dx và ảnh dy. Sau đó ta lấy tổng 2 ảnh lại, đặt giới hạn là chỉ lấy những điểm ảnh lớn hơn 15% của giá trị điểm ảnh lớn nhất để giảm bớt nhiễu.

Hình 2.6: Kết quả sau khi áp dụng Sobel kernel để xác định biên cạnh

2.11 laplace

Hàm này dùng để xác định cạnh bằng Laplacian kernel 3 x 3. Đấu tiên ta phải chuyển ảnh rgb sang ảnh xám. Ta có Laplacian kernel [1] là

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Ta lần lượt duyệt cửa sổ và lấy tích chập với Laplacian sẽ cho được ảnh kết quả.

Hình 2.7: Kết quả sau khi áp dụng Laplacian kernel để xác định biên cạnh

Tài liệu tham khảo

References

[1] Tran Thai Son. *Tim kiem bien canh va dac trung trong anh.* URL: https://courses.fit.hcmus.edu.vn/pluginfile.php/204766/mod_resource/content/1/CV_Week_2_VN_update_03Mar2024.pdf.