|  |  |
| --- | --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG THÔN VÀ PTNT** |
| **TRƯỜNG ĐẠI HỌC THỦY LỢI** | |



NGUYỄN DOANH CHÍNH

**TÌM HIỂU VÀ XÂY DỰNG PHẦN MỀM HIỆU CHỈNH ÁNH SÁNG TRONG ẢNH**

**ĐỒ ÁN TỐT NGHIỆP**

**HÀ NỘI, NĂM 2024**

|  |  |
| --- | --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG THÔN VÀ PTNT** |
| **TRƯỜNG ĐẠI HỌC THỦY LỢI** | |

NGUYỄN DOANH CHÍNH

**TÌM HIỂU VÀ XÂY DỰNG PHẦN MỀM HIỆU CHỈNH ÁNH SÁNG TRONG ẢNH**

|  |  |
| --- | --- |
| Ngành: | Hệ thống thông tin |
| Mã số: |  |

|  |  |
| --- | --- |
| Người hướng dẫn: | THS. Nguyễn Ngọc Quỳnh Châu |

**HÀ NỘI, NĂM 2024**

|  |  |
| --- | --- |
| Description: Logo-WRU | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập - Tự do - Hạnh phúc**  ----------★----------  **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP** |

|  |  |  |  |
| --- | --- | --- | --- |
| Họ tên sinh viên: | Nguyễn Doanh Chính | **Hệ đào tạo:** | Đại học chính quy |
| Lớp: | 61HT | **Ngành:** | Hệ thống thông tin |
| Khoa: | Công nghệ thông tin |  |  |

**1 – TÊN ĐỀ TÀI**

**TÌM HIỂU VÀ XÂY DỰNG PHẦN MỀM HIỆU CHỈNH ÁNH SÁNG TRONG ẢNH**

.

**2 – NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN**

|  |  |
| --- | --- |
| **Nội dung các phần** | **Tỷ lệ** |
| MỞ ĐẦU | 5% |
| Chương 1: Tổng quan về xử lý ảnh | 30% |
| Chương 2: Các phương pháp hiệu chỉnh ảnh | 30% |
| Chương 3: Xây dựng thử nghiệm phần mềm xử lý ảnh | 30% |
| KẾT LUẬN | 5% |

**3 – GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN**

|  |  |
| --- | --- |
| **Nội dung các phần** | **GVHD** |
| MỞ ĐẦU | THS. Nguyễn Ngọc Quỳnh Châu |
| Chương 1: Tổng quan về xử lý ảnh |
| Chương 2: Các phương pháp hiệu chỉnh ảnh |
| Chương 3: Xây dựng thử nghiệm phần mềm xử lý ảnh |
| KẾT LUẬN |

**5 – NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Ngày ………… tháng ………… năm 20…

|  |  |
| --- | --- |
| **Trưởng bộ môn**  *(Ký và ghi rõ Họ tên)* | **Giáo viên hướng dẫn chính**  *(Ký và ghi rõ Họ tên)* |

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua.

Ngày ……tháng …… năm 20…

|  |  |
| --- | --- |
|  | **Chủ tịch Hội đồng**  *(Ký và ghi rõ Họ tên)* |

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày … tháng … năm 20…

|  |  |
| --- | --- |
|  | **Sinh viên làm Đồ án tốt nghiệp**  *(Ký và ghi rõ Họ tên)*  **Chính**  **Nguyễn Doanh Chính** |

|  |  |
| --- | --- |
| Description: logo | TRƯỜNG ĐẠI HỌC THỦY LỢI **KHOA CÔNG NGHỆ THÔNG TIN**  --------------------  BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP |
|  |  |
| **TÊN ĐỀ TÀI** | **TÌM HIỂU VÀ XÂY DỰNG PHẦN MỀM HIỆU CHỈNH ÁNH SÁNG TRONG ẢNH** |
| *Sinh viên thực hiện* | : **Nguyễn Doanh Chính** |
| *Lớp* | : **61HT** |
| *Giáo viên hướng dẫn* | : THS.Nguyễn Ngọc Quỳnh Châu |
| *Thời gian thực hiện* | : … tuần – *từ ngày ../../2023 đến ngày ../../2023* |

**TÓM TẮT ĐỀ TÀI**

Dưới tác động ngày càng lớn của công nghệ, ảnh số đã trở thành một phần không thể thiếu trong đời sống hàng ngày và đóng góp quan trọng vào sự phát triển của nhiều lĩnh vực khoa học. Trong mỗi bức ảnh, ánh sáng đóng vai trò quyết định, là yếu tố tác động trực tiếp lên chất lượng và cảm xúc mà bức ảnh mang lại.Việc điều chỉnh ánh sáng trong ảnh trở thành một nhiệm vụ không thể thiếu. Từ việc tinh chỉnh độ sáng và độ tương phản đến những thao tác phức tạp như hồi phục màu sắc của vật thể, tất cả đều nhằm mục đích tối ưu hóa trải nghiệm hình ảnh. Nhưng điều này không dừng lại ở đó. Đằng sau mỗi bức ảnh được cải thiện, là sự kết hợp tinh tế giữa kiến thức về xử lý ảnh, thuật toán và ứng dụng của học máy. Đề tài "**Tìm hiểu và Xây Dựng Phần Mềm Hiệu Chỉnh Ánh Sáng Trong Ảnh**" không chỉ đơn thuần là nghiên cứu, mà còn là quá trình triển khai và đánh giá kết quả, từ đó mang lại những sự cải thiện rõ rệt đối với mỗi bức ảnh.

**CÔNG NGHỆ SỬ DỤNG**

* Môi trường lập trình: **IDLE**,**VSCode**.
* Ngôn ngữ lập trình: **Python**.
* Framework**: OpenCV, Tkinter, PIL (Python Imaging Library), Matplotlib.**

**CÁC MỤC TIÊU CHÍNH**

* Khái quát về xử lý ảnh.
* Hiểu về khái niệm ánh sáng trong ảnh: Nắm khái niệm cơ bản về ánh sáng trong ảnh bao gồm độ sáng, độ tương phản, cân bằng màu sắc.
* Hiểu về cách màu sắc được biểu diễn trong không gian màu sắc (RGB, HSV, HSI) và cách chúng tương tác.
* Nguyên cứu phương pháp hiệu chỉnh ánh sáng :Nắm cách điều chỉnh độ sáng tổng thể và sự tương phản trong ảnh .
* Cân bằng màu sắc để loại bỏ sự biến đổi màu không mong muốn.
* Sử dụng histogram để phân tích phân bố màu sắc và độ sáng trong ảnh.

**KẾT QUẢ DỰ KIẾN**

**1. Kiến thức:**

* Sinh viên hiểu được về những kiến thức tìm hiểu trong báo cáo.
* Nắm bắt về kỹ thuật xử lý ảnh.
* Nắm bắt khái niệm ánh sáng trong ảnh và các yếu tố liên quan như độ sáng, độ tương phản và màu sắc.
* Các kỹ thuật điều chỉnh ánh sáng.

1. **Kỹ năng:**

* Có khả năng sử dụng Python và các thư viện xử lý ảnh như OpenCV và thư viện để thực hiện các thao tác xử lý ảnh.
* Xây dựng giao diện người dùng trực quan bằng cách sử dụng framework như Tkinter.

**3. Sản phẩm:**

* Hoàn thành báo cáo đồ án.
* Sản phảm hiệu chỉnh trong ảnh.

**LỜI CAM ĐOAN**

Em tên là Nguyễn Doanh Chính, em xin cam đoan đây là Đồ án tốt nghiệp, Luận án tốt nghiệp do chính bản thân em làm. Các kết quả trong Đồ án tốt nghiệp, Khóa luận tốt nghiệp này em làm là trung thực, không lẫy sẵn những nguồn đã có sẵn. Việc tham khảo các nguồn tài liệu ( nếu có ) đã được em thực hiện chích dẫn và ghi nguồn tài liệu tham khảo một cách rõ ràng và đúng quy định.

|  |  |
| --- | --- |
|  | **Tác giả ĐATN / KLTN** |
|  | *Chính* |
|  | **Nguyễn Doanh Chính** |

LỜI CẢM ƠN

Sau hơn 4 năm học tập và rèn luyện tại trường Đại học Thủy Lợi nói chung và Khoa Công nghệ thông tin nói riêng, em đã nhận được rất nhiều sự hướng dẫn, giúp đỡ tận tình từ thầy cô và các bạn trong ngành học. Nhờ vào những lời khuyên và chỉ bảo của thầy, cô em đã cố gắng vượt qua những khó khan để thực hiện bài đồ án tốt nghiệp của mình.

Tiếp đến, em xin gửi lời chi ân tới các thầy, cô trong khoa Công nghệ thông tin trường Đại học Thủy Lợi đã tạo điều kiện thuận lợi cho em trong quá trình học tập,truyền đạt những kiến thức để giúp em có được nền tảng kiến thức tốt tại trường cũng như thời gian thực hiện làm đồ án tốt nghiệp. Đặc biệt em muốn gửi lời cảm ơn tới cô **Ths.Nguyễn Ngọc Quỳnh Châu** – giáo viên trực tiếp hướng dẫn, chỉ bảo để em khắc phục khó khăn, thiếu xót và hoàn thành các phần trong đồ án tốt nghiệp từ lý thuyết cho tới thực hành một cách thuận lợi. Ngoài ra không thể không nhắc tới gia đình ,bạn bè người thân đã là chỗ dựa tinh thần của em trong thời gian qua để thực hiện tốt phần đồ án tốt nghiệp lần này.

Mặc dù em đã cố gắng hết sức để hoàn thành đồ án này ,em nhận thức rằng với lượng kiến thức và kinh nghiệm ít ỏi của bản than nên đồ án tốt nghiệp không tránh khỏi thiếu sót. Em mong nhận được những sự đóng góp tới từ thầy cô để em có thể ngày càng hoàn thiện hơn.

*Em xin chân thành cảm ơn!*

Hà Nội, ngày … tháng … năm 2024

Sinh viên thực hiện

**Nguyễn Doanh Chính**

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc156248748)

[1. Lý do chọn đề tài 1](#_Toc156248749)

[2. Mục tiêu nghiên cứu 1](#_Toc156248750)

[3. Phương pháp nghiên cứu 2](#_Toc156248751)

[4. Phạm vi nghiên cứu 2](#_Toc156248752)

[DANH SÁCH HÌNH ẢNH 3](#_Toc156248753)

[CHƯƠNG 1: TỔNG QUAN XỬ LÝ ẢNH 4](#_Toc156248754)

[1.1 Cơ sở lý thuyết 4](#_Toc156248755)

[1.1.1 Tổng quan về xử lý ảnh 4](#_Toc156248756)

[1.1.2 Màu sắc trong ảnh 8](#_Toc156248757)

[1.1.3 Các bộ lọc 12](#_Toc156248758)

[1.2. Nền tảng công nghệ sử dụng 16](#_Toc156248759)

[1.2.1 Ngôn ngữ python 16](#_Toc156248760)

[1.2.2 Thư viện 18](#_Toc156248761)

[CHƯƠNG 2: CÁC PHƯƠNG PHÁP HIỆU CHỈNH ẢNH 22](#_Toc156248762)

[2.1 Hiệu chỉnh độ sáng 22](#_Toc156248763)

[2.2 Hiệu chỉnh độ tương phản 24](#_Toc156248764)

[2.3 Hiệu chỉnh gamma 28](#_Toc156248765)

[2.4 Cân bằng màu 31](#_Toc156248766)

[2.5 Hiệu ứng mờ viền 34](#_Toc156248767)

[2.6 Xoay ảnh 37](#_Toc156248768)

[CHƯƠNG 3: XÂY DỰNG THỬ NGHIỆM PHẦN MỀM XỬ LÝ ẢNH 41](#_Toc156248769)

[3.1 Môi trường thực hiện 41](#_Toc156248770)

[3.2 Chương trình thử nghiệm 41](#_Toc156248771)

[3.3 Code các module chương trình 42](#_Toc156248772)

[KẾT LUẬN 56](#_Toc156248773)

[4.1 Kết quả đạt được 56](#_Toc156248774)

[4.2 Một số hạn chế trong ứng dụng 56](#_Toc156248775)

[4.3 Hướng phát triển ứng dụng trong tương lai 56](#_Toc156248776)

[DANH MỤC TÀI LIỆU THAM KHẢO 57](#_Toc156248777)

MỞ ĐẦU

1. Lý do chọn đề tài

Tiến bộ công nghệ đã tạo ra một không gian rộng lớn cho việc nghiên cứu và ứng dụng hiệu chỉnh ánh sáng trong ảnh. Đề tài này không chỉ là quy trình nhằm cải thiện chất lượng hình ảnh, mà còn mang lại nhiều lợi ích toàn diện.

Lý do chính để chọn đề tài “**Tìm hiểu và xây dựng phần mềm Hiệu chỉnh ánh sáng trong ảnh**” là nhu cầu thực tế. Trong thời đại số hóa ngày nay, việc chụp và chia sẻ hình ảnh trở nên phổ biến hơn bao giờ hết. Hiệu chỉnh ánh sáng trong ảnh giúp cải thiện chất lượng hình ảnh, tạo ra những bức ảnh ấn tượng và thu hút hơn.

Hiệu chỉnh ánh sáng trong ảnh không chỉ hữu ích trong lĩnh vực nhiếp ảnh, mà còn có ứng dụng trong nhiều lĩnh vực khác như quảng cáo, truyền thông, giáo dục, nghiên cứu khoa học, và thậm chí là y tế. Điều này cho thấy sự đa dạng và phong phú của đề tài, mở ra nhiều cơ hội cho việc nghiên cứu và ứng dụng.

Hiệu chỉnh ánh sáng trong ảnh là một lĩnh vực rộng lớn với nhiều phương pháp và kỹ thuật khác nhau. Điều này tạo ra nhiều cơ hội để mở rộng nghiên cứu và phát triển các giải pháp mới. Điều này không chỉ thúc đẩy sự sáng tạo và đổi mới, mà còn giúp cải thiện và nâng cao chất lượng của các sản phẩm và dịch vụ liên quan đến hình ảnh.

Tổng quan, việc “**Tìm hiểu và xây dựng phần mềm Hiệu chỉnh ánh sáng trong ảnh**” không chỉ đem lại lợi ích cá nhân mà còn đóng góp đến sự phát triển và cải thiện trong nhiều lĩnh vực trong cuộc sống hàng ngày của con người.

2. Mục tiêu nghiên cứu

* Nghiên cứu là phát triển cải tiến các phương pháp hiệu chỉnh ánh sáng trong ảnh, nâng cao chất lượng hình ảnh và tạo những bức ảnh sống động, hấp dẫn hơn.
* Khôi phục và tái tạo thông tin bị mất do ánh sáng không đồng đều hoặc do quá trình chụp ảnh.
* Nhằm làm nổi bật các đối tượng và chi tiết quan trọng trong ảnh, giúp chuyền đạt thông tin một cách hiệu quả.
* Khám phá các ứng dụng mới của việc hiệu chỉnh ánh sáng trong ảnh từ truyền thông, giáo dục đến nghiên cứu khoa học và y tế.
* Chia sẻ kết quả nghiên cứu với cộng đồng nhiếp ảnh và công nghệ, giúp mọi người tạo ra những bức ảnh chất lượng cao hơn.

3. Phương pháp nghiên cứu

* Tìm hiểu và đánh giá các công trình nghiên cứu, bài báo, và tài liệu liên quan đến “**Hiệu chỉnh ánh sáng trong ảnh**”. Điều này giúp hiểu rõ hơn về lĩnh vực nghiên cứu, cũng như những tiến bộ và hạn chế của các phương pháp hiện có.
* Thu thập thông tin các tính năng cần thiết cho việc hiệu chỉnh ánh sáng trong ảnh. Xác định các yếu tố ảnh hưởng đến ảnh ,cũng như các kỹ thuật và công nghệ được sử dụng để điều chỉnh.
* Dựa trên thông tin thu thập, tiến hành xây dựng thiết kế cho ứng dụng “**Hiệu chỉnh ánh sáng trong ảnh**”.
* Mô tả cách thức hoạt động của ứng dụng và các tính năng có trong ứng dụng.

4. Phạm vi nghiên cứu

* Nghiên cứu về “**Hiệu chỉnh ánh sáng trong xử lý ảnh** “ đặt ra các thách thức và khía cạnh quan trọng trong lĩnh vực xử lý ảnh. Việc hiểu rõ ảnh hưởng của ánh sáng đến chất lượng hình ảnh, độ tương phản, và màu sắc là trọng tâm của đề tài.
* Một phần quan trọng khác là phát triển các kỹ thuật điều chỉnh độ sáng không chỉ để cải thiện chết lượng tổng thể của hình ảnh mà còn để khôi phục thông tin mất mát.
* Ứng dụng thực tế của nghiên cứu mở ra nhiều cơ hội trong các lĩnh vực như nhiếp ảnh, quảng cáo, thiết kế, y học và khoa học.
* Tập trung vào các khía cạnh kỹ thuật , ứng dụng cũng chú trọng dến tác động của việc hiệu chỉnh ánh sáng đối với trải nghiệm người dung. Giúp cho người dùng có trải nhiệm tốt với giao diện dễ dung, hỗ trợ dễ dàng người sử dụng trong quá trình.
* Đánh giá khả năng ứng dụng và linh hoạt trong kỹ thuật “**Hiệu chỉnh ánh sáng trong ảnh**” trong nhiều tình huống sẽ giúp xác định tiềm năng và quyết định hướng phát triển trong tương lại.Tuy nhiên ứng dụng có khả năng mở rộng để phục vụ các đối tượng và tổ chức có nhu cầu trong việc xử lý ảnh. Đặc biệt trong nhận diện trong điều kiện thiếu ánh sáng.

DANH SÁCH HÌNH ẢNH

[Hình 1.1: Sơ đồ mô tả các bước cơ bản xử lý ảnh 5](#_Toc6716)

[Hình 1.2 : Chuyển đổi giữa các mô hình biểu diễn ảnh 7](#_Toc10303)

[Hình 1.3 : Ảnh biểu diễn dưới dạng ma trận 2 chiều 7](#_Toc13526)

[Hình 1.4 :Hệ màu HSI 10](#_Toc24785)

[Hình 1.5 :Hệ màu HSV 12](#_Toc31076)

[Hình 1.6 :Biểu đồ hình Gaussian 1D 13](#_Toc10712)

[Hình 1.7 :Đồ thị hình Gausian 14](#_Toc12224)

[Hình1. 8 :Ngôn ngữ Python 16](#_Toc19977)

[Hình 2.1 :Ví dụ về chứ năng “Hiệu chỉnh ánh sáng tăng” với tham số là 72 22](#_Toc23901)

[Hình 2.2 :Ví dụ về chứ năng “Hiệu chỉnh ánh sáng tăng” với tham số là -72 23](#_Toc3240)

[Hình 2.3 :Hình ảnh với độ tương phản 2.0 26](#_Toc15045)

[Hình 2.4 :Hình ảnh với độ tương phản -2.0 27](#_Toc23198)

[Hình 2.5 : Biểu diễn các giá trị gamma 27](#_Toc7493)

[Hình 2.6:Ví dụ chức năng “Hiệu chỉnh gamma” với tham số là 2.2 30](#_Toc14796)

[Hình 2.7 :Hình ảnh trước và sau khi cân bằng màu với ảnh xám 32](#_Toc10470)

[Hình 2.8 :Hình ảnh trước và sau khi cân bằng màu với ảnh màu 33](#_Toc6815)

[Hình 2.9 : Phép quay hình 37](#_Toc24224)

[Hình 2.10 :Phép đối xứng 37](#_Toc31716)

[Hình 2.11 : ví dụ chức năng xoay ảnh 39](#_Toc23243)

[Hình 3.1 : Giao diện chính chương trình 41](#_Toc15582)

[Hình 3.2 : Hiển thị thư mục chọn ảnh 42](#_Toc22486)

[Hình 3.3 :Hình khi chọn ảnh chỉnh sửa 43](#_Toc21910)

[Hình 3.4 :Kết quả chức năng điều chỉnh độ sáng 44](#_Toc18747)

[Hình 3.5 :Kết quả chức năng điều chỉnh tương phản 46](#_Toc26995)

[Hình 3.6 : Kết quả chức năng gamma 47](#_Toc8882)

[Hình 3.7 : Kết quả chức năng cân bằng màu 49](#_Toc26634)

[Hình 3.8 :Hình: Kết quả chức năng hiệu ứng mờ viền 51](#_Toc8102)

[Hình 3.9 :Kết quả chức năng xoay ảnh và đổi màu nền 54](#_Toc12343)

CHƯƠNG 1: TỔNG QUAN XỬ LÝ ẢNH

1.1 Cơ sở lý thuyết

1.1.1 Tổng quan về xử lý ảnh

Xử lý ảnh đại diện cho quá trình biến đổi hình ảnh số, nhằm nâng cao chất lượng, rút trích thông tin hoặc sáng tạo hình ảnh mới từ dữ liệu hình ảnh ban đầu. Lĩnh vực này đóng vai trò quan trọng trong thị giác máy tính và trí tuệ nhân tạo, có ứng dụng rộng rãi trong nhiều lĩnh vực như y tế, công nghiệp, nhận dạng khuôn mặt, ô tô tự hành và nhiều ứng dụng khác. Điều này mang lại sự tiện lợi và cải thiện hiệu suất trong xử lý hình ảnh, giúp tối ưu hóa các ứng dụng và dịch vụ từ dữ liệu hình ảnh.

Các bước cơ bản xử lý ảnh:

Hình 1.1: Sơ đồ mô tả các bước cơ bản xử lý ảnh

**Nạp Ảnh (Loading Image)**: Đây là bước đầu tiên, nơi chúng ta tải ảnh vào hệ thống để xử lý. Có thể cần chuyển đổi định dạng ảnh hoặc điều chỉnh kích thước ảnh.

**Tiền Xử Lý (Preprocessing)**: Trong bước này, chúng ta chuẩn bị ảnh cho các bước xử lý chính. Điều này có thể bao gồm việc điều chỉnh độ sáng, độ tương phản, loại bỏ nhiễu, hoặc thậm chí là phân đoạn ảnh để tách các đối tượng quan tâm.

**Xử Lý Chính (Processing)**: Đây là nơi các biến đổi chính diễn ra. Các bước này có thể bao gồm việc áp dụng các bộ lọc (như bộ lọc trung bình), thực hiện phép biến đổi Fourier hoặc phép biến đổi Hough, hoặc thực hiện các thuật toán nhận dạng đối tượng.

**Hậu Xử Lý (Postprocessing):** Sau khi các biến đổi chính được áp dụng, có thể cần một số bước hậu xử lý để chuẩn bị ảnh cho việc hiển thị hoặc lưu trữ. Điều này có thể bao gồm việc chuyển đổi ảnh về định dạng phù hợp, điều chỉnh kích thước, hoặc áp dụng các hiệu ứng cuối cùng.

**Ứng Dụng Cuối Cùng (Final Application):** Cuối cùng, ảnh đã được xử lý có thể được sử dụng trong ứng dụng hoặc quá trình cụ thể. Điều này có thể bao gồm hiển thị kết quả cho người dùng, lưu lại để sử dụng sau này, hoặc tích hợp vào các hệ thống lớn hơn.

1.1.1.1 Ảnh và Điểm ảnh

Ảnh tự nhiên, thường được biểu diễn như một hàm liên tục của không gian và độ sáng, đòi hỏi quá trình số hóa để trở thành một tập hợp điểm ảnh, hay còn gọi là pixel, đại diện cho vị trí và độ sáng tương ứng. Mỗi pixel là một điểm ảnh, và khoảng cách giữa chúng được thiết lập sao cho mắt người không phân biệt được sự chuyển động giữa các pixel khi nhìn bình thường.

Quá trình số hóa này thường được thực hiện thông qua mô hình toán học, trong đó ảnh có thể được xem như là một hàm hai biến f(x, y), với giá trị xám giới hạn từ 0 đến 255 trong trường hợp ảnh xám.Điểm ảnh được xem như là dấu hiệu hay cường độ sáng tại một toạ độ trong không gian của đối tượng.

Ảnh được xem như là một tập hợp các điểm ảnh. Khi số hoá, nó thường được biểu diễn bởi bảng hai chiều I(n,p): n dòng và p cột. Ta nói ảnh gồm n x p điểm ảnh. Người ta thường kí hiệu I(x,y) để chỉ một điểm ảnh.

Mỗi pixel trong ảnh được coi như một dấu hiệu hoặc cường độ sáng tại một vị trí cụ thể trong không gian của đối tượng. Việc giới hạn giá trị của hàm f(x, y) trong khoảng 0 đến 255 giúp đảm bảo rằng mỗi điểm ảnh có thể được biểu diễn bằng một byte. Đối với ảnh màu, mỗi màu (R, G, B) có thể được mô tả thông qua các hàm riêng biệt của (x, y).

Số điểm ảnh tạo nên một ảnh gọi là độ phân giải (resolusion). Độ phân giải thường được biểu thị bằng số điểm ảnh theo chiều dọc và chiều ngang của ảnh. Ảnh có độ phân giải càng cao càng rõ nét. Như vậy, ảnh càng to thì càng bị vỡ hạt, độ mịn càng kém. Ảnh có thể được biểu diễn theo mô hình Vector hoặc mô hình Raster.

* **Mô hình Vector và Mô hình Raster trong Biểu Diễn Ảnh:**
* **Mô hình Raster:**

Đây là cách biểu diễn ảnh thông dụng nhất hiện nay,Mô hình Raster là một tập hợp hữu các giá trị số,gọi là điểm ảnh (pixel) . Mỗi pixel đều chứa thông tin về độ sáng và màu sắc tại một vị trí xác định trên lưới hình ảnh. Cả ảnh được chia thành một số lượng hữu hạn các pixel, và mỗi pixel đại diện cho một điểm ảnh riêng lẻ. Tuỳ theo yêu cầu thực thế mà mỗi điểm ảnh được biểu diễn qua 1 hay nhiều bit. Mật độ điểm ảnh trên một đơn vị kích thước vật lý được gọi là độ phân giải. Ảnh có độ phân giải cao thì càng đẹp, càng mịn và càng thể hiện rõ chi tiết. Đây là mô hình phổ biến trong xử lý ảnh và đồ họa máy tính,thuận lợ cho việc hiển thị và in ấn.

Ảnh raster thường được thu từ camera,các máy chiếu chụp quét… và chinh là đối tượng chinh của xử lý ảnh và thị giác máy tính.

* **Mô hình Vector:**

Ảnh vector là một loại ảnh được tạo thành từ các thành phần hình học đơn giản như điểm, đường thẳng, hình khối, v.v. Thay vì được lưu trữ dưới dạng ma trận điểm ảnh như ảnh raster, ảnh vector được biểu diễn dưới dạng tọa độ các thành phần trong ảnh.

Điều đặc biệt của ảnh vector là khả năng thay đổi kích thước mà không bị mất chất lượng, không xuất hiện hiện tượng răng cưa như ảnh raster. Điều này là do dữ liệu trong ảnh vector chỉ gồm các thông tin về tọa độ, không phụ thuộc vào độ phân giải như trong ảnh raster. Do đó, ảnh vector thường tiết kiệm dung lượng lưu trữ hơn.Màu sắc trong ảnh vector thường kém hơn so với ảnh raster, vì chỉ được biểu diễn dưới dạng một số màu cơ bản hoặc những phần tử màu sắc đơn giản khác.

Ảnh vector được sử dụng trong thiết kế logo, banner, giao diện đồ họa và các ứng dụng tương tự. Tuy nhiên, trong lĩnh vực xử lý ảnh và thị giác máy tính, thường không đề cập đến xử lý ảnh vector, mà tập trung vào xử lý ảnh raster.

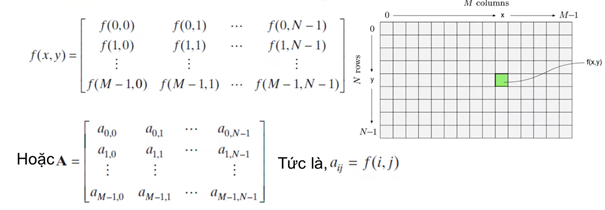


Hình 1.2: Chuyển đổi giữa các mô hình biểu diễn ảnh

1.1.1.2 Ánh Số

Ảnh số là một tập hợp hữu hạn các điểm ảnh, hay còn gọi là pixel. Mỗi pixel trong ảnh số có thể được biểu diễn dưới dạng một phần tử trong ma trận hai chiều.

Ảnh số cũng có thể được hiểu như một hàm hai chiều, f(x, y), trong đó x và y là tọa độ không gian và biên độ của f tại mỗi cặp tọa độ (x, y) biểu diễn cường độ hoặc mức độ màu xám của hình ảnh tại điểm đó..



Hình 1.3: Ảnh biểu diễn dưới dạng ma trận 2 chiều

Khi tất cả các giá trị x, y và cường độ của f đều là các đại lượng rời rạc, hữu hạn, chúng ta gọi đó là hình ảnh kỹ thuật số. Độ lớn (amplitude) của hàm f được gọi là độ sáng (intensity) hay độ xám (gray level) của ảnh tại điểm đó

Ảnh số thành 2 loại : ảnh đen trắng và ảnh màu:

* **Ảnh đen trắng**

Ảnh đen trắng là loại ảnh chỉ bao gồm ảnh đen và ảnh trắng, không chứa bất kỳ màu nào khác. Thực tế ảnh đen trắng được phân loại thành 2 loại khác nhau là ảnh nhị phân và ảnh đa mức xám. Người ta phân mức đen trắng của ảnh thành L mức. Nếu L bằng 2 nghĩa là chỉ có 2 mức 0 và 1, hay còn được gọi là ảnh nhị phân. Nếu L lớn hơn 2 thì ta có ảnh đa cấp xám.

Nói cách khác, mỗi điểm ảnh nhị phân được mã hóa trên 1 bit còn ảnh xám 256 mức đã được mã hóa 8 bit (1byte).

* **Ảnh màu**

Ảnh màu là một dạng biểu diễn của ảnh số, trong đó mỗi pixel không chỉ biểu diễn một giá trị độ sáng, mà còn biểu diễn màu sắc. Mỗi pixel trong ảnh màu được biểu diễn bằng một bộ ba giá trị (r, g, b), tương ứng với mức độ của màu đỏ (red), xanh lục (green), và xanh lam (blue). [Khi kết hợp các màu này theo tỷ lệ nhất định, ta có thể tạo ra nhiều màu sắc khác nhau](https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/" \t "_blank).

[Để dễ cho việc xử lý ảnh, ma trận pixel của ảnh màu thường được chia thành ba ma trận riêng biệt, mỗi ma trận biểu diễn một kênh màu gồm: kênh đỏ (red), kênh xanh lục (green), và kênh xanh lam (blue)](https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/" \t "_blank). [Ngoài ra, còn có các hệ màu khác như HSV (Hue, Saturation, Value) được định nghĩa theo cách tương tự như cách con người cảm nhận màu sắc](https://vi.eyewated.com/mo-hinh-mau-hsv-la-gi/" \t "_blank).

[Độ sâu màu, hay bit màu, là thước đo số lượng màu sắc mà một bức ảnh có thể dùng để hiển thị](https://thuvien.hocviennhiepanh.com/chon-lua-8-bits-hay-16-bits-khi-hau-ky/" \t "_blank). [Độ sâu màu cao hơn sẽ cho phép hiển thị nhiều sắc thái màu sắc hơn](https://quantrimang.com/cong-nghe/do-sau-mau-la-gi-182177" \t "_blank).

1.1.1.3 Mức xám

* **Mức xám:**

Mức xám trong ảnh là kết quả của quá trình mã hóa cường độ sáng tương ứng của mỗi điểm ảnh bằng một giá trị số. Cách mã hóa thường sử dụng một số mức khác nhau, ví dụ như 16, 32, hoặc 64 mức. Tuy nhiên, mã hóa với 256 mức là phổ biến nhất do lợi ích kỹ thuật. Với 256 mức, mỗi điểm ảnh được biểu diễn bằng 8 bit, từ [0 ,255].

Ảnh có chỉ hai mức xám được gọi là ảnh nhị phân, mỗi điểm ảnh chỉ có thể có giá trị 0 hoặc 1. Nếu ảnh có mức xám lớn hơn 2, nó được gọi là ảnh đa cấp xám hoặc ảnh màu, trong đó mỗi điểm ảnh có thể có giá trị mức xám khác nhau.

Với ảnh màu, mọi màu có thể được tạo ra từ ba màu cơ bản: Đỏ, Lục (Xanh lá cây), và Lam (Xanh dương). Mỗi điểm ảnh của ảnh màu được lưu trữ trong 3 byte, tương ứng với 24 bit, và do đó, có 224 = 16,7 triệu màu khả dụng. Ảnh xám, tuy nói là ảnh màu, thực chất chỉ sử dụng màu xám, có các thành phần R, G, B trong hệ thống màu RGB có cùng cường độ. Điều này có nghĩa là mỗi điểm ảnh sẽ có một mức xám xác định.

Mức xám không chỉ là cách biểu diễn sự đơn giản của ảnh, mà còn liên quan đến việc truyền đạt thông tin về cường độ sáng tại mỗi điểm ảnh. Trong quá trình số hóa, mức xám được ánh xạ từ giá trị cường độ sáng của điểm ảnh, và nó thường được biểu diễn dưới dạng giá trị số từ 0 đến 255 trong trường hợp ảnh xám.

Ảnh xám thường được coi như là biểu hiện của không gian hai chiều, trong đó mỗi điểm ảnh tương ứng với một giá trị cụ thể của hàm hai biến. Quá trình này là quan trọng để mắt người không phân biệt được sự chuyển động giữa các điểm ảnh khi nhìn bình thường.

Nếu nhìn sâu hơn vào ảnh xám, nó thực chất là một dạng đặc biệt của ảnh màu, trong đó cả ba thành phần màu (R, G, B) có giá trị bằng nhau. Điều này tạo ra hiệu ứng màu xám tại mỗi điểm ảnh.

Mức xám không chỉ quan trọng trong việc truyền đạt thông tin về sự sáng tối của ảnh, mà còn là cơ sở cho nhiều phương pháp xử lý ảnh, bao gồm điều chỉnh độ tương phản, làm sắc nét, phát hiện cạnh, và nhiều ứng dụng khác.

1.1.2 Màu sắc trong ảnh

1.1.2.1 Hệ màu RGB

RGB được viết tắt của Red (màu đỏ), Green (màu xanh lá), và Blue (màu xanh dương), là 3 màu gốc trong mô hình ánh sáng bổ sung. Các màu này tương ứng với tia sáng đỏ, xanh lá và xanh dương trong ánh sáng trắng. Mỗi màu sẽ đại diện cho 1 giá trị từ 0 – 255, trong đó, 0 là không có màu và 255 là màu sáng nhất. Sự kết hợp của 3 màu cơ bản RGB theo cường độ khác nhau cho ra các màu sắc diễn tả các màu sắc khác nhau trong tự nhiên.

Tuy nhiên, 3 màu cơ bản RGB không định nghĩa “đỏ”, “xanh lá” và “xanh dương” chính xác. Với cùng các giá trị, RGB có thể mô tả các màu khác nhau trên các thiết bị khác nhau.

Hệ màu RGB hoạt động theo cơ chế cộng để tạo ra màu mới. Tùy vào tỷ lệ pha trộn giữa 3 màu cơ bản RGB, mỗi màu sẽ có một độ sáng khác nhau.  Khi 3 màu Red, Green và Blue hòa trộn theo tỉ lệ 1:1:1 sẽ tạo thành màu trắng trong mô hình ánh sáng bổ sung (màu sáng hơn màu gốc). Hệ thống mã hóa màu RGB (Red-Green-Blue) được sử dụng trong đồ họa máy tính sử dụng ba byte (28)3, tương đương khoảng 16 triệu màu khác nhau.

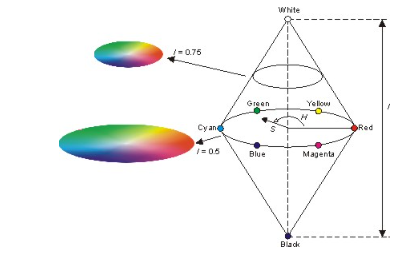
Đặc biệt là các file hệ RGB sẽ thể hiện tốt trên các thiết bị phát quang có sử dụng ánh sáng trắng làm cơ sở. Vì thế, hệ màu RGB thường được sử dụng cho các màu được thể hiện trên màn hình máy tính hoặc các màu trong lĩnh vực thiết kế digital trên website được chiếu qua các màn hình hoặc máy chiếu dùng ánh sáng.

Hiện nay, mã Hex 6 chữ số là phổ biến nhất. Tiếp đến là mã thập lục phân byte 3 chữ số, mỗi byte đại diện cho cường độ của màu đỏ, xanh lá và xanh dương. Giá trị byte của mã Hex nằm trong cường độ từ 00 (thấp nhất) đến FF (cao nhất)..

1.1.2.2 Hệ màu HSI

**Hệ màu HSI (Hue, Saturation, Intensity)** là một mô hình màu được sử dụng rộng rãi trong xử lý ảnh số. [Hệ màu này mã hóa thông tin màu sắc theo cách mà con người thường mô tả màu sắc](https://viblo.asia/p/mo-hinh-mau-hsi-toan-tap-va-ung-dung-demo-tren-thiet-bi-di-dong-android-ZDEvLRJpeJb" \t "_blank).

* **[Hue (H)](https://viblo.asia/p/mo-hinh-mau-hsi-toan-tap-va-ung-dung-demo-tren-thiet-bi-di-dong-android-ZDEvLRJpeJb" \t "_blank)**[: Hue mang thông tin về màu sắc và có giá trị từ 0 đến 360 độ](https://viblo.asia/p/mo-hinh-mau-hsi-toan-tap-va-ung-dung-demo-tren-thiet-bi-di-dong-android-ZDEvLRJpeJb" \t "_blank).
* **Saturation (S)**: Saturation, hay độ bão hòa, cho biết mức độ mà một màu sắc được pha loãng bằng ánh sáng trắng. [Giá trị của S nằm trong khoảng từ 0 đến 1](https://viblo.asia/p/mo-hinh-mau-hsi-toan-tap-va-ung-dung-demo-tren-thiet-bi-di-dong-android-ZDEvLRJpeJb" \t "_blank).
* **Intensity (I)**: Intensity, hay cường độ, liên quan đến mức độ sáng của màu sắc. [Giá trị của I cũng nằm trong khoảng từ 0 đến 1](https://viblo.asia/p/mo-hinh-mau-hsi-toan-tap-va-ung-dung-demo-tren-thiet-bi-di-dong-android-ZDEvLRJpeJb" \t "_blank).



Hình 1.4:Hệ màu HSI

Hệ màu HSI hoạt động dựa trên cơ sở hấp thụ ánh sáng. Màu mà chúng ta nhìn thấy là từ phần của ánh sáng không bị hấp thụ. [Để thay đổi màu trong hệ HSI, chúng ta không tăng thêm ánh sáng mà bản thân màu HSI sẽ tự loại bỏ đi ánh sáng đi từ ánh sáng gốc để thay đổi thành các màu sắc khác nhau](https://viblo.asia/p/mo-hinh-mau-hsi-toan-tap-va-ung-dung-demo-tren-thiet-bi-di-dong-android-ZDEvLRJpeJb).

**Chuyển đổi giữa không gian màu :**

**Chuyển đổi từ HSI sang RGB:**

* (H) là giá trị màu Hue nằm trong khoảng [0, 360).
* (S) là giá trị độ bão hòa Saturation nằm trong khoảng [0, 1].
* I là giá trị Intensity nằm trong khoảng [0, 1].

Tính toán hệ số R, G, B từ H, S, I theo các công thức:

Đối với các khu vực màu Hue, ánh xạ chúng vào các giá trị R, G, B như sau:

* + - Khu vực 0° ≤ (H) < 60°: (R = C, G = X, B = 0)
    - Khu vực 60° ≤ (H) < 120°: (R = X, G = C, B = 0)
    - Khu vực 120° ≤ (H) < 180°: (R = 0, G = C, B = X)
    - Khu vực 180° ≤ (H) < 240°: (R = 0, G = X, B = C)
    - Khu vực 240° ≤ (H) < 300°: (R = X, G = 0, B = C)
    - Khu vực 300° ≤ (H) < 360°: (R = C, G = 0, B = X)

Áp dụng dời và tụ điện để có giá trị cuối cùng:

* + - R = (R + m) ×255
    - G = (G + m) ×255
    - B = (B + m) ×255

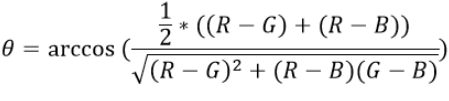
**Chuyển đổi từ RGB sang HSI:**

* + (R, G, B) là giá trị màu Red, Green, Blue nằm trong khoảng [0, 255].

- Tính toán (I) (Intensity): )

- Nếu (R = G = B), có thể xác định (H) (Hue) là 0 vì ảnh đen trắng.

- Nếu (R G B), tính toán (H) từ các giá trị (R, G, B) theo các công thức:



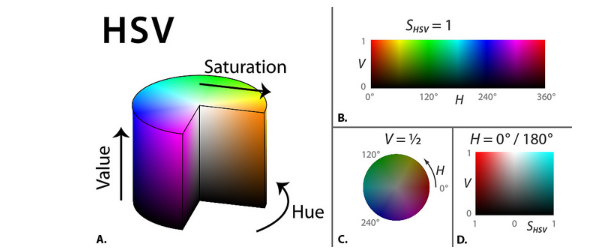
- Tính toán (S) (Saturation):

1.1.2.3 Hệ màu HSV

Hệ màu HSV (Hue, Saturation, Value) là một phương pháp biểu diễn màu sắc trong không gian màu ba chiều, nơi màu sắc được mô tả bằng các yếu tố chính là màu sắc chính (Hue), độ bão hòa (Saturation), và độ sáng (Value). Hệ màu HSV thường được sử dụng trong xử lý ảnh, đồ họa máy tính và nhiều ứng dụng khác do tính linh hoạt và dễ hiểu của nó.

* + **[Hue (H)](https://vi.eyewated.com/mo-hinh-mau-hsv-la-gi/" \t "_blank)**[: Hue mang thông tin về màu sắc và có giá trị từ 0 đến 360 độ](https://vi.eyewated.com/mo-hinh-mau-hsv-la-gi/" \t "_blank).
  + **Saturation (S)**: Saturation, hay độ bão hòa, cho biết mức độ mà một màu sắc được pha loãng bằng ánh sáng trắng. [Giá trị của S nằm trong khoảng từ 0 đến 1](https://vi.eyewated.com/mo-hinh-mau-hsv-la-gi/" \t "_blank).
  + **Value (V)**: Value, hay cường độ, liên quan đến mức độ sáng của màu sắc. [Giá trị của V cũng nằm trong khoảng từ 0 đến 1](https://vi.eyewated.com/mo-hinh-mau-hsv-la-gi/" \t "_blank).

.



Hình 1.5:Hệ màu HSV

Màu HSV cũng cho phép dễ dàng thực hiện được các phép toán trên màu sắc, như việc tăng hoặc giảm độ sáng, thay đổi màu sắc chính và tạo ra các hiệu ứng khác nhau.

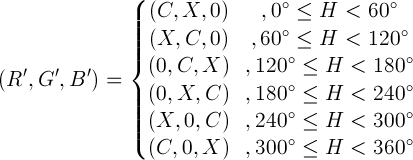
Công thức chuyển đổi HSV sang RGB:

Khi **0 ≤ *H* <360, 0 ≤ *S* ≤ 1 và 0 ≤ *V* ≤ 1:**

***C* = *V* × *S***

***X* = *C* × (1 - | ( *H* / 60 °) mod 2 - 1 |)**

***m* = *V* - *C***



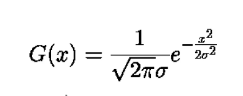
( *R* , *G* , *B* ) = (( *R* '+ *m* ) × 255, ( *G* ' + *m* ) × 255, ( *B* '+ *m* ) × 255)

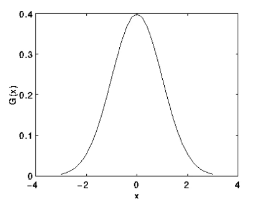
1.1.3 Các bộ lọc

1.1.3.1 Bộ lọc Gausian

Lọc Gaussian là một phương pháp quan trọng trong xử lý ảnh, được sử dụng để làm mờ và giảm nhiễu hình ảnh. Nó dựa trên lý thuyết hàm Gaussian, một hàm mượt có hình dạng giống hình chuông.

Công thức hàm Gaussian 1D có dạng như sau:



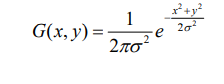


Hình 1.6:Biểu đồ hình Gaussian 1D

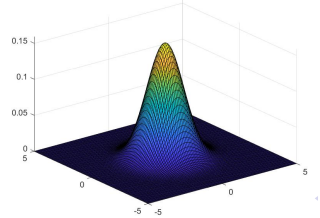
**Trong đó :**

* **G(x)** là giá trị của hàm tại điểm x.
* **σ** (sigma) là độ lệch chuẩn (standard deviation).
* **π** là số pi.
* **(*e*)** là số Euler (gần giá trị 2.71828).

Trong không gian 2 chiều, hàm Gaussian 2 chiều với trung bình (0,0) và độ lệch chuẩn bằng **σ** có công thức sau:



Đồ thị Gaussian có hình dạng sau:



Hình 1.7:Đồ thị hình Gausian

**Trong đó:**

* **(x, y)** là tọa độ của pixel đang xét.
* **σ** là tham số định rõ độ mờ của hiệu ứng. Giá trị sigma càng lớn, độ mờ càng cao.
* **e** là số Euler, xấp xỉ 2.71828.

Hàm Gaussian tạo ra một kernel Gaussian, có dạng hình tròn đồng tâm, với giá trị giảm dần từ trung tâm ra ngoài.

Lọc Gaussian, hay còn gọi là Gaussian Blur là một phép biến đổi sử dụng kernel Gaussian để làm mờ chi tiết và giảm nhiễu trong ảnh. Khi áp dụng Gaussian Blur, mỗi pixel trong ảnh mới được tính toán dựa trên giá trị trung bình có trọng số của các pixel xung quanh, với trọng số được xác định bởi hàm Gaussian.

Công thức áp dụng Gaussian Blur trong không gian 2 chiều:

I'(x, y) =

Trong đó (I(x, y)) là giá trị pixel tại tọa độ ((x, y)) trong ảnh gốc, (I'(x, y)) là giá trị pixel sau khi áp dụng Gaussian Blur, và (G(i, j)) là giá trị của kernel Gaussian tại tọa độ ((i, j)).

Lọc Gaussian giúp làm mờ chi tiết và giảm độ tương phản trong ảnh, tạo hiệu ứng mịn màng và làm đẹp hình ảnh. Độ lệch chuẩn (sigma) của kernel quyết định mức độ làm mờ, với (sigma) lớn tạo ra hiệu ứng làm mờ mạnh mẽ hơn.

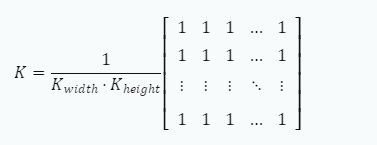
1.1.3.2 Bộ lọc trung bình

Bộ lọc trung bình là một kỹ thuật lọc đơn giản nhưng hiệu quả trong xử lý ảnh. Nó được xây dựng dựa trên ý tưởng tính giá trị một điểm ảnh bằng trung bình cộng các điểm ảnh xung quanh nó.

Giả sử chúng ta có ảnh đầu vào I kích thước M×N và mặt nạ K kích thước m×n. Khi đó, phép lọc ảnh sẽ được thực hiện như sau:

1. Đặt mặt nạ K tại mỗi vị trí (x,y) của ảnh I sao cho điểm trung tâm của K trùng với (x,y).
2. Tính giá trị mới của pixel tại (x,y) bằng cách lấy trung bình cộng các giá trị pixel của I tương ứng với các vị trí của K.

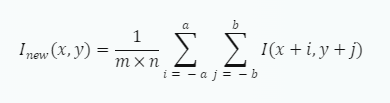
Ma trận lọc, còn được gọi là mặt nạ, được sử dụng trong bộ lọc trung bình thường có dạng như sau:



*Trong đó Kwidth và Kheight* là chiều rộng và chiều cao của ma trận lọc.

Bộ lọc trung bình hoạt động dựa trên nguyên tắc lấy trung bình cộng các giá trị pixel xung quanh một pixel cụ thể. Điều này được thực hiện bằng cách sử dụng một ma trận (3x3), với các pixel được xét ở trung tâm.

Khi áp dụng bộ lọc trung bình, giá trị mới cho một pixel cụ thể (x,y) trong ảnh được tính như sau:



Trong đó :

* *Inew(x,y)* là giá trị mới cửa pixel tại vị trí (x,y).
* *l*(x + i,y + j) là giá trị của pixel tại vị trí (x + i,y + j) trong ảnh gốc.
* m và n là kích thước của ma trận lọc.
* a = và b = = .

1.2. Nền tảng công nghệ sử dụng

1.2.1 Ngôn ngữ python

Python là một ngôn ngữ lập trình hướng đối tượng bậc cao, mạnh mẽ được đưa ra bởi Guido van Rossum. Nó dễ học và cú pháp ngắn gọn giúp cho người học dễ dàng tiếp cận. Python là ngôn ngữ lập trình bậc cao lại có cách tiếp cận đơn giản, hiệu quả để lập trình hướng đối tượng. Với cú pháp đơn giản của python mà vẫn linh hoạt, rõ ràng, dễ hiểu để viết kịch bản và phát triển ứng dụng trong nhiều ngành nghề, lĩnh vực của cuộc sống.



Hình1. 8:Ngôn ngữ Python

=Python là một ngôn ngữ lập trình nổi bật với các tính năng:

* Ngôn ngữ bậc cao nhưng lại đơn giản và dễ hiểu: Python với cú pháp đơn giản và gọn gàng. Nó dễ học và lập trình hơn nhiều so với nhiều ngôn ngữ lập trình khác như C++, Java, C#. Ngôn ngữ python giúp cho việc lập trình trở nên dễ dàng, cho phép người lập trình có những giải pháp tốt hơn trong các lĩnh vực của cuộc sống.
* Mã nguồn mở là miễn phí: Ngôn ngữ sử dụng tự do,miễn phí cho mọi mục đích.Nó là mã nguồn mở, nên có thể sửa đổi một cách đơn giản. Ngôn ngữ lập trình python có một cộng đồng lớn đang liên tục được cập nhật mới.
* Khả năng lập trình trên nhiều hệ điều hành: Các mã chương trình python có thể di chuyển từ nền tảng này sang nền tảng khác mà không thay đổi cấu trúc của chương trình. Nó chạy được hầu hết trên các hệ điều hành phổ biến như Windows, macOs, Linux.
* Khả năng nhúng và mở rộng của chương trình: Python dễ dàng kết hợp các ngôn ngữ khác C, C++ và những ngôn ngữ lập trình khác vào cùng python. Điều này giúp mở rộng tính năng của ứng dụng.
* Ngôn ngữ thông dịch cấp cao: Khác với C/C++, python không khó khăn trong việc quản lý bộ nhớ. Python tự động quản lý bộ nhớ, giúp giải quyết các vấn đề phức tạp mà không cần quan tâm nhiều đến việc quản lý bộ nhớ.
* Thư viện tiêu chuẩn lớn và phổ biến: Python có một lượng lớn các thư viện tiêu chuẩn . Chỉ việc hiểu và ứng dụng các thư viện có sẵn được liên kết với python.
* Hướng đối tượng: Là ngôn ngữ lập trình hướng đối tượng (OOP) giúp giải quyết nhiều vấn đề phức tạp một cách trực quan.

Hiện nay python vẫn là ngôn ngữ lập trình được sử dụng phổ biến trong các lĩnh vực của đời sống như: Lập trình ứng dụng web, khoa học tính toán, dự báo thiên nhiên, chẩn đoán trong y học, phát triển trò chơi.

Ngoài những tính năng nổi bật như trên python còn có một số hạn chế đáng kể:

* Tốc độ tính toán, thực thi chậm.
* Sử dụng nhiều không gian bộ nhớ.
* Hạn chế trong việc phát triển ứng dụng trên thiết bị di động.
* Hạn chế trong việc thiết kế ứng dụng.

1.2.2 Thư viện

1.2.2.1 Thư viện OpenCV

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở sử dụng phổ biến trong 2 lĩnh vực chính là : thị giác máy tính và xử lý hình ảnh.Được phát triển bởi Intel và được nhà phát triển đóng góp từ khắp nơi trên thế giới. OpenCV được viết bằng ngôn ngữ C++, nhưng cung cấp các giao diện lập trình cho nhiều ngôn ngữ như Python, Java, và MATLAB.

OpenCV cung cấp các chức năng và công cụ mạnh mẽ để phân tích, xử lý. Dưới đây là một số tính năng chính của OpenCV:

* Xử lý ảnh: OpenCV cung cấp các chức năng để đọc, ghi, và xử lý hình ảnh. Thực hiện các thao tác như lọc ảnh, biến đổi hình ảnh, phát hiện cạnh, phát hiện đặc trưng, phân đoạn ảnh và nhiều hơn nữa.
* Xử lý video: OpenCV hỗ trợ xử lý video và phân tích video trực tiếp. Thực hiện các thao tác như ghi video, đọc video, trích xuất khung hình, xử lý khung hình, theo dõi đối tượng, và nhận dạng chuyển động.
* Machine learning: OpenCV tích hợp các thuật toán học máy và học sâu. Sử dụng các mô hình đã được huấn luyện trước hoặc huấn luyện các mô hình mới để phân loại, nhận dạng, và dự đoán đối tượng .
* Xử lý thời gian thực: OpenCV hỗ trợ xử lý hình ảnh và video trong thời gian thực. Tạo các ứng dụng thị giác máy tính có khả năng xử lý theo thời gian thực, chẳng hạn như xử lý hình ảnh từ camera trực tiếp.
* Tương tác với phần cứng: OpenCV cho phép tương tác với các thiết bị phần cứng như camera và cảm biến. Đọc dữ liệu từ camera, điều khiển camera và thực hiện các thao tác khác với các thiết bị ngoại vi.
* Đa nền tảng: OpenCV hỗ trợ nhiều trên các nền tảng như: Windows, Linux, macOS, iOS và Android. Có thể phát triển ứng dụng sử dụng OpenCV trên nhiều nền tảng khác nhau mà không cần thay đổi mã nguồn.

OpenCV là một thư viện mạnh và linh hoạt, được sử dụng trong nhiều các lĩnh vực như xử lý ảnh y tế, nhận dạng khuôn mặt, tự động hóa công nghiệp, xe tự hành.. Nó cung cấp tập hợp các công cụ và thuật toán tiện lợi để giúp bạn xây dựng ứng dụng thị giác máy tính mạnh mẽ.

1.2.2.2 Thư viện Numpy

Thư viện NumPy (Numerical Python) là một thư viện mã nguồn mở cho ngôn ngữ lập trình Python, cung cấp các công cụ và chức năng để làm việc với mảng và ma trận đa chiều. NumPy là một thư viện quan trọng trong cộng đồng khoa học dữ liệu và tính toán số, và nó là một phần không thể thiếu trong nhiều ứng dụng Python.

Các tính năng chính của thư viện NumPy áp dụng vào bài toán :

* Mảng đa chiều: NumPy tạo ra một cấu trúc dữ liệu gọi là ndarray (n-dimensional array) để lưu trữ mảng đa chiều. Mảng ndarray trong NumPy cho phép lưu trữ và xử lý dữ liệu nhanh chóng, hiệu quả và linh hoạt. Mảng này có thể có nhiều chiều, cho phép bạn thực hiện các phép toán trên các phần tử mảng .
* Phép toán số học và logic: NumPy cung cấp các hàm số học và logic để thực hiện các phép toán cơ bản như cộng, trừ, nhân, chia, luỹ thừa, căn bậc hai,… . Các phép toán này có thể được áp dụng cho mảng array và được thực hiện nhanh chóng và hiệu quả.
* Truy cập và cắt mảng: NumPy cho phép truy cập và cắt mảng dễ dàng thông qua các chỉ số và đánh dấu vị trí. Bạn có thể truy cập các phần tử, hàng, cột hoặc các khối con của mảng một cách linh hoạt và hiệu quả.
* Hỗ trợ cho ma trận và tính toán ma trận: NumPy cung cấp chức năng mạnh mẽ để làm việc với ma trận. Thao tác và xử lý dữ liệu: NumPy cung cấp nhiều công cụ và các chức năng để thao tác và xử lý dữ liệu trong mảng ndarray.

NumPy là một thư viện rất mạnh mẽ và quan trọng cho tính toán số và xử lý dữ liệu trong Python. Nó cung cấp một cơ sở chắc chắn cho việc phân tích dữ liệu, tính toán khoa học, và xử lý hình ảnh, và là một công cụ quan trọng trong lĩnh vực khoa học dữ liệu và machine learning.

1.2.2.3 Thư viện Tkinter

Tkinter là thư viện được sử dụng phổ biến nhất để phát triển GUI (Giao diện người dùng đồ họa) bằng Python. Nó là một giao diện Python tiêu chuẩn cho bộ công cụ Tk GUI đi kèm với Python. Vì Tk và Tkinter có sẵn trên hầu hết các nền tảng Unix cũng như trên hệ thống Windows, việc phát triển các ứng dụng GUI với Tkinter trở nên nhanh và dễ dàng.

Tkinter là mô-đun python có sẵn được sử dụng để tạo các ứng dụng GUI. Đây là một trong những mô-đun được sử dụng phổ biến nhất để tạo các ứng dụng GUI bằng Python vì nó đơn giản và dễ làm việc. Nó cung cấp một giao diện hướng đối tượng cho bộ công cụ Tk GUI.

Dưới đây là một số trường hợp sử dụng phổ biến cho Tkinter:

* Tạo cửa sổ và hộp thoại: Tkinter có thể được sử dụng để tạo cửa sổ và hộp thoại cho phép người dùng tương tác với chương trình của bạn. Chúng có thể được sử dụng để hiển thị thông tin, thu thập đầu vào hoặc trình bày các tùy chọn cho người dùng.
* Xây dựng GUI cho ứng dụng máy tính để bàn: Tkinter có thể được sử dụng để tạo giao diện cho ứng dụng máy tính để bàn, bao gồm các nút, menu và các yếu tố tương tác khác.
* Thêm GUI vào chương trình dòng lệnh: Tkinter có thể được sử dụng để thêm GUI vào chương trình dòng lệnh, giúp người dùng tương tác với chương trình và nhập đối số dễ dàng hơn.
* Tạo các widget tùy chỉnh: Tkinter bao gồm nhiều widget tích hợp, chẳng hạn như nút, nhãn và hộp văn bản, cho phép bạn tạo các widget tùy chỉnh của riêng mình.
* Tạo mẫu GUI: Tkinter có thể được sử dụng để nhanh chóng tạo nguyên mẫu GUI, cho phép bạn kiểm tra và lặp lại các ý tưởng thiết kế khác nhau .

1.2.2.4 Thư viện Pillow

Pillow là một thư viện xử lý ảnh mạnh mẽ và dễ sử dụng trong ngôn ngữ lập trình Python. Nó được phát triển như một nhánh của thư viện PIL (Python Imaging Library)

Pillow cung cấp nhiều tính năng mạnh mẽ như thao tác hình ảnh cơ bản, chuyển đổi định dạng ảnh, vẽ và văn bản, hiệu chỉnh màu sắc và độ sáng, và nhiều chức năng khác. Điều này giúp cho việc xử lý hình ảnh trở nên dễ dàng và linh hoạt.

Các tính năng chính:

* Thao tác với ảnh : Pillow cho phép thao tác cơ bản trên ảnh như cắt, xoay, lật, thay đổi kích thước và chỉnh sửa độ phân giải. Gíup quá trình xử lý ảnh trở nên thuận tiện, đặc biệt là trong việc chuẩn bị dữ liệu đầu vào cho các mô hình máy học.
* Chuyển đổi định dạng ảnh: Thư viện này hỗ trợ nhiều định dạng ảnh như JPEG, PNG, GIF, và BMP. Bạn có thể dễ dàng chuyển đổi giữa các định dạng này.

Ứng dụng của Pillow:

* Xử Lý Dữ Liệu Hình Ảnh: Trong khoa học dữ liệu và máy học, Pillow được sử dụng để xử lý và chuẩn bị dữ liệu hình ảnh cho việc huấn luyện mô hình.
* Đồ Họa Máy Tính: Cho phép tạo và chỉnh sửa đồ họa trực quan, biểu đồ, và hình ảnh cho ứng dụng đồ họa.

CHƯƠNG 2: CÁC PHƯƠNG PHÁP HIỆU CHỈNH ẢNH

2.1 Hiệu chỉnh độ sáng

Trong ảnh kỹ thuật số, giá trị tại mỗi điểm ảnh thể hiện cường độ sáng tại điểm đó. Do đó, phương pháp đơn giản nhất để hiệu chỉnh ánh sáng trong ảnh là thay đổi giá trị điểm ảnh của ảnh. Ý tưởng của phương pháp này là thay đổi một cách đồng đều giá trị tại mỗi điểm ảnh. Phương pháp này được thực hiện bằng cách cộng giá trị mỗi điểm ảnh với một số nguyên nằm trong khoảng [-255, 255].

* **Công thức điều chỉnh ánh sáng**

Công thức điều chỉnh ánh sáng được biểu diễn như sau:

Trong đó:

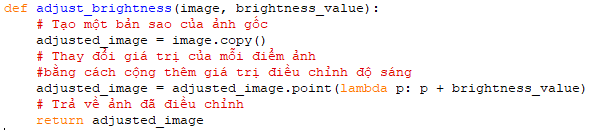
* *g(x,y)*đại diện cho giá trị cường độ điều chỉnh của pixel tại tọa độ *(x,y)* trong ảnh đầu ra.
* *f(x,y)*là giá trị cường độ ban đầu của pixel tại tọa độ *(x,y)* trong ảnh đầu vào.
* *b* là thành phần điều chỉnh độ sáng (khi *b* là giá trị âm, *g(x,y)* sẽ giảm so với *f(x,y),* đẫn đến việc làm tối ảnh , người lại *b* là giá trị dương ảnh sẽ sáng lên).
* **Thuật toán điều chỉnh ánh sáng**

Dưới đây là một cách tổng quát để mô tả hoạt động của đoạn mã điều chỉnh độ sáng trong ảnh:

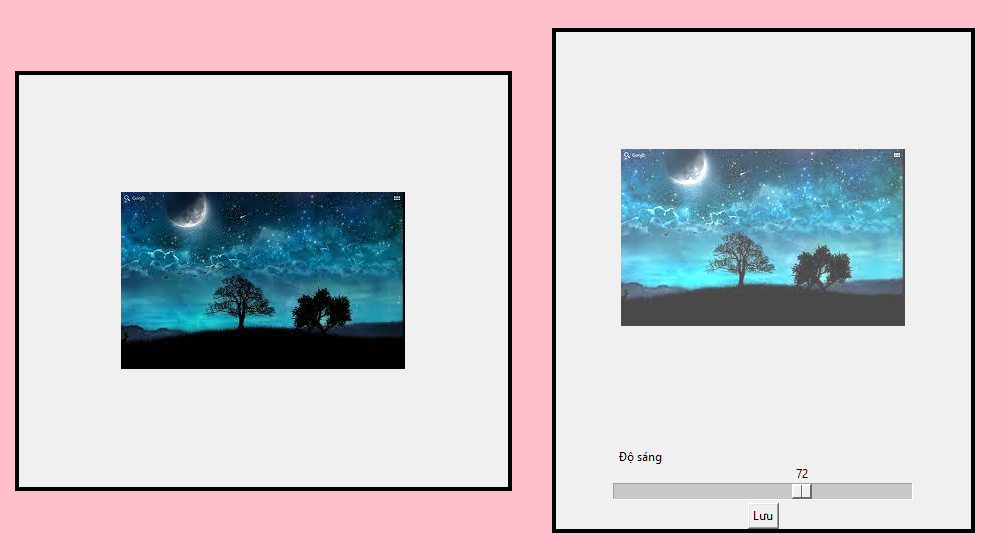
* **Input:**
* **Ảnh gốc**: Đây là ảnh mà chúng ta muốn điều chỉnh độ sáng. Ảnh này có thể có kích thước và độ phân giải khác nhau.
* **Giá trị điều chỉnh độ sáng**: Đây là giá trị mà chúng ta muốn thêm vào mỗi điểm ảnh của ảnh gốc để tăng hoặc giảm độ sáng.
* **Output:**
* **Ảnh đã điều chỉnh**: Đây là ảnh sau khi đã được điều chỉnh độ sáng. Ảnh này có cùng kích thước và độ phân giải với ảnh gốc, nhưng giá trị của mỗi điểm ảnh đã được thay đổi.
* **Hoạt động của thuật toán:**

Thuật toán điều chỉnh độ sáng hoạt động bằng cách thay đổi giá trị của mỗi điểm ảnh trong ảnh gốc. Điều này được thực hiện bằng cách cộng thêm giá trị điều chỉnh độ sáng vào giá trị ban đầu của mỗi điểm ảnh.

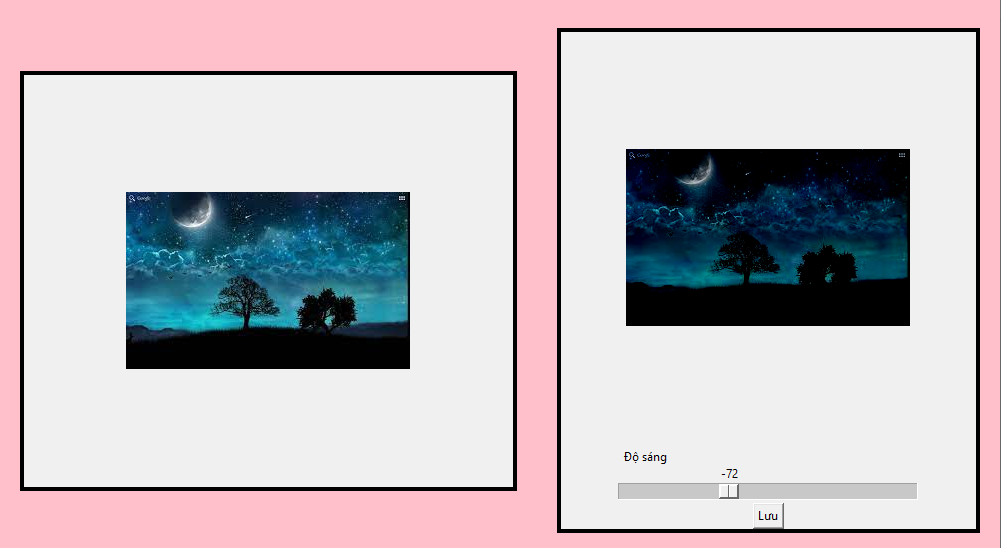
**Python:**



Trong đoạn mã trên, image là ảnh gốc và *brightness*\_value là giá trị điều chỉnh độ sáng. Hàm adjust\_*brightness*sẽ trả về ảnh sau khi đã được điều chỉnh độ sáng. Kết quả cuối cùng là ảnh đã điều chỉnh, với mỗi điểm ảnh có giá trị cường độ sáng đã được thay đổi theo giá trị điều chỉnh độ sáng. Nếu *brightness\_value* là một số âm, ảnh sẽ tối đi. Nếu *brightness\_value* là một số dương, ảnh sẽ sáng lên. Ảnh gốc không bị thay đổi trong quá trình này.



Hình 2.1:Ví dụ về chứ năng “Hiệu chỉnh ánh sáng tăng” với tham số là 72



Hình 2.2:Ví dụ về chứ năng “Hiệu chỉnh ánh sáng tăng” với tham số là -72

2.2 Hiệu chỉnh độ tương phản

Độ tương phản trong ảnh đóng vai trò quan trọng trong việc xác định chất lượng hình ảnh. Nó biểu thị sự khác biệt về độ sáng giữa các điểm ảnh trong hình ảnh. Một hình ảnh với độ tương phản cao thường sắc nét hơn và dễ nhìn hơn so với hình ảnh có độ tương phản thấp.

* **Điều chỉnh độ tương phản**

Điều chỉnh độ tương phản là quá trình thay đổi mức độ tương phản của hình ảnh để cải thiện chất lượng hình ảnh hoặc để phù hợp với yêu cầu cụ thể.

* **Input**
* **Ảnh gốc**: Một ma trận m x n, với m là chiều rộng và n là chiều cao của ảnh. Mỗi phần tử của ma trận biểu diễn một pixel, bao gồm giá trị màu đỏ, xanh lục và xanh lam.
* **Giá trị điều chỉnh độ tương phản C**: Một số nguyên biểu thị mức độ tương phản mong muốn.
* **Output**
* **Ảnh đã điều chỉnh**: Một ma trận m x n, tương tự như ảnh gốc nhưng giá trị của mỗi pixel đã được thay đổi theo công thức điều chỉnh độ tương phản.
* **Công thức điều chỉnh độ tương phản**

Bước đầu tiên trong quá trình này là tính toán hệ số hiệu chỉnh độ tương phản (F), được tính theo công thức sau:

Trong đó, C biểu thị mức độ tương phản mong muốn. Để thuật toán hoạt động chính xác, giá trị cho F cần phải được lưu trữ dưới dạng số dấu phẩy động và không phải là một số nguyên.

Bước tiếp theo là thực hiện việc điều chỉnh độ tương phản thực tế. Công thức sau đây cho thấy sự điều chỉnh độ tương phản được thực hiện đối với thành phần màu đỏ của màu:

Tương tự, công thức cho thành phần màu xanh lục và xanh lam là:

Trong đó, ***Truncate*** là một thủ tục đảm bảo rằng các giá trị mới của màu đỏ, xanh lục và xanh lam nằm trong phạm vi hợp lệ từ 0 đến 255.Trong không gian màu RGB, giá trị màu của mỗi kênh (đỏ, xanh lục, xanh lam) nằm trong khoảng [0,255].Gía trị trung bình trong phạm vi này là 128.

Khi chúng ta trừ đi 128 từ giá trị màu gốc, chúng ta đang dịch chuyển giá trị màu sao cho giá trị trung bình (128) trở thành 0. Điều này tạo ra một phạm vi mới từ -128 đến 127.

Sau đó, chúng ta nhân giá trị màu đã được dịch chuyển này với hệ số hiệu chỉnh độ tương phản F. Điều này làm thay đổi độ tương phản của hình ảnh.

Cuối cùng, chúng ta cộng thêm 128 vào giá trị màu đã được điều chỉnh. Điều này dịch chuyển phạm vi giá trị màu trở lại phạm vi ban đầu từ [0, 255].

Vì vậy, giá trị 128 được sử dụng như một điểm trung tâm để dịch chuyển giá trị màu trước và sau khi điều chỉnh độ tương phản. Điều này giúp đảm bảo rằng giá trị màu sau khi điều chỉnh vẫn nằm trong phạm vi hợp lệ từ [0, 255].

* **Phạm vi của hệ số hiệu chỉnh độ tương phản F**

Phạm vi của hệ số hiệu chỉnh độ tương phản F phụ thuộc vào giá trị của C, mức độ tương phản mong muốn :

* **Khi C = -255**: Điều này có nghĩa là bạn muốn giảm độ tương phản xuống mức tối thiểu. Khi đó, F sẽ tiến dần về 0. Khi F = 0, mọi giá trị màu sẽ trở thành 128, tức là tất cả các pixel đều có cùng mức độ sáng, làm cho hình ảnh trở nên hoàn toàn mờ.
* **Khi C = 0**: Điều này có nghĩa là bạn không muốn thay đổi độ tương phản của hình ảnh. Khi đó, F = 1, tức là hình ảnh sẽ không thay đổi.
* **Khi C = 255**: Điều này có nghĩa là bạn muốn tăng độ tương phản lên mức tối đa. Khi đó, F sẽ tiến dần về vô cùng. Khi F rất lớn, mọi giá trị màu nhỏ hơn 128 sẽ trở thành 0 và mọi giá trị màu lớn hơn 128 sẽ trở thành 255, làm cho hình ảnh trở nên rất sắc nét nhưng cũng có thể làm mất đi chi tiết.
* **Thuật toán điều chỉnh độ tương phản**

Giả sử bạn có một hình ảnh I với kích thước m×n và một số nguyên b biểu thị mức độ tương phản mong muốn. Dưới đây là thuật toán để điều chỉnh độ tương phản của hình ảnh:

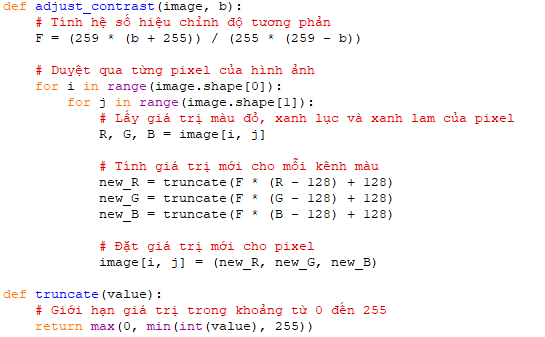
* Tính hệ số hiệu chỉnh độ tương phản F:
* Duyệt qua từng pixel (i,j) của hình ảnh I:

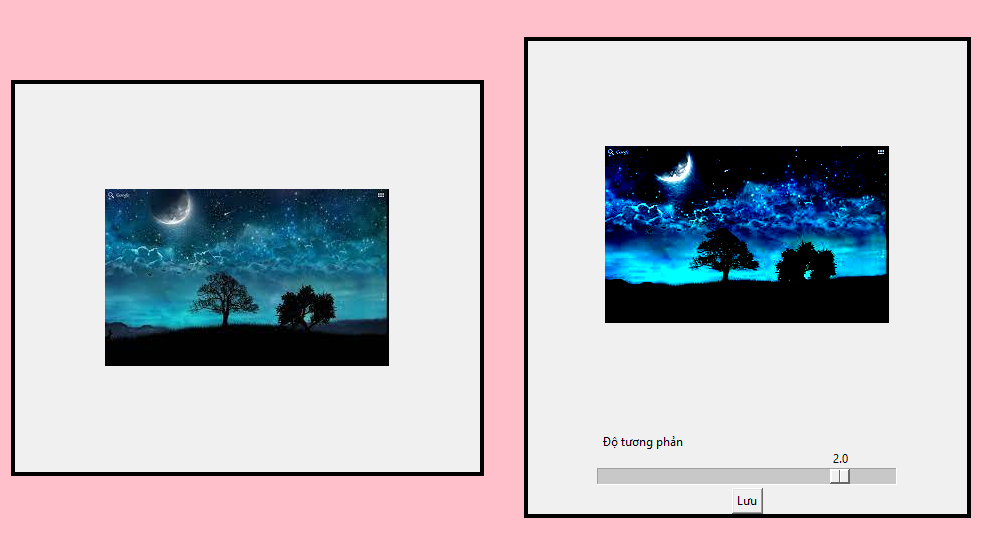
Lấy giá trị màu đỏ, xanh lục và xanh lam của pixel (i,j), ký hiệu là R, G, B.

* Tính giá trị mới cho mỗi kênh màu.
* Đặt giá trị mới cho pixel (i,j) là (R′**,**G′**,**B′).

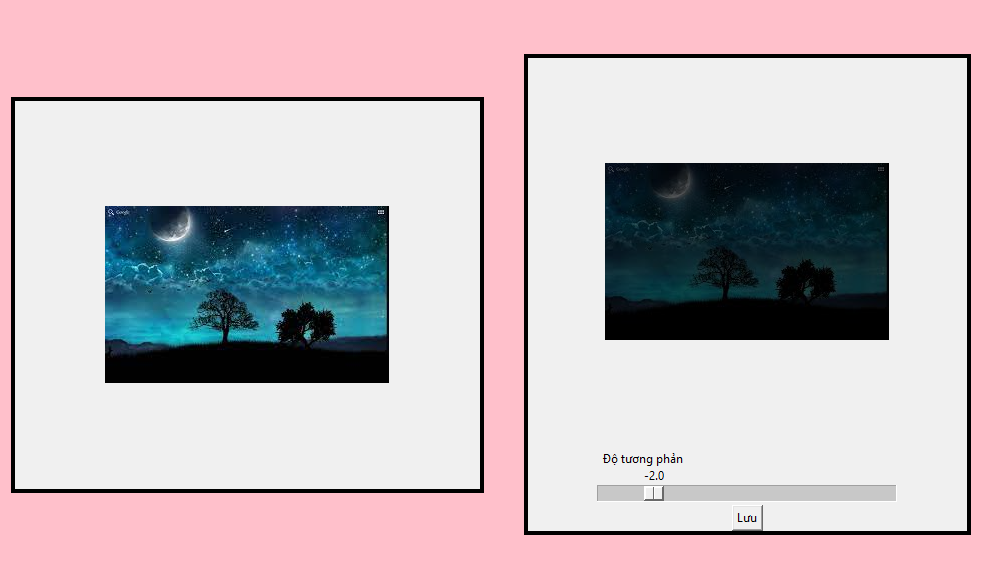
Trong đó, hàm ***Truncate*** được sử dụng để đảm bảo rằng giá trị mới của màu đỏ, xanh lục và xanh lam nằm trong phạm vi hợp lệ từ 0 đến 255.

* **Python:**





Hình 2.3:Hình ảnh với độ tương phản 2.0

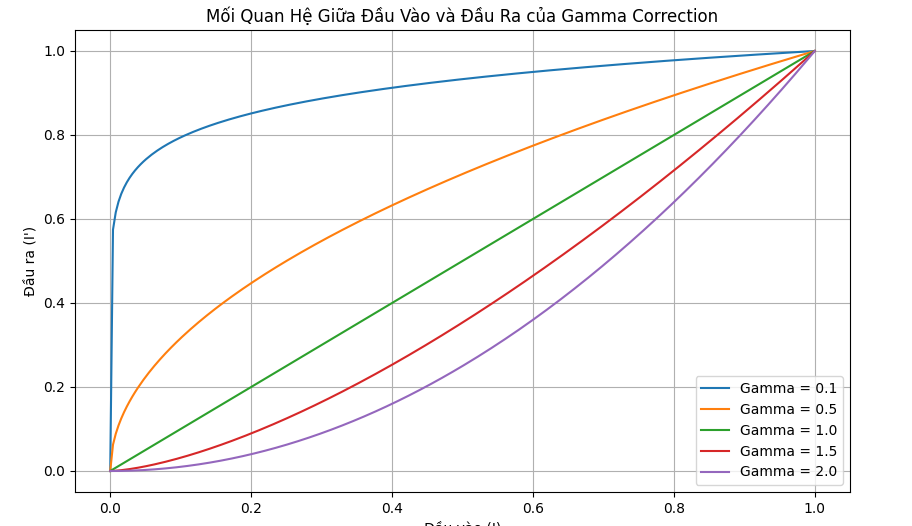


Hình 2.4:Hình ảnh với độ tương phản -2.0

2.3 Hiệu chỉnh gamma

Khi hiển thị hình ảnh trên màn hình máy tính, một vấn đề thường gặp phải là độ nhạy sáng . Đặc điểm chung của hầu hết các loại màn hình là khi xuất kết quả, giá trị đầu ra thường là một hàm mũ của giá trị đầu vào. Điều này có thể làm giảm chất lượng hình ảnh, khiến hình ảnh hiển thị trên màn hình thường tối hơn so với bình thường.

Gamma mô tả mối quan hệ giữa giá trị đầu vào (I) và kết quả đầu ra (I’). Trong trường hợp này, đầu vào (I) là giá trị cường độ RGB của hình ảnh.



Hình 2.5: Biểu diễn các giá trị gamma

* **Mối Quan Hệ Giữa Đầu Vào và Đầu Ra**

Mối quan hệ giữa đầu vào (I) và đầu ra (I’) được biểu diễn bằng công thức sau:

**I′=Iγ**

Khi gamma bằng 1, đầu vào (I) sẽ bằng đầu ra (I’), tạo ra một đường thẳng. Để thực hiện hiệu chỉnh gamma, giá trị đầu vào (I) được nâng lên lũy thừa của nghịch đảo gamma. Công thức cho việc này như sau:

**I′=I(1/γ)**

* **Ảnh Hưởng Của Giá Trị Gamma**
* Khi **γ>1**, mối quan hệ giữa (I) và (I’) trở nên nhạy cảm hơn, với giá trị (I’) tăng nhanh hơn so với (I). Điều này dẫn đến việc tăng độ tương phản và làm nổi bật các chi tiết ở các mức độ sáng hoặc tối.
* Khi **0<γ<1**, mối quan hệ trở nên giảm nhạy, với giá trị (I’) tăng chậm hơn so với (I). Điều này có thể dẫn đến việc giảm độ tương phản và làm mịn các chi tiết.
* Khi **γ=1**, mối quan hệ trở thành một đường thẳng, và không có sự biến đổi ngoại trừ sự biến đổi tỷ lệ thuận tuyến tính.
* **Điều chỉnh gamma**

Điều chỉnh gamma là một phương pháp xử lý hình ảnh phổ biến, được sử dụng để thay đổi độ sáng của hình ảnh. Nó hoạt động bằng cách thay đổi giá trị màu sắc của từng pixel trong hình ảnh theo một hàm mũ.

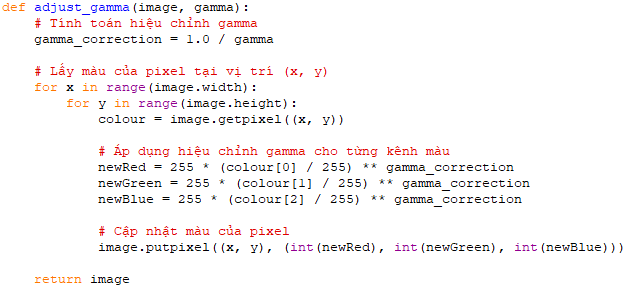
* **Input:**
* **Ảnh gốc:** Một ma trận **m x n**, với **m** là chiều rộng và **n** là chiều cao của ảnh. Mỗi phần tử của ma trận biểu diễn một pixel, bao gồm giá trị màu đỏ, xanh lục và xanh lam.
* **Giá trị gamma (G):** Một số thực biểu thị mức độ điều chỉnh gamma mong muốn.
* **Output:**
* **Ảnh đã điều chỉnh:** Một ma trận **m x n**, tương tự như ảnh gốc nhưng giá trị của mỗi pixel đã được thay đổi theo công thức điều chỉnh gamma.
* **Công thức Điều Chỉnh Gamma**

Cụ thể hơn, giá trị gammaCorrection được sử dụng trong công thức hiệu chỉnh gamma như sau:

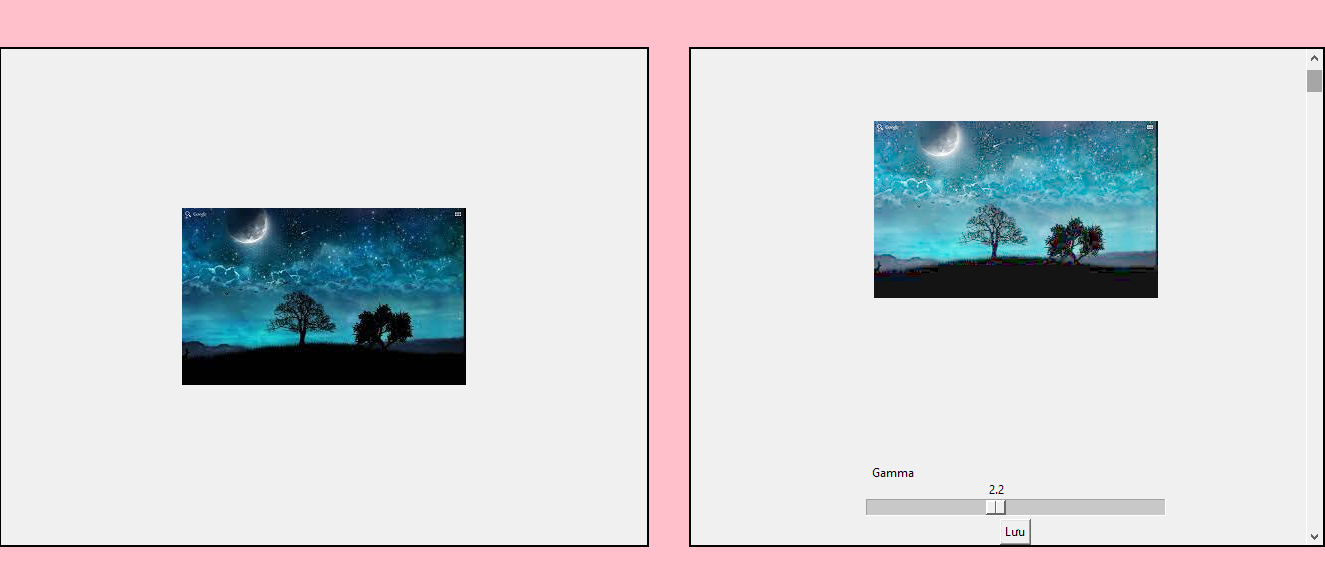
Trong đó, “I” là giá trị cường độ RGB ban đầu của một pixel (trong khoảng từ 0 đến 255), và “I’” là giá trị cường độ RGB mới sau khi đã được hiệu chỉnh gamma.

* **Biểu diễn Thuật Toán**

Dưới đây là mã giả mô tả cách thực hiện hiệu chỉnh gamma:

****

Trong thuật toán này, chúng ta duyệt qua từng pixel của hình ảnh và áp dụng công thức điều chỉnh gamma cho mỗi kênh màu của mỗi pixel. Phạm vi giá trị được sử dụng cho gamma sẽ tùy thuộc vào ứng dụng trong xử lý ảnh.



Hình 2.6:Ví dụ chức năng “Hiệu chỉnh gamma” với tham số là 2.2

2.4 Cân bằng màu

Cân bằng màu là một kỹ thuật xử lý ảnh để cải thiện độ tương phản và sắc thái của ảnh. Có nhiều phương pháp cân bằng màu khác nhau, nhưng một trong những phương pháp phổ biến nhất là cân bằng histogram. Trong bài viết này, chúng ta sẽ tìm hiểu về histogram, cách tính và cân bằng histogram bằng Python. Chúng ta cũng sẽ thử áp dụng phương pháp này cho một ảnh mẫu và so sánh kết quả.

* **Histogram**

Histogram là biểu đồ thống kê số lần xuất hiện của các mức sáng trong ảnh. Một ảnh có histogram cân bằng là ảnh có phân bố đều các mức sáng từ đen đến trắng, không bị chệch về một phía nào. Histogram biểu diễn độ tương phản và sắc thái của ảnh. Một ảnh có độ tương phản cao sẽ có histogram có nhiều cột cao và rộng, phân bố trên toàn dải mức sáng. Một ảnh có độ tương phản thấp sẽ có histogram có nhiều cột thấp và hẹp, tập trung ở một vùng nhỏ mức sáng. Một ảnh có sắc thái đa dạng sẽ có histogram có nhiều cột có giá trị khác nhau. Một ảnh có sắc thái đơn điệu sẽ có histogram có ít cột có giá trị gần nhau.

Cân bằng histogram là quá trình biến đổi các giá trị pixel của ảnh sao cho histogram của ảnh mới có phân bố đều hơn. Mục tiêu của cân bằng histogram là làm cho ảnh có độ tương phản cao hơn, sắc thái rực rỡ hơn và không bị chệch về một phía nào.

* **Hàm biến đổi**

Để cân bằng histogram, ta cần xác định một hàm biến đổi **K(i)** với i là giá trị pixel ban đầu, và **K(i)** là giá trị pixel mới. Hàm **K(i)** có thể được tính theo công thức sau:

Trong đó **Z(i)** là hàm tích lũy của histogram, được tính bằng tổng số pixel có giá trị nhỏ hơn hoặc bằng **i**. **min(Z)** và **max(Z)** là giá trị nhỏ nhất và lớn nhất của **Z(i).**

Để tính toán hàm biến đổi **K(i)**, ta cần xác định hàm tính lũy **Z(i)** được tính như sau:

* Tính histogram của ảnh gốc và lưu vào mảng **H[0…L-1].**
* Khởi tạo mảng **Z[0…L-1]** với giá trị **Z(0) = H(0).**
* Duyệt qua từng mức sáng **i** từ **1** đến **L-1**: a. Tính **Z(i) = Z(i-1) + H(i).**

Hàm biến đổi **K(i)** dựa vào hàm tích lũy của histogram để chuẩn hóa các giá trị pixel về khoảng [0, 255]. Hàm biến đổi **K(i)** có tính chất là đồng biến, nghĩa là nếu **i <** **j** thì **K(i) < K(j).** Hàm biến đổi **K(i)** cũng có tính chất là bảo toàn diện tích, nghĩa là tổng số pixel của ảnh gốc và ảnh mới bằng nhau.

* **Cách áp dụng hàm biến đổi**

Sau khi có hàm **K(i),** ta thay thế mỗi pixel của ảnh gốc bằng giá trị mới theo hàm **K(i).** Kết quả là ảnh mới có histogram cân bằng hơn. Cách áp dụng hàm biến đổi có thể được mô tả bằng thuật toán sau:

**Input:** Ảnh gốc **f(x, y)** có kích thước **M x N**, số mức sáng **L**.

**Output:** Ảnh mới **g(x, y)** có histogram cân bằng.

* Bước 1: Tính histogram của ảnh gốc, lưu vào mảng **H[0…L-1].**
* Bước 2: Tính hàm tích lũy của histogram, lưu vào mảng **Z[0…L-1].**
* Bước 3: Tìm giá trị nhỏ nhất và lớn nhất của hàm tích lũy, lưu vào biến **minZ** và **maxZ**.
* Bước 4: Tính hàm biến đổi **K(i)** cho mỗi mức sáng **i**, lưu vào mảng **K[0…L-1].**
* Bước 5: Duyệt qua mỗi pixel của ảnh gốc, lấy giá trị pixel là **i**, thay thế bằng giá trị mới là **K[i],** lưu vào ảnh mới.
* Bước 6: Trả về ảnh mới.
* **Chọn không gian màu**

Cân bằng histogram có thể được áp dụng cho ảnh xám hoặc ảnh màu.Tuy nhiên, với ảnh màu , việc chọn không gian màu đúng là quan trọng để tránh thay đổi màu sắc không mong muốn. Một số không gian màu phổ biến như RGB,CMYK, HSV,HSL,…

* **Ảnh xám**

Đối với ảnh xám, quá trình này được thực hiện trên một kênh duy nhất đại diện cho độ sáng.

Trước tiên, chúng ta chuyển đổi ảnh màu sang ảnh xám để làm giảm chiều sâu màu sắc, chỉ giữ lại thông tin độ sáng.



Sau khi có ảnh xám, chúng ta sử dụng hàm cv2.equalizeHist() để áp dụng cân bằng histogram.





Hình 2.7:Hình ảnh trước và sau khi cân bằng màu với ảnh xám

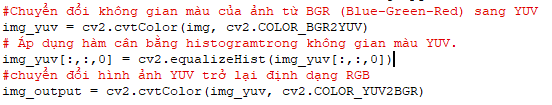
* **Ảnh màu**

Với ảnh màu, không nên cân bằng histogram trên không gian màu RGB trực tiếp. Bởi vì RGB biểu diễn màu sắc bằng ba kênh đỏ, lục và lam, mỗi kênh có ảnh hưởng đến cả độ sáng và độ bão hòa của ảnh. Nếu cân bằng histogram trên từng kênh RGB riêng biệt, màu sắc của ảnh có thể thay đổi không mong muốn.

Mặc định ảnh màu là hệ RGB hoặc BGR, muốn cân bằng sáng ta cần biến đổi về hệ màu HSV. Hệ màu HSV bao gồm 3 chanel:

* H-HUE: giá trị màu
* S-SATURATION: độ bảo hòa.
* V- VALUE: độ sáng của màu sắc.

Ta sẽ áp dụng cân bằng histogram chỉ trên độ sáng V của ảnh:





Hình 2.8:Hình ảnh trước và sau khi cân bằng màu với ảnh màu

2.5 Hiệu ứng mờ viền

"Hiệu ứng mờ viền" trong ảnh (Vignette) là một kỹ thuật tạo ra một hiệu ứng mờ ở các cạnh của ảnh. Điều này được thực hiện bằng cách áp dụng một mặt nạ lên ảnh, với giá trị pixel giảm dần từ trung tâm ra các cạnh.. Kỹ thuật này tập trung vào việc làm mờ các vùng ở góc ngoài cùng của hình ảnh, tạo ra một hiệu ứng tập trung ánh sáng ở trung tâm và tăng cường sự chú ý đối với các phần quan trọng hoặc trung tâm của hình ảnh.

* **Thuật toán và Công thức Toán học**

Thuật toán và công thức toán học được áp dụng trong hiệu ứng Vignette như sau:

**Input:**

* **Hình ảnh gốc**: Hình ảnh muốn áp dụng hiệu ứng Vignette. Hình ảnh được thực hiện vào chương trình thông qua một đường dẫn tệp hoặc đối tượng hình ảnh.
* **Tham số radius:** Tham số này điều chỉnh độ lớn của vùng mờ trong hiệu ứng Vignette. Giá trị lớn hơn của **radius** sẽ tạo vùng mờ lớn hơn.
* **Tham số focus\_x** và **focus\_y:** Điều chỉnh vị trí của vùng tập trung (vùng không bị mờ) trong hiệu ứng. Giá trị của **focus\_x** và **focus\_y** thay đổi tương ứng với trung tâm hình ảnh

**Output:**

* **Hình ảnh đã được áp dụng hiệu ứng Vignette:** Đây là hình ảnh sau khi đã được áp dụng hiệu ứng Vignette. Hình ảnh này sẽ có các cạnh mờ, tạo ra một hiệu ứng như đang nhìn qua ống kính máy ảnh hoặc ống nhòm. Hình ảnh này có thể được lưu lại dưới dạng tệp tin hoặc được hiển thị trực tiếp trong chương trình.
* **Chuyển đổi hình ảnh thành mảng NumPy**: Hình ảnh ban đầu được chuyển đổi thành một mảng NumPy để dễ dàng cho việc xử lý vì **Numpy** là thư viện mạnh mẽ để hỗ trợ tính toán trên mảng nhiều chiều. Nếu không chuyển đổi hình ảnh thành mảng **Numpy,** việc thao tác trên các pixel của hình ảnh sẽ trở lên khó khăn và không hiệu quả.
* **Tính toán mặt nạ Vignette:** Một mặt nạ với kích thước tương tự như hình ảnh được tạo ra. Trung tâm của mặt nạ được xác định, và sau đó, khoảng cách từ mỗi pixel đến trung tâm được tính toán. Công thức toán học tính khoảng cách Euclidean trong không gian hai chiều là:

Trong đó:

* ***i*** và ***j*** là tọa độ của pixel.
* **center** là tọa độ của trung tâm mặt nạ, với **center[1]** tương ứng với tọa độ **x** và **center[0]** tương ứng với tọa độ **y**.
* **Cập nhật giá trị của mặt nạ:** Giá trị của mỗi pixel trong mặt nạ được cập nhật dựa trên khoảng cách này. Công thức toán học để cập nhật giá trị pixel trong mặt nạ là:

Trong đó:

* **mask[i, j]** là giá trị của pixel tại vị trí**(i, j)** trong mặt nạ.
* **distance**là khoảng cách từ pixel tại vị trí**(i, j)** đến trung tâm của mặt nạ.
* **radius**và focus là các tham số điều chỉnh mức độ của hiệu ứng Vignette.
* **255**là giá trị pixel tối đa trong một hình ảnh 8-bit.

Nếu **distance**tăng ( tức là pixel càng xa trung tâm), thì giá trị trong ngoặc **(1 - distance / (radius \* focus))** sẽ giảm, do đó giá trị của **mask[i, j]** cũng giảm, tạo ra hiệu ứng mờ.

Nếu **radius**hoặc **focus**tăng, thì giá trị trong ngoặc sẽ tăng (do mẫu số lớn hơn), do đó giá trị của **mask[i, j]** cũng tăng, làm giảm hiệu ứng mờ.

* **Áp dụng mặt nạ lên hình ảnh:** Mặt nạ sau đó được áp dụng lên hình ảnh bằng cách nhân mỗi kênh màu của hình ảnh với mặt nạ. Công thức toán học để áp dụng mặt nạ lên hình ảnh là:

Trong đó:

* **Img\_array** là mảng Numpy biểu diễn hình ảnh gốc.
* **Mask** là mặt nạ đã Vignette đã tạo ra.
* **255** là giá trị pixel tối đa trong một hình ảnh 8-bit.



* **Chuyển đổi không gian màu:** Việc chuyển đổi không gian màu từ RGB sang HSV giúp chúng ta chỉnh sửa độ sáng (Value) của hình ảnh mà không ảnh hưởng đến màu sắc. Nếu chúng ta giữ nguyên không gian màu RGB hoặc chuyển sang một không gian màu khác không phân tách rõ ràng giữa độ sáng và màu sắc, việc áp dụng mặt nạ có thể làm thay đổi màu sắc của hình ảnh, không chỉ độ sáng. Điều này có thể tạo ra hiệu ứng không mong muốn.

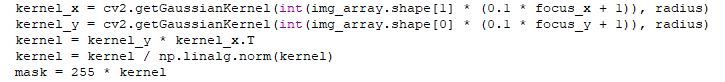


* **Áp dụng mặt nạ lên kênh giá trị (Value):** Mặt nạ được áp dụng chỉ lên kênh giá trị (Value) của hình ảnh, thay vì tất cả các kênh màu. Điều này tạo ra hiệu ứng mà độ sáng của hình ảnh giảm dần từ trung tâm ra các cạnh, trong khi màu sắc của hình ảnh vẫn được giữ nguyên.



**Tạo mặt nạ Gaussian:** Mặt nạ Gaussian được tạo ra bằng cách nhân hai kernel Gaussian, một theo chiều ngang và một theo chiều dọc. Kernel Gaussian được tạo ra bằng hàm cv2.getGaussianKernel, với đầu vào là kích thước của kernel và độ lệch chuẩn (được xác định bởi tham số radius).

Đầu tiên, chúng ta tạo kernel Gaussian theo chiều ngang (kernel\_x) và chiều dọc (kernel\_y).Tiếp theo, chúng ta tạo ra mặt nạ Gaussian bằng cách nhân kernel\_y với chuyển vị của kernel\_x.Cuối cùng, chúng ta chuẩn hóa mặt nạ Gaussian để có tổng bằng 1 và nhân với 255 để có giá trị pixel trong phạm vi từ 0 đến 255:



2.6 Xoay ảnh

Chức năng xoay ảnh là một công cụ hữu ích trong việc chỉnh sửa ảnh số, cho phép người dùng tùy chỉnh góc quay của ảnh theo ý muốn của họ. Điều này giúp cải thiện chất lượng hình ảnh và tạo ra hiệu ứng nghệ thuật độc đáo. Người dùng có thể dễ dàng điều chỉnh góc quay của ảnh và xem trước kết quả ngay trên màn hình.

Phát triển chức năng xoay ảnh trong xử lý ảnh số đòi hỏi sử dụng các thuật toán và phép biến đổi xử lý ảnh. Các thuật toán này có thể áp dụng để đảm bảo tính chính xác và hiệu suất của chức năng.

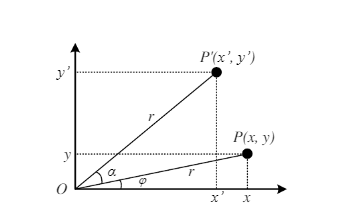
Chức năng xoay ảnh được xây dựng trên một số phép biến đổi sau :

* **Phép biến đổi hình học:** Phép xoay ảnh là một đa dạng của phép biến đổi hình học, nơi mà mỗi điểm ảnh được di chuyển đến một vị trí mới dựa trên quy tắc phép biến đổi tọa độ trong không gian hai chiều( phép xoay quay quanh tâm).

Phép xoay trong không gian 2 chiều ta xét phép quay (Rotation) quanh gốc toạ độ O một góc .

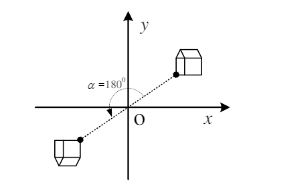
Xét điểm P(x, y) , quay đến vị trí P’(x’, y’) với góc quay . Ta quy ước phép quay ngược chiều kim đồng hồ là chiều dương và cùng chiều kim đồng hồ là chiều âm.Xét phép quay quanh gốc toạ độ.Được biểu diễn qua phép toán học của phép quay:

Trong đó góc quay và là góc quay ban đầu.



Hình 2.9: Phép quay hình

Phép đối xứng quay quanh gốc tọa độ :



Hình 2.10:Phép đối xứng

Trong hệ trục tọa độ ta có:

Biểu diễn qua (x,y) ta có :

Trong đó (x,y) là tọa độ ban đầu của điểm, là góc xoay, và (x′,y′) là tọa độ mới của điểm sau khi xoay.

* **Phép dịch chuyển:** Trước khi xoay, chúng tôi dịch chuyển ảnh sao cho tâm của ảnh trùng với gốc tọa độ. Sau khi xoay, chúng tôi dịch chuyển ảnh trở lại vị trí ban đầu.
* **Điều chỉnh kích thước ảnh:** Khi xoay ảnh, kích thước của ảnh sẽ thay đổi. Để giữ nguyên tỷ lệ khung hình, chúng tôi sử dụng công thức sau để tính toán chiều rộng và chiều cao mới của ảnh sau khi xoay:

**new\_width=width.cos(angle\_rad)+height.sin(angle\_rad)​**

**new\_height=width.sin(angle\_rad)+height.cos(angle\_rad)**​

Ở đây, **angle\_rad** là góc xoay được chuyển đổi từ độ sang radian, width và height là chiều rộng và chiều cao ban đầu của ảnh.

* **Thuật toán**

**Input:**

* **Một ảnh:** Đây là ảnh mà người dùng muốn xoay.
* **Góc xoay:** Góc mà người dùng muốn xoay ảnh, được nhập qua thanh trượt hoặc hộp nhập.

**Output:**

* **Ảnh đã được xoay:** Đây là ảnh sau khi đã được xoay theo góc mà người dùng đã chọn. Ảnh này sẽ được hiển thị trên giao diện người dùng và có thể được lưu lại nếu người dùng muốn.

Phép xoay trong không gian hai chiều có một số tính chất quan trọng:

**Tính liên tục:** Phép xoay là một phép biến đổi liên tục, tức là nếu ta xoay một điểm dần dần, điểm đó sẽ di chuyển một cách liên tục trong không gian.

**Tính khả nghịch**: Phép xoay là khả nghịch. Nếu ta xoay một điểm theo góc , ta có thể xoay nó trở lại vị trí ban đầu bằng cách xoay theo góc −.

**Tính kết hợp:** Phép quay tuân theo tính chất kết hợp. Nghĩa là, nếu ta xoay một điểm theo góc 1 và sau đó xoay theo góc 2, kết quả sẽ giống như khi ta xoay điểm đó một lần theo góc 1+2​

**Bảo toàn khoảng cách:** Khoảng cách giữa hai điểm không thay đổi sau khi cả hai điểm đều bị xoay.

**Bảo toàn góc:** Góc giữa hai đường thẳng không thay đổi sau khi cả hai đường thẳng đều bị xoay.



Hình 2.11: ví dụ chức năng xoay ảnh

CHƯƠNG 3: XÂY DỰNG THỬ NGHIỆM PHẦN MỀM XỬ LÝ ẢNH

3.1 Môi trường thực hiện

* Máy phát triển : Windows 10 i7- 10510U, CPU@ 1.80GHZ, RAM: 8GB.
* Môi trường phát triển trên local: PYCHAMP và IDLE
* Ngôn ngữ phát triển: PYTHON.

3.2 Chương trình thử nghiệm

Chương trình ”Phần mềm xử lý ảnh” ứng dụng các thuật toán để thực hiện các chức năng xử lý ảnh được trình bày trong báo cáo đồ án. Chương trình được xây dựng trên nền tảng ngôn ngữ lập trình python và trên nền tảng Window.

Chương trình được xây dựng với các chức năng sau :

Chức năng “Tệp” xử lý gồm các chức năng :

* Chức năng “Mở “: thực hiên mở file tệp tin để xử lý.
* Chức năng “Thoát”: thực hiện chức năng thoát chương trình.

Chức năng “Chỉnh sửa” xử lý gồm các chức năng :

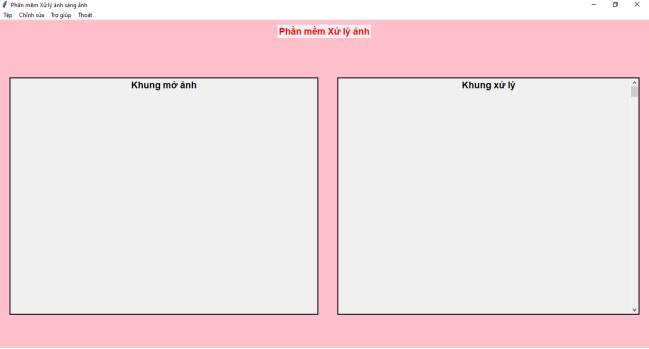
* Chức năng “Điều chỉnh độ sáng”: thực hiện việc tăng giảm độ sáng của trong bức ảnh, áp dụng thuật toán “ hiệu chỉnh độ sáng”.
* Chức năng “Điều chỉnh độ tương phản”: thực hiện việc tăng giảmtương phản của trong bức ảnh, áp dụng thuật toán “ hiệu chỉnh độ tương phản”.
* Chức năng “Điều chỉnh gamma”: thực hiện điều chỉnh gamma,áp dụng thuật toán “ hiệu chỉnh gamma”.
* Chức năng “Cân bằng màu”: thực hiện điều chỉnh cân bằng màu trong bức ảnh giúp cho ảnh có độ sáng đồng đều, áp dụng thuật toán “ cân bằng màu”.
* Chức năng “Hiệu ứng mờ viền”: thực hiện điều chỉnh hiệu ứng mờ viền xung quanh bức ảnh, áp dụng thuật toán “ Hiệu ứng mờ viền ”.

Chức năng “Trợ giúp”: Giúp người dùng hiểu rõ về ứng dụng.

Chức năng “Thoát”:Giúp người dùng.

3.3 Code các module chương trình

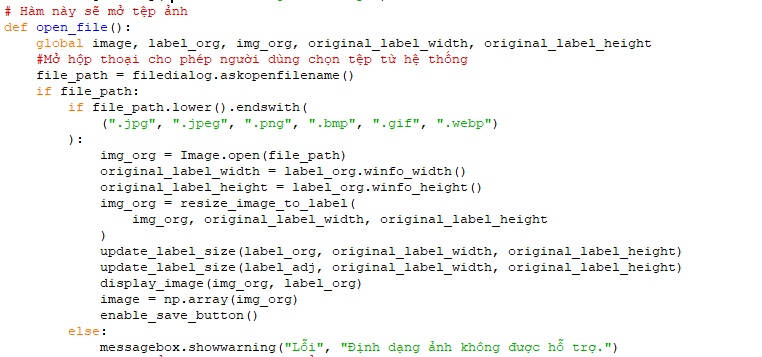
Giao diện chính của trương trình xây dựng bằng python.

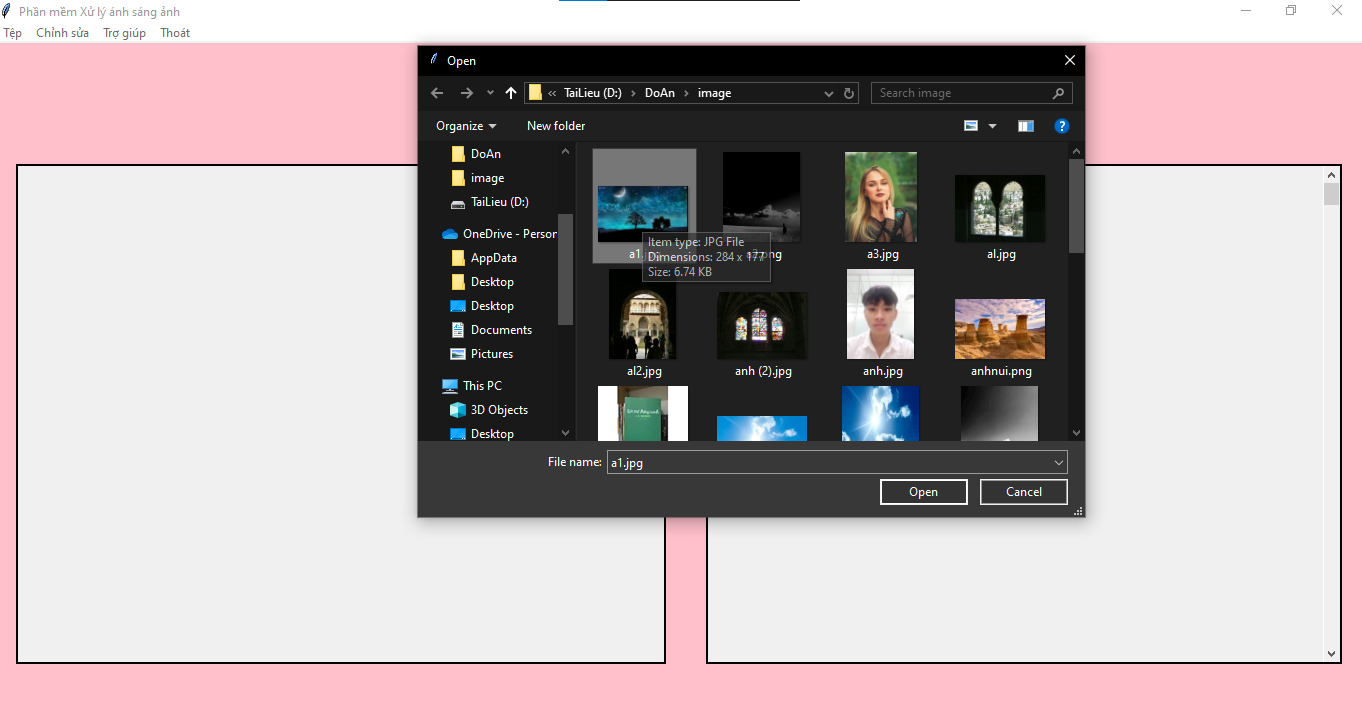


Hình 3.1: Giao diện chính chương trình

Đầu tiên cần mở ảnh để chỉnh sửa chọn vào menu “Tệp” và chọn vào phần “Mở” để chỉnh hiển thị thư mục chọn ảnh để chỉnh sủa:

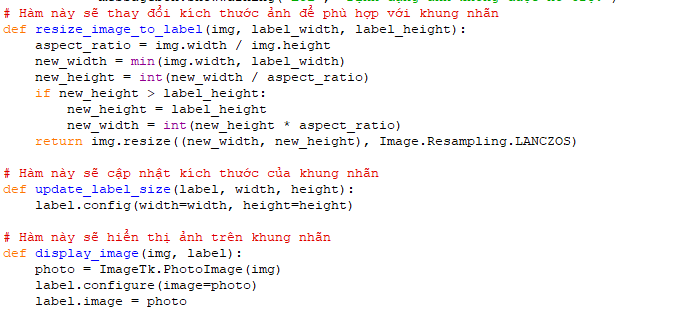
Tronng đó hàm mở được định nghĩa một hàm “***open\_file***” cho phép người dùng chọn ảnh từ hệ thống. Kiểm tra xem ảnh được chọn có phải là ảnh hợp lệ hay không. Nếu không hợp lệ sẽ thông báo cho người dùng với thông báo cảnh báo.





Hình 3.2: Hiển thị thư mục chọn ảnh

Để ảnh hiển thị phù hợp với khung hình trong ứng dụng sẽ tạo một hàm.Hàm này nhận vào một ảnh và kích thước của nhãn mà ảnh sẽ được hiển thị. Nó tính tỷ lệ giữa chiều rộng và chiều cao của ảnh, sau đó điều chỉnh kích thước của ảnh để phù hợp với nhãn, đồng thời giữ nguyên tỷ lệ này. Kết quả là một ảnh đã được điều chỉnh kích thước mà không làm thay đổi tỷ lệ giữa chiều rộng và chiều cao.Sau đó cập nhật kích thước của nhãn để phù hợp kích thước mới của ảnh. Đảm bảo rằng nhãn luân có kích thước phù hợp để hiển thị ảnh.





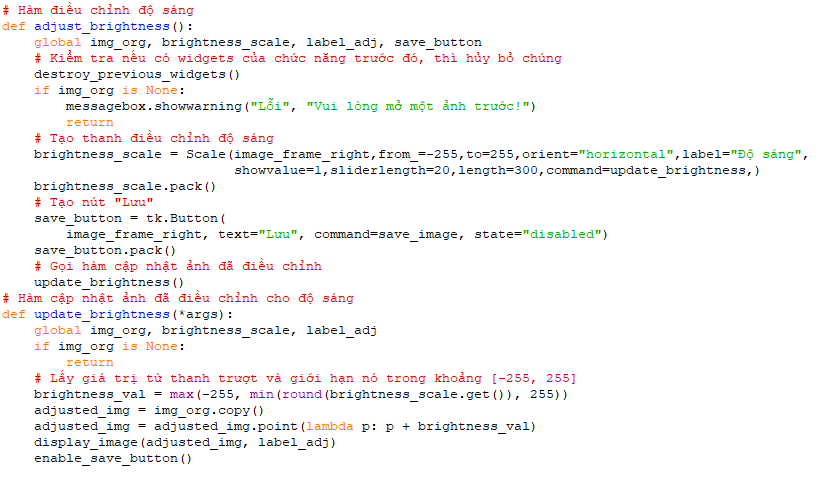
Hình 3.3:Hình khi chọn ảnh chỉnh sửa

* **Module xử lý chức năng điều chỉnh độ sáng :**

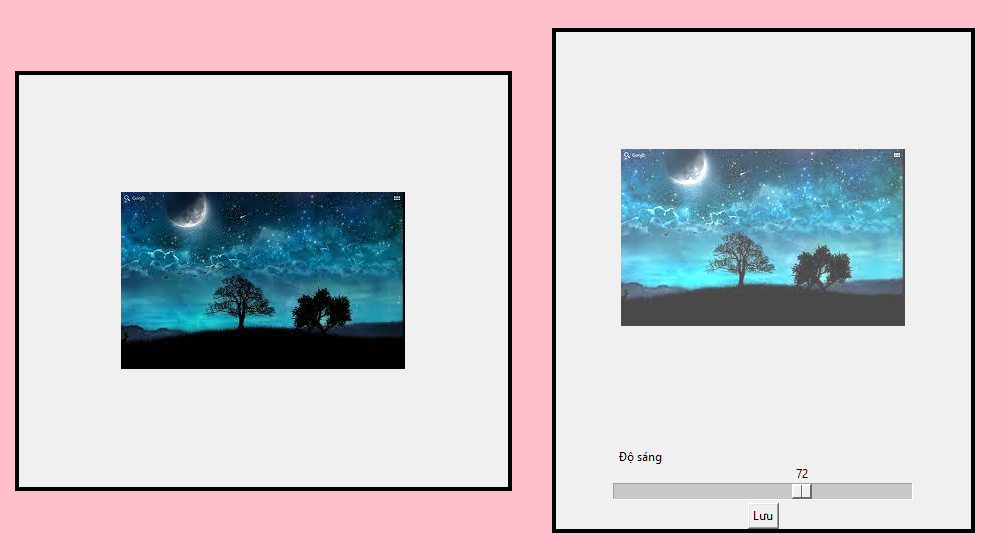
Hàm***adjust\_brightness()*** được gọi khi người dùng muốn điều chỉnh độ sáng của ảnh. Đầu tiên, hàm này kiểm tra xem có ảnh nào đang được mở hay không. Nếu không, hàm sẽ hiển thị một thông báo cảnh báo yêu cầu người dùng mở một ảnh.

Nếu có ảnh, hàm sẽ tạo một thanh trượt cho phép người dùng điều chỉnh độ sáng của ảnh. Thanh trượt này cho phép người dùng chọn một giá trị từ -255 đến 255, tương ứng với việc giảm hoặc tăng độ sáng. Hàm cũng tạo một nút “Lưu” để người dùng có thể lưu ảnh sau khi đã điều chỉnh.

Hàm***update\_brightness()*** được gọi mỗi khi người dùng thay đổi giá trị trên thanh trượt độ sáng. Hàm này lấy giá trị từ thanh trượt, sau đó điều chỉnh độ sáng của ảnh tương ứng. Điều này được thực hiện bằng cách thêm giá trị độ sáng vào mỗi pixel của ảnh. Sau khi điều chỉnh, hàm hiển thị ảnh đã điều chỉnh trên giao diện người dùng và kích hoạt nút “Lưu”.



Thực hiện chức năng điều chỉnh độ sáng :

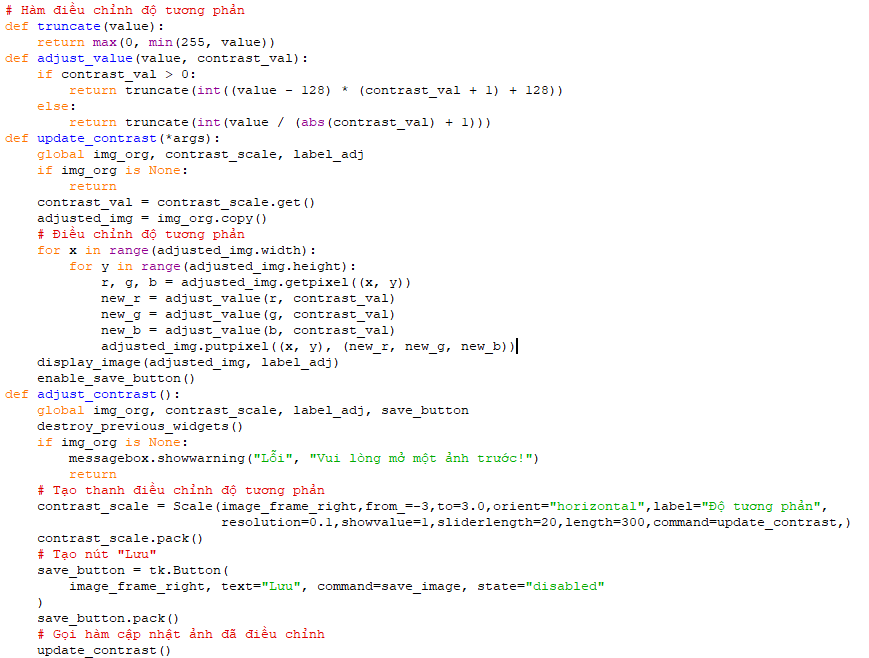


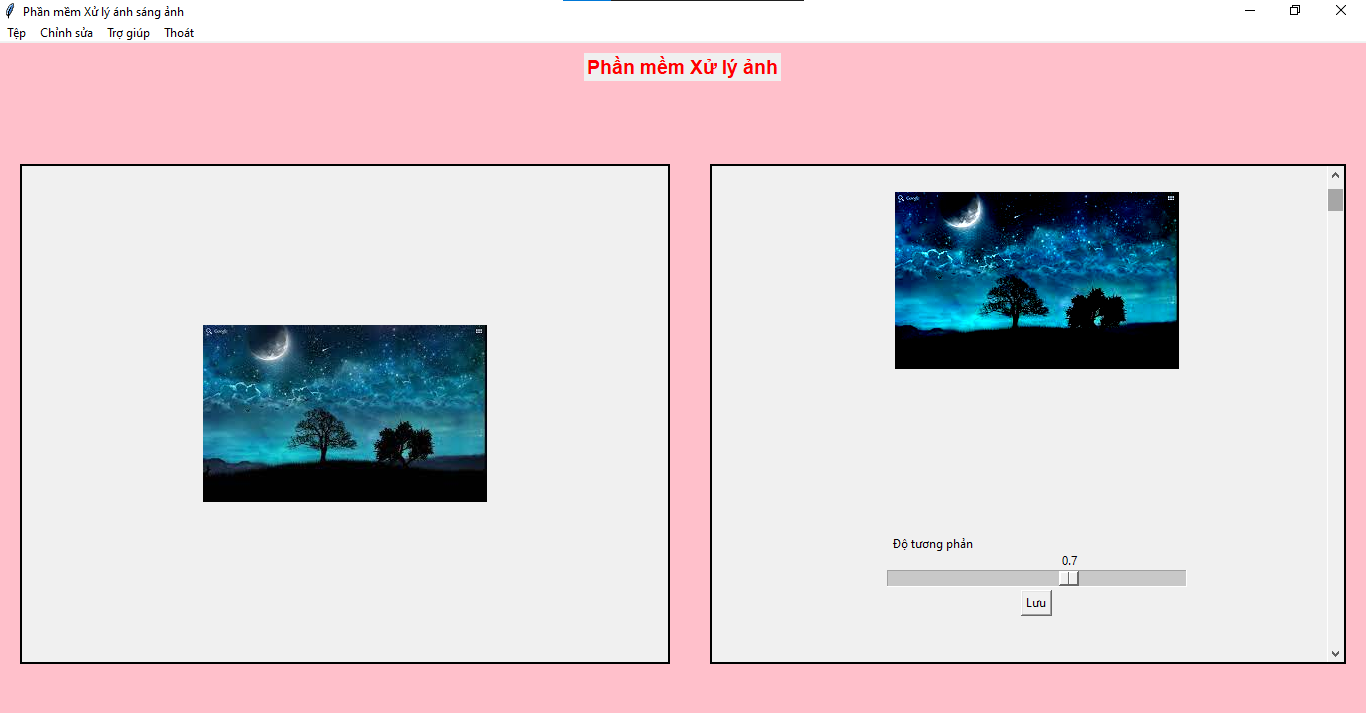
Hình 3.4:Kết quả chức năng điều chỉnh độ sáng

* **Module xử lý chức năng điều chỉnh độ tương phản:**

Hàm ***adjust\_contrast()*** được gọi khi người dùng muốn điều chỉnh độ tương phản của ảnh. Đầu tiên, hàm này kiểm tra xem có ảnh nào đang được mở hay không. Nếu không, hàm sẽ hiển thị một thông báo cảnh báo yêu cầu người dùng mở một ảnh.

Nếu có ảnh, hàm sẽ tạo một thanh trượt cho phép người dùng điều chỉnh độ tương phản của ảnh. Thanh trượt này cho phép người dùng chọn một giá trị , tương ứng với việc giảm hoặc tăng độ tương phản. Hàm ***update\_contrast()*** được gọi mỗi khi người dùng thay đổi giá trị trên thanh trượt độ tương phản. Hàm này lấy giá trị từ thanh trượt, sau đó điều chỉnh độ tương phản của ảnh tương ứng. Điều này được thực hiện bằng cách thay đổi giá trị của mỗi pixel của ảnh dựa trên giá trị độ tương phản được chọn. Sau đó có thể lưu ảnh sau khi đã chỉnh sửa.





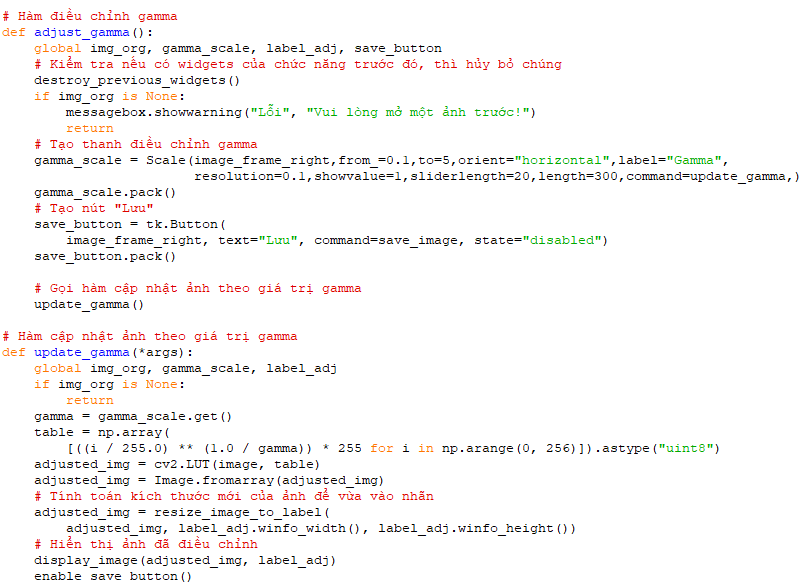
Hình 3.5:Kết quả chức năng điều chỉnh tương phản

* **Module xử lý chức năng điều chỉnh gamma:**

Hàm ***adjust\_gamma()*** được gọi khi người dùng muốn điều chỉnh Gamma của ảnh. Đầu tiên, hàm này kiểm tra xem có ảnh nào đang được mở hay không. Nếu không, hàm sẽ hiển thị một thông báo cảnh báo yêu cầu người dùng mở một ảnh.

Nếu có ảnh, hàm sẽ tạo một thanh trượt cho phép người dùng điều chỉnh Gamma của ảnh. Thanh trượt này cho phép người dùng chọn một giá trị từ 0.1 đến 5(tùy thuộc vào người dùng có thể đặt cho chúng không được dùng giá trị âm nếu là giá trị âm công thức sẽ không còn hợp lệ, vì không thể lấy một số thực bất kỳ mũ cho một số âm).

Hàm***update\_gamma()*** được gọi mỗi khi người dùng thay đổi giá trị trên thanh trượt Gamma. Hàm này lấy giá trị từ thanh trượt, sau đó điều chỉnh Gamma của ảnh tương ứng. Điều này được thực hiện bằng cách thay đổi giá trị của mỗi pixel của ảnh dựa trên giá trị Gamma được chọn. Sau khi điều chỉnh, hàm hiển thị ảnh đã điều chỉnh trên giao diện người dùng và kích hoạt nút “Lưu”.



Thực hiện chức năng hiệu chỉnh gamma:



Hình 3.6: Kết quả chức năng gamma

* **Module xử lý chức năng điều chỉnh cân bằng màu:**

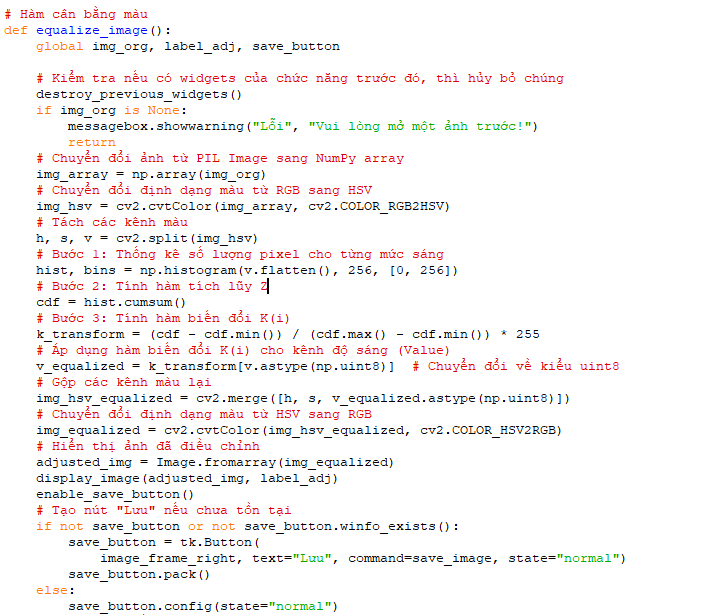
Điều này được thực hiện thông qua hàm ***equalize\_image().***

Hàm ***equalize\_image()*** được gọi khi người dùng muốn cân bằng màu của ảnh. Đầu tiên, hàm này kiểm tra xem có ảnh nào đang được mở hay không. Nếu không, hàm sẽ hiển thị một thông báo cảnh báo yêu cầu người dùng mở một ảnh.

Nếu có ảnh, hàm sẽ chuyển đổi ảnh từ định dạng PIL Image sang NumPy array. Sau đó, hàm chuyển đổi định dạng màu từ RGB sang HSV và tách các kênh màu.

Hàm sau đó thống kê số lượng pixel cho từng mức sáng và tính hàm tích lũy. Dựa trên hàm tích lũy này, hàm tính hàm biến đổi K(i) và áp dụng hàm biến đổi này cho kênh độ sáng (Value) của ảnh.

Sau cùng, hàm gộp các kênh màu lại, chuyển đổi định dạng màu từ HSV sang RGB, và hiển thị ảnh đã điều chỉnh trên giao diện người dùng.



Thực hiện chức năng cân bằng màu:



Hình 3.7: Kết quả chức năng cân bằng màu

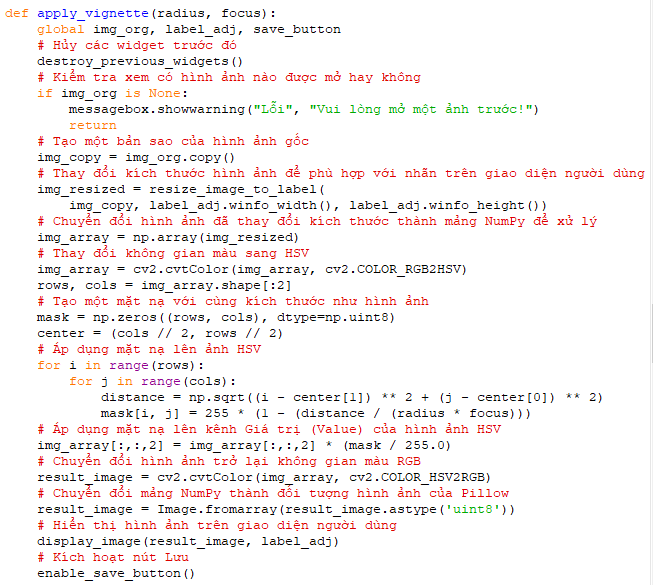
**Module Hiệu ứng mờ viền:**

Hiệu ứng mờ viền là một kỹ thuật nhiếp ảnh phổ biến, nhằm làm tối hoặc làm mờ các cạnh của một bức ảnh trong khi vùng trung tâm được tập trung và sáng hơn. Hiệu ứng này có thể thêm một nét tinh tế và nghệ thuật cho các bức ảnh.Nó được thông qua các hàm :

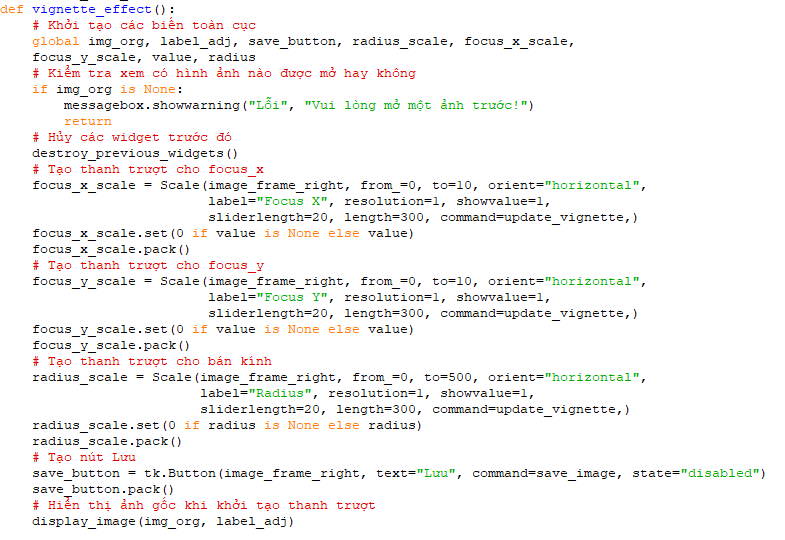
Hàm **apply\_vignette(radius, focus):** Hàm này áp dụng hiệu ứng mờ viền lên ảnh dựa trên bán kính (radius) và tập trung (focus) được cung cấp. Nó chuyển đổi ảnh thành không gian màu HSV, tạo một mặt nạ dựa trên bán kính và tập trung, áp dụng mặt nạ lên kênh Giá trị (Value) của ảnh và chuyển đổi trở lại không gian màu RGB. Kết quả được hiển thị trên giao diện người dùng.

Đầu vào của hàm gồm hai tham số:

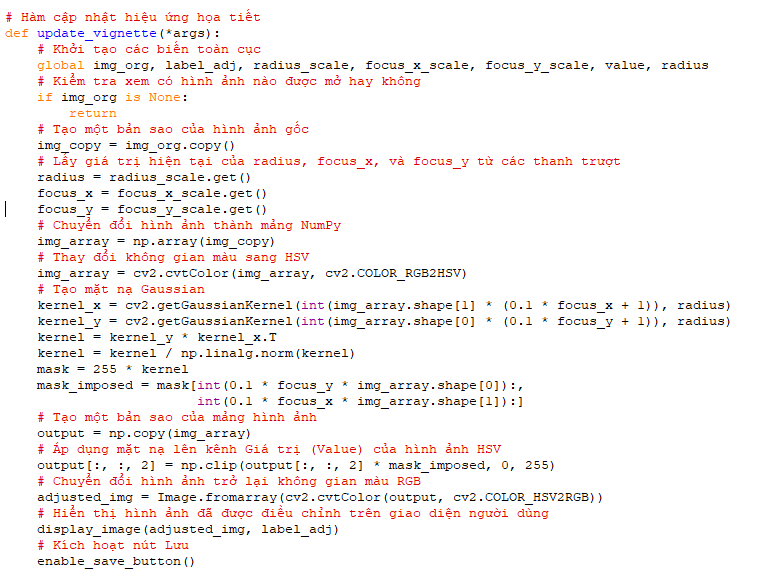
* **radius**: Bán kính của hiệu ứng mờ viền. Giá trị bán kính lớn tạo ra một hiệu ứng mạnh hơn.
* **focus**: Yếu tố tập trung của hiệu ứng mờ viền. Nó xác định mức độ tập trung vào vùng trung tâm. Giá trị tập trung cao làm cho vùng trung tâm nổi bật hơn.



Hàm **vignette\_effect():** Hàm này tạo giao diện người dùng cho việc áp dụng hiệu ứng họa tiết lên ảnh. Nó tạo các thanh trượt cho bán kính (radius), tập trung X và tập trung Y, và một nút Lưu. Khi các thanh trượt thay đổi, hàm update\_vignette() được gọi để cập nhật hiệu ứng họa tiết dựa trên giá trị của các thanh trượt.



Hàm **update\_vignette(\*args):** Hàm này được gọi khi các thanh trượt thay đổi giá trị. Nó lấy giá trị từ các thanh trượt và cập nhật hiệu ứng họa tiết dựa trên các giá trị đó. Nó tạo một kernel dựa trên bán kính và tập trung X và Y, tạo một mặt nạ từ kernel, áp dụng mặt nạ lên kênh Giá trị của ảnh và hiển thị kết quả trên giao diện người dùng.



Thực hiện chức năng hiệu ứng mờ viền:

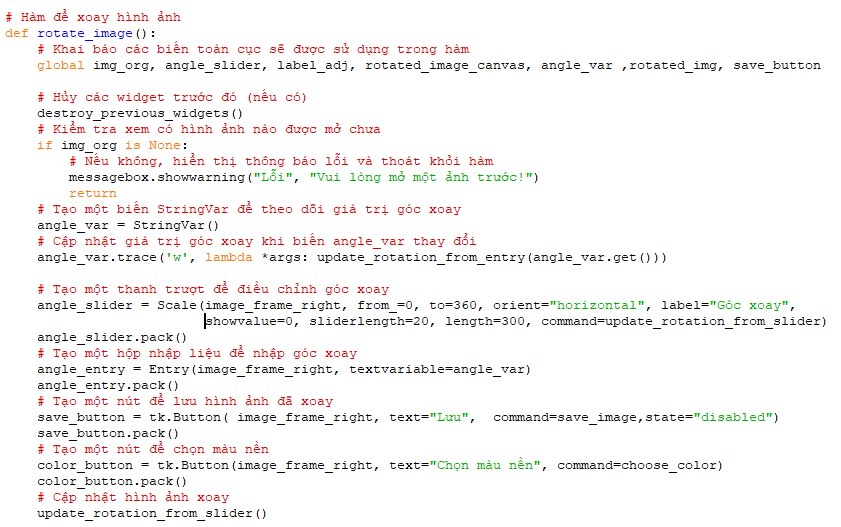


Hình 3.8: Kết quả chức năng hiệu ứng mờ viền

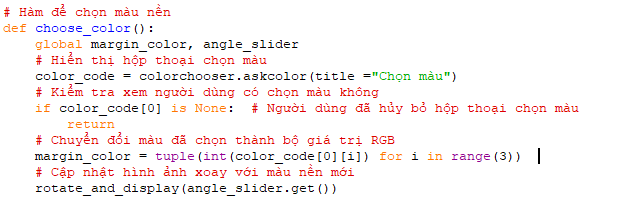
**Module xoay ảnh:**

 Đoạn mã này sử dụng các module và chức năng để cho phép người dùng xoay ảnh theo góc độ mong muốn và hiển thị kết quả trực tiếp trên giao diện người dùng.Được thông qua các hàm chức năng sau:

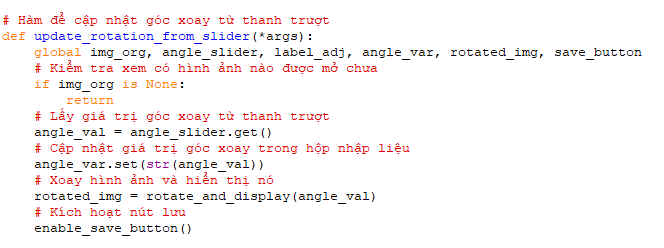
Hàm**rotate\_image():** Hàm này được gọi khi người dùng muốn xoay ảnh. Nó tạo giao diện người dùng với một thanh trượt để chọn góc xoay, một ô nhập giá trị góc xoay, nút "Lưu" để lưu ảnh sau khi xoay, và nút "Chọn màu nền" để chọn màu nền cho ảnh. Hàm này cũng gọi các hàm khác để cập nhật giao diện người dùng và hiển thị ảnh sau khi xoay.



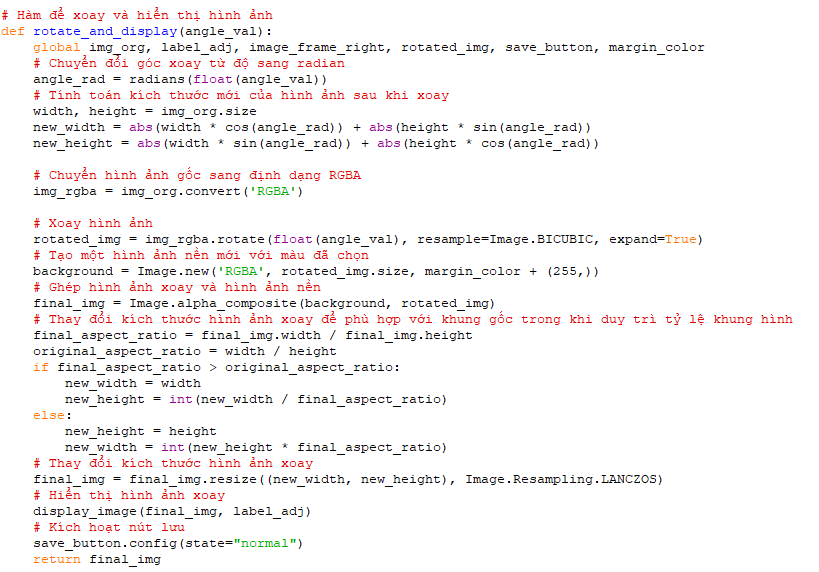
Hàm **choose\_color():** Hàm này được gọi khi người dùng nhấp vào nút "Chọn màu nền". Nó hiển thị một hộp thoại cho phép người dùng chọn màu từ bảng màu và lưu giá trị màu nền được chọn.



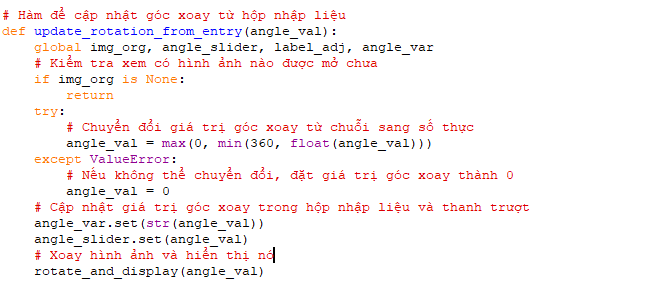
Hàm**update\_rotation\_from\_slider():** Hàm này được gọi khi giá trị thanh trượt góc xoay thay đổi. Nó cập nhật giá trị góc xoay từ thanh trượt và gọi hàm rotate\_and\_display() để xoay ảnh và hiển thị kết quả.



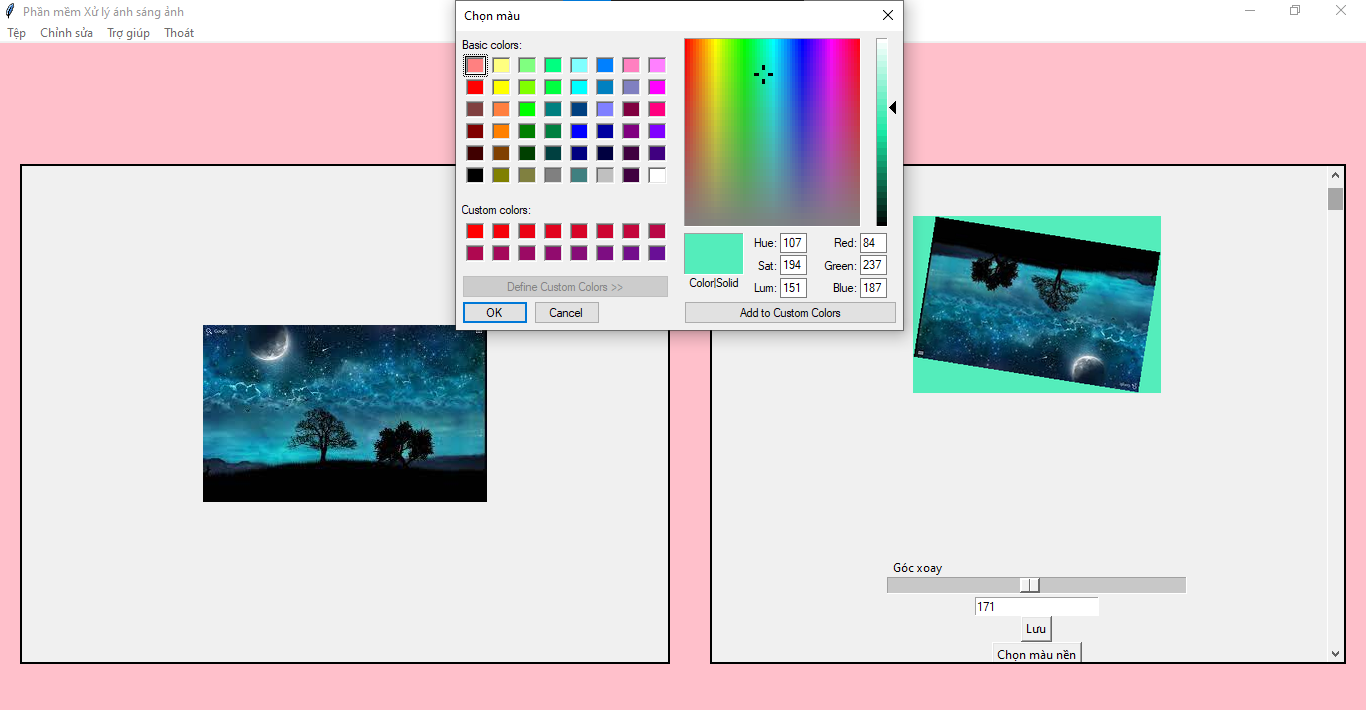
Hàm**rotate\_and\_display(angle\_val):** Hàm này thực hiện việc xoay ảnh dựa trên góc xoay được cung cấp. Nó tính toán kích thước mới cho ảnh xoay, xoay ảnh sử dụng phương pháp lặp lại Bicubic, tạo ảnh nền với màu nền đã chọn và kết hợp ảnh xoay với ảnh nền. Cuối cùng, nó điều chỉnh kích thước ảnh để phù hợp với kích thước ban đầu và hiển thị ảnh kết quả.



Hàm**update\_rotation\_from\_entry(angle\_val):** Hàm này gọi khi giá trị góc xoay nhập từ ô để thay đổi Nó cập nhật giá trị góc xoay từ ô nhập và gọi hàm **rotate\_and\_display(**) để xoay ảnh và hiển thị kết quả.



Thực hiện chức năng xoay ảnh:



Hình 3.9:Kết quả chức năng xoay ảnh và đổi màu nền

KẾT LUẬN

4.1 Kết quả đạt được

Sau quá trình thực hiện đồ án “Hiệu chỉnh ánh sáng ảnh”, em đã học hỏi đón nhận tiếp thu được một số kết quả sau:

* Hiểu và triển khai cấu trúc một ứng dụng bằng ngôn ngữ python.
* Sử dụng các thư viện để áp dụng vào các chức năng cụ thể trong xử lý.
* Quy trình để xử lý ảnh để thay đổi bức ảnh.

4.2 Một số hạn chế trong ứng dụng

Vì thời gian thực hiện ngắn tuy đề tài đã hoàn thành nhưng vẫn còn chứa hạn chế:

* Sản phẩm chưa được trải qua quy trình kiểm thử nghiêm ngặt.
* Giao diện còn sơ sài.
* Tính bảo mật chưa được tối ưu và là một trong những hạn chế của hệ thống.
* Chức năng còn chưa ít chưa có nhiều để người dùng có trải nghiệm tốt hơn.

4.3 Hướng phát triển ứng dụng trong tương lai

Trong thời gian tới em sẽ tiếp tục phát triển ứng dụng với những yêu cầu như:

* Thêm một số chức năng cho người dùng trải nghiệm tốt hơn.
* Thêm chức năng trừ điểm bài thi nếu cảnh báo quá nhiều.
* Có thể áp dụng các phương pháp mô hình để đưa ra những bức ảnh có chất lượng tốt.

**DANH MỤC TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] |  |
| [1] |  |
| [1] |  |
| [1] | Wikipedia. [Online] Available: https://vi.wikipedia.org/wiki/python |
| [2] | Mozilla. [Online] Available: https://developer.mozilla.org/en-US/docs/Glossary/Python |
| [3] | W3Schools. [Online]. Available: https://www.w3schools.com/. |
| [4] | Stackoverflow. [Online]. Available: https://stackoverflow.com/. |
| [5] | https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-5-contrast-adjustment/ |
| [6] | https://www.geeksforgeeks.org/digital-image-processing-basics/ |
|  |  |