

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**XÁC SUẤT VÀ THỐNG KÊ ỨNG DỤNG CHO  
CÔNG NGHỆ THÔNG TIN**

**BÀI TIỂU LUẬN GIỮA KÌ**

*Người hướng dẫn:* **THẦY MAI DUY TÂN**

*Người thực hiện:* **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**

**Lớp : 22050201**

**Khoá : 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**XÁC SUẤT VÀ THỐNG KÊ ỨNG DỤNG CHO  
CÔNG NGHỆ THÔNG TIN**

**BÀI TIỂU LUẬN GIỮA KÌ**

*Người hướng dẫn:* **THẦY MAI DUY TÂN**

*Người thực hiện:* **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**

**Lớp : 22050201**

**Khoá : 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

## LỜI CẢM ƠN

Để hoàn thành đề tài, bài tiểu luận Python Giữa kì I năm học 2023 - 2024 môn Xác suất và thống kê ứng dụng cho Công nghệ thông tin lần này.

Lời đầu tiên, em xin gửi lời cảm ơn chân thành của mình đến Ban giám hiệu trường Đại học Tôn Đức Thắng, quý thầy cô giáo giảng viên của khoa Công nghệ thông tin vì đã tạo điều kiện tốt nhất về cơ sở vật chất và hệ thống thư viện hiện đại, đa dạng đầy đủ các loại sách giúp em có thể trau dồi kiến thức, tìm kiếm thông tin, tư liệu để hoàn thành bài Tiểu luận này. Đây thực sự là một cơ hội tuyệt vời giúp cho nghề nghiệp của em trong tương lai rộng mở hơn khi được tiếp xúc với sự hiện đại, nhiều kiến thức.

Bên cạnh đó, em cũng nhận được rất nhiều sự giúp đỡ tận tình của quý thầy cô. Thầy cô giảng viên là những người hướng dẫn và truyền cảm hứng cho em trong học tập trong suốt thời gian qua ở môi trường đại học.

Với lòng biết ơn sâu sắc và vô cùng đặt biệt, em xin gửi lời cảm ơn của mình đến thầy Mai Duy Tân– Giảng viên thực hành môn Xác suất và thống kê ứng dụng cho Công nghệ thông tin, người đã luôn đồng hành, dẫn dắt và giúp đỡ em trong việc hoàn thành bài Tiểu luận. Từ những kiến thức thầy đã giảng dạy trên những giờ học để em có thể áp dụng những kiến thức đã được vào bài Tiểu luận lần này. Một lần nữa em xin chân thành cảm ơn thầy vì sự hỗ trợ của thầy ạ.

Vì kiến thức bản thân còn hạn chế nên trong quá trình giải quyết vấn đề nên khi hoàn thành bài tiểu luận lần này em không tránh khỏi những sai sót, em kính mong nhận được những lời nhận xét, đóng góp ý kiến từ thầy ạ.

Lời cuối cùng, em xin gửi lời cảm ơn chân thành và gửi ngàn lời chúc tốt đẹp đến với quý thầy cô khi đã tạo cơ hội cho chúng em nâng cấp kiến thức trong môn học này.

## **ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của thầy Mai Duy Tân;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 03 tháng 11 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*



*Huỳnh Hoàng Tiến Đạt*

## **PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

### **Phần xác nhận của GV hướng dẫn**

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày      tháng      năm  
(kí và ghi họ tên)

### **Phần đánh giá của GV chấm bài**

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày      tháng      năm  
(kí và ghi họ tên)

## **TÓM TẮT**

Tiểu luận này gồm có **04** chương:

**Chương 1: Statistics library in Python (Khảo sát các hàm trong thư viện Statistics trong Python).**

**Chương 2: Histogram equalization algorithm (Khảo sát các thuật toán cân bằng Histogram và đối sánh Histogram trong xử lý ảnh).**

**Chương 3: Histogram matching algorithm (Thuật toán đối sánh biểu đồ)**

**Chương 4: Implementation (Hiện thực code)**

# MỤC LỤC

|  |    |
|--|----|
| LỜI CẢM ƠN .....   | 1  |
| PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN .....                         | 3  |
| TÓM TẮT .....  | 4  |
| DANH SÁCH CÁC HÌNH ẢNH VÀ BIỂU ĐỒ.....                                 | 7  |
| LỜI MỞ ĐẦU .....   | 7  |
| CHƯƠNG 1: KHẢO SÁT CÁC HÀM TRONG THƯ VIỆN STATISTICS TRONG PYTHON..... | 9  |
| 1.1 Thư viện Statistic trong Python:.....                              | 9  |
| 1.2 Các hàm thường sử dụng trong Module Statistic trong Python: .....  | 9  |
| 1.2.1 Hàm mean(): .....  | 9  |
| 1.2.2 Hàm fmean():.....  | 10 |
| 1.2.3 Hàm geometric_mean(): .....                                      | 11 |
| 1.2.4 Hàm harmonic_mean(): .....                                       | 11 |
| 1.2.5 Hàm median(): .....  | 12 |
| 1.2.6 Hàm median_low(): .....  | 13 |
| 1.2.7 Hàm median_high(): .....   | 14 |
| 1.2.8 Hàm median_grouped(): .....                                      | 14 |
| 1.2.9 Hàm mode(): .....  | 15 |
| 1.2.10 Hàm multimode(): .....  | 16 |
| 1.2.11 Hàm quantiles(): .....  | 16 |
| 1.2.12 Hàm pstdev(): .....   | 17 |
| 1.2.13 Hàm pvariance(): .....  | 17 |
| 1.2.14 Hàm stdev(): .....  | 18 |
| 1.2.15 Hàm variance(): .....   | 19 |
| 1.2.16 Hàm covariance(): .....   | 19 |
| 1.2.17 Hàm correlation(): .....  | 20 |
| 1.2.18 Hàm linear_regression(): .....                                  | 20 |
| CHƯƠNG 2: HISTOGRAM EQUALIZATION.....                                  | 22 |
| 2.1 Giải thích thuật toán Histogram equalization:.....                 | 22 |
| 2.2 Điều kiện: .....   | 22 |
| 2.3 Phương pháp của thuật toán Histogram equalization:.....            | 23 |
| 2.4 Minh họa:.....   | 24 |
| 2.6 Đánh giá: .....  | 25 |

|  |    |
|--|----|
| CHƯƠNG 3: HISTOGRAM MATCHING ALGORITHMS.....                                   | 26 |
| 3.1 Giải thích thuật toán Histogram Matching: .....                            | 26 |
| 3.2 Điều kiện: .....   | 26 |
| 3.3 Phương pháp của thuật toán Histogram Matching Algorithms:.....             | 26 |
| 3.4 Minh họa:.....   | 27 |
| 3.5 Đánh giá: .....  | 27 |
| CHƯƠNG 4: IMPLEMENTATION .....   | 28 |
| 4.1 Code cho ví dụ minh họa Chương 2:.....                                     | 28 |
| 4.1.1 Input – Output:.....   | 29 |
| 4.1.2 Phân tích code: .....  | 30 |
| 4.1.2.1 Nhập thư viện: .....   | 30 |
| 4.1.2.2 Thực hiện cân bằng biểu đồ tần số: .....                               | 31 |
| 4.1.2.3 Tính hàm phân phối tích lũy của biểu đồ tần số.....                    | 31 |
| 4.1.2.4 Tạo mảng có cùng kích thước với ảnh “img” để lưu ảnh đã cân bằng:..... | 32 |
| 4.1.2.5 Đọc đường dẫn hình ảnh đầu vào (input): .....                          | 32 |
| 4.1.2.6 Điều chỉnh lại kích thước của ảnh: .....                               | 32 |
| 4.1.2.7 Cân bằng biểu đồ tần số:.....  | 32 |
| 4.1.2.8 Lưu ảnh ra tệp:.....   | 33 |
| 4.1.2.9 Tính biểu đồ tần số của ảnh gốc và ảnh đã cân bằng: .....              | 33 |
| 4.1.2.10 Tạo lại khung: .....  | 33 |
| 4.1.2.11 Hiển thị ảnh vào khung và sắp xếp thứ tự của chúng:.....              | 34 |
| 4.2 Code cho ví dụ minh họa cho chương 3:.....                                 | 35 |
| 4.2.1 Input – Output: .....  | 37 |
| 4.2.2 Phân tích code: .....  | 38 |
| 4.2.2.1 Nhập thư viện: .....   | 38 |
| 4.2.2.2 Tính histogram của ảnh: .....  | 38 |
| 4.2.2.3 Tính tích lũy histogram của ảnh: .....                                 | 39 |
| 4.2.2.4 Áp dụng thuật toán histogram matching: .....                           | 39 |
| 4.2.2.5 Thêm ảnh input đầu vào và ảnh tham chiếu: .....                        | 40 |
| 4.2.2.6 Áp dụng thuật toán Histogram Matching: .....                           | 40 |
| 4.2.2.7 Tạo khung: .....   | 40 |
| 4.2.2.8 Vẽ biểu đồ của ảnh gốc, ảnh tham chiếu và kết quả:.....                | 41 |
| 4.2.2.9 Lưu ảnh đã cân bằng ra tệp:.....                                       | 41 |
| TÀI LIỆU THAM KHẢO (TIẾNG VIỆT).....   | 42 |
| TÀI LIỆU THAM KHẢO (TIẾNG ANH) .....   | 42 |
| SELF-EVALUATION FORM.....  | 43 |



## **DANH SÁCH CÁC HÌNH ẢNH VÀ BIỂU ĐỒ**

1. Hình 1: Các bước thực hiện của thuật toán Histogram Equalization
2. Hình 2: Ví dụ trực quan của thuật toán Histogram Equalization
3. Hình 3: Công thức tính toán
4. Hình 4: Hình ảnh ví dụ cho thuật toán Histogram Equalization tương ứng với các ảnh dark, light, low – contrast và high contrast
5. Hình 5: Hình ảnh bài tập minh họa gồm hình ảnh và biểu đồ của nó khi sử dụng thuật toán Histogram Equalization
6. Hình 6: Biểu đồ của thuật toán Histogram Matching
7. Hình 7: Hình ảnh bài tập minh họa gồm hình ảnh gốc, ảnh tham chiếu và biểu đồ của nó khi sử dụng thuật toán Histogram Matching
8. Hình 8: Hình ảnh input đầu vào để áp dụng thuật toán
9. Hình 9: Hình ảnh bài tập minh họa gồm hình ảnh và biểu đồ của nó khi sử dụng thuật toán Histogram Equalization
10. Hình 10: Hình ảnh input đầu vào (ảnh gốc)
11. Hình 11: Hình ảnh thứ hai input đầu vào (ảnh tham chiếu)
12. Hình 12: Hình ảnh bài tập minh họa gồm hình ảnh gốc, ảnh tham chiếu và biểu đồ của nó khi sử dụng thuật toán Histogram Matching

## LỜI MỞ ĐẦU

Trong thời đại công nghệ số phát triển mạnh mẽ, nhu cầu và đời sống của con người ngày càng được cải thiện. Con người có xu hướng trải nghiệm, tìm kiếm những điều mới mẻ, tận hưởng cuộc sống hiện đại, ứng dụng những công nghệ hiện đại như robot,... Điều này dẫn đến việc ứng dụng, sử dụng ngôn ngữ lập trình để tạo ra các trang web, trò chơi, ứng dụng là điều không thể tránh khỏi. Thậm chí là các loại AI chatbot để tư vấn những thắc mắc của con người chúng ta.

Trong số các loại ngôn ngữ lập trình như C, C#, C++, Pascal, ... thì Python là một ngôn ngữ lập trình phổ biến và được sử dụng rộng rãi trong nhiều lĩnh vực. Nó được đánh giá cao về tính linh hoạt và dần trở thành công cụ quan trọng trong việc giải quyết các bài toán liên quan đến khoa học cụ thể như khoa học dữ liệu, trí tuệ nhân tạo AI, xử lý ảnh, kỹ thuật số,... Ngoài ra, Python còn là một công cụ hữu ích, mang lại nhiều lợi ích cho người dùng khi nó được sử dụng để tạo ra các ứng dụng web, desktop, mobile, chatbot và một vài ứng dụng phần mềm khác.

# CHƯƠNG 1: KHẢO SÁT CÁC HÀM TRONG THƯ VIỆN STATISTICS TRONG PYTHON

## 1.1 Thư viện Statistic trong Python:

- Thư viện statistics trong Python cung cấp các hàm để tính toán các thống kê toán học của dữ liệu số. Thư viện này không nhằm cạnh tranh với các thư viện bên thứ ba như NumPy, SciPy, hoặc các gói thống kê chuyên nghiệp đầy đủ tính năng như Minitab, SAS và Matlab. Nó hướng đến mức độ của máy tính biểu đồ và máy tính khoa học.

- Nguồn tài liệu tham khảo: <https://docs.python.org/3/library/statistics.html#>

## 1.2 Các hàm thường sử dụng trong Module Statistic trong Python:

### 1.2.1 Hàm mean():

- Công dụng: Hàm mean() được sử dụng để tính giá trị trung bình cộng số học của các số trong danh sách.
- Cú pháp: statistics.mean(data)
- Tham số đầu vào: list\_of\_numbers(danh sách) – một danh sách các giá trị số, cũng có thể là bộ dữ liệu hoặc các giá trị số có thể lặp lại khác chứa các giá trị số (gồm số nguyên, số thực và các đối tượng số học như 'Fraction' và 'Decimal').
- Kết quả trả về: Giá trị trung bình cộng của danh sách các số bằng cách lấy tổng các giá trị số trong dãy chia cho số lượng các số trong dãy.
- Ví dụ minh họa cho hàm statistic.mean() trong module statistic.

```
Chapter1_StatisticFunction.py > ...
1  import statistics
2  list_of_numbers = [1, 2, 3, 4, 5]
3  mean = statistics.mean(list_of_numbers)
4  print('Mean is:')
5  print(mean) #Output: 3
6
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
3
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Mean is:
3
```

- Nếu giá trị đầu vào là rỗng thì kết quả trả về sẽ là “StatisticsError”.
- Ghi chú: Giá trị trung bình cộng trong hàm mean() bị ảnh hưởng bởi giá trị ngoại lai vì khi đó giá trị ngoại lai sẽ khiến cho giá trị trung bình cộng bị chênh lệch mạnh.
- Ví dụ minh họa cho trường hợp này:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 list_of_numbers = [1, 2, 3, 4, 5, 1000]
3 mean = statistics.mean(list_of_numbers)
4 print("Mean: ")
5 print(mean) # Output: 169.16666666666666
6
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Mean:
169.16666666666666
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Mean:
169.16666666666666
```

### 1.2.2 Hàm fmean():

- Công dụng: Chuyển đổi dữ liệu thành số thực và tính trung bình cộng
- Cú pháp: statistics.fmean(list\_of\_numbers, weights = None)
- Tham số đầu vào:
  - o list\_of\_numbers(danh sách) – một danh sách các giá trị số, cũng có thể là bộ dữ liệu hoặc các giá trị số có thể lặp lại.
  - o weights(tùy chọn) là danh sách các trọng số tương ứng với từng phần tử trong list\_of\_numbers, đại diện cho mức độ quan trọng khi thực hiện phép tính. Nếu không cung cấp hoặc bỏ trống thì sẽ mặc định là tất cả các phần tử trong list\_of\_numbers là giống nhau (cùng trọng số).
- Kết quả trả về: Giá trị trung bình cộng của các giá trị số ở dạng số thực
- Ví dụ minh họa cho hàm statistic.fmean() khi không có trọng số:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 list_of_numbers = [1.2, 2.5, 3.4]
3 fmean = statistics.fmean(list_of_numbers)
4 print("Fmean: ")
5 print(fmean)
6
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Fmean:
2.3666666666666667
```

- Ví dụ minh họa cho hàm `statistics.fmean()` khi có trọng số:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 list_of_numbers = [1.2, 2.5, 3.4]
3 weights = [0.2, 0.3, 0.22]
4 fmean = statistics.fmean(list_of_numbers, weights)
5 print("Fmean: ")
6 print(fmean)
7
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Fmean:
2.4138888888888888
```

- Nếu giá trị đầu vào rỗng thì kết quả trả về là “`StatisticsError`”
- Nếu giá trị trọng số không cùng độ dài với dữ liệu thì sẽ ra lỗi “`ValueError`”

### 1.2.3 Hàm `geometric_mean()`:

- Công dụng: Chuyển dữ liệu thành dạng số thực và tính giá trị trung bình hình học
- Cú pháp: `statistics.geometric_mean(list_of_numbers)`
- Tham số đầu vào: Là một chuỗi hoặc tập hợp các giá trị số
- Kết quả trả về: Giá trị trung bình hình học của các giá trị số bằng cách chuyển đổi chúng thành dạng số thực và tính bằng cách lấy căn bậc n của tích các giá trị số.
- Ví dụ minh họa cho hàm `statistics.geometric_mean()`:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 list_of_numbers = [1, 2, 3]
3 geometric_mean = statistics.geometric_mean(list_of_numbers)
4 print("Geometric_mean: ")
5 print(geometric_mean)
6
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Geometric_mean:
1.8171205928321397
```

- Nếu dữ liệu đầu vào rỗng hoặc giá trị bằng không, giá trị âm thì sẽ đưa ra ngoại lệ “`StatisticsError`”

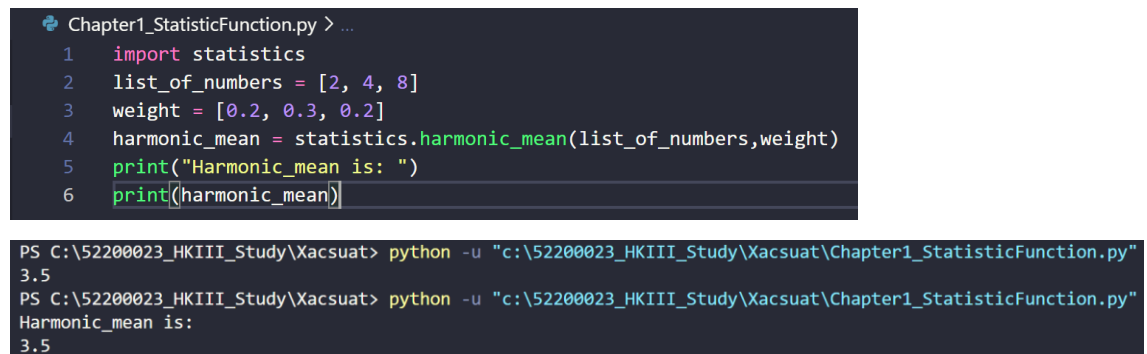
### 1.2.4 Hàm `harmonic_mean()`:

- Công dụng: tính giá trị trung bình hình học của một tập dữ liệu.

- Cú pháp: `statistics.harmonic_mean(list_of_numbers, weight)`
- Tham số đầu vào:
  - o `list_of_numbers` là một chuỗi hoặc một tập hợp các giá trị số
  - o `weight` là danh sách các trọng số tương ứng với từng phần tử trong `list_of_numbers`. Nếu không cung cấp hoặc bỏ trống thì sẽ mặc định là tất cả các phần tử trong `list_of_numbers` là giống nhau (cùng trọng số).
- Kết quả trả về: Giá trị trung bình hình học của tập dữ liệu. Cụ thể, giá trị trả về như sau:

$$H = n / ((\frac{1}{x_1}) + (\frac{1}{x_2}) + \dots + (\frac{1}{x_n}))$$

- Ví dụ minh họa cho hàm `statistics.harmonic_mean()` trong module `statistics`.



```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 list_of_numbers = [2, 4, 8]
3 weight = [0.2, 0.3, 0.2]
4 harmonic_mean = statistics.harmonic_mean(list_of_numbers, weight)
5 print("Harmonic_mean is: ")
6 print(harmonic_mean)

PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
3.5
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Harmonic_mean is:
3.5
```

- Nếu giá trị đầu vào rỗng hoặc giá trị âm thì trả về ngoại lệ “`StatisticsError`”

### 1.2.5 Hàm `median()`:

- Công dụng: là đại lượng đo lường trung vị để thể hiện giá trị ở giữa của một tập dữ liệu
- Cú pháp: `statistics.median(list_of_numbers)`
- Tham số truyền vào: `list_of_numbers` (danh sách) – một danh sách các giá trị số, cũng có thể là bộ dữ liệu hoặc các giá trị số có thể lặp lại khác.
- Kết quả trả về: Giá trị trung vị của dữ liệu
- Khi data point là số lẻ, hàm sẽ trả về ở giữa
- Ví dụ minh họa cho hàm `statistics.median()`:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 data = [10, 30, 20, 1000, 50]
3 median = statistics.median(data)
4 print("Median: ")
5 print(median)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Median:
3
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Median:
30
```

- Khi data point là số chẵn, trả về trung bình cộng của hai điểm ở giữa.
- Ví dụ minh họa:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 data = [10, 30, 20, 1000, 50, 99]
3 median = statistics.median(data)
4 print("Median: ")
5 print(median)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Median:
30
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Median:
40.0
```

- Nếu dữ liệu đầu vào rỗng, hàm trả về giá trị ngoại lệ “StatisticsError”.

### 1.2.6 Hàm median\_low():

- Công dụng: tính giá trị trung vị thấp của một tập dữ liệu
- Cú pháp: statistics.median\_low(list\_of\_numbers)
- Tham số đầu vào: list\_of\_numbers(danh sách) – một danh sách các giá trị số, cũng có thể là bộ dữ liệu hoặc các giá trị số có thể lặp lại khác.
- Kết quả trả về: Giá trị trung vị thấp và luôn thuộc tập dữ liệu.
- Khi data point là số lẻ, hàm trả về điểm giữa. Khi data point là số chẵn, hàm sẽ trả về giá trị nhỏ hơn trong hai giá trị ở giữa.
- Nếu dữ liệu đầu vào rỗng, hàm sẽ đưa ra một ngoại lệ “StatisticsError”
- Ví dụ minh họa cho hàm statistics.median\_low()

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 data = [10, 30, 20, 40, 50, 60]
3 median_low = statistics.median_low(data)
4 print("Median Low is: ")
5 print(median_low)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Median Low is:
30
```

### 1.2.7 Hàm median\_high():

- Công dụng: Tính giá trị trung vị cao của một tập dữ liệu
- Cú pháp: statistics.median\_high(list\_of\_numbers)
- Tham số đầu vào: list\_of\_numbers(danh sách) – một danh sách các giá trị số, cũng có thể là bộ dữ liệu hoặc các giá trị số có thể lặp lại khác.
- Kết quả trả về: Giá trị trung vị cao và luôn nằm trong tập dữ liệu
- Khi data point là dữ liệu số lẻ, hàm sẽ trả về điểm ở giữa. Khi là số chẵn thì hàm sẽ trả về giá trị lớn hơn trong hai giá trị ở giữa.
- Nếu dữ liệu đầu vào rỗng, hàm sẽ đưa ra một ngoại lệ “StatisticsError”
- Ví dụ minh họa cho hàm statistics.median\_high():

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 data = [10, 30, 20, 40, 50, 60]
3 median_high = statistics.median_high(data)
4 print("Median High is: ")
5 print(median_high)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Median High is:
40
```

### 1.2.8 Hàm median\_grouped():

- Công dụng: Tính giá trị trung vị của một tập dữ liệu liên tục được nhóm bằng phương pháp nội suy
- Cú pháp: statistics.median\_grouped(list\_of\_numbers, interval = 1)
- Tham số đầu vào:
  - o list\_of\_numbers: Một chuỗi số (danh sách, bộ, mảng) hoặc dữ liệu lặp lại có chứa giá trị trung tâm của mỗi lớp hoặc khoảng nào đó.



- interval: Khoảng cách giữa các lớp và mặc định là 1. Thay đổi khoảng đồng nghĩa thay đổi quá trình nội suy cho phép tính.

- Kết quả trả về: Giá trị trung vị của dữ liệu liên tục được nhóm với:

$$\bullet \text{ Gmedian} = l + \left( \frac{n/2 - cf}{f} \right) * h$$

- Ví dụ minh họa cho hàm `statistics.median_grouped()`:

```
Chapter1_StatisticFunction.py > ...
1  import statistics
2  data = [10, 30, 20, 40, 50, 60, 40, 30, 10]
3  median_grouped = statistics.median_grouped(data,2)
4  print("Median Grouped is: ")
5  print(median_grouped)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Median Grouped is:
30.5
```

### 1.2.9 Hàm `mode()`:

- Công dụng: xác định điểm dữ liệu xuất hiện nhiều nhất trong một tập dữ liệu hoặc một danh sách các số.
- Cú pháp: `statistics.mode(list_of_numbers)`
- Tham số đầu vào: một danh sách, một bộ hoặc bất kỳ dạng dữ liệu có thể lặp lại
- Kết quả trả về: điểm dữ liệu phổ biến nhất từ dữ liệu rời rạc cũng như dữ liệu không phải số hoặc danh nghĩa
- Ví dụ minh họa cho `statistics.mode()`

```
1  import statistics
2  data = [10, 30, 20, 40, 50, 60, 40, 30, 10]
3  data_strings = ["comic", "book", "book", "comic", "paper", "comic", "comic"]
4  result1 = statistics.mode(data)
5  result2 = statistics.mode(data_strings)
6  print("Mode is: ")
7  print(result1)
8  print("Mode with strings are: ")
9  print(result2)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Mode is:
10
Mode with strings are:
comic
```

- Nếu có nhiều tần số, hàm sẽ trả về giá trị đầu tiên mà hàm gặp phải

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 data = [10, 30, 20, 40, 50, 60, 40, 30, 10, 20, 50, 10, 40]
3 result1 = statistics.mode(data)
4 print("Mode is: ")
5 print(result1)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Mode is:
10
```

- Nếu dữ liệu đầu vào rỗng, hàm sẽ đưa ra một ngoại lệ “StatisticsError”

### 1.2.10 Hàm multimode():

- Công dụng: tìm các giá trị xuất hiện nhiều nhất theo thứ tự gặp đầu tiên
- Cú pháp: statistics.multimode(data)
- Tham số đầu vào: một danh sách hoặc một đối tượng có thể lặp lại
- Kết quả trả về: một danh sách chứa các giá trị xuất hiện phổ biến nhất trong dữ liệu theo thứ tự gặp đầu tiên. Nếu có nhiều hơn một giá trị, hàm trả về tất cả giá trị. Nếu dữ liệu rỗng thì hàm trả về danh sách rỗng.
- Ví dụ minh họa cho hàm statistics.multimode()

```
1 import statistics
2 data = 'dddooooooooohhhhhhhhhh iiiiii'
3 result1 = statistics.multimode(data)
4 print("Multimode is: ")
5 print(result1)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Multimode is:
h
```

### 1.2.11 Hàm quantiles():

- Công dụng: Tính các phân vị của một tập dữ liệu, chia thành n khoảng liên tục có xác suất gần bằng nhau
- Cú pháp: statistics.quantiles(ldata, \*, n = 4, method = 'exclusive')
- Tham số đầu vào:
  - o ldata: một danh sách hoặc dãy dữ liệu mẫu
  - o n: Số khoảng hoặc điểm cắt
  - o method: phương pháp tính phân vị. Mặc định là 'exclusive' hoặc 'inclusive'

- Kết quả trả về: một danh sách gồm  $n - 1$  điểm cắt để tách dữ liệu thành các khoảng với kích thước bằng nhau
- Ví dụ minh họa cho hàm `statistics.quantiles()`:

```
Chapter1_StatisticFunction.py > ...
1  import statistics
2  data = [10, 15, 20, 25, 30, 45, 50, 60, 70, 85, 65]
3  result = statistics.quantiles(data, n = 4)
4  print("Quantiles are: ")
5  print(result)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\X
acsuat\Chapter1_StatisticFunction.py"
Quantiles are:
[20.0, 45.0, 65.0]
```

### 1.2.12 Hàm `pstdev()`:

- Công dụng: Tính độ lệch chuẩn của tổng thể (căn bậc hai của phương sai tổng thể)
- Cú pháp: `statistics.pstdev(data, mu = None)`
- Tham số đầu vào:
  - o `data`: là một danh sách, bộ dữ liệu hoặc các giá trị số có thể lặp lại khác
  - o `mu`: giá trị trung bình tổng thể tùy chọn. Nếu không cung cấp, giá trị trung bình sẽ được tính từ chuỗi đã cho sẵn
- Kết quả trả về: Độ lệch chuẩn tổng thể của một chuỗi dữ liệu.
- Ví dụ minh họa cho hàm `statistics.pstdev()`

```
Chapter1_StatisticFunction.py > ...
1  import statistics
2  data = [10, 20, 30, 40, 50]
3  pstdev = statistics.pstdev(data)
4  print("Pstdev is: ")
5  print(pstdev)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\X
acsuat\Chapter1_StatisticFunction.py"
Pstdev is:
14.142135623730951
```

### 1.2.13 Hàm `pvariance()`:

- Công dụng: tính phương sai của tổng thể
- Cú pháp: `statistics.pvariance(data, mu = none)`

- Tham số đầu vào:
  - data: là một danh sách, bộ dữ liệu hoặc các giá trị số có thể lặp lại khác
  - mu: giá trị trung bình tổng thể tùy chọn. Nếu không cung cấp, giá trị trung bình sẽ được tính từ chuỗi đã cho sẵn
- Kết quả trả về: Phương sai tổng thể của dữ liệu
- Ví dụ minh họa cho hàm `statistics.pvariance()`:

```
Chapter1_StatisticFunction.py > ...
1  import statistics
2  data = [10, 20, 30, 40, 50]
3  pvariance = statistics.pvariance(data)
4  print("Pvariance is: ")
5  print(pvariance)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Pvariance is:
200
```

#### 1.2.14 Hàm `stdev()`:

- Công dụng: tính độ lệch chuẩn (căn bậc hai của phương sai) của mẫu – một tập hợp nhỏ dữ liệu được lựa chọn từ quần thể lớn hơn. Tương tự `pstdev()` nhưng sử dụng dữ liệu từ tổng thể lớn hơn.
- Cú pháp: `statistics.stdev(data, xbar = none)`
- Tham số đầu vào:
  - data là một danh sách, bộ dữ liệu hoặc các giá trị số có thể lặp lại khác
  - xbar là giá trị trung bình tổng thể tùy chọn. Nếu không cung cấp, giá trị trung bình sẽ được tính từ chuỗi đã cho sẵn
- Kết quả trả về: Giá trị phương sai của mẫu
- Ví dụ minh họa cho hàm `statistics.stdev()`:

```
Chapter1_StatisticFunction.py > ...
1  import statistics
2  data = [10, 20, 30, 40, 50]
3  stdev = statistics.stdev(data)
4  print("Stdev is: ")
5  print(stdev)
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Stdev is:
15.811388300841896
```

### 1.2.15 Hàm variance():

- Công dụng: tính phương sai của mẫu. Tương tự pvariance() nhưng sử dụng dữ liệu từ một tổng thể lớn hơn
- Cú pháp: statistics.variance(data, xbar = none)
- Tham số đầu vào:
  - o data là một danh sách, bộ dữ liệu hoặc các giá trị số có thể lặp lại khác
  - o xbar là giá trị trung bình tổng thể tùy chọn. Nếu không cung cấp, giá trị trung bình sẽ được tính từ chuỗi đã cho sẵn
- Kết quả trả về: giá trị phương sai của mẫu
- Ví dụ minh họa cho hàm statistics.variance():

```
Chapter1_StatisticFunction.py > ...
1  import statistics
2  data = [10, 20, 30, 40, 50]
3  variance = statistics.variance(data)
4  print("Variance is: ")
5  print([variance])
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Variance is:
250
```

### 1.2.16 Hàm covariance():

- Công dụng: tính hiệp phương sai mẫu giữa hai tập dữ liệu, đo lường sự biến thiên của hai biến ngẫu nhiên
- Cú pháp: statistics.covariance(x,y,/)
- Tham số đầu vào:
  - o x là một danh sách, một dãy dữ liệu chứa các giá trị số của biến thứ nhất
  - o y là một danh sách, một dãy dữ liệu chứa các giá trị số của biến thứ hai

- Kết quả trả về: Hiệp phương sai mẫu giữa x và y
- Ví dụ minh họa cho hàm `statistics.covariance()`:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 x = [1, 3, 5, 7, 9]
3 y = [2, 4, 6, 8, 10]
4 covariance = statistics.covariance(x,y)
5 print("Covariance is: ")
6 print([covariance])

PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\X
acsuat\Chapter1_StatisticFunction.py"
Covariance is:
10.0
```

### 1.2.17 Hàm `correlation()`:

- Công dụng: tính hệ số tương quan Pearson cho hai đầu vào (hai tập dữ liệu)
- Cú pháp: `statistics.correlation(x,y,/)`
- Tham số đầu vào:
  - o x là một danh sách, một dãy dữ liệu chứa các giá trị số của biến thứ nhất
  - o y là một danh sách, một dãy dữ liệu chứa các giá trị số của biến thứ hai
- Kết quả trả về: hệ số tương quan Pearson của x và y
- Ví dụ minh họa cho hàm `statistics.correlation()`:

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 x = [1, 3, 5, 7, 9]
3 y = [2, 4, 6, 8, 10]
4 correlation = statistics.correlation(x,y)
5 print("Correlation is: ")
6 print([correlation])

PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\X
acsuat\Chapter1_StatisticFunction.py"
Correlation is:
1.0
```

### 1.2.18 Hàm `linear_regression()`:

- Công dụng: tính độ dốc và giao điểm của các tham số hồi quy tuyến tính đơn giản, dự đoán biến phụ thuộc có giá trị liên tục (Y) dựa trên biến (X)
- Cú pháp: `statistics.linear_regression(x,y,/,*,proportional = false)`

- Tham số đầu vào:
  - x là một danh sách, một dãy dữ liệu chứa các giá trị số của biến thứ nhất
  - y là một danh sách, một dãy dữ liệu chứa các giá trị số của biến thứ hai
  - proportional: xác định mối quan hệ tuyến tính giữa hai biến đi qua gốc tọa độ (0,0) hay không. Nếu proportional = true thì mối quan hệ tuyến tính giữa hai biến là tỷ lệ thuận và đi qua gốc tọa độ (0,0). Nếu proportional = false, tỷ lệ sẽ được dựa trên dữ liệu thực tế
- Kết quả trả về: Độ dốc và giao điểm (điểm cắt) ước tính của đường hồi quy tuyến tính.
- Ví dụ minh họa cho hàm statistics.linear\_regression():

```
Chapter1_StatisticFunction.py > ...
1 import statistics
2 x = [1, 2, 3, 4, 5]
3 y = [6, 7, 8, 9, 10]
4 result1, result2 = statistics.linear_regression(x,y)
5 print("Results 1 without proportional is: ", result1)
6 print("Results 2 without proportional is: ",result2)]
```

```
PS C:\52200023_HKIII_Study\Xacsuat> python -u "c:\52200023_HKIII_Study\Xacsuat\Chapter1_StatisticFunction.py"
Results 1 without proportional is: 1.0
Results 2 without proportional is: 5.0
```

## CHƯƠNG 2: HISTOGRAM EQUALIZATION

### 2.1 Giải thích thuật toán Histogram equalization:

- Thuật toán cân bằng biểu đồ (Histogram equalization) là một kỹ thuật xử lý ảnh được sử dụng để cải thiện độ tương phản (contrast) cho ảnh và độ sáng của ảnh bằng cách dẫn cường độ của ảnh. Thuật toán này hoạt động bằng cách phân tích biểu đồ tần suất của ảnh và sau đó phân bố lại các giá trị pixel theo cách sao cho biểu đồ tần suất mới có phân phối đồng đều hơn, thường là phân bố đều theo dạng histogram chuẩn. Kết quả là ảnh sẽ có độ tương phản cao hơn và trở nên sáng sủa hơn, rõ nét hơn.
- Khi một tấm ảnh có phân bố xám không đều, điều này có thể dẫn đến việc bị mất vài thông tin trên ảnh hoặc không nhìn thấy rõ vì độ tương phản thấp. Đối với ảnh xám, histogram là biểu đồ tần số của các mức xám từ 0 đến 255. Thuật toán Histogram equalization sẽ giải quyết vấn đề này bằng cách phân bố lại mức xám cho đồng đều hơn và thông thường sẽ phân bố ở dạng uniform histogram (histogram chuẩn).

### 2.2 Điều kiện:

- Histogram là một biểu đồ thống kê hiển thị số lần xuất hiện của từng mức giá trị độ sáng hoặc màu sắc trong ảnh. Nó là công cụ quan trọng trong xử lý ảnh vì nó giúp chúng ta hiểu được phân bố giá trị mức độ sáng hoặc màu sắc trong ảnh. Các điều kiện sau phải được đáp ứng:
  1. Ảnh input: Đảm bảo phải có một tấm ảnh đầu vào để tính toán histogram. Ảnh có thể xám hoặc cũng có thể là ảnh màu
  2. Mức độ sáng và màu sắc: Histogram đo tần suất xuất hiện của các mức độ sáng (trong ảnh xám) hoặc mức độ màu sắc (trong ảnh màu).
  3. Biểu đồ tần suất: thống kê hiển thị số lần xuất hiện của từng mức giá trị độ sáng hoặc màu sắc trong ảnh
  4. Tần suất: tính toán histogram, ta đếm số lần xuất hiện từng mức sáng hoặc mức màu trong ảnh input.



- Histogram là một công cụ quan trọng trong xử lý ảnh vì nó giúp chúng ta hiểu được phân bố các giá trị mức độ sáng hoặc màu sắc trong ảnh.

### 2.3 Phương pháp của thuật toán Histogram equalization:

1. Tính toán biểu đồ: đầu tiên là cần phải tính toán biểu đồ của hình ảnh (đếm số lượng điểm ảnh tương ứng với mỗi mức xám). Biểu đồ là một biểu diễn đồ họa phân phối cường độ hình ảnh.
2. Tính toán biểu đồ tích lũy (CDF) của histogram: CDF cho ta biết tỷ lệ phần trăm các giá trị pixel có cường độ nhỏ hơn hoặc bằng một giá trị cường độ cụ thể.
3. Chuẩn hóa CDF: chuẩn hóa nó sao cho giá trị lớn nhất là 255 (giá trị trong khoảng từ 0 đến 1)
4. Ánh xạ lại các giá trị pixel: sử dụng hàm ánh xạ để lấy các giá trị cường độ của hình ảnh sau khi cân bằng.

Ánh xạ mới thay thế mỗi điểm ảnh trong ảnh ban đầu bằng giá trị tương ứng với mỗi điểm trong ánh xạ mới.

#### \*Các bước làm tham khảo từ tài liệu:

1. Tính toán histogram  $p_r(r)$
2. Chuẩn hóa histogram cho về khoảng  $[0, 1]$ :  $p_r(r_k) = \frac{n_k}{MN}$
3. Tính hàm xác suất mật độ

$$s_k = T(r_k) = (L-1) \sum_{j=0}^k p_r(r_j) \quad k = 0, 1, 2, \dots, L-1$$

4. Tính giá trị mức xám cho từng điểm ảnh:  $O(x, y) = \text{round}(T(I(x, y)))$

Ta có ví dụ để cho trực quan hơn. Ta có bảng phân phối cường độ và giá trị histogram cho hình ảnh kỹ thuật số 3 bit ảnh  $64 \times 64$

**TABLE 3.1**  
Intensity  
distribution and  
histogram values  
for a 3-bit,  $64 \times 64$   
digital image.

| $r_k$     | $n_k$ | $p_r(r_k) = n_k / MN$ |
|-----------|-------|-----------------------|
| $r_0 = 0$ | 790   | 0.19                  |
| $r_1 = 1$ | 1023  | 0.25                  |
| $r_2 = 2$ | 850   | 0.21                  |
| $r_3 = 3$ | 656   | 0.16                  |
| $r_4 = 4$ | 329   | 0.08                  |
| $r_5 = 5$ | 245   | 0.06                  |
| $r_6 = 6$ | 122   | 0.03                  |
| $r_7 = 7$ | 81    | 0.02                  |

Theo công thức ta sẽ tính được

$$L = 8$$

$$s_0 = T(r_0) = 7 * p_r(n_0) = 1.33$$

$$s_1 = T(r_1) = 3.08, s_2 = 4.55, s_3 = 5.67, s_4 = 6.23, s_5 = 6.65, s_6 = 6.86, \text{ and } s_7 = 7.00.$$

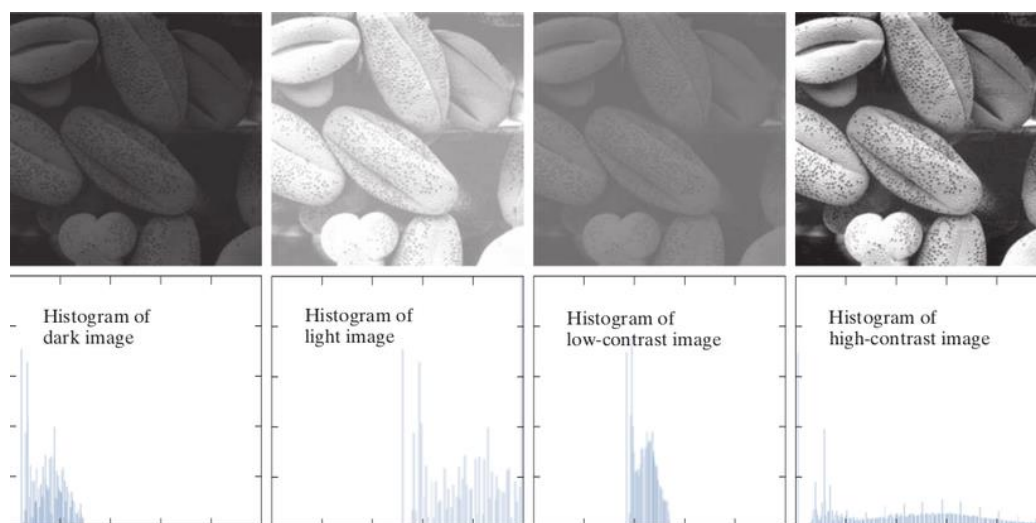
Sau đó ta sẽ chuyển đổi giá trị của từng điểm ảnh như sau:

$$\begin{array}{llll} s_0 = 1.33 \rightarrow 1 & s_2 = 4.55 \rightarrow 5 & s_4 = 6.23 \rightarrow 6 & s_6 = 6.86 \rightarrow 7 \\ s_1 = 3.08 \rightarrow 3 & s_3 = 5.67 \rightarrow 6 & s_5 = 6.65 \rightarrow 7 & s_7 = 7.00 \rightarrow 7 \end{array}$$

Hình 1, 2, 3

## 2.4 Minh họa:

Ta lần lượt có ảnh và histogram tương ứng với các ảnh dark, light, low – contrast và high – contrast:

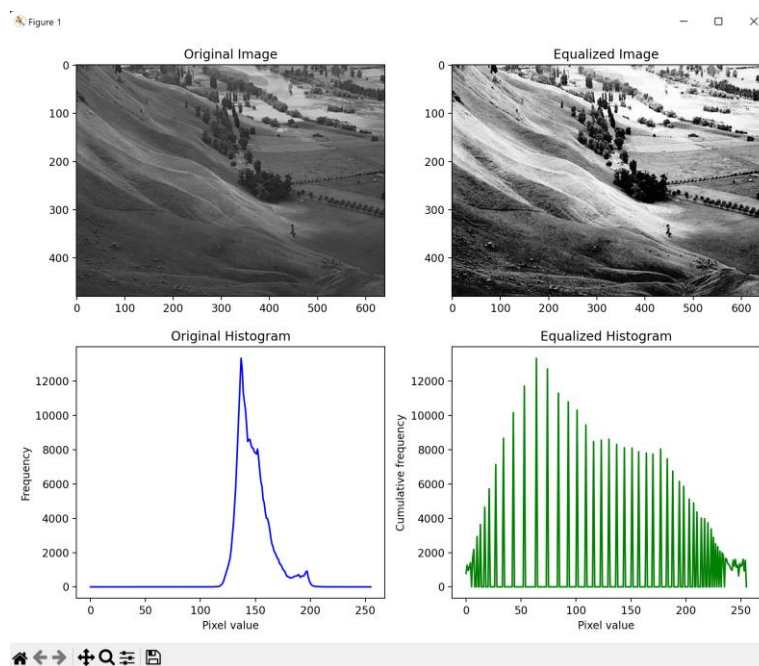


Hình 4

### ❖ Nhận xét:

- Với ảnh dark thì histogram có các cột tập trung vào bên trái tương ứng với màu tối
- Với ảnh light thì histogram có tập trung vào bên phải chứa các pixel có màu trắng

- Với ảnh có độ tương phản thấp (low – contrast) thì histogram có các cột tập trung xít nhau và ở giữa
- Với ảnh độ tương phản cao (high – contrast) thì histogram san đều với các giá trị



Hình 5

#### ❖ Nhận xét:

- Có thể thấy từ ảnh input đầu vào thì ảnh có vẻ hơi xám màu và tối (dark) nhưng sau khi qua sử dụng Histogram Equalization thì ảnh có vẻ sáng hơn và khả quan hơn.

#### 2.6 Đánh giá:

- Cân bằng biểu đồ là một kỹ thuật xử lý ảnh giúp cải thiện độ tương phản, làm cho hình ảnh trở nên sống động và hấp dẫn hơn. Tuy nhiên, nếu không sử dụng một cách cẩn thận, kỹ thuật này có thể gây mất cân bằng và giảm chất lượng của hình ảnh.
- Cân bằng biểu đồ hoạt động bằng cách phân tích độ sáng của từng pixel trong hình ảnh và điều chỉnh độ sáng của các pixel đó sao cho chúng có độ tương phản tốt hơn. Các cách điều chỉnh cụ thể độ sáng này có thể khác nhau, tùy thuộc vào thuật toán cân bằng biểu đồ được sử dụng.

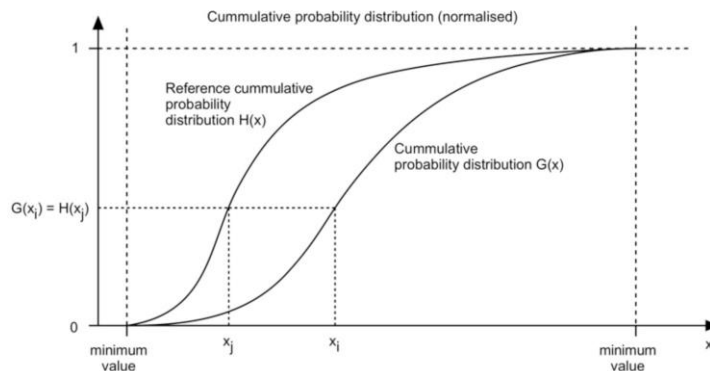
## CHƯƠNG 3: HISTOGRAM MATCHING ALGORITHMS

### 3.1 Giải thích thuật toán Histogram Matching:

- Histogram Matching Algorithms là một kỹ thuật xử lý ảnh được sử dụng để cải thiện độ tương phản của hình ảnh bằng cách biến đổi biểu đồ phân bố độ sáng của hình ảnh ban đầu sao cho khớp với biểu đồ phân bố độ sáng của một hình ảnh tham chiếu.
- Thuật toán hoạt động bằng cách tìm ra một hàm ánh xạ từ các giá trị độ sáng của hình ảnh ban đầu sang các giá trị độ sáng của hình ảnh tham chiếu sao cho biểu đồ phân bố độ sáng của hình ảnh ban đầu sau khi ánh xạ khớp với biểu đồ phân bố độ sáng của hình ảnh tham chiếu.

### 3.2 Điều kiện:

- Histogram Matching Algorithms yêu cầu hai hình ảnh đầu vào là: hình ảnh ban đầu và hình ảnh tham chiếu. Cả hai hình ảnh phải có cùng kích thước và độ sâu màu.
- Ghi chú: Ánh xạ cho mỗi trạng thái trong 256 trạng thái khác nhau:



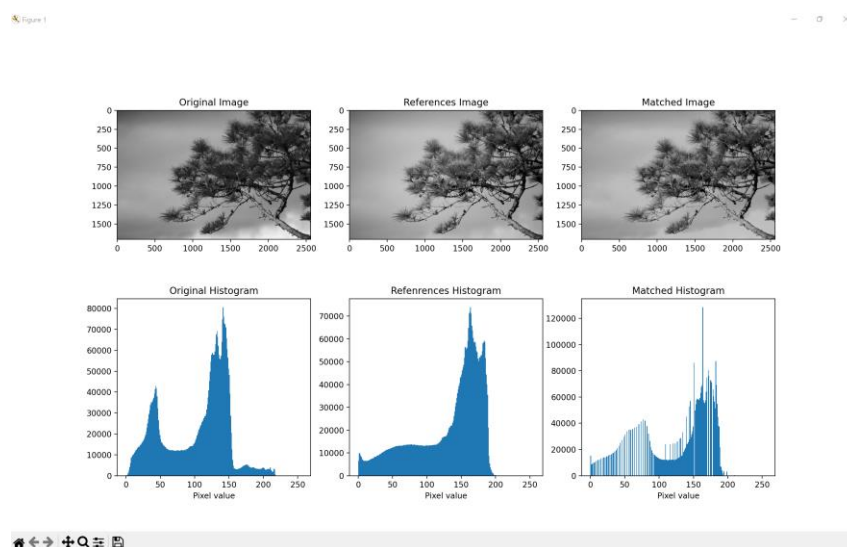
Hình 6

### 3.3 Phương pháp của thuật toán Histogram Matching Algorithms:

1. **Tạo biểu đồ phân bố độ sáng (histogram)** là biểu đồ mô tả sự phân bố các giá trị độ sáng trong một hình ảnh. Để tạo biểu đồ phân bố độ sáng của hình ảnh, cần tính toán số lượng pixel có giá trị độ sáng cụ thể trong hình ảnh đó.

2. **Tìm hàm biến đổi ánh sáng phù hợp nhất** là sau khi tạo biểu đồ độ sáng của hình ảnh ban đầu và hình ảnh tham chiếu, chúng ta cần tìm ra hàm biến đổi ánh sáng phù hợp nhất để biến đổi biểu đồ phân bố độ sáng của hình ảnh tham chiếu. Một cách phổ biến là sử dụng thuật toán tối ưu hóa, nó sẽ tìm ra hàm biến đổi ánh sáng và giảm thiểu sự khác biệt giữa hai biểu đồ phân bố độ sáng.
3. **Áp dụng hàm biến đổi ánh sáng** là sau khi tìm được hàm phù hợp, ta áp dụng vào hình ảnh ban đầu để tạo ra hình ảnh kết quả. Hàm biến đổi ánh sáng sẽ được sử dụng để tính toán giá trị độ sáng mới của mỗi pixel trong hình ảnh ban đầu. Giá trị độ sáng mới được tính bằng cách áp dụng hàm biến đổi ánh sáng cho giá trị độ sáng ban đầu của pixel đó.

### 3.4 Minh họa:



Hình 7

### 3.5 Đánh giá:

- Histogram Matching Algorithms là một kỹ thuật xử lý ảnh hữu ích có thể được sử dụng để cải thiện độ tương phản của hình ảnh. Tuy nhiên, thuật toán này cũng có một số nhược điểm như có thể khiến một số vùng của ảnh quá sáng hoặc quá tối gây mất cân đối và giảm chất lượng của hình ảnh, mất nhiều thời gian để thực hiện, đặc biệt là các hình ảnh lớn. Do đó, cần lựa chọn thuật toán cho phù hợp với các nhu cầu cụ thể.

## CHƯƠNG 4: IMPLEMENTATION

### 4.1 Code cho ví dụ minh họa Chương 2:

```
Chapter2.py > ...
1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  def hist_equalization(img):
6      # Tính toán lược đồ histogram
7      hist = np.zeros(256, dtype= int)
8      for i in range(img.shape[0]):
9          for j in range(img.shape[1]):
10             hist[img[i, j]] += 1
11
12     # Tính toán CDF
13     cdf = np.zeros(256, dtype = int)
14     cdf = np.cumsum(hist) / hist.sum()
15
16     # Giá trị pixel
17     equalized_img = np.zeros_like(img)
18     for i in range(img.shape[0]):
19         for j in range(img.shape[1]):
20             value_new = cdf[img[i, j]] * 255
21             equalized_img[i, j] = value_new
22
23     return equalized_img
24
25 if __name__ == "__main__":
26     # Đọc ảnh image
27     img = cv2.imread("input.jpg", cv2.IMREAD_GRAYSCALE)
28
29     # Điều chỉnh lại kích thước image
30     img = cv2.resize(img, (640, 480))
31
32     # Áp dụng thuật toán histogram equalization
33     equalized_img = hist_equalization(img)
34
35     # Lưu ảnh đã cân bằng ra tệp "equalized_image.jpg"
36     cv2.imwrite("equalized_image.jpg", equalized_img)
37
38
39     # Tính lược đồ tần số cho ảnh gốc và ảnh đã cân bằng
40     old_hist = cv2.calcHist([img], [0], None, [256], [0,256])
41     hist_equalized = cv2.calcHist([equalized_img], [0], None, [256], [0,256])
42
```

```

Chapter2.py > ...
43     # Hiện thị ảnh
44     plt.figure(figsize=(10, 8))
45     plt.subplot(2, 2, 1)
46     plt.imshow(img, cmap="gray")
47     plt.title("Original Image")
48
49     plt.subplot(2, 2, 2)
50     plt.imshow(equalized_img, cmap="gray")
51     plt.title("Equalized Image")
52
53     # Lược đồ nguyên bản
54     plt.subplot(2, 2, 3)
55     plt.plot(old_hist, color = 'blue')
56     plt.title("Original Histogram")
57     plt.xlabel("Pixel value")
58     plt.ylabel("Frequency")
59
60     # Lược đồ đã điều chỉnh
61     plt.subplot(2, 2, 4)
62     plt.plot(hist_equalized, color = 'green')
63     plt.title("Equalized Histogram")
64     plt.xlabel("Pixel value")
65     plt.ylabel("Cumulative frequency")
66
67     plt.tight_layout()
68     plt.show()

```

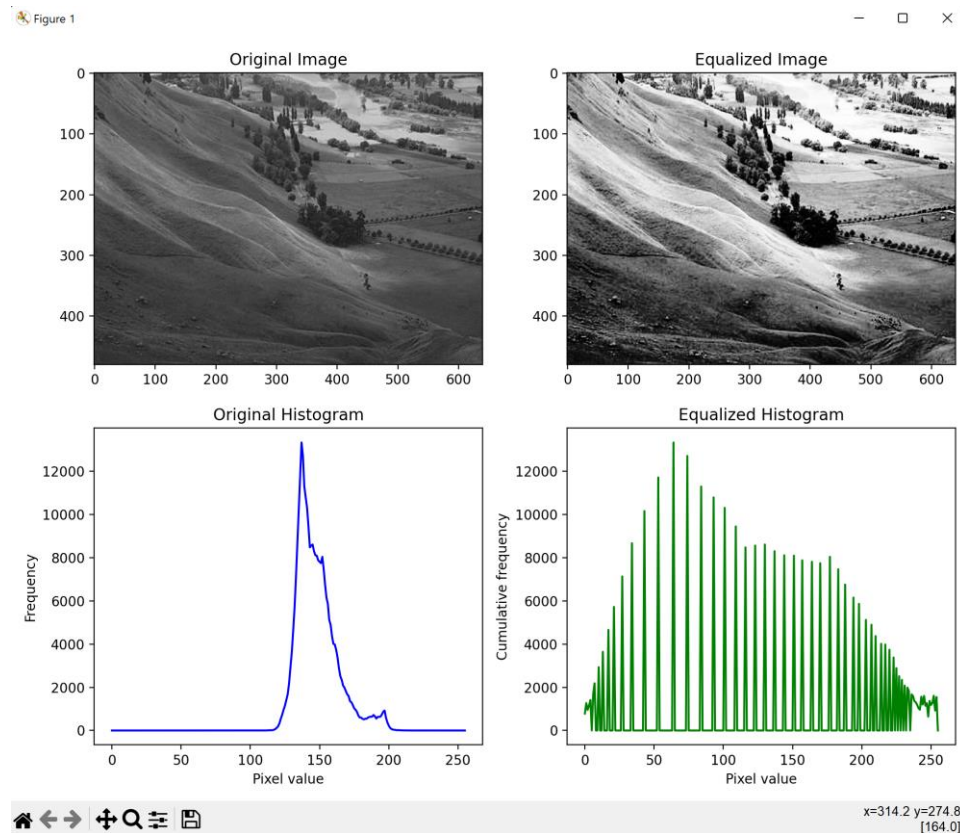
#### 4.1.1 Input – Output:

##### - Input:



Hình 8

- **Output:**



Hình 9

## 4.1.2 Phân tích code:

### 4.1.2.1 Nhập thư viện:

```
Chapter2.py > hist_equalization
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
```

- Thêm các thư viện cần thiết:
- “Numpy” (“np”): Thư viện được sử dụng cho xử lý mảng và ma trận số học và tính toán
- “matplotlib.pyplot” (“plt”): Thư viện được sử dụng để vẽ biểu đồ và hiển thị hình ảnh, biểu đồ
- “OpenCV” (“cv2”): Thư viện OpenCV, được sử dụng để xử lý và làm việc với ảnh



#### 4.1.2.2 Thực hiện cân bằng biểu đồ tần số:

- Định nghĩa hàm “histogram\_equalization(image)” để tiến hành cân bằng biểu đồ tần số của ảnh đầu vào

```
5 def hist_equalization(img):  
6     # Tính toán lược đồ histogram  
7     hist = np.zeros(256, dtype= int)  
8     for i in range(img.shape[0]):  
9         for j in range(img.shape[1]):  
10             hist[img[i, j]] += 1  
11
```

- Tạo một mảng “hist” có 256 phần tử (mức độ sáng từ 0 – 255) để lưu tần suất xuất hiện (đếm số lần xuất hiện) của mỗi giá trị pixel trong ảnh.
- Sử dụng vòng lặp để duyệt qua từng phần tử pixel trong ảnh và tăng giá trị tương ứng trong mảng “hist”.
  - o “img.shape[0]”: trả về chiều cao của hình ảnh img bằng vòng lặp “for i in range(img.shape[0])” sẽ duyệt qua tất cả các hàng của hình ảnh.
  - o “img.shape[1]”: trả về chiều rộng của hình ảnh img bằng vòng lặp “for j in range(img.shape[1])” sẽ duyệt qua tất cả các cột của ảnh.
- Tại (i, j) trong ảnh, tăng giá trị tương ứng trong mảng

#### 4.1.2.3 Tính hàm phân phối tích lũy của biểu đồ tần số

```
# Tính toán CDF  
cdf = np.zeros(256, dtype = int)  
cdf = np.cumsum(hist) / hist.sum()
```

- Tạo một mảng CDF (cumulative distribution function) với 256 phần tử để lưu giá trị hàm phân phối tích lũy.
- Đặt giá trị đầu tiên của CDF bằng giá trị tần số của pixel đầu tiên.
- Sử dụng “np.cumsum” để tính CDF bằng cách tích lũy giá trị trong mảng “hist” và chia cho tổng số giá trị trong histogram.

#### 4.1.2.4 Tạo mảng có cùng kích thước với ảnh “img” để lưu ảnh đã cân bằng:

```
# Giá trị pixel và tạo mảng để lưu ảnh đã cân bằng
equalized_img = np.zeros_like(img)
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        value_new = cdf[img[i, j]] * 255
        equalized_img[i, j] = value_new

return equalized_img
```

- Duyệt qua từng pixel trong ảnh và tính giá trị mới cho từng pixel dựa trên giá trị CDF đã tính
- Cú pháp “value\_new = cdf[img[i,j]] \* 255” tính giá trị mới của pixel(i,j) dựa trên giá trị pixel ban đầu của “img[i,j]” và giá trị CDF tương ứng. Giá trị này sẽ nằm trong khoảng 0 – 255
- Sau đó ta gán giá trị mới cho pixel(i,j) trong ảnh “equalized\_img”
- Trả về equalized\_img

#### 4.1.2.5 Đọc đường dẫn hình ảnh đầu vào (input):

```
# Đọc ảnh image
img = cv2.imread("input.jpg", cv2.IMREAD_GRAYSCALE)
```

- Đọc ảnh đầu vào từ tệp đường dẫn (ảnh phải nằm trong cùng thư mục với mã code thì mới hợp lệ) bằng cách sử dụng “cv2.imread”
- Chuyển đổi thành hình ảnh mức xám bằng hàm trong OpenCV “cv2.IMREAD\_GRAYSCALE”

#### 4.1.2.6 Điều chỉnh lại kích thước của ảnh:

```
# Điều chỉnh lại kích thước image
img = cv2.resize(img, (640, 480))
```

- Điều chỉnh lại kích thước của ảnh thành 640x480 pixels

#### 4.1.2.7 Cân bằng biểu đồ tần số:

```
# Áp dụng thuật toán histogram equalization
equalized_img = hist_equalization(img)
```

- Áp dụng thuật toán Histogram Equalization bằng cách gọi hàm “hist\_equalization” đã định nghĩa để cân bằng biểu đồ tần số cho ảnh gốc tạo ảnh đã cân bằng là “equalized\_img”

#### 4.1.2.8 Lưu ảnh ra tệp:

```
# Lưu ảnh đã cân bằng ra tệp "equalized_image.jpg"
cv2.imwrite("equalized_image.jpg", equalized_img)
```

- Tiến hành lưu ảnh ra tệp “equalized\_image.jpg” bằng hàm “cv2.imwrite”

#### 4.1.2.9 Tính biểu đồ tần số của ảnh gốc và ảnh đã cân bằng:

```
#Tính lược đồ tần số cho ảnh gốc và ảnh đã cân bằng
old_hist = cv2.calcHist([img], [0], None, [256], [0,256])
hist_equalized = cv2.calcHist([equalized_img], [0], None, [256], [0,256])
```

- Sử dụng hàm “cv2.calcHist” để tính biểu đồ tần số của ảnh gốc và ảnh đã cân bằng, bằng cách đếm số lần xuất hiện của mỗi giá trị pixel từ 0 đến 255.

#### 4.1.2.10 Tạo lại khung:

```
# Hiện thị ảnh
plt.figure(figsize=(10, 8))
```

- Tạo một khung hình với kích thước là 10x8 inch bằng thư viện Matplotlib để chứa các hình ảnh và biểu đồ output.

#### 4.1.2.11 Hiển thị ảnh vào khung và sắp xếp thứ tự của chúng:

```
45 plt.subplot(2, 2, 1)
46 plt.imshow(img, cmap="gray")
47 plt.title("Original Image")
48
49 plt.subplot(2, 2, 2)
50 plt.imshow(equalized_img, cmap="gray")
51 plt.title("Equalized Image")
52
53 # Lược đồ nguyên bản
54 plt.subplot(2, 2, 3)
55 plt.plot(old_hist, color = 'blue')
56 plt.title("Original Histogram")
57 plt.xlabel("Pixel value")
58 plt.ylabel("Frequency")
59
60 # Lược đồ đã điều chỉnh
61 plt.subplot(2, 2, 4)
62 plt.plot(hist_equalized, color = 'green')
63 plt.title("Equalized Histogram")
64 plt.xlabel("Pixel value")
65 plt.ylabel("Cumulative frequency")
66
67 plt.tight_layout()
68 plt.show()
```

- Sử dụng “matplotlib” để tạo một khung hình với 4 phần tử (2 hàng, 2 cột) – đồ thị 2x2 ở 4 góc để hiển thị ảnh và đồ thị.
- Sử dụng “plt.imshow” để hiển thị ảnh.
- Sử dụng “plt.plot” để vẽ biểu đồ tần số.
- Sử dụng “plt.tight\_layout()” sắp xếp các subplot một cách hợp lý để không bị chồng chéo lên nhau
- Sử dụng “plt.title” để viết tiêu đề
- Sử dụng “plt...label” để thêm nhãn
- Sử dụng “plt.show” để hiển thị khung hình chứa ảnh và biểu đồ.

## 4.2 Code cho ví dụ minh họa cho chương 3:

```
52200023_Part3.py > ...
1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # Hàm để tính histogram của ảnh
6  def compute_histogram(image):
7      histogram = cv2.calcHist([image], [0], None, [256], [0, 256])
8      histogram = histogram / histogram.sum()
9      return histogram
10
11 # Hàm để tính tích lũy histogram của ảnh
12 def compute_cumulative_histogram(histogram):
13     cumulative_histogram = histogram.cumsum()
14     return cumulative_histogram
15
16 # Hàm để áp dụng thuật toán histogram matching
17 def histogram_matching(input_image, reference_image):
18     input_hist = compute_histogram(input_image)
19     reference_hist = compute_histogram(reference_image)
20
21     input_cdf = compute_cumulative_histogram(input_hist)
22     reference_cdf = compute_cumulative_histogram(reference_hist)
23
24     mapping = np.interp(input_cdf, reference_cdf, range(256))
25
26     matched_image = mapping[input_image]
27
28     return matched_image
29
```

```

30 # Load ảnh gốc và ảnh tham chiếu
31 input_image = cv2.imread("image3.jpg", cv2.IMREAD_GRAYSCALE)
32 reference_image = cv2.imread("image4.jpg", cv2.IMREAD_GRAYSCALE)
33
34 # Áp dụng thuật toán histogram matching
35 matched_image = histogram_matching(input_image, reference_image)
36
37 # Vẽ biểu đồ của ảnh gốc, ảnh tham chiếu và kết quả
38 plt.figure(figsize=(15, 8))
39
40 plt.subplot(2, 3, 1)
41 plt.title("Original Image")
42 plt.imshow(input_image, cmap='gray')
43
44 plt.subplot(2, 3, 2)
45 plt.title("References Image")
46 plt.imshow(reference_image, cmap='gray')
47
48 plt.subplot(2, 3, 3)
49 plt.title("Matched Image")
50 plt.imshow(matched_image, cmap='gray')
51
52 plt.subplot(2, 3, 4)
53 plt.title("Original Histogram")
54 plt.hist(input_image.ravel(), 256, [0, 256])
55 plt.xlabel("Pixel value")
56

```

```

● 57 plt.subplot(2, 3, 5)
58 plt.title("Referrences Histogram")
59 plt.hist(reference_image.ravel(), 256, [0, 256])
60 plt.xlabel("Pixel value")
61
62 plt.subplot(2, 3, 6)
63 plt.title("Matched Histogram")
64 plt.hist(matched_image.ravel(), 256, [0, 256])
65 plt.xlabel("Pixel value")
66
67 plt.show()
68
69 #Lưu ảnh đã cân bằng ra tệp với tên: "matched_image.jpg"
70 cv2.imwrite("matched_image.jpg", matched_image)

```

#### 4.2.1 Input – Output:

- Input:

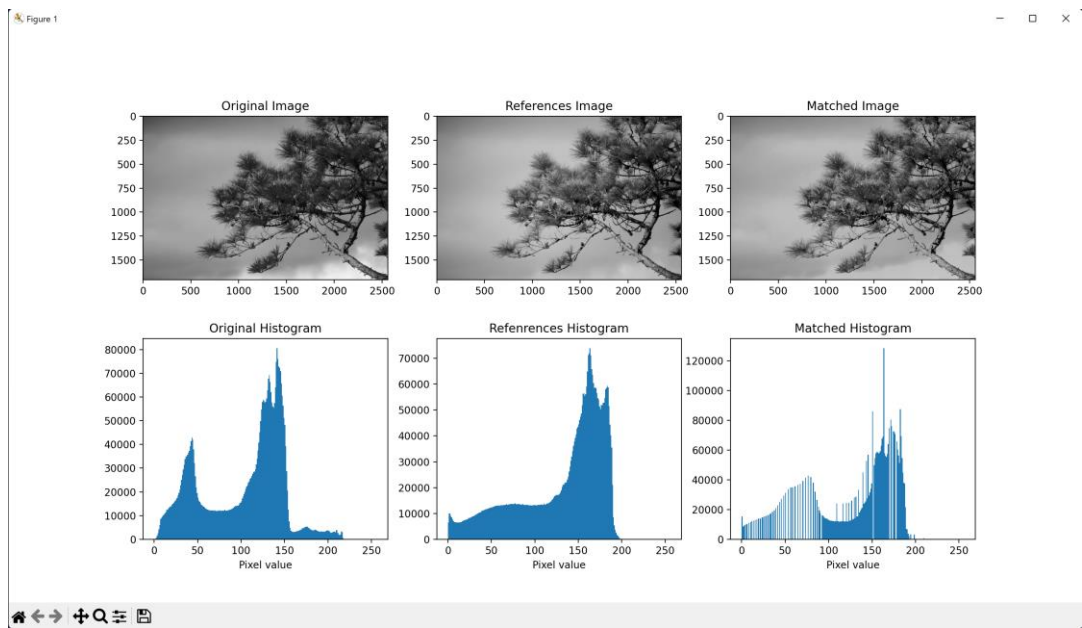


Hình 10



Hình 11

- Output:



Hình 12

#### 4.2.2 Phân tích code:

##### 4.2.2.1 Nhập thư viện:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
```

- Thêm các thư viện cần thiết:
- “Numpy” (“np”): Thư viện được sử dụng cho xử lý mảng và ma trận số học và tính toán
- “matplotlib.pyplot” (“plt”): Thư viện được sử dụng để vẽ biểu đồ và hiển thị hình ảnh, biểu đồ
- “OpenCV” (“cv2”): Thư viện OpenCV, được sử dụng để xử lý và làm việc với ảnh

##### 4.2.2.2 Tính histogram của ảnh:

```
# Hàm để tính histogram của ảnh
def compute_histogram(image):
    histogram = cv2.calcHist([image], [0], None, [256], [0, 256])
    histogram = histogram / histogram.sum()
    return histogram
```



- Hàm “compute\_histogram(image)” dùng để tính histogram của ảnh đầu vào theo biểu đồ tần số của mỗi giá trị pixel từ 0 đến 255
- Sử dụng cú pháp “cv2.calHist” để tính histogram của ảnh “image”
- Histogram được chuẩn hóa bằng cách chia mỗi giá trị trong histogram cho tổng số lần xuất hiện của tất cả các giá trị pixel, để đảm bảo tổng tỷ lệ của tất cả các giá trị trong histogram là 1.

#### 4.2.2.3 Tính tích lũy histogram của ảnh:

```
# Hàm để tính tích lũy histogram của ảnh
def compute_cumulative_histogram(histogram):
    cumulative_histogram = histogram.cumsum()
    return cumulative_histogram
```

- Sử dụng hàm “compute\_cumulative\_histogram(histogram)” để tính tích lũy histogram dựa trên histogram đã tính ở bước đầu tiên.
- Tích lũy histogram là tổng tích lũy của các giá trị trong histogram. Nó đại diện cho việc phân phối tích lũy của các giá trị pixel.

#### 4.2.2.4 Áp dụng thuật toán histogram matching:

```
# Hàm để áp dụng thuật toán histogram matching
def histogram_matching(input_image, reference_image):
    input_hist = compute_histogram(input_image)
    reference_hist = compute_histogram(reference_image)

    input_cdf = compute_cumulative_histogram(input_hist)
    reference_cdf = compute_cumulative_histogram(reference_hist)

    mapping = np.interp(input_cdf, reference_cdf, range(256))

    matched_image = mapping[input_image]

    return matched_image
```

- Ta có hàm “histogram\_matching(input\_image, reference\_image)” thực hiện quá trình cân bằng đồ tần số giữa ảnh đầu vào (“input\_image”) và ảnh tham chiếu (“reference\_image”)
- Tính histogram cho ảnh đầu vào và ảnh tham chiếu bằng cách gọi hàm “compute\_histogram”.

- Tính tích lũy histogram cho ảnh đầu vào và ảnh tham chiếu bằng cách gọi hàm “compute\_cumulative\_histogram”
- Tạo một ánh xạ (mapping) từ tích lũy histogram của ảnh đầu vào và sang tích lũy histogram của ảnh tham chiếu bằng cách sử dụng “np.interp”. Điều này đảm bảo rằng giá trị mới của pixel sau khi cân bằng sẽ phù hợp với phân phối của ảnh tham chiếu.
- Sử dụng ánh xạ này để áp dụng phép biến đổi lên ảnh đầu vào và trả về ảnh đã cân bằng.

#### 4.2.2.5 Thêm ảnh input đầu vào và ảnh tham chiếu:

```
# Load ảnh gốc và ảnh tham chiếu
input_image = cv2.imread("image3.jpg", cv2.IMREAD_GRAYSCALE)
reference_image = cv2.imread("image4.jpg", cv2.IMREAD_GRAYSCALE)
```

- Đọc ảnh đầu vào là ảnh gốc và ảnh tham chiếu từ tệp đường dẫn (ảnh phải nằm trong cùng thư mục với mã code thì mới hợp lệ) sử dụng “cv2.imread”
- Chuyển đổi thành hình ảnh mức xám bằng hàm trong OpenCV “cv2.IMREAD\_GRAYSCALE”

#### 4.2.2.6 Áp dụng thuật toán Histogram Matching:

```
# Áp dụng thuật toán histogram matching
matched_image = histogram_matching(input_image, reference_image)
```

- Gọi hàm “histogram\_matching” để cân bằng histogram ảnh đầu vào dựa trên tham chiếu. Kết quả thu được lưu trong biến “matched\_image”

#### 4.2.2.7 Tạo khung:

```
# Vẽ biểu đồ của ảnh gốc, ảnh tham chiếu và kết quả
plt.figure(figsize=(15, 8))
```

- Tạo một khung hình với kích thước là 15x8 inch bằng thư viện Matplotlib để chứa các hình ảnh và biểu đồ output.

#### 4.2.2.8 Vẽ biểu đồ của ảnh gốc, ảnh tham chiếu và kết quả:

```
40 plt.subplot(2, 3, 1)
41 plt.title("Original Image")
42 plt.imshow(input_image, cmap='gray')
43
44 plt.subplot(2, 3, 2)
45 plt.title("References Image")
46 plt.imshow(reference_image, cmap='gray')
47
48 plt.subplot(2, 3, 3)
49 plt.title("Matched Image")
50 plt.imshow(matched_image, cmap='gray')
51
52 plt.subplot(2, 3, 4)
53 plt.title("Original Histogram")
54 plt.hist(input_image.ravel(), 256, [0, 256])
55 plt.xlabel("Pixel value")
56
57 plt.subplot(2, 3, 5)
58 plt.title("References Histogram")
59 plt.hist(reference_image.ravel(), 256, [0, 256])
60 plt.xlabel("Pixel value")
61
62 plt.subplot(2, 3, 6)
63 plt.title("Matched Histogram")
64 plt.hist(matched_image.ravel(), 256, [0, 256])
65 plt.xlabel("Pixel value")
66
67 plt.show()
```

- Sử dụng thư viện “matplotlib” để tạo khung hình với 2 hàng và 3 cột
- Sử dụng “plt.subplot” để đặt các khu vực vẽ hình ảnh và biểu đồ tương ứng.
- Vẽ hình ảnh gốc, hình ảnh tham chiếu và hình ảnh đã cân bằng ở các vị trí tương ứng.
- Vẽ biểu đồ histogram cho hình ảnh gốc, hình ảnh tham chiếu và hình ảnh đã cân bằng ở các vị trí tương ứng với hình ảnh.
- Sử dụng “plt.show” để hiển thị khung hình chứa ảnh và biểu đồ.

#### 4.2.2.9 Lưu ảnh đã cân bằng ra tệp:

```
69 #Lưu ảnh đã cân bằng ra tệp với tên: "matched_image.jpg"
70 cv2.imwrite("matched_image.jpg", matched_image)
```

- Sử dụng “cv2.imwrite” để lưu ảnh đã cân bằng (biến “matched\_image”) thành tệp “matched\_image.jpg”.

## TÀI LIỆU THAM KHẢO (TIẾNG VIỆT)

- “Tuần 3: Histogram - Histogram equalization.” *Viblo*, [https://viblo.asia/p/tuan-3-histogram-histogram-equalization-3P0lPnxmKox#\\_3-code-9](https://viblo.asia/p/tuan-3-histogram-histogram-equalization-3P0lPnxmKox#_3-code-9). Accessed 8 November 2023.
- “Xử lý ảnh - OpenCV cân bằng sáng (histogram equalization).” *Minh Nguyen*, 27 October 2018, <https://minhng.info/tutorials/xu-ly-anh-opencv-can-bang-sang-histogram-equalization.html>. Accessed 8 November 2023.

## TÀI LIỆU THAM KHẢO (TIẾNG ANH)

- “‘;.’” ‘;.’ - *YouTube*, 9 March 2019, [https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram\\_equalization/histogram\\_equalization.html#how-does-it-work](https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html#how-does-it-work). Accessed 8 November 2023.
- Bourke, Paul, and Fred Weinhaus. “Histogram Matching.” *Paul Bourke*, <https://paulbourke.net/miscellaneous/equalisation/>. Accessed 8 November 2023.
- “How to Find the Median of Grouped Data (With Examples).” *Statology*, 11 February 2022, <https://www.statology.org/median-of-grouped-data/>. Accessed 8 November 2023.
- Huamán, Ana. “OpenCV: Histogram Equalization.” *OpenCV Documentation*, [https://docs.opencv.org/3.4/d4/d1b/tutorial\\_histogram\\_equalization.html](https://docs.opencv.org/3.4/d4/d1b/tutorial_histogram_equalization.html). Accessed 8 November 2023.
- Jain, Sandeep. “Histogram matching with OpenCV, scikit-image, and Python.” *GeeksforGeeks*, 3 January 2023, <https://www.geeksforgeeks.org/histogram-matching-with-opencv-scikit-image-and-python/>. Accessed 8 November 2023.
- “(PDF) The Integrated Usage of LBP and HOG Transformations and Machine Learning Algorithms for Age Range Prediction from Facial Images.” *ResearchGate*, [https://www.researchgate.net/publication/329732956\\_The\\_Integrated\\_Usage\\_of\\_LBP\\_and\\_HOG\\_Transformations\\_and\\_Machine\\_Learning\\_Algorithms\\_for\\_Age\\_Range\\_Prediction\\_from\\_Facial\\_Images](https://www.researchgate.net/publication/329732956_The_Integrated_Usage_of_LBP_and_HOG_Transformations_and_Machine_Learning_Algorithms_for_Age_Range_Prediction_from_Facial_Images). Accessed 8 November 2023.

### SELF-EVALUATION FORM

| Criteria     |                      | Scale | 0 score               | 1/2 score                                  | Full score                                  | Self-evaluation | Reason                                      |
|--------------|----------------------|-------|-----------------------|--|---|-----------------|---|
| Task 1       | Theoretical research | 1     | Do nothing or wrongly | Not enough details, no example, no comment | Correct calculations, detailed explanations | 1               | Correct calculations, detailed explanations |
|              | Implementation       | 2     | Error                 | Correct but bad performance                | Correct and good performance                | 2               | Correct and good performance                |
|              | Test                 | 1     | No test               | Test without verification                  | Test and verification                       | 1               | Test and verification                       |
|              | Theoretical research | 2     | Do nothing or wrongly | Not enough details, no example, no comment | Correct calculations, detailed explanations | 2               | Correct calculations, detailed explanations |
| Task 2       | Implementation       | 1     | Error                 | Correct but bad performance                | Correct and good performance                | 1               | Correct and good performance                |
|              | Test                 | 1     | No test               | Test without verification                  | Test and verification                       | 1               | Test and verification                       |
|              | Analysis             | 0.5   | Do nothing or wrongly | Not enough details, no example, no comment | Correct, detailed explanations              | 0.5             | Correct, detailed explanations              |
|              | Discussion           | 0.5   | Do nothing or wrongly | Not enough details, no example, no comment | Correct, detailed explanations              | 0.5             | Correct, detailed explanations              |
|              | Recommendation       | 0.5   | Do nothing or wrongly | Not enough details, no example, no comment | Correct, detailed explanations              | 0.5             | Correct, detailed explanations              |
| Reference    |                      | 0.5   | No reference          | Wrong format                               | Right format                                | 0.5             | Right format                                |
| <b>Total</b> |                      | 10    | Result                |  |   | 10              |   |