N22DCCN053 - Ngô Quang Minh - D22CQCN01-N

Homework 2

Bài 1. Xử lý ảnh “Mammogram.bin” – Nhị phân hóa và phát hiện biên xấp xỉ

I. Mục tiêu

- **Chuyển ảnh xám `Mammogram.bin` (256×256, 8-bit) thành ảnh nhị phân bằng phương pháp thresholding đơn giản.**
- **Áp dụng Approximate Contour Generation để tạo ảnh biên xấp xỉ của vùng mô.**
- **Đánh giá khả năng dùng chain code để biểu diễn biên chính.**

II. Lý thuyết

1. **Nhị phân hóa (Thresholding):**
   Ảnh xám f(x,y)f(x,y)f(x,y) được chuyển thành ảnh nhị phân g(x,y)g(x,y)g(x,y) theo ngưỡng
   Mục đích: tách nền tối và vùng mô sáng trong ảnh chụp X-quang.

2. **Phát hiện biên xấp xỉ (Approximate Contour):**
   Dựa trên phép co (Erosion) trong hình thái học.
   Biên xấp xỉ được tính bằng hiệu giữa ảnh nhị phân ban đầu và ảnh sau khi co

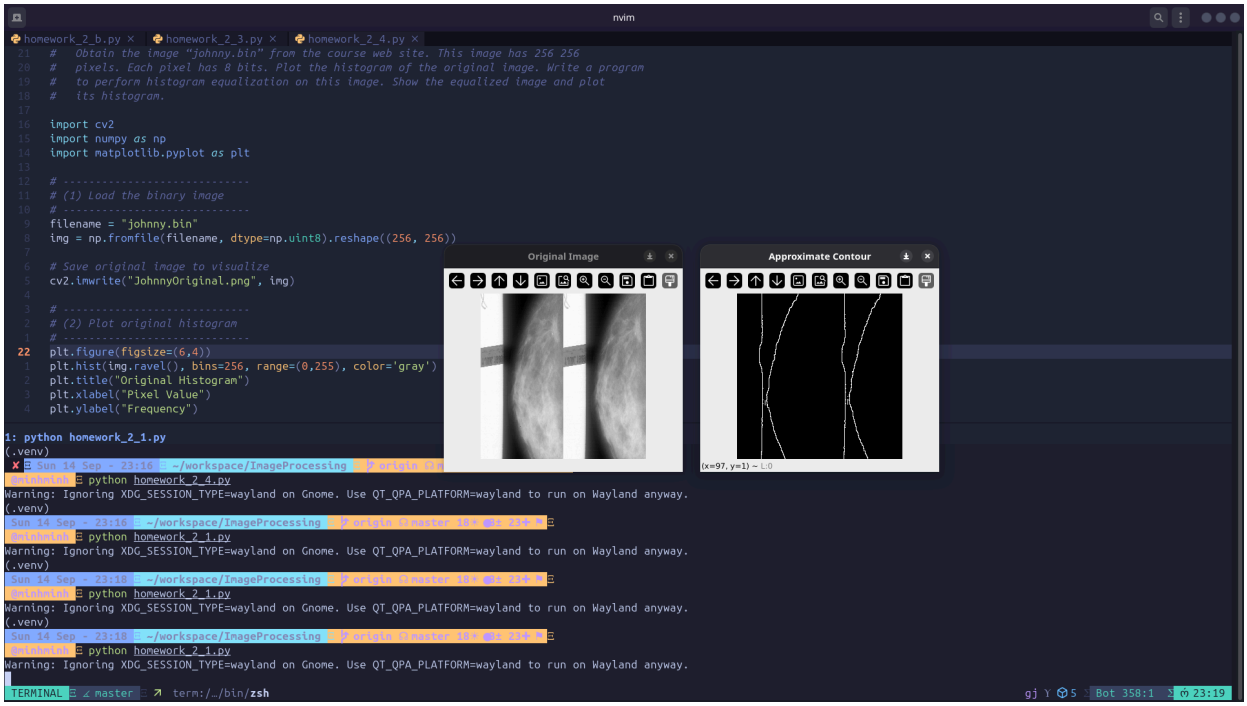
   Chain code:
   Là phương pháp mã hóa biên dạng theo chuỗi hướng (0–7) trong lân cận 8 điểm.
   Phù hợp để biểu diễn các đường biên liên tục và khép kín.

III. Các bước thực hiện

1. **Đọc ảnh nhị phân thô `Mammogram.bin` bằng `numpy.fromfile()` và reshape thành 256×256.**
2. **Chuẩn hóa dữ liệu về dạng 8-bit nếu cần.**
3. **Chọn ngưỡng T=70T = 70T=70 (xác định bằng quan sát histogram).**
4. **Tạo ảnh nhị phân: giá trị ≥70 → 255 (vùng mô), ngược lại → 0 (nền).**
5. **Thực hiện phép Erosion 3×3 bằng `cv2.erode()` để co vùng sáng.**
6. **Tính ảnh biên: `Contour = Binary - Eroded`.**
7. **Lưu và hiển thị kết quả:**
    - **Ảnh gốc**
    - **Ảnh nhị phân**
    - **Ảnh biên xấp xỉ**

IV. Kết luận

- **Ảnh biên thu được thể hiện rõ ranh giới vùng mô.**
- **Có thể dùng chain code để biểu diễn biên vì đường bao là liên tục, khép kín và đơn giá trị.**



Bài 2. Xử lý ảnh “lady.bin” – Kéo giãn tương phản toàn phần

I. Mục tiêu

- **Thực hiện Full-Scale Contrast Stretching trên ảnh `lady.bin`.**
- **So sánh histogram trước và sau khi kéo giãn.**

II. Lý thuyết

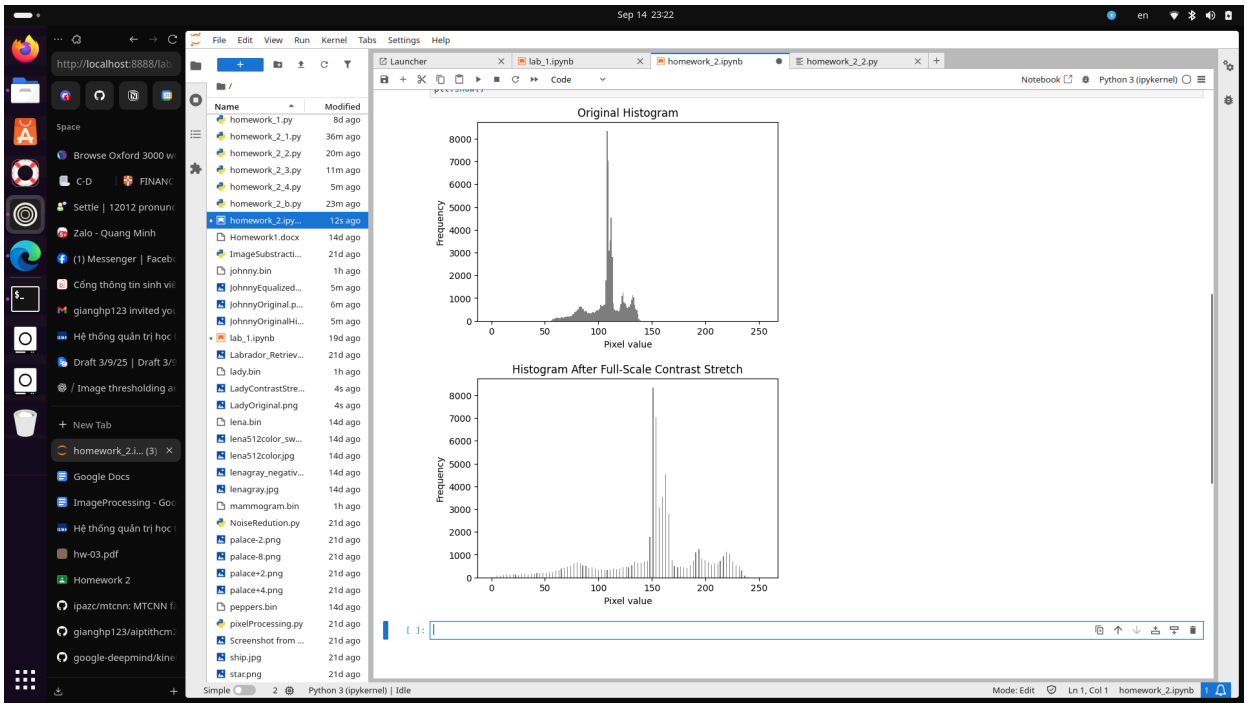
1. **Kéo giãn tương phản (Contrast Stretching):**
   Dùng để tăng độ tương phản bằng cách ánh xạ miền giá trị xám [rmin,rmax][r_{min}, r_{max}][rmin,rmax] → [0,255]:
   s=(r−rmin)(rmax−rmin)×255s = \frac{(r - r_{min})}{(r_{max} - r_{min})} \times 255s=(rmax−rmin)(r−rmin)×255
2. **Histogram:**
   Biểu đồ phân bố giá trị xám của ảnh.
   Sau khi kéo giãn, histogram được dàn đều hơn trên toàn miền 0–255.

III. Các bước thực hiện

1. **Đọc ảnh `lady.bin` (256×256, 8-bit).**
2. **Tính rminr_{min}rmin và rmaxr_{max}rmax của ảnh.**
3. **Áp dụng công thức kéo giãn tương phản để tạo ảnh mới.**
4. **Vẽ histogram trước và sau khi kéo giãn bằng `matplotlib`.**
5. **Lưu ảnh kết quả và biểu đồ histogram.**

IV. Kết luận

- **Ảnh sau khi kéo giãn trở nên sáng hơn, các vùng tối/sáng phân biệt rõ.**
- **Histogram trải đều từ 0 đến 255, chứng tỏ ảnh đã đạt tương phản tối đa.**



Bài 3. Xử lý ảnh “actontBin.bin” – Nhận dạng ký tự “T” bằng Binary Template Matching

I. Mục tiêu

- **Phát hiện các vị trí chứa ký tự “T” trong ảnh nhị phân.**
- **Ứng dụng thuật toán Binary Template Matching (M2 Measure).**

II. Cơ sở lý thuyết

1. **Binary Template Matching:**
   Dò tìm mẫu TTT bằng cách trượt cửa sổ mẫu trên ảnh.
   Mức độ tương đồng tại mỗi vị trí được đo bằng chỉ số M2:
   M2=1−∑(I⊕T)NM2 = 1 - \frac{\sum (I \oplus T)}{N}M2=1−N∑(I⊕T)
   Trong đó ⊕\oplus⊕ là phép XOR, NNN là tổng số điểm trong mẫu.
   Khi M2 gần 1 → vùng ảnh trùng khớp với mẫu.

2. **Ngưỡng (Thresholding):**
   Sau khi tính M2, ta ngưỡng ảnh để tạo ảnh nhị phân thể hiện vị trí có ký tự “T”.

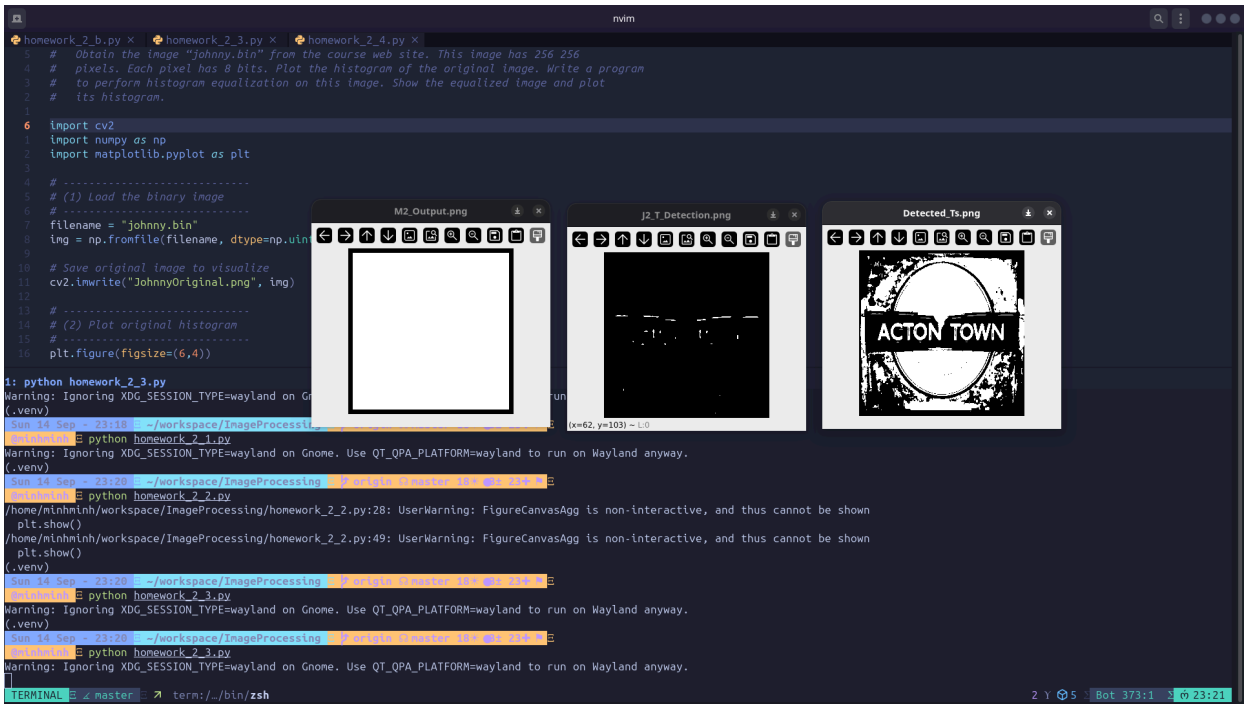
III. Các bước thực hiện

1. **Đọc ảnh nhị phân `actontBin.bin` và chuẩn hóa về giá trị 0–1.**
2. **Tạo mẫu chữ “T” kích thước 15×15 gồm thanh ngang trên và thanh dọc giữa.**
3. **Trượt mẫu trên ảnh:**
   - **Tại mỗi vị trí, tính phép XOR giữa mẫu và vùng con.**
   - **Tính giá trị M2 = 1 - (tổng XOR / N).**
4. **Gán kết quả M2 vào ma trận J1 tại vị trí trung tâm.**
5. **Dùng ngưỡng 0.8 để tạo ảnh nhị phân J2 (nơi có “T”).**

6. **Hiển thị và lưu các ảnh:**
    - **Ảnh M2 (mức tương đồng)**
    - **Ảnh J2 (kết quả nhị phân)**
    - **Ảnh đánh dấu các chữ "T" phát hiện được**

IV. Kết luận

- **Phương pháp template matching cho kết quả chính xác khi kích thước mẫu phù hợp.**
- **M2 cao (≈1) tại vị trí có chữ "T", thấp hơn tại vùng khác.**
- **Có thể mở rộng để phát hiện các ký tự khác bằng cách thiết kế template tương ứng.**



Bài 4. Xử lý ảnh "johnny.bin" – Cân bằng Histogram

I. Mục tiêu

- **Áp dụng Histogram Equalization cho ảnh `johnny.bin` để cải thiện độ tương phản.**
- **So sánh histogram trước và sau khi cân bằng.**

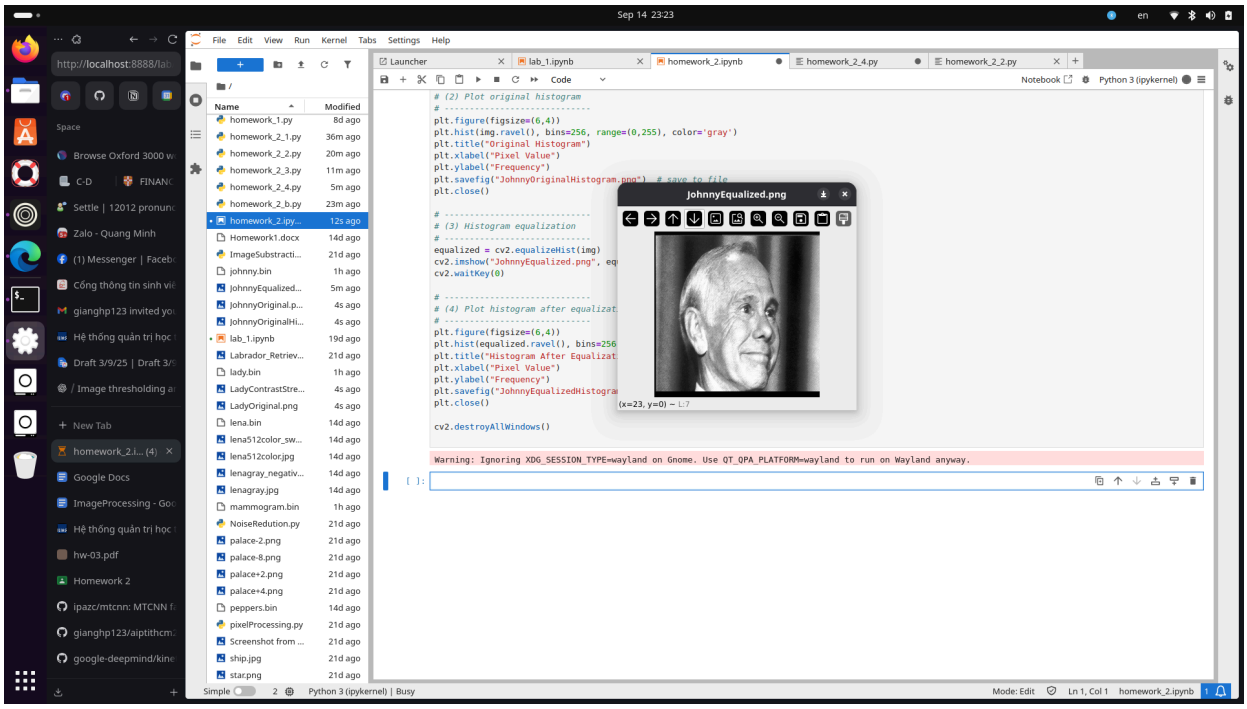
II. Cơ sở lý thuyết

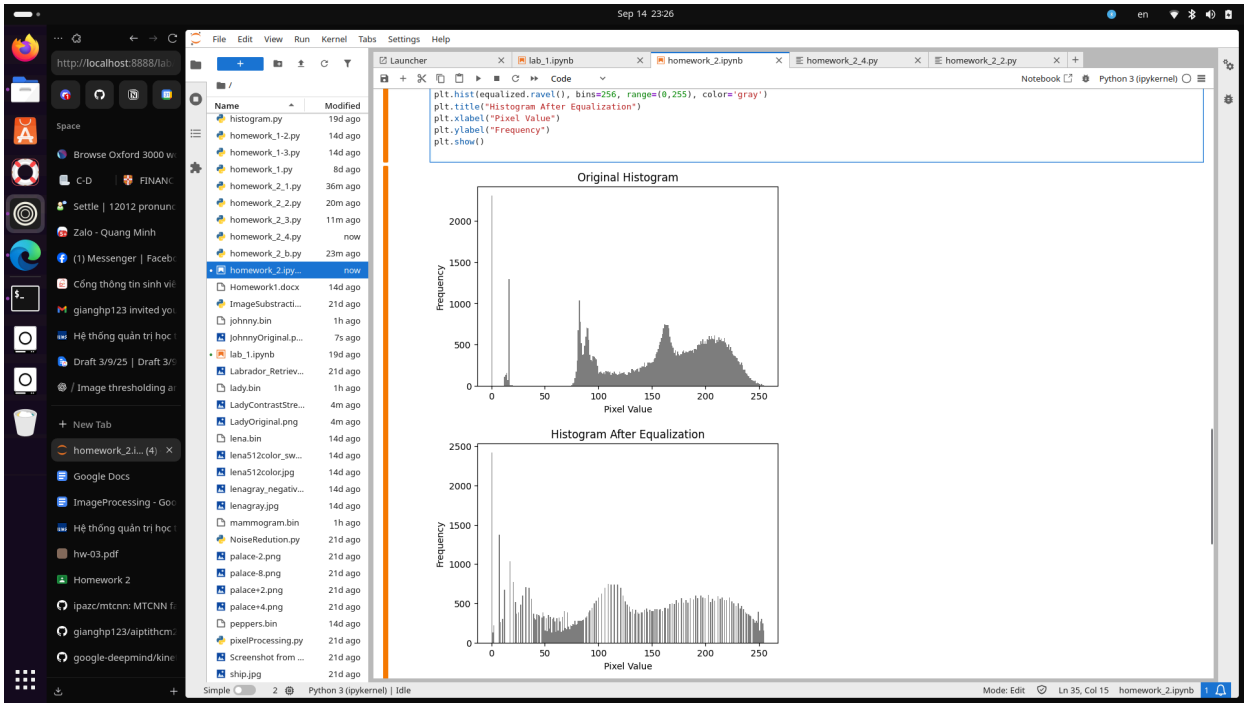
1. **Histogram Equalization:**
   Là kỹ thuật biến đổi giá trị xám để phân bố histogram đồng đều hơn, giúp ảnh có độ tương phản cao hơn.
   Công thức biến đổi:
   s=T(r)=(L−1)∑j=0rpr(rj)s = T(r) = (L-1) \sum_{j=0}^{r} p_r(r_j)s=T(r)=(L−1)j=0∑rpr(rj)
   với pr(rj)p_r(r_j)pr(rj) là xác suất xuất hiện mức xám rjr_jrj, L=256L=256L=256.
2. **Lợi ích:**
   - **Tăng cường chi tiết ở vùng tối/sáng.**
   - **Cân bằng phân bố mức xám trên toàn ảnh.**

III. Các bước thực hiện

1. **Đọc ảnh `johnny.bin` (256×256, 8-bit).**
2. **Vẽ histogram ban đầu bằng `matplotlib`.**
3. **Dùng hàm `cv2.equalizeHist()` để cân bằng histogram.**
4. **Hiển thị ảnh đã cân bằng và vẽ lại histogram sau khi xử lý.**
5. **Lưu các ảnh kết quả:**
   - **`JohnnyOriginal.png`**

- **JohnnyEqualized.png**
- **JohnnyOriginalHistogram.png**
- **JohnnyEqualizedHistogram.png**





IV. Kết luận

- **Histogram sau khi cân bằng phân bố đều hơn, độ tương phản ảnh cải thiện rõ rệt.**
- **Phù hợp cho các ảnh có dải cường độ hẹp hoặc bị tối/sáng cục bộ.**