

ỨNG DỤNG WINDOWS CHỈNH SỬA VÀ TRÍCH XUẤT THÔNG TIN TỪ ẢNH

LUẬN VĂN CỬ NHÂN

Cao Thành Danh – 1912836

Giảng viên hướng dẫn

ThS. Nguyễn Khánh Lợi



ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ, BỘ MÔN VIỄN THÔNG

11 – 2023

Số: _____ /BKĐT

Khoa: **Điện – Điện tử**

Bộ Môn: **Viễn Thông**

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. Họ và tên: Cao Thành Danh MSSV: 1912836
2. Ngành: Điện – Điện tử Chuyên ngành: Kỹ thuật Điện tử - Truyền thông
3. Đề tài: Ứng dụng windows chỉnh sửa và trích xuất thông tin từ ảnh
4. Nhiệm vụ:
 - Nghiên cứu thuật toán xử lý ảnh liên quan đến đề tài.
 - Thực nghiệm đánh giá kết quả trên Python.
 - Xây dựng ứng dụng Windows, kết hợp với Python.
 - Phân tích và so sánh kết quả.
5. Ngày giao nhiệm vụ luận văn: 02/10/2023
6. Ngày hoàn thành nhiệm vụ: 18/12/2023
7. Họ và tên người hướng dẫn: Phản hướng dẫn
ThS. Nguyễn Khánh Lợi,
BM Viễn Thông, Khoa Điện – Điện Tử 100%
Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

TP.HCM, ngày 04 tháng 12 năm 2023

CHỦ NHIỆM BỘ MÔN

PGS. TS. Hà Hoàng Kha

NGƯỜI HƯỚNG DẪN CHÍNH

ThS. Nguyễn Khánh Lợi

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):

Đơn vị:

Ngày bảo vệ :

Điểm tổng kết:

Nơi lưu trữ luận văn:

LỜI CẢM ƠN

Lời đầu tiên em xin được cảm ơn tập thể các thầy cô của trường Đại học Bách Khoa, các thầy cô của khoa Điện – Điện tử và các thầy cô ở bộ môn Viễn thông vì đã tận tình chỉ dạy và trang bị cho em những kiến thức về các môn đại cương cũng như các môn chuyên ngành để làm nền tảng cho việc thực hiện Đồ án cũng như những kinh nghiệm quý báu để nhóm em vững tin hơn trong môi trường làm việc sau này.

Đặc biệt, em xin chân thành cảm ơn thầy Nguyễn Khánh Lợi, người đã tận tình hướng dẫn, giúp đỡ, định hướng, góp ý và cung cấp ý tưởng cũng như tài liệu tham khảo trong thời gian làm Đồ án.

Vì kiến thức bản thân còn nhiều hạn chế, trong quá trình học tập, hoàn thiện báo cáo này em không tránh khỏi những sai lầm. Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để em có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công tác thực tế sau này.

Em xin chân thành cảm ơn.

TP. HCM, ngày 18, tháng 12 năm 2023

Cao Thành Danh

LỜI CAM ĐOAN

Tôi tên: Cao Thành Danh là sinh viên chuyên ngành Kỹ thuật Điện tử - Truyền thông, khóa 2019, tại Đại học Quốc gia thành phố Hồ Chí Minh – Trường Đại học Bách Khoa. Tôi xin cam đoan những nội dung sau đây là sự thật: (i) Công trình nghiên cứu này hoàn toàn do chính tôi thực hiện; (ii) Các tài liệu và trích dẫn trong luận văn này được tham khảo từ các nguồn thực tế, có uy tín và độ chính xác cao; (iii) Các số liệu và kết quả của công trình này được tôi tự thực hiện một cách độc lập và trung thực.

TP. HCM, ngày 18, tháng 12 năm 2023

Cao Thành Danh

TÓM TẮT LUẬN VĂN

Đề luận văn xây dựng “Ứng dụng windows chỉnh sửa và trích xuất thông tin từ ảnh”. Mục tiêu của đề tài tiến hành nghiên cứu và áp dụng các kỹ thuật xử lý ảnh, nghiên cứu thuật toán trên Python, tiến hành triển khai ứng dụng thực tế trên nền tảng Windows. Ứng dụng được xây dựng có thể đáp ứng được các nhiệm vụ như xóa vật thể, thực hiện chỉnh màu cho ảnh, và trích xuất thông tin từ ảnh. Trước tiên, đề tài nghiên cứu về thuật toán Inpaint được OpenCV hỗ trợ, tiến hành nghiên cứu sâu hơn để nâng cao hiệu quả xử lý qua bài toán Inpaint Anything[2]. Tìm hiểu về thuật toán xử lý màu ảnh thư viện Pilgram. Bên cạnh đó, kỹ thuật trích xuất thông tin từ ảnh thông qua Nhận dạng kí tự quang học (OCR) để trích xuất thông tin nhanh từ ảnh. Qua đó xây dựng được một ứng dụng đáp ứng các nhu cầu về ảnh từ chỉnh sửa ảnh đến trích xuất thông tin từ ảnh.

Sau quá trình nghiên cứu, đề tài đạt được gần như các mục tiêu đề ra. Thực hiện nghiên cứu thuật toán xóa vật thể (Inpaint), chỉnh sửa màu ảnh và trích xuất thông tin đạt được hiệu quả xử lý tốt trong Python. Quá trình xây dựng ứng dụng trên Windows xây dựng tốt giao diện và xử lý trong ứng dụng, tuy nhiên vẫn còn vấn đề trong quản lý hệ thống. Kết quả chỉnh sửa ảnh gồm Inpaint và chỉnh màu ảnh đạt kết quả tốt như mong đợi, thời gian đáp ứng nhanh. Việc trích xuất thông tin từ ảnh đạt hiệu quả trong trường hợp tương đối lý tưởng, cải thiện hơn trong các trường hợp phức tạp.

ABSTRACT

Thesis on developing “Windows Application To Edit And Extract Information From Photos”. The goal of the project is to research and apply image processing techniques, research algorithms on Python, and deploy practical applications on the Windows platform. The built application can handle tasks such as removing objects, performing color correction on photos, and extracting information from photos. First, the research project on the Inpaint algorithm is supported by OpenCV, conducting further research to improve processing efficiency through the Inpaint Anything Paper[2]. Learn about Pilgram library's image color processing algorithm. Besides, the technique of extracting information from images is through Optical Character Recognition (OCR) to quickly extract information from images. Thereby building an application that meets photo needs from photo editing to extracting information from photos.

After the research process, the project achieved almost the set goals. Conduct research on algorithms for object deletion (Inpaint), image color correction and information extraction to achieve good processing efficiency in Python. The process of building applications on Windows creates a good interface and processing in the application, but there are still problems in system management. Photo editing results including Inpaint and photo color correction achieved good results as expected, with fast response time. Extracting information from images is effective in relatively ideal cases, but needs to be improved in complex cases.

MỤC LỤC

LỜI CẢM ƠN	i
LỜI CAM ĐOAN	ii
TÓM TẮT LUẬN VĂN	iii
ABSTRACT	iv
DANH SÁCH HÌNH ẢNH	vii
DANH SÁCH TỪ VIẾT TẮT	x
CHƯƠNG 1. GIỚI THIỆU	1
1.1 ĐẶT VẤN ĐỀ.....	1
1.2 PHẠM VI VÀ PHƯƠNG PHÁP NGHIÊN CỨU	2
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	3
2.1 XỬ LÝ ẢNH SỐ (DIGITAL IMAGE PROCESSING).....	3
2.1.1 Giới thiệu về ảnh số	3
2.1.2 Biến đổi miền không gian (Spatial Domain Transformation)	7
2.2 INPAINT.....	17
2.2.1 Inpaint trong OpenCV.....	18
2.2.2 Inpaint Anything	23
2.3 KỸ THUẬT PHÂN ĐOẠN ẢNH (IMAGE SEGMENTATION).....	25
2.3.1 Đầu vào và đầu ra của phân đoạn ảnh.....	26
2.3.2 Các dạng phân đoạn ảnh	27
2.3.3 Đánh giá bài toán Segmentation	27
2.4 NHẬN DẠNG KÝ TỰ QUANG HỌC (OCR).....	29
2.4.1 Giới thiệu OCR	29
2.4.2 Phân loại OCR	30
2.4.3 Ứng dụng OCR	30
2.4.4 Đánh giá bài toán OCR	30
2.5 THƯ VIỆN SỬ DỤNG	33
2.5.1 WinUI 3 – .NET Framework 4.8	33
2.5.2 OpenCV Sharp	35
2.5.3 Tesseract Python và IronOCR	36
2.5.4 Pilgram Python.....	38
2.5.5 Biểu thức chính quy – Regex	38
CHƯƠNG 3. KẾT QUẢ VÀ PHÂN TÍCH.....	40
3.1 PHƯƠNG PHÁP TIẾP CẬN.....	40
3.1.1 Thử nghiệm trên Python	40

3.1.2	Thử nghiệm trên C#	43
3.2	KẾT QUẢ VÀ PHÂN TÍCH.....	46
3.2.1	Thực hiện Inpaint trong Python	46
3.2.2	Thực hiện chỉnh màu ảnh - Filter trong Python	57
3.2.3	Thực hiện OCR trong Python	60
3.2.4	Thực hiện Inpaint trong C Sharp	70
3.2.5	Thực hiện Filter trong C Sharp	76
3.2.6	Thực hiện OCR trong C Sharp.....	79
3.3	KẾT LUẬN CHƯƠNG	81
CHƯƠNG 4. KẾT LUẬN.....		82
4.1	TÓM TẮT VÀ KẾT LUẬN CHUNG	82
4.2	HƯỚNG PHÁT TRIỂN	82
PHỤ LỤC A.....		84
A.1	Code chương trình Filter Python	84
A.2	Code chương trình xử lý C Sharp	86
TÀI LIỆU THAM KHẢO		91

DANH SÁCH HÌNH ẢNH

Hình 2-1 Hình ảnh mặt cắt ngang đơn giản của mắt người [3]	3
Hình 2-2 Minh họa mắt người nhìn vào cây cọ. Điểm C là tâm của thấu kính.[3]	4
Hình 2-3 Cảm biến ảnh kỹ thuật số	5
Hình 2-4 Hình ảnh minh họa hình thành ảnh số [3]	5
Hình 2-5 Ma trận ảnh trước và sau khi lấy mẫu và lượng tử hóa	6
Hình 2-6 Phép xử lý điểm và Phép xử lý lân cận điểm.	8
Hình 2-7 Ví dụ về xử lý miền không gian (Spatial Domain).....	9
Hình 2-8 Một vài phép toán điểm cơ bản trong xử lý ảnh.....	10
Hình 2-9 Ảnh chụp X – Quang tuyến vú. a) Ảnh gốc. b) Ảnh âm bản [3].....	11
Hình 2-10 Minh họa về phép biến đổi Log a) Phổ của Fourier. b) Ảnh sau khi áp dụng phép biến đổi Logarithm với $c = 1$	12
Hình 2-11 Đồ thị với biểu thức tổng quát hàm lũy thừa $s = c r\gamma$	12
Hình 2-12 Ảnh chụp cộng hưởng từ - MRI cột sống trước ngực [3].....	13
Hình 2-13 Minh họa cơ chế bộ lọc tuyến tính sử dụng Kernel 3×3	14
Hình 2-14 Kích thước kernel của bộ lọc tuyến tính.....	15
Hình 2-15 Xấp xỉ kernel của bộ lọc sắc nét	17
Hình 2-16 Minh họa thuật toán Inpaint sử dụng Fast Marching Method [4]	18
Hình 2-17 Nguyên lý hoạt động của kỹ thuật Inpaint FFM [4].....	19
Hình 2-18 Các vùng liên quan trong xử lý Inpaint sử dụng FFM	20
Hình 2-19 Mã giả mô tả xử lý chi tiết Inpaint với FFM	21
Hình 2-20 Tác động của các yếu tố $dir(p, q)$, $dst(p, q)$ và $lev(p, q)$ đến $w(p, q)$	23
Hình 2-21 Quy trình hoạt động với ba tính năng chính của Inpaint Anything	24
Hình 2-22 Trực quan kết quả xử lý của Remove Anything.....	25
Hình 2-23 Phân biệt giữa thuật toán phát hiện vật thể và phân đoạn ảnh.....	26
Hình 2-24 Input (bên trái) và Output (bên phải) của mô hình Image Segmentation	27
Hình 2-25 Semantic segmentation và Instance segmentation.....	27
Hình 2-26 Minh họa cách tính IoU	28
Hình 2-27 Minh họa công thức tính IoU	28
Hình 2-28 Quy trình xử lý OCR	29
Hình 2-29 Ba lỗi cơ bản trong đánh giá bài toán OCR.....	31
Hình 2-30 Ví dụ tính Tỷ lệ lỗi ký tự CER	32
Hình 2-31 Ví dụ tính Tỷ lệ lỗi từ WER	33
Hình 2-32 .NET Hỗ trợ đa nền tảng.....	34
Hình 3-1 Quy trình xử lý Inpaint trong Python	41
Hình 3-2 Quy trình chỉnh màu ảnh trong Python	42
Hình 3-3 Quy trình trích xuất thông tin từ ảnh trong Python	43
Hình 3-4 Quy trình xử lý Inpaint trong ứng dụng Windows	44
Hình 3-5 Quy trình trích xuất thông tin từ ảnh trong ứng dụng Windows	46
Hình 3-6 Tập ảnh đơn giản trong xử lý Inpaint trong Python	47
Hình 3-7 Tập ảnh trung bình trong xử lý Inpaint trong Python	47
Hình 3-8 Tập ảnh phức tạp trong xử lý Inpaint trong Python.....	48
Hình 3-9 Tổ chức thư mục Inpaint trong Python.....	49
Hình 3-10 Kết quả ảnh đơn giản Inpaint trong Python.....	49

Hình 3-11 Danh sách ảnh đầu vào xử lý Inpaint Python	51
Hình 3-12 Kết quả tạo mặt nạ cho xử lý Inpaint Python	51
Hình 3-13 Minh họa kết quả xử lý Inpaint Python	52
Hình 3-14 Xác định vị trí vật thể trong Inpaint Anything Python	53
Hình 3-15 Kết quả xử lý Inpaint Anything Python	53
Hình 3-16 Kết quả xử lý ảnh đơn giản Inpaint Python	54
Hình 3-17 Kết quả xử lý ảnh đơn giản Inpaint Anything Python	54
Hình 3-18 Kết quả xử lý ảnh trung bình Inpaint Python	55
Hình 3-19 Kết quả xử lý ảnh trung bình Inpaint Anything Python	55
Hình 3-20 Kết quả xử lý ảnh phức tạp Inpaint Python	56
Hình 3-21 Kết quả xử lý ảnh phức tạp Inpaint Anything Python	56
Hình 3-22 Các định dạng ảnh khác nhau chỉnh sửa màu ảnh Python	58
Hình 3-23 Kết quả xử lý các định dạng ảnh có thể chỉnh sửa màu trong Python	58
Hình 3-24 Ảnh đầu vào chỉnh sửa màu ảnh Python	59
Hình 3-25 Kết quả chỉnh sửa màu ảnh hàng loạt Python	59
Hình 3-26 Kết quả chỉnh sửa màu ảnh bitmap	60
Hình 3-27 Tập dữ liệu cơ bản trong xử lý OCR Python	61
Hình 3-28 Tập dữ liệu trung bình trong xử lý OCR Python	61
Hình 3-29 Tập dữ liệu phức tạp trong xử lý OCR Python	62
Hình 3-30 Tô chức thư mục xử lý OCR Python	63
Hình 3-31 Kết quả xử lý ảnh cơ bản Tesseract OCR Python lần 1	64
Hình 3-32 Kết quả xử lý ảnh cơ bản Tesseract OCR Python lần 2	65
Hình 3-33 Kết quả lý ảnh trung bình Tesseract OCR Python lần 1	65
Hình 3-34 Kết quả xử lý ảnh trung bình Tesseract OCR Python lần 2	66
Hình 3-35 Kết quả xử lý ảnh phức tạp Tesseract OCR Python lần 1	66
Hình 3-36 Kết quả xử lý ảnh phức tạp Tesseract OCR Python lần 2	67
Hình 3-37 Kết quả xử lý ảnh đơn giản VietOCR Python lần 1	67
Hình 3-38 Kết quả xử lý ảnh đơn giản VietOCR Python lần 2	68
Hình 3-39 Kết quả xử lý ảnh trung bình VietOCR lần 1	68
Hình 3-40 Kết quả xử lý ảnh phức tạp VietOCR Python lần 1	69
Hình 3-41 Kết quả xử lý ảnh phức tạp VietOCR Python lần 2	69
Hình 3-42 Kết quả xây dựng Inpaint trong ứng dụng Windows	71
Hình 3-43 Minh họa xử lý đa luồng (multi threads) trong ứng dụng Windows	72
Hình 3-44 Hướng dẫn chọn dạng hình cho việc tạo Mask	73
Hình 3-45 Minh họa việc tạo Mask trong ứng dụng Windows	73
Hình 3-46 Kết quả xử lý Inpaint trong ứng dụng Windows	74
Hình 3-47 Lưu ảnh sau khi xử lý Inpaint trong ứng dụng Windows	75
Hình 3-48 Minh họa đặt tên và chọn nơi lưu sau khi xử lý Inpaint Windows	75
Hình 3-49 Lấy đường dẫn tới ảnh xử lý chỉnh màu trong Windows	76
Hình 3-50 Danh sách các Filter trong ứng dụng Windows	77
Hình 3-51 Kết quả sau khi xử lý chỉnh màu trong Windows	78
Hình 3-52 So sánh kết quả chỉnh màu ảnh trong Windows	78
Hình 3-53 Thêm ảnh trích xuất thông tin vào ứng dụng Windows	79
Hình 3-54 Minh họa chọn ngôn ngữ OCR trong Windows	80
Hình 3-55 Kết quả trích xuất thông tin trong ứng dụng Windows	80

DANH SÁCH TỪ VIẾT TẮT

Inpaint	Kỹ thuật xử lý ảnh xóa đi các chi tiết không mong muốn trong ảnh.
OCR	Optical Character Recognition
OpenCV	Open Computer Vision Library, thư viện xử lý ảnh.
CSS Gram	Thư viện xử lý màu ảnh dựa trên ngôn ngữ thiết kế CSS, Cascading Style Sheets
Windows OS	Windows Operating System, hệ điều hành Windows.
.NET Framework	Một nền tảng phát triển ứng dụng mã nguồn mở do Microsoft xây dựng.
WinUI 3	Nền tảng xây dựng giao diện người dùng cung cấp các APIs, các bộ công cụ giúp phát triển ứng dụng máy tính trên Windows.
Inpaint Anything	Thuật toán xử lý xóa vật thể, có bài báo khoa học [3].
Tesseract	Một thư viện xử lý nhận dạng ký tự quang học.
APIs	Application Programming Interfaces. Cơ chế giao tiếp giữ các thành phần phần mềm với nhau.
Nuget	Trình quản lý gói (packages) cho phát triển phần mềm trên nền tảng .NET
UWP	Universal Windows Platform, một nền tảng xây dựng ứng dụng Windows cho Windows 10, trên đa dạng thiết bị khác nhau.
CLR	The Common Language Runtime. Một dạng máy ảo để chạy các phần mềm được viết bằng .NET Framework
Regex	Regular Expression. Biểu thức chính quy là các dạng mẫu ký tự được sử dụng trong xử lý chuỗi ký tự.
MIT	the Massachusetts Institute of Technology. Giấy phép sử dụng phần mềm mã nguồn mở được Viện công nghệ Massachusetts cung cấp.
Fluent Design	Microsoft Fluent Design System là một ngôn ngữ thiết kế giao diện hiện đại được Microsoft phát triển từ năm 2017.
CRT	Cathode Ray Tube. CRT là một dạng đèn điện tử chân không, chứa nguồn phát ra các hạt điện tử và màn hình lân quang. Các tia hạt tác động vào các điểm ảnh hình thành sự phản xạ ánh sáng.
MRI	Magnetic Resonance Imaging. MRI phương pháp thu thập hình ảnh trong y học dùng để chuẩn đoán tình trạng của các cơ quan trong cơ thể sống dựa trên hiện

tương Công hưởng từ hạt nhân.

DPI Dots Per Inch. Chỉ số đo lường độ phân giải không gian của hình ảnh. Biểu thị số lượng Pixel trên một Inch vuông (in^2) .

CHƯƠNG 1. GIỚI THIỆU

1.1 ĐẶT VĂN ĐỀ

Trong thời đại số hóa hiện nay, sự phát triển của công nghệ khiến việc sở hữu những thiết bị thông minh với các tính năng như quay phim, chụp ảnh vô cùng phổ biến. Các nhu cầu tương tác với hình ảnh đang được sinh ra đồng thời với nhịp độ phát triển trên. Trên thị trường về phần mềm đang có rất nhiều ứng dụng khác nhau hỗ trợ việc xử lý hình ảnh và video để đáp ứng được nhu cầu của người dùng. Tuy nhiên việc tiếp cận với các ứng dụng khác nhau và đa dạng như vậy yêu cầu người dùng cần phải có kiến thức về sử dụng phần mềm, màu sắc, đồ họa và thuật ngữ xử lý ảnh cụ thể. Với các nhu cầu chỉnh sửa cơ bản và trích xuất thông tin từ ảnh, việc tìm hiểu các phần mềm chuyên dụng tốn nhiều thời gian và chi phí.

Trong vai trò là một người dùng có nhu cầu cơ bản và sự đơn giản khi sử dụng. Một ứng dụng cần phải đáp ứng các nhu cầu sau:

- Đơn giản, dễ dùng: Việc sử dụng phần mềm cần có sự quen thuộc và tương đồng với cần phần mềm mà người dùng thường sử dụng. Giao diện đơn giản, không quá nhiều công cụ nhưng vẫn đáp ứng đúng nhu cầu.
- Xử lý hiệu quả: Xử lý các tác vụ xử lý ảnh cơ bản như xóa vật thể, thay đổi kích thước ảnh, thay đổi màu ảnh cần được đáp ứng một cách chính xác.
- Quen thuộc: Ứng dụng phải được xây dựng trên nền tảng quen thuộc và các thao tác tương tự các ứng dụng mà người dùng thường sử dụng.

Một ứng dụng có sự kết hợp giữa xử lý ảnh và thị giác máy tính đã mở ra hướng giải quyết tìm năn. Cụ thể trong việc chỉnh sửa ảnh, xóa các chi tiết vật thể dùng thư viện OpenCV, một thư viện xử lý ảnh mạnh mẽ. Thay đổi màu ảnh để tạo ra các bức ảnh đẹp hơn với sự hỗ trợ của thư viện Pilgram, được lấy ý tưởng từ CSS Gram, một thư viện xây dựng các tính năng lọc màu cho ảnh. Cuối cùng việc trích xuất văn bản từ hình ảnh thông qua nhận dạng ký tự quang học OCR.

Ngoài việc ứng dụng có thể đáp ứng được các yếu tố xử lý về mặt kỹ thuật, việc lựa chọn nền tảng quen thuộc với người dùng cũng là khía cạnh quan trọng. Để tài chọn nền tảng Windows OS để xây dựng ứng dụng dựa trên .Net Framework để xây dựng phần xử lý lỗi, và dùng WinUI 3 thiết kế giao diện người dùng. Hệ điều hành Windows là hệ điều hành được dùng phổ biến nhất thế giới chiếm 64.27% [1] trong lĩnh vực máy tính cá nhân, máy tính bảng và console trong thống kê thị trường hệ điều hành vào tháng 6 năm 2023. Vì vậy đây là hệ điều hành rất quen thuộc với đại đa số người dùng hiện nay.

Câu hỏi nghiên cứu đặt ra của luận văn tốt nghiệp là:

- Xử lý việc xóa các vật thể và chỉnh sửa có đạt được hiệu quả hình ảnh, có đáp ứng thời gian thực hay không?
- Trích xuất thông tin văn bản từ ảnh liệu có khả thi không? Kết quả có đầy đủ và chính xác không?
- Xây dựng ứng dụng Windows có thực sự đơn giản và quen thuộc với người sử dụng hay không?

Trong chương 2, cơ sở lý thuyết về ứng dụng sẽ được trình bày. Trong chương 3, kết quả thử nghiệm sẽ được so sánh và phân tích. Cuối cùng, chương 4 sẽ đưa ra kết luận chung.

1.2 PHẠM VI VÀ PHƯƠNG PHÁP NGHIÊN CỨU

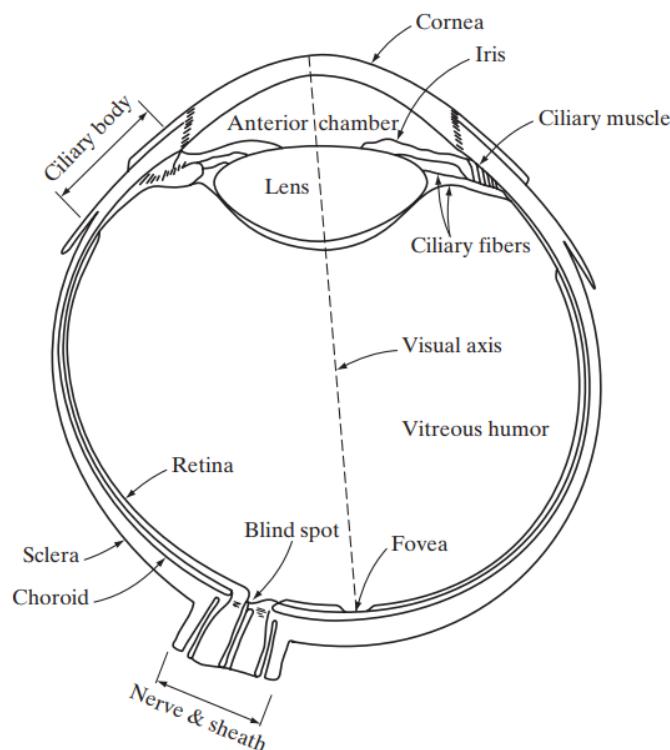
- Nghiên cứu việc xóa vật thể khỏi hình ảnh bằng thư viện OpenCV Python, nghiên cứu mở rộng thêm bằng giải thuật Inpaint Anything [2].
- Chỉnh sửa ảnh với sự hỗ trợ của thư viện Pillow trong Python, thay đổi màu ảnh bởi thư viện Pilgram được lấy ý tưởng từ CSS gram.
- Trích xuất thông tin từ ảnh bằng nhận dạng ký tự quang học OCR với sự hỗ trợ của thư viện Tesseract trong Python.
- Xây dựng ứng dụng trong hệ điều hành Windows dựa trên .NET Framework và thiết kế giao diện người dùng bằng WinUI 3.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 XỬ LÝ ẢNH SỐ (DIGITAL IMAGE PROCESSING)

2.1.1 Giới thiệu về ảnh số

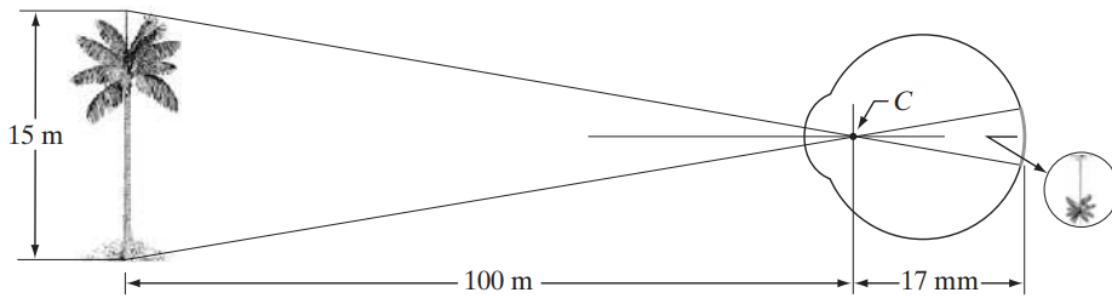
Mặc dù xử lý ảnh số được xây dựng dựa trên nền tảng toán học và xác suất, trực giác và khả năng phân tích của con người đóng vai trò quan trọng. Tuy nhiên ban đầu được lấy ý tưởng từ những hiểu biết về thị giác của con người. Nguyên tắc hình thành hình ảnh dựa trên các hoạt động cơ học và hóa học. Trước hết hãy điểm qua về các yếu tố thị giác con người trong việc thu thập hình ảnh.



Hình 2-1 Hình ảnh mặt cắt ngang đơn giản của mắt người [3]

Mắt người có hình dạng gần như hình cầu, có đường kính trung bình 20mm. Ba màng bao quanh mắt gồm: Giác mạc(Cornea) và cung mạc(Sclera), màng mạc(Choroid) và võng mạc (Retina). Giác mạc là một mô cứng trong suốt bao phủ bì mặt trước của mắt, và liên tục với giác mạc là cung mạc, cung mạc là một màng mờ đục bao quanh

phần còn lại của nhĩn cầu. Màng mạch nằm ngay dưới cung mạc. Màng này chứa một mạng lưới các mạch máu đóng vai trò là nguồn dinh dưỡng chính cho mắt. Võng mạc nơi có chức năng dẫn truyền các tín hiệu thần kinh để nhận biết ánh sáng để hình thành hình ảnh.

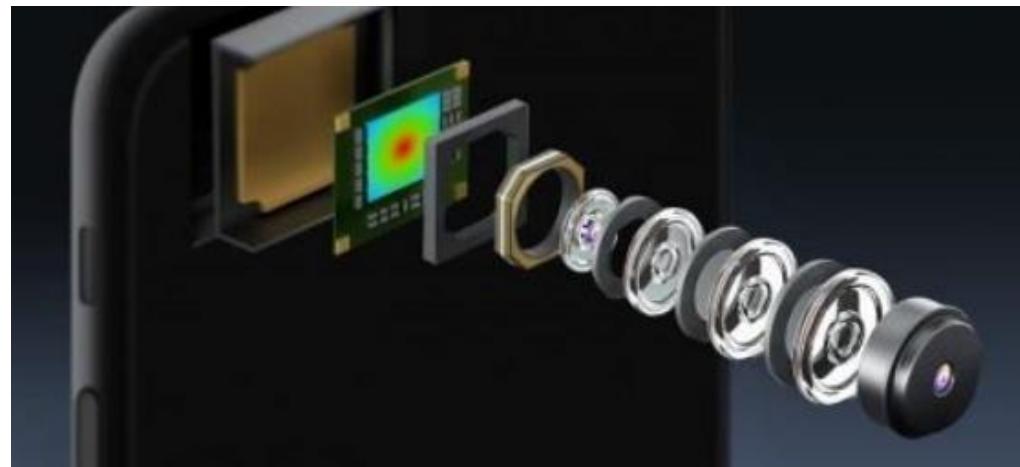


Hình 2-2 Minh họa mắt người nhìn vào cây cọ. Điểm C là tâm của thấu kính.[3]

Trong máy ảnh thông thường, ống kính có tiêu cự cố định, việc lấy nét với khoảng cách khác nhau bằng cách thay đổi khoảng cách giữa thấu kính và màn ảnh. Mắt người lại hoạt động điều chỉnh lại, khoảng cách giữa thấu kính (thủy tinh thể) và vùng thu ảnh (võng mạc) được cố định, tiêu điểm có thể thay đổi bằng cách thay đổi hình dạng của thủy tinh thể, thay đổi lỗ đồng tử.

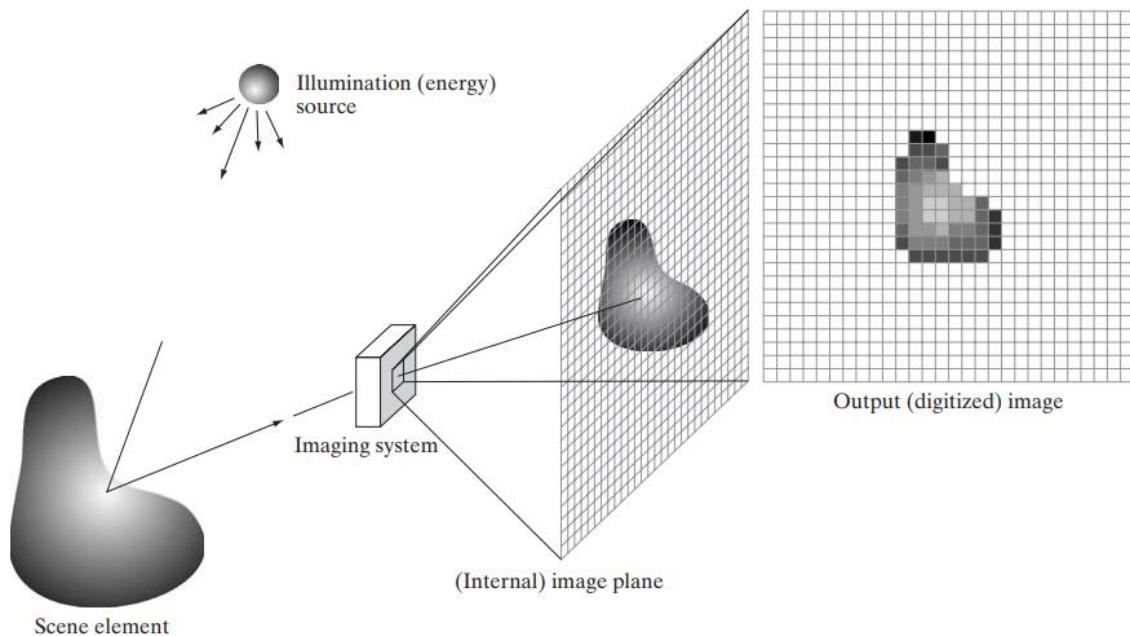
Khoảng cách từ tâm của thủy tinh thể đến võng mạc khoảng 17mm, có thể thay đổi trong khoảng 14mm đến 17mm. Giả sử khoảng cách từ mắt tới vật thể là 100m, chiều cao của vật thể là 15m, gọi chiều cao của hình ảnh được hình thành trong võng mạc là h (mm). Ta được: $15/100 = h/17$ hay $h = 2.55$ mm. Sau đó hình ảnh được mắt biến đổi thành xung điện được gửi lên não xử lý thông tin với tín hiệu này.

Vậy mắt có cơ chế thu thập hình ảnh với dạng thấu kính, hình ảnh được thu thập và xử lý ở võng mạc của mắt, nơi có các tế bào cảm thụ ánh sáng. Từ ý tưởng đó, các nhà khoa học dần nghiên cứu và triển các kỹ thuật thu thập hình ảnh. Cùng sự phát triển của công nghiệp bán dẫn và trình độ công nghệ, việc xử lý ảnh số được ra đời. Việc hình ảnh số trong các máy ảnh hiện nay được thu thập từ các cảm biến ảnh được sắp xếp với dạng một mảng hai chiều.



Hình 2-3 Cảm biến ảnh kỹ thuật số

Khi năng lượng ánh sáng đi vào trong máy ảnh đến thiết bị cảm biến sẽ tạo ra các mức điện áp tương ứng, tập hợp các cảm biến được sắp xếp để thu nhận ảnh. Hình ảnh được hình thành tạo thành một mảng các điểm ảnh.



Hình 2-4 Hình ảnh minh họa hình thành ảnh số [3]

Trong trường hợp đơn giản, ảnh được là một hàm độ sáng hai chiều $f(x, y)$. Giá trị hay cường độ tại vị trí (x, y) là một giá trị không âm được xác định từ quá trình xử lý từ cảm biến ảnh.

$$0 < f(x, y) < \infty \quad (2.1)$$

Giá trị của hàm f tại toạ độ (x, y) thể hiện độ sáng của ảnh tại điểm đó, được đặt trung bởi hai thành phần là cường độ chiếu sáng $i(x, y)$ và độ phản xạ $r(x, y)$, trong đó 0 là hấp thụ hoàn toàn và 1 là phản xạ hoàn toàn.

$$f(x, y) = i(x, y)r(x, y) \quad (2.2)$$

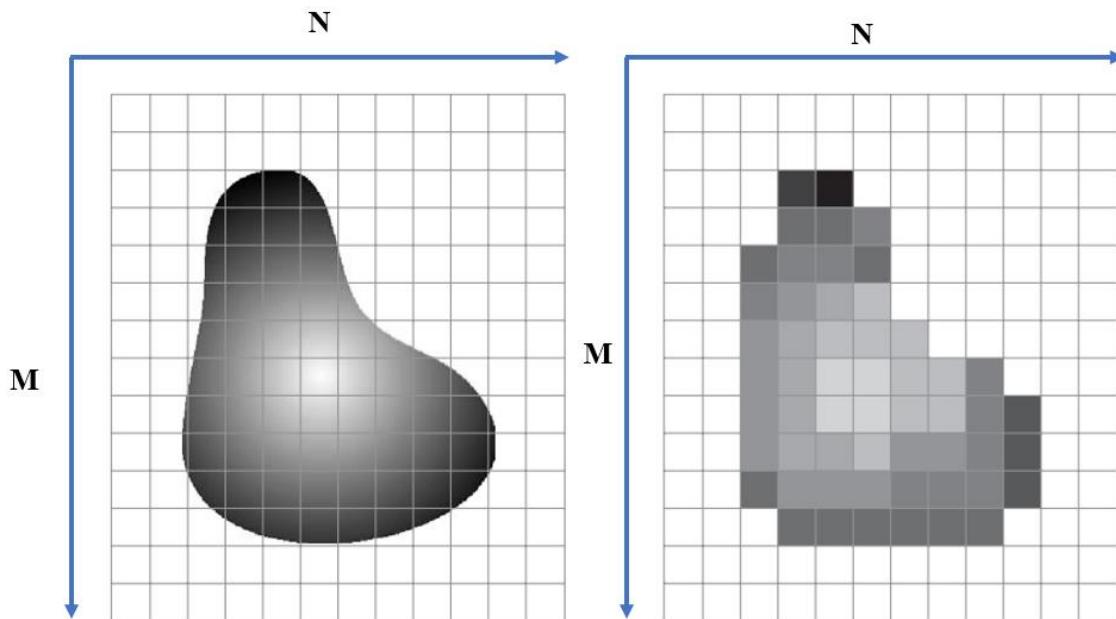
Trong đó;

$$0 < i(x, y) < \infty \quad (2.3)$$

Và

$$0 < r(x, y) < 1 \quad (2.4)$$

Việc xử lý ảnh số trong máy tính, tiến hành xấp xỉ giá trị $f(x, y) = f[x, y]$ bằng cách lấy mẫu và lượng tử hóa trong không gian giá trị độ sáng để được một ma trận có kích thước là (M, N) .



Hình 2-5 Ma trận ảnh trước và sau khi lấy mẫu và lượng tử hóa

Trong đó giá trị của đọa độ mỗi điểm $[x, y]$, với $x \in \{0, 1, 2, \dots, M - 1\}$ và $y \in \{0, 1, 2, \dots, N - 1\}$. Ma trận ảnh đơn sắc hai chiều (trường hợp ảnh đơn giản nhất) được biểu diễn với dạng ma trận sau:

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \dots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \dots & f(1, N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1, 0) & f(M-1, 1) & \dots & f(M-1, N-1) \end{bmatrix} \quad (2.5)$$

Mỗi thành phần trên ảnh số tại toạ độ (x, y) bất kì được gọi là 1 pixel hay picture element. Đây là đơn vị nhỏ nhất của bức ảnh.

Tiếp theo tìm hiểu về Độ phân giải của ảnh gồm: Mức độ chi tiết của bức ảnh được xác định dựa trên số lượng mẫu (độ phân giải không gian) và số mức xám (độ phân giải độ sáng). Trong đó độ phân giải không gian là $M \times N \times 1$ (ảnh đơn sắc) và độ phân giải độ sáng bằng $L = 2^k$, giá trị mức sáng trong thang xám là $[0, L-1]$.

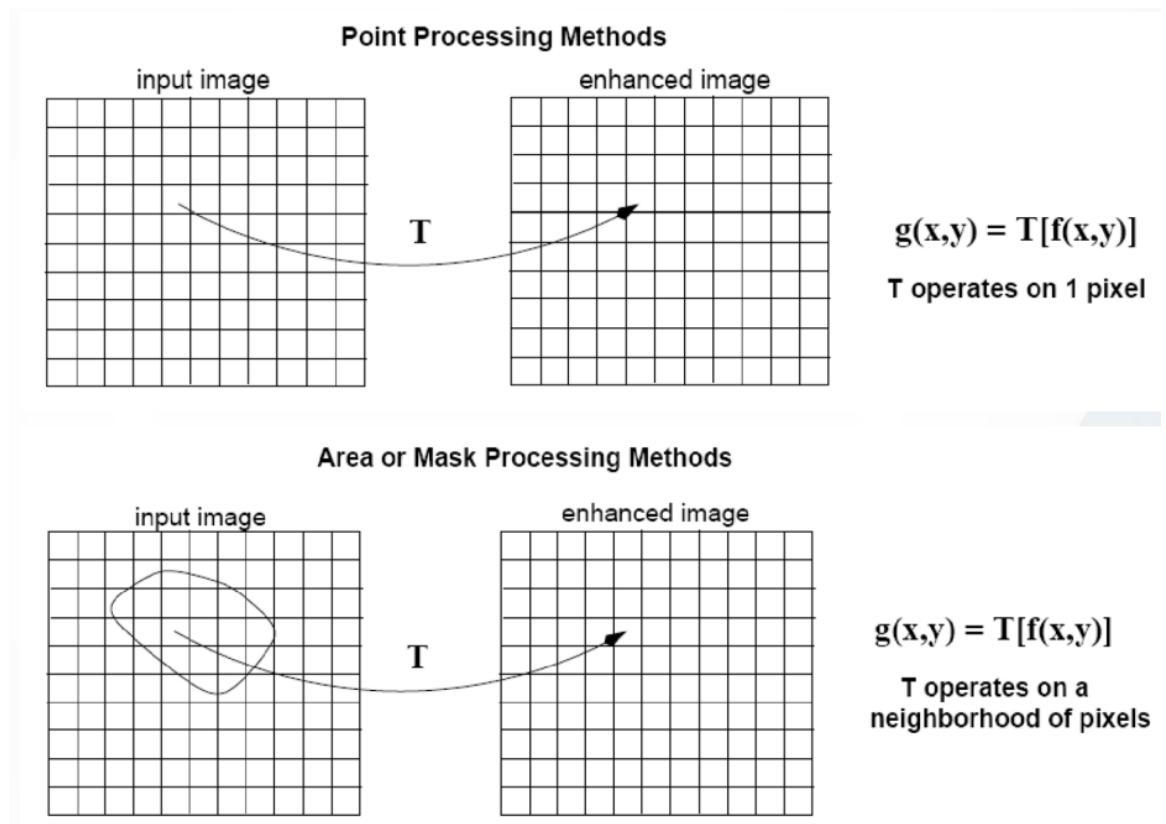
Vậy qua quá trình tìm hiểu trên đã biết được ảnh số được hình thành từ các cảm biến ảnh được lấy mẫu và lượng tử hóa tương đương một ma trận số. Sau đó được lưu trữ và xử lý các thiết bị kỹ thuật số, gồm các thông số như độ phân giải, mức sáng, kích thước lưu trữ. Ngoài ra, còn nhiều thông tin khác như cách mã hóa ảnh trong tệp tin, codec và nhiều yếu tố khác về ảnh số khác không thể trình bày hết ở đây. Tiếp theo chúng ta tìm hiểu về các kỹ thuật xử lý ảnh cơ bản, những kỹ thuật đóng vai trò cốt lõi trong xử lý ảnh hiện nay.

2.1.2 Biến đổi miền không gian (Spatial Domain Transformation)

Xử lý miền không gian là thực hiện các toán tử biến đổi ảnh đầu vào để có được ảnh đầu ra được biểu diễn tổng quát theo biểu thức sau:

$$g(x, y) = T[f(x, y)] \quad (2.6)$$

Trong đó $f(x, y)$ là ảnh đầu vào, $g(x, y)$ là ảnh đầu ra. Toán tử $T[f(x, y)]$ có thể thực hiện đổi với ảnh đơn hoặc tập hợp các ảnh. Với phần giới thiệu này, chúng ta tìm hiểu về xử lý ảnh đơn.

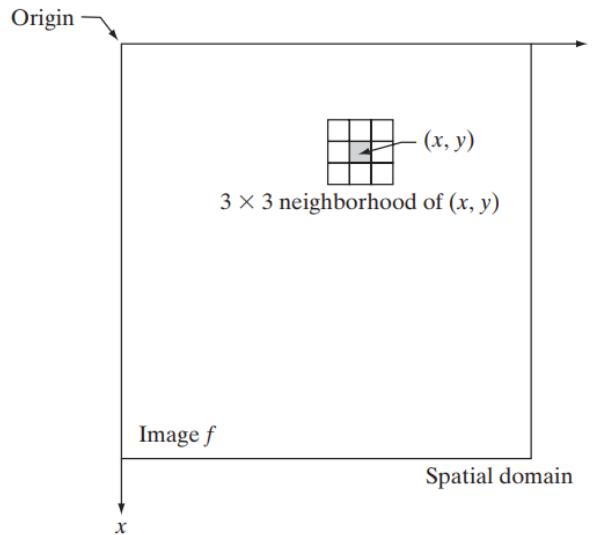


Hình 2-6 Phép xử lý điểm và Phép xử lý lân cận điểm.

Để có được ảnh sau xử lý là $g(x, y)$, chúng ta thực hiện đưa ảnh đầu vào qua toán tử $T[f(x, y)]$. Toán tử $T[f(x, y)]$ được định nghĩa là phép biến đổi giá trị f dựa trên các giá trị lân cận tại điểm có tọa độ (x, y) trong ảnh. Ví dụ: Toán tử xử lý ảnh đầu vào $T[f(x, y)]$ là trung bình cộng của các pixel lân cận với điểm có tọa độ (x, y) , với kích thước kernel là 3×3 . Ta có công thức xử lý được minh họa bên dưới.

$$g(x, y) = T[f(x, y)] \quad (2.7)$$

$$g(x, y) = \frac{1}{3 \times 3} \sum_{i=-1}^1 \sum_{j=-1}^1 f(x + i, y + j) \quad (2.8)$$



Hình 2-7 Ví dụ về xử lý miền không gian (Spatial Domain)

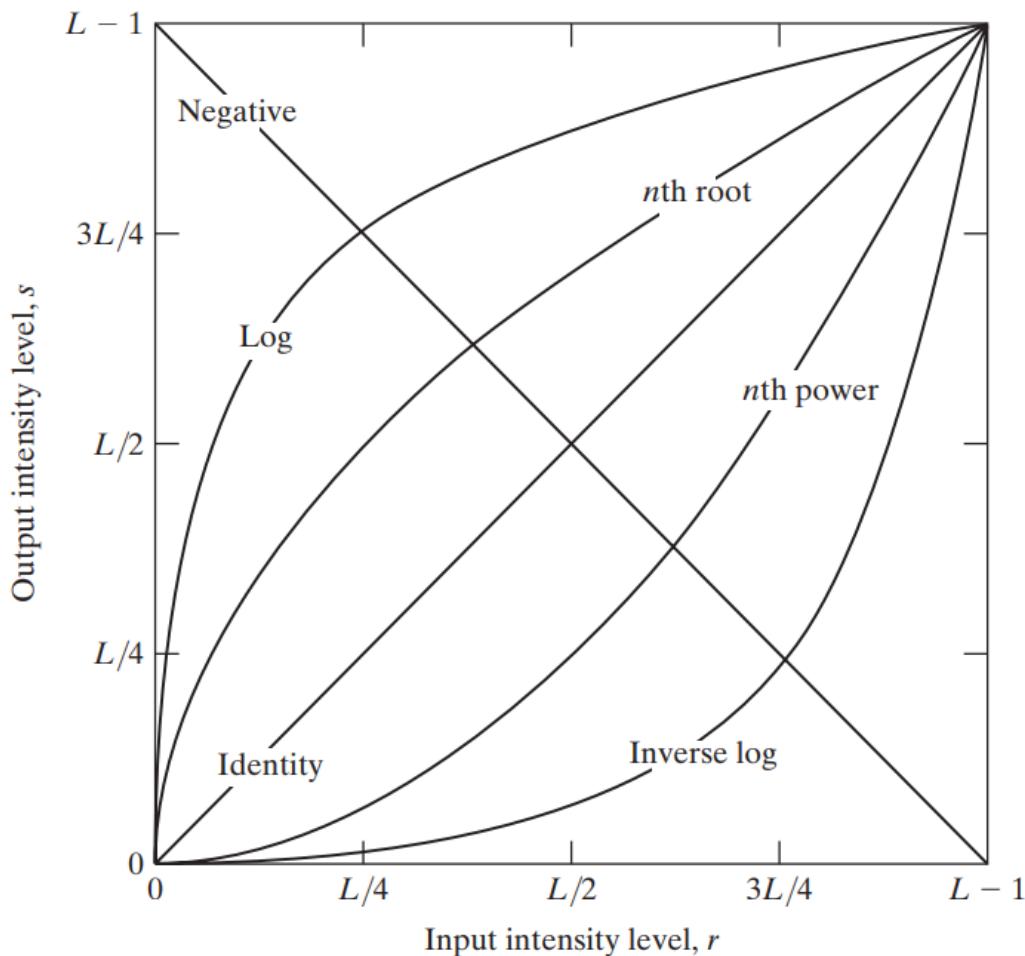
Vậy toán tử $T[f(x, y)]$ xử lý với giá trị vùng lân cận là nhỏ nhất với kích thước kernel là 1×1 . Lúc này toán tử $T[f(x, y)]$ chỉ sử dụng một giá trị của f tại vị trí (x, y) . Phép biến đổi trở thành phép biến đổi cường độ (Intensity Transformation Function) và có thể viết gọn biết thúc với dạng:

$$s = T(r) \quad (2.9)$$

Trong đó, s và r biểu thị cường độ sáng của $g(x, y)$ và $f(x, y)$ tại vị trí bất kì (x, y) . Sau đây chúng ta tìm hiểu các phép toán điểm thông dụng.

2.1.2.1 Phép toán điểm (Intensity Transformation Function)

Phép toán điểm là phép toán đơn giản nhất trong xử lý ảnh. Với việc thực hiện ánh xạ từng giá trị cường độ r trong ảnh đầu vào sang giá trị cường độ s ở ảnh đầu ra. Giả sử cường độ sáng được lưu với 8 bit, vậy các giá trị ánh xạ có tới 256 giá trị. Trong chuyển đổi cường độ sáng có thể chia ra các loại phép toán cơ bản như: Tuyến tính (âm bản và đệm), Logarithm (log và \log^{-1}) và phép toán mũ.



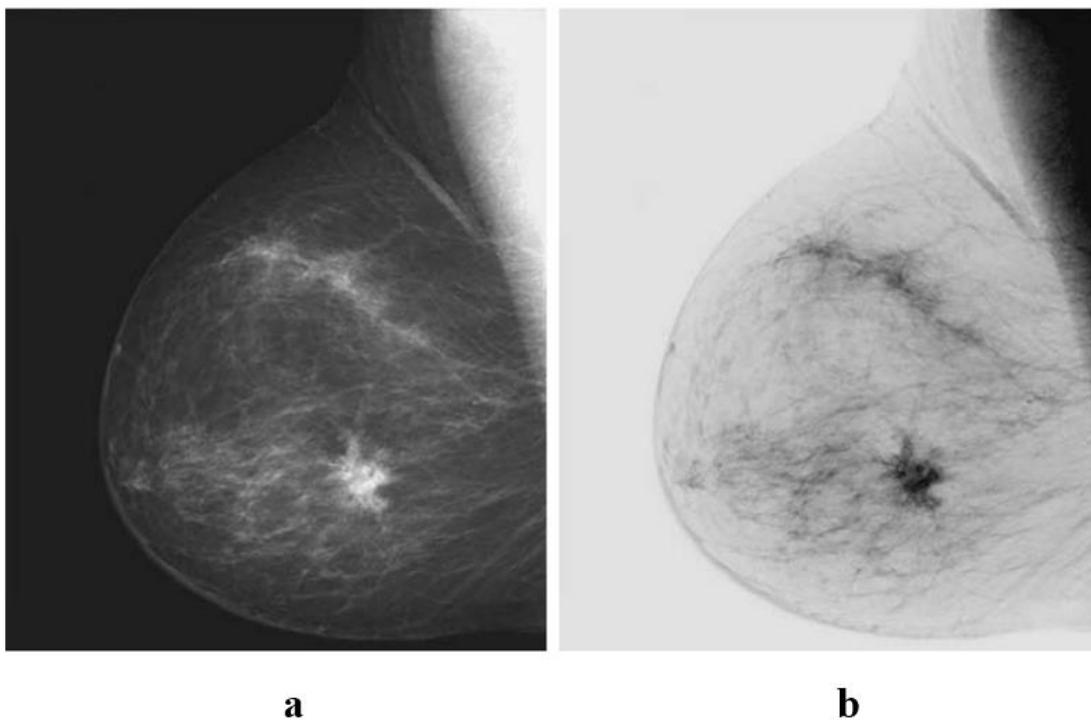
Hình 2-8 Một vài phép toán điểm cơ bản trong xử lý ảnh

- Ảnh âm bản (Image Negatives):

Ảnh đầu vào với giá trị cường độ sáng thuộc thang xám $[0, L - 1]$ khi xử dụng phép toán âm bản sẽ được thể hiện với biểu thức sau:

$$s = L - 1 - r \quad (2.10)$$

Phép toán âm bản được sử dụng để tăng cường các chi tiết màu trắng hoặc xám bị ẩn trong vùng tối, đặc biệt khi các vùng màu tối chiếm ưu thế hơn về mặt kích thước.



Hình 2-9 Ảnh chụp X – Quang tuyến vú. a) Ảnh gốc. b) Ảnh âm bản [3]

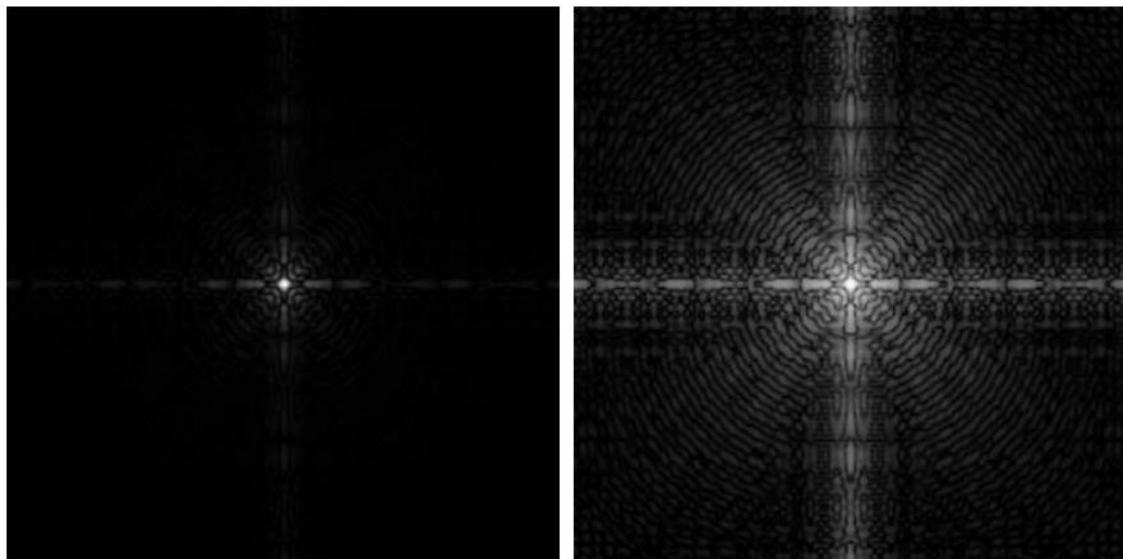
Hình ảnh chụp X – Quang cho thấy tuyến vú có một vết thương nhỏ. Mặc dù dùng mắt thường thấy rằng hai ảnh là giống nhau, nhưng việc phân tích trên ảnh âm bản sẽ trong các trường hợp cụ thể sẽ dễ dàng hơn nhiều.

- Logarithm và Inverse Logarithm

Phép biến đổi Logarithm có công thức tổng quát sau:

$$s = c \log(1 + r) \quad (2.11)$$

Trong đó, c là hằng số và giả xử thang xám của đầu vào là $[0, L - 1]$ thì giá trị của ảnh đầu ra tại điểm bất kì s cũng thuộc thang xám trên. Từ hình dạng đường cong của phép biến đổi Logarithm ở Hình 2-8, cho thấy rằng hàm \log thực hiện làm sáng lên các vùng có mức xám ở giữ của giá trị thang xám, giúp trải rộng vùng sáng trên ảnh. Ngược lại hàm \log^{-1} nén các vùng sáng của ảnh xuống, làm giảm độ sáng của các vùng sáng.



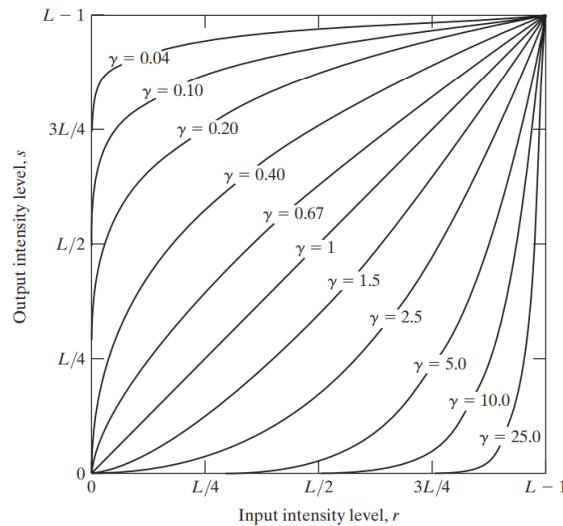
Hình 2-10 Minh họa về phép biến đổi Log a) Phổ của Fourier. b) Ảnh sau khi áp dụng phép biến đổi Logarithm với $c = 1$.

- Biến đổi lũy thừa (Power-Law Transformations)

Phép biến đổi lũy thừa có công thức tổng quát sau:

$$s = c r^\gamma \quad (2.12)$$

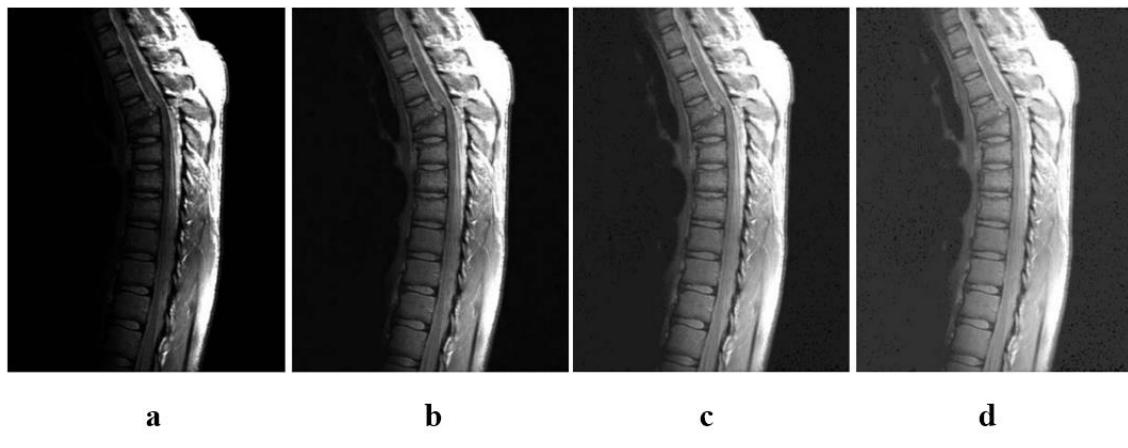
Trong đó c và γ là hai hằng số dương được định trước.



Hình 2-11 Đồ thị với biểu thức tổng quát hàm lũy thừa $s = c r^\gamma$

Tương tự như biến đổi Logarithm, biến đổi hàm lũy thừa vững có thể mở rộng hoặc thu hẹp vùng sáng. Tuy nhiên sử dụng biến đổi lũy thừa giúp lý hoạt hơn trong việc tùy chỉnh bằng cách thay đổi giá trị của γ , khi $c = \gamma = 1$ thì phép biến đổi sẽ quay về Identity transformation.

Rất nhiều thiết bị chụp ảnh tuân theo phép biến đổi lũy thừa, quá trình điều chỉnh hiện thị của ảnh các thiết bị này được gọi là “hiệu chỉnh Gamma”[3]. Ví dụ các thiết bị ống tia âm cực CRT được sử dụng các trong thiết bị hiển thị hình ảnh có cường độ điện áp với hàm số mũ thay đổi từ giá trị 1.8 đến 2.5. Với sự tham chiếu vào đồ thị Hình 2 – 11, ảnh được tạo ra từ thiết bị trên có xu hướng tối hơn ảnh mong muốn. Việc của chúng ta là thực hiện phép biến đổi trên với số mũ $\gamma < 1$ để có được ảnh có vùng sáng trải rộng hơn.



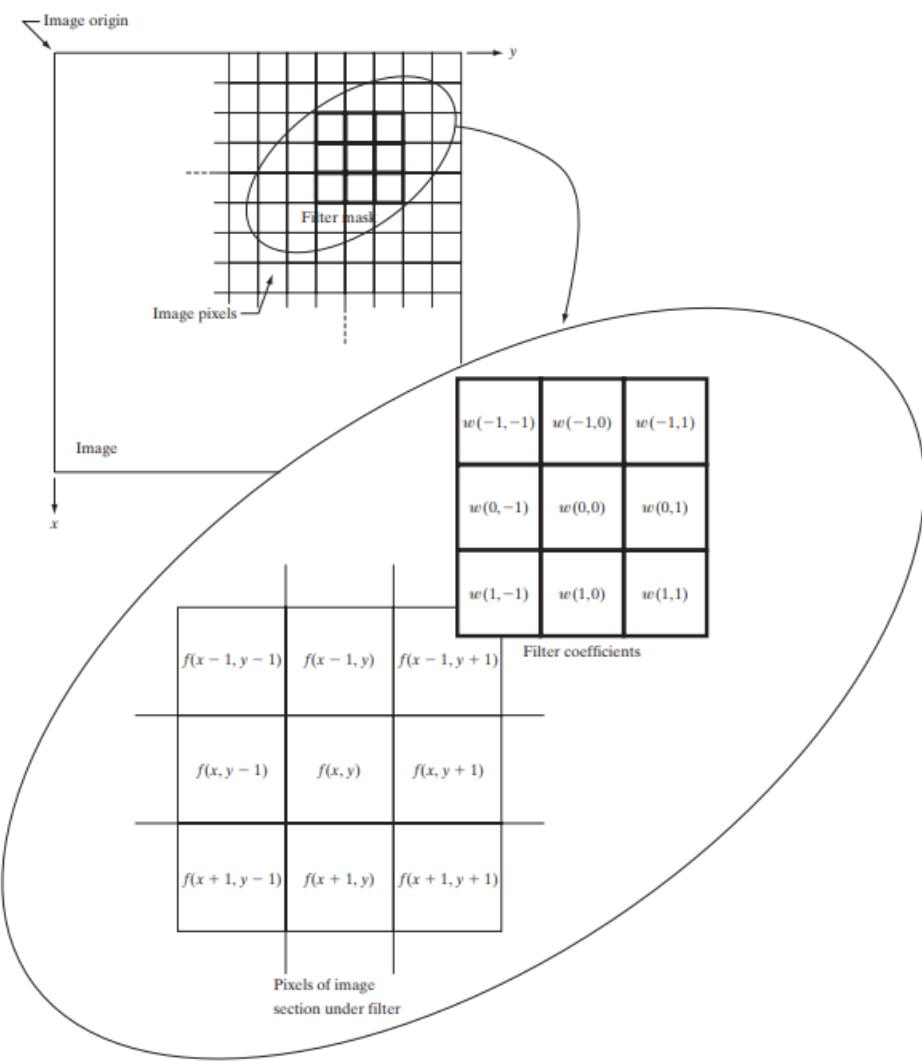
Hình 2-12 Ảnh chụp cộng hưởng từ - MRI cột sống trước ngực [3]

Trong ví dụ trên, ảnh chụp cộng hưởng từ MRI của người bị trật khớp xương và chèn tủy sống, Vết nứt nằm ở vị trí $\frac{1}{4}$ từ trên xuống từ đỉnh cột sống. a) Ảnh chụp cột sống từ máy MRI, b, c, d lần lượt là kết quả của phép biến đổi lũy thừa với $c = 1, \gamma = 0.6, 0.4, 0.3$. Việc sử dụng hiệu chỉnh γ xuống 0.4 đã mang lại kết quả hiệu quả ở ảnh c, thấy được rõ vết nứt nằm ở cột sống, sử dụng giá trị 0.3 là để thêm một chút chi tiết cho ảnh. Qua ví dụ trên cho thấy được phép biến đổi lũy thừa được dùng nhiều trong thực tế.

Bên cạnh đó, còn rất nhiều phép toán điểm khác được sử dụng trong xử lý ảnh như: Biến đổi tuyến tính từng phần, cân bằng histogram, lấy ngưỡng và lấy ngưỡng toàn cục Otsu. Do giới hạn của đề tài, các thuật toán này có thể tham khảo thêm ở tài liệu tham khảo [3].

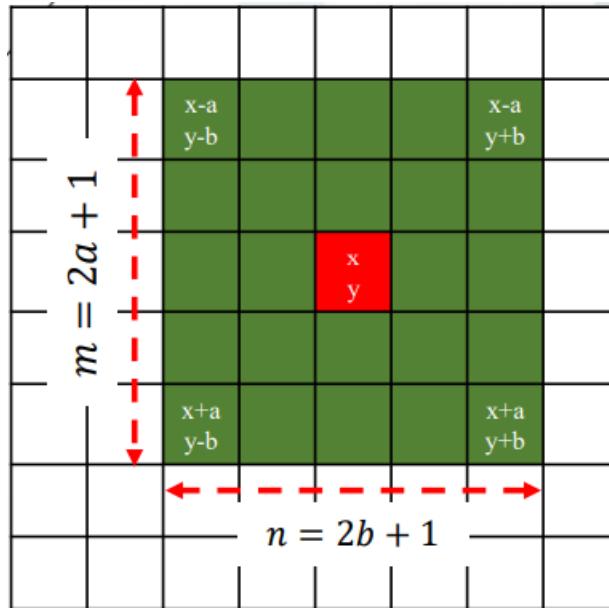
2.1.2.2 Phép toán lân cận – Xử lý không gian (Spatial Filtering)

Phép toán lân cận – xử lý không gian bao gồm một vùng lân cận, (thường là hình chữ nhật) và phép toán xác định trước lên pixel hình ảnh được bao quanh bởi vùng lân cận. Một ảnh đã qua lọc (filter), xử lý được tạo ra khi trung tâm của bộ lọc truy cập vào mỗi pixel đầu vào. Nếu thao tác thực hiện trên các pixel ảnh là tuyến tính thì được gọi là bộ lọc không gian tuyến tính (Linear Spatial Filter). Ngược lại là bộ lọc phi tuyến (Nonlinear Spatial Filter). Vậy phép toán lân cận được gọi là bộ lọc (Filter).



Hình 2-13 Minh họa cơ chế bộ lọc tuyến tính sử dụng Kernel 3×3 .

Giả sử ảnh đầu vào có kích thước $M \times N$, kernel của bộ lọc có kích thước $m \times n$, ($m = 2a + 1, n = 2b + 1; a, b > 0$).



Hình 2-14 Kích thước kernel của bộ lọc tuyến tính

Biểu thức tổng quát về bộ lọc tuyến tính trong xử lý ảnh được thể hiện như sau:

$$g(x, y) = \sum_{i=-a}^a \sum_{j=-b}^b w(i, j) \times f(x + i, y + j) \quad (2.13)$$

Trong đó, $g(x, y)$ ảnh sau khi xử lý và $w(x, y)$ là hệ số của mỗi pixel lân cận. Sau đây chúng ta tìm hiểu vài bộ lọc thông dụng tuyến tính cơ bản trong xử lý ảnh.

- Bộ lọc tương quan (Correlation Filter):

Bộ lọc tương quan di chuyển mặt nạ lọc trên hình ảnh và tính tổng giá trị cường độ sáng tại điểm (x, y) bất kì. Biểu thức tổng quát sau:

$$g(x, y) = \sum_{i=-a}^a \sum_{j=-b}^b w(i, j) \times f(x + i, y + j) \quad (2.14)$$

- Bộ lọc tích chập (Convolution Filter):

Bộ lọc tích chập cũng tương tự bộ lọc tương quan nhưng chỉ khác kernel được quay với góc 180° . Biểu thức của bộ lọc tích chập:

$$g(x, y) = \sum_{i=-a}^a \sum_{j=-b}^b w(i, j) \times f(x - i, y - j) \quad (2.15)$$

- Bộ lọc trung bình (Mean Filter)

Bộ lọc trung bình là thực hiện phép tính trung bình các giá trị pixel lân cận, có chức năng làm mờ ảnh, làm mịn ảnh.

$$g(x, y) = \frac{\sum_{i=-a}^a \sum_{j=-b}^b w(i, j) \times f(x - i, y - j)}{\sum_{i=-a}^a \sum_{j=-b}^b w(i, j)} \quad (2.16)$$

- Bộ lọc max và bộ lọc min (Max Filter và Min Filter)

Bộ lọc max tìm điểm sáng nhất trong vùng lân cận.

$$g(x, y) = \max f(x + i, y + j) \quad (2.17)$$

Bộ lọc min tìm điểm tối nhất trong vùng lân cận.

$$g(x, y) = \min f(x + i, y + j) \quad (2.18)$$

- Bộ lọc sắc nét (Sharp Filter)

Mục tiêu của bộ lọc sắc nét làm nổi bật sự chuyển tiếp về cường độ sáng. Bộ lọc sắc nét được ứng dụng nhiều trong điện tử, trong y tế, trong công nghiệp và trong quân sự. Trong bộ lọc làm mịn thực hiện phép tính lây trung bình các pixel lân cận, việc này tương tự phép toán tích phân. Vậy bộ lọc sắc nét có thể được thực hiện dựa trên phép toán vi phân.

Bộ lọc sắc nét sử dụng Laplacian, đạo hàm bậc hai của hàm hai biến $f(x, y)$.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (2.19)$$

$$\nabla^2 f = [f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1)] - 4f(x, y) \quad (2.21)$$

Việc sử dụng đạo hàm bậc hai của hàm hai biến trên giúp pháp hiện được cạnh theo chiều dọc và chiều ngang. Việc thực hiện nhận dạng được cạnh ở các góc 45° . Thực hiện điều chỉnh hệ số xuất hiện ở các góc như sau:

0	1	0
1	-4	1
0	1	0

a

1	1	1
1	-8	1
1	1	1

b

0	-1	0
-1	4	-1
0	-1	0

c

-1	-1	-1
-1	8	-1
-1	-1	-1

d

Hình 2-15 Xấp xỉ kernel của bộ lọc sắc nét

Trong đó, a) kernel được triển khai trực tiếp từ công thức Laplacian, b) là kernel có thêm thành phần phát hiện cạnh ở các đường chéo. Và c) và d) là hai ma trận được sử dụng nhiều trong thực tế.

Ngoài ra còn rất nhiều phép toán xử lý lân cận khác mà giới hạn về trình bày nên không thể đề cập hết trong phần này. Để biết thêm các phép toán bộ lọc khác tham khảo thêm tài liệu tham khảo [3].

2.2 INPAINT

Thuật toán Inpainting là một kỹ thuật xử lý ảnh được sử dụng để điền vào các vùng bị thiếu hoặc bị hỏng trong ảnh một cách tự động và hợp lý. Các vùng này có thể bị mất thông tin do nhiễu, bị che khuất, hoặc bị xóa bỏ có ý. Mục tiêu của thuật toán Inpainting là tái tạo nội dung trong các vùng này một cách thẩm mỹ và có logic để ảnh trở nên hoàn chỉnh và tự nhiên.

Với hướng phát triển của đề tài, việc sử dụng kỹ thuật Inpainting chủ yếu giúp xóa chi tiết và tái tạo vùng ảnh bị thiếu một cách có chủ ý. Kỹ thuật Inpaint qua nghiên cứu và sử dụng được hỗ trợ sẵn trong thư viện OpenCV với hai thuật toán chính: Fast

Marching Method [4] và Navier-Stokes [5]. Do kết quả xử lý của thuật toán này trong các ngữ cảnh phức tạp chưa đạt hiệu quả quá tốt. Để cải thiện hiệu quả xử lý trong ngữ cảnh phức tạp, đề tài nghiên cứu thêm kỹ thuật Inpaint Anything [2]. Chúng ta tiến hành đi vào tìm hiểu nguyên lý hoạt động của các thuật toán này.

2.2.1 *Inpaint trong OpenCV*

2.2.1.1 The Fast Marching Method (cv.INPAINT_TELEA)

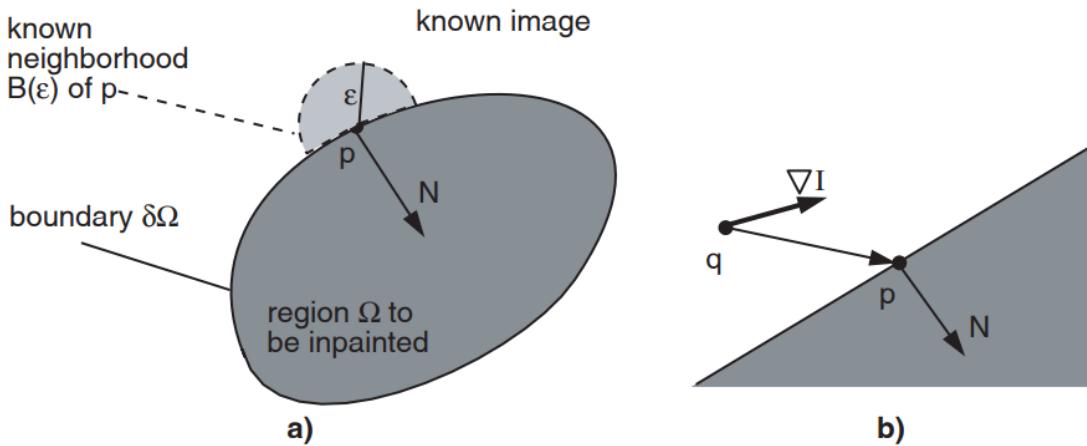
Thuật toán Inpaint đầu tiên này được lấy ý tưởng từ bài báo nghiên cứu “*An Image Inpainting Technique Based on the Fast Marching Method*” của Alexandru Telea vào năm 2004 [4].

Thuật toán Inpaint nào cũng gồm các bước tiếp cận sau: Trước hết cần xác định được vùng cần xử lý Inpaint. Tiếp theo sử dụng thông tin màu sắc của các vùng lân cận tiến hành tái tạo khu vực còn thiếu trong ảnh.



Hình 2-16 Minh họa thuật toán Inpaint sử dụng Fast Marching Method [4]

- Mô hình toán học:



Hình 2-17 Nguyên lý hoạt động của kĩ thuật Inpaint FFM [4]

Giả sử bài toán Inpaint được mô tả như trên Hình 2 -17 a), mục tiêu bài toán là tái tạo là vùng bị thiếu Ω . Trước tiên thuật toán xử lý các điểm nằm trên đường bao $\delta\Omega$.

Để vẽ được điểm p nằm trên đường bao $\delta\Omega$, thực hiện lấy một vùng lân cận nhỏ $B(\varepsilon)_p$ có kích thước là ε xung quanh điểm p như trên hình. Trong các thuật toán Inpaint nói chung để tái tạo được điểm p phải dựa vào vùng lân cận của nó, tức là $B(\varepsilon)_p$. Bài toán được thực hiện trên ảnh xám, đơn sắc. Việc thực hiện cho ảnh màu tương tự cho các kênh màu.

Xét Hình 2 – 17 b), với ε đủ nhỏ thì giá trị $I_q(p)$ (cường độ sáng) đạt giá trị gần đúng tại điểm p . Cho trước giá trị của điểm ảnh $I(q)$ và gradient $\nabla I(q)$ của điểm q , Thực hiện tính giá trị $I_q(p)$ theo công thức sau:

$$I_q(p) = I(q) + \nabla I(q)(p - q) \quad (2.22)$$

Việc thực hiện tính giá trị độ sáng tại điểm p phụ thuộc giá trị độ sáng điểm q lân cận bất kì. Chúng ta thực hiện Inpaint tái tạo điểm p dựa trên tất cả điểm q trong vùng $B(\varepsilon)_p$ bằng các tính tổng có trọng số $\omega(p, q)$ theo công thức sau:

$$I(p) = \frac{\sum_{q \in B_\varepsilon(p)} \omega(p, q) [I(q) + \nabla I(q)(p - q)]}{\sum_{q \in B_\varepsilon(p)} \omega(p, q)} \quad (2.23)$$

- Sử dụng Inpaint kết hợp với “Phương pháp bắt điểm nhanh” FFM

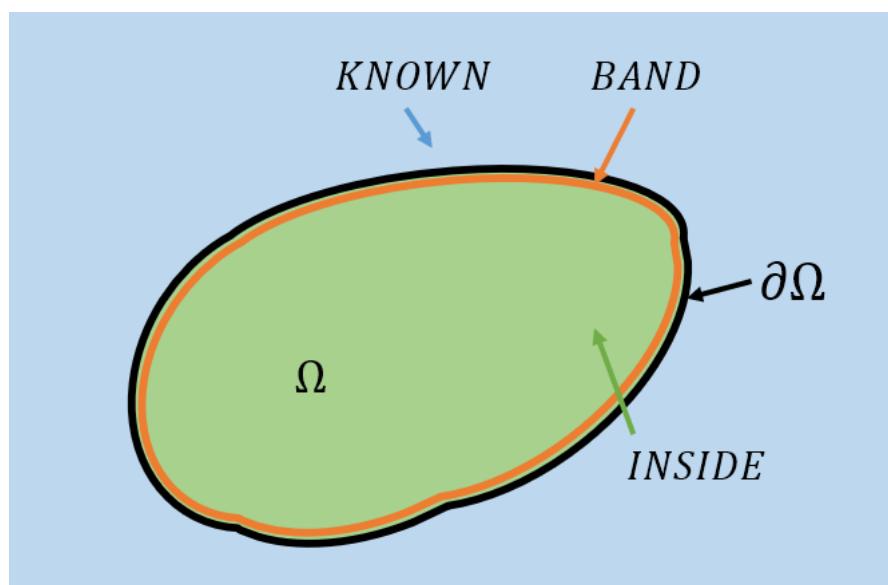
Việc tính toán giá trị độ sáng theo công thức (2.23) tính được giá trị độ sáng của điểm p . Việc thực hiện Inpaint toàn bộ vùng Ω cần thực hiện từ ngoài vào trong, việc

chọn những Pixel nào xử lý tiếp theo, bài báo sử dụng “Phương pháp bắt điểm nhanh” FFM. FFM được mô tả theo biểu thức sau:

$$|\nabla T| = 1 \quad \text{on } \Omega, \quad \text{with } T = 0 \text{ on } \partial\Omega \quad (2.24)$$

Trong đó T khoảng cách từ Pixel bên trong Ω , việc thực hiện xác định ranh giới mới $\partial\Omega_i$ chính xác dựa trên giá trị chính xác của $|\nabla T|$. Ý nghĩa của FFM là đảm bảo việc xử lý Inpaint sẽ triển khai theo thứ tự tăng dần về khoảng đến vùng ranh giới (T), tức là luôn vẽ các pixel gần nhất trước.

Việc triển khai thuật toán Inpaint sẽ cần tìm hiểu các vùng được xác định như sau:



Hình 2-18 Các vùng liên quan trong xử lý Inpaint sử dụng FFM

Với mỗi Pixel trong ảnh, thực hiện lưu trữ giá trị T , giá trị độ sáng I và một flag với ba giá trị có thể có: BAND là những pixel nằm liền kề $\partial\Omega$ (giá trị T đang được cập nhật), KNOWN là những pixel nằm ngoài $\partial\Omega$ (giá trị T và I đã biết), INSIDE là những pixel nằm trong $\partial\Omega$ (vùng cần vẽ, T và I chưa biết).

```

while (NarrowBand not empty)
{
    extract P(i,j) = head(NarrowBand);           /* STEP 1 */
    f(i,j) = KNOWN;
    for (k,l) in (i1,j1),(i,j1),(i+1,j),(i,j+1)
    if (f(k,l)!=KNOWN)
    {
        if (f(k,l)==INSIDE)
        {
            f(k,l)=BAND;                      /* STEP 2 */
            inpaint(k,l);                    /* STEP 3 */
        }
        T (k,l) = min(solve(k1,l,k,l1),
                      solve(k+1,l,k,l1),
                      solve(k1,l,k,l+1),
                      solve(k+1,l,k,l+1));
        insert(k,l) in NarrowBand;           /* STEP 5 */
    }
}

float solve(int i1,int j1,int i2,int j2)
{
    float sol = 1.0e6;
    if (f(i1,j1)==KNOWN)
        if (f(i2,j2)==KNOWN)
    {
        float r = sqrt(2(T(i1,j1)T(i2,j2))*(T(i1,j1)T(i2,j2)));
        float s = (T(i1,j1)+T(i2,j2)r)/2;
        if (s>=T(i1,j1) && s>=T(i2,j2)) sol = s;
        else
            { s += r; if (s>=T(i1,j1) && s>=T(i2,j2)) sol = s; }
    }
    else sol = 1+T(i1,j1);
    else if (f(i2,j2)==KNOWN) sol = 1+T(i1,j2));
    return sol;
}

```

Hình 2-19 Mã giả mô tả xử lý chi tiết Inpaint với FFM

Kĩ thuật FFM có quá trình khởi tạo và lan truyền như sau. Đầu tiên chúng ta tiến hành đặt giá trị T bằng 0 cho các pixel nằm trên và ngoài đường bao $\partial\Omega$, và T có giá trị rất lớn(Khoảng 10^6) nằm trong $\partial\Omega$. Sau đó tiến hành khởi tạo flag cho toàn bộ các pixel trong ảnh. Được thực hiện dựa trên cách đánh nhãn Hình 2 -18.

Chúng ta sẽ xác định được vùng dãy băng hẹp “NarrowBand” (các pixel được xác định xử lý dựa trên FFM) bằng cách sắp xếp theo thứ tự tăng dần của giá trị T . Thực hiện lan truyền xử lý vào bên trong bằng cách lặp lại việc tính toán T , f và I sử dụng đoạn mã giả trên Hình 2 – 19.

Bước 1, xác định vùng BAND lấy các điểm có giá trị T nhỏ nhất ($T = 1$). Bước 2, thực hiện tạo một đường bao mới $\partial\Omega_i$. Bước 3, thực hiện quá trình Inpaint cho các pixel trong vùng BAND. Bước 4, truyền giá trị T tại điểm (i, j) đến điểm lân cận (k, l) bằng biểu thức sau:

$$\max(D^{-x}T, -D^{+x}T, 0)^2 + \max(D^{-y}T, -D^{+y}T, 0)^2 = 1 \quad (2.25)$$

Trong đó:

$$D^{-x}T(i, j) = T(i, j) - T(i - 1, j) \quad \text{và} \quad D^{+x}T(i, j) = T(i + 1, j) - T(i, j) \quad (2.26)$$

Tiến hành giải phương trình trên với bốn góc phần tư của điểm (k, l) và giữ lại nghiệm nhỏ nhất. Bước 5, thực hiện thêm giá trị mới vào điểm (k, l) .

- Chi tiết triển khai Inpaint từng Pixel

Việc thực hiện quá trình Inpaint ở bước 3 liên quan đến việc xác định trọng số $w(p, q)$ rất quan trọng trong việc tái tạo chi tiết sắc nét và làm mịn vào vùng Inpaint. $w(p, q)$ được tính theo biểu thức sau:

$$w(p, q) = dir(p, q) \cdot dst(p, q) \cdot lev(p, q) \quad (2.27)$$

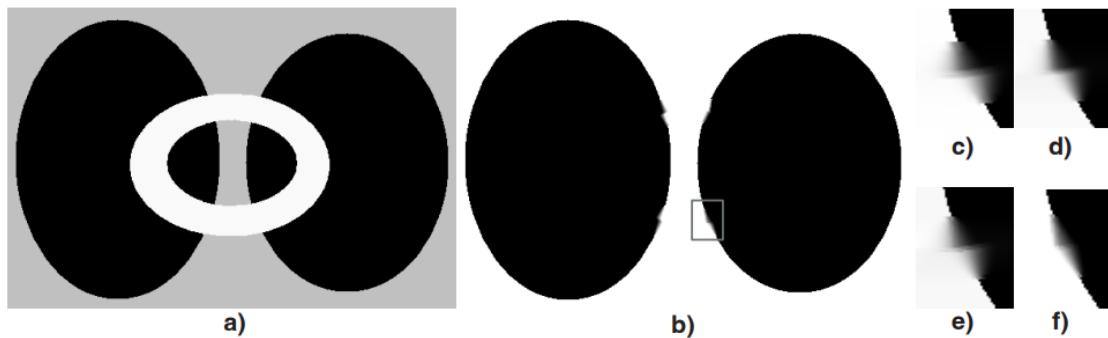
Trong đó:

$$dir(p, q) = \frac{p - q}{\|p - q\|} \cdot N(p) \quad (2.28)$$

$$dst(p, q) = \frac{d_0^2}{\|p - q\|^2} \quad (2.29)$$

$$lev(p, q) = \frac{T_0}{1 + |T(p) - T(q)|} \quad (2.30)$$

Thành phần $dir(p, q)$ đảm bảo sự đóng góp của các pixel gần với hướng thông thường $N(p) = \nabla T$, hướng lan truyền FFM. Thành phần $dst(p, q)$ khoảng cách hình học làm giảm sự đóng góp của điểm ảnh q xa hơn về mặt hình học so với p . Khoảng cách đặt mức $lev(p, q)$ đảm bảo rằng pixel nằm gần đường bao sẽ đóng góp nhiều hơn các Pixel ở xa. Cả hai thành phần $dst(p, q)$ và $lev(p, q)$ để tương đối tôn trọng về khoảng cách tham chiếu d_0 và T_0 , trong thực tế hai giá trị này sẽ được đặt thành 1.



Hình 2-20 Tác động của các yếu tố $dir(p, q)$, $dst(p, q)$ và $lev(p, q)$ đến $w(p, q)$

Hình 2-20 a) Ảnh đầu vào, với vùng cần Inpaint là vùng màu trắng như trên hình.
b) Ảnh kết quả đầu ra sau xử lý với $w(p, q)$ được trình bày ở biểu thức (2.27).

Đối với các vùng ảnh có kích thước $\varepsilon = 6$ (6 pixel), việc ảnh hưởng của hai thông số $dst(p, q)$ và $lev(p, q)$ tương đối yếu. Đối với các vùng ảnh dày hơn khoảng $\varepsilon = 12$, việc sử dụng $dst(p, q)$ và $lev(p, q)$ mang lại hiệu quả tốt hơn chỉ sử dụng $dir(p, q)$.

Hình 2-20 c) Kết quả chỉ sử dụng $dir(p, q)$, kết quả bị mờ nhiều. Hình 2-20 d) Kết quả khi sử dụng $dir(p, q)$ và $dst(p, q)$ và Hình 2-20 e) Kết quả khi sử dụng $dir(p, q)$ và $lev(p, q)$ cho kết quả tốt hơn. Hình 2-20 f) Kết quả trực quan tốt khi thực hiện Inpaint với 3 thông số trên.

- Nhận xét thuật toán

Phương pháp Inpaint sử dụng kĩ thuật FFM đạt được kết quả tốt khi kết cấu ảnh là đồng nhất giữa các vùng, hoặc các vùng có độ dày nhỏ khoảng 10 – 15 pixel. Ưu điểm lớn nhất là xử lý tốt các chi tiết nhỏ hoặc đồng nhất với thời gian xử lý nhanh.

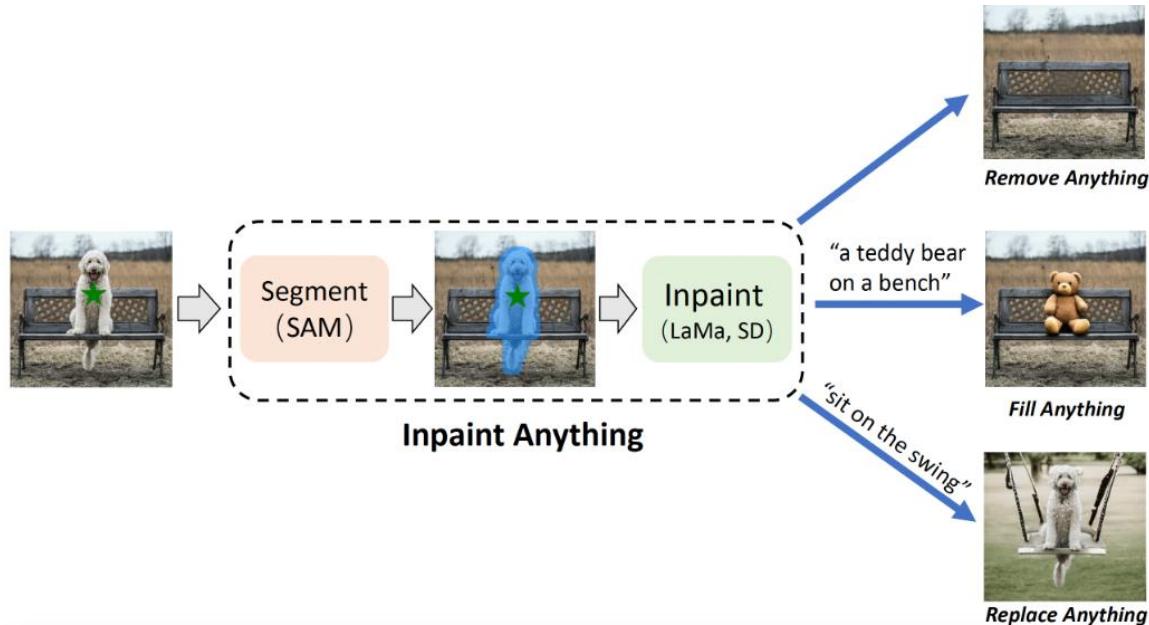
Hạn chế lớn nhất của thuật toán này là hiện tượng mờ có thể tạo ra khi các chi tiết sắc nét sát đường bao gần như tiếp tuyến. Hiện tượng này là do đặc tính tuyến tính và xử lý cục bộ của phương pháp FFM.

2.2.1.2 Navier-Stokes (cv.INPAINT_NS)

2.2.2 *Inpaint Anything*

Inpaint Anything[2] là thuật toán được xây dựng như một hệ thống Inpaint hiện đại. Xử lý bao gồm cả quá trình xác định vùng cần Inpaint và xử lý Inpaint cho vùng đó.

Thuận toán này ưu điểm các thuật toán có điển trong việc xác định vùng bao vật thể trong ngữ cảnh và tạo vùng ảnh đó phù hợp với ngữ cảnh của bức ảnh. Inpaint Anything hệ gồm ba tính năng chính: Remove Anything (thực hiện xóa vật thể), Fill Anything (điền vào dựa trên văn bản gợi ý truyền vào hệ thống) và Replace Anything (thay thế một đối tượng này bằng một đối tượng mới). Trong giới hạn của đề tài, Inpaint Anything sử dụng tính năng Remove Anything, xóa vật thể ra khỏi ngữ cảnh.



Hình 2-21 Quy trình hoạt động với ba tính năng chính của Inpaint Anything

Inpaint Anything sử dụng “*Segment-Anything Model*”[7] (SAM) dùng để xác định vùng cần tạo quanh đối tượng được chọn. Sau đó tiến hành lắp đầy vùng đó bằng “*Large Mask Inpainting*”[8] LaMa. Chúng ta tìm hiểu sơ bộ về nguyên lý hoạt động của Inpaint Anything, tính năng Remove Anything.

- Remove Anything

Remove Anything tập trung vào vấn đề loại bỏ đối tượng bằng cách cho phép người dùng loại bỏ bất kỳ đối tượng nào khỏi hình ảnh trong khi vẫn đảm bảo rằng kết quả hình ảnh vẫn hợp lý về mặt trực quan. Xóa mọi thứ bao gồm ba bước: nhấp vào, phân đoạn và xóa.

Trong bước đầu tiên, người dùng chọn đối tượng họ muốn xóa khỏi hình ảnh bằng cách nhấp vào đối tượng đó. Tiếp theo sử dụng mô hình phân đoạn như “*Segment Anything*” (SAM) được sử dụng để tự động phân đoạn đối tượng dựa trên vị trí nhấp

chuột và tạo mặt nạ. Cuối cùng, sử dụng mô hình Inpaint tiên tiến nhất chǎng hạn như LaMa là được sử dụng để lấp đầy lỗ được tạo bởi đối tượng bị loại bỏ bằng cách sử dụng mặt nạ. Vì đối tượng không còn hiện diện trong ảnh nữa, mô hình inpainting sẽ lấp đầy lỗ hỏng bằng thông tin ảnh lân cận. Lưu ý rằng, trong toàn bộ quá trình, người dùng chỉ cần nhấp vào đối tượng họ muốn xóa khỏi hình ảnh.

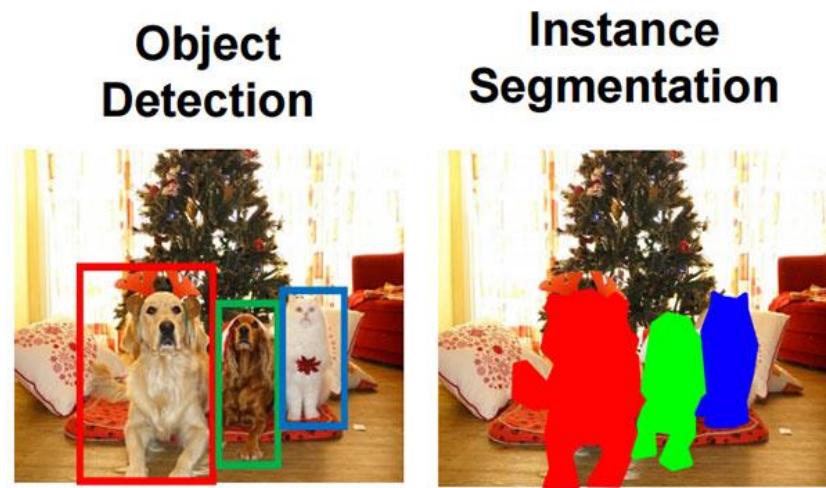


Hình 2-22 Trực quan kết quả xử lý của Remove Anything

Với mô hình xử lý Inpaint hiện đại, sử dụng Inpaint Anything trong xóa vật thể hiệu quả tốt. Để tài tiền hành nghiên cứu để áp dụng kĩ thuật này vào Ứng dụng Windows để mang lại trải nghiệm tốt nhất cho người dùng.

2.3 KỸ THUẬT PHÂN ĐOẠN ẢNH (IMAGE SEGMENTATION)

Trong lĩnh vực thị giác máy, phân đoạn ảnh là một dạng bài toán phổ biến giúp xác định vật thể hay phân loại ảnh. Các hệ thống tìm kiếm bằng hình ảnh hay các trang thương mại điện tử cũng cần trích xuất thông tin thông qua kĩ thuật phân đoạn ảnh. Ngoài ra, kĩ thuật OCR bài toán nhận diện chữ quang học, góp phần trích xuất thông tin từ chữ viết phát hiện được nhằm số hóa văn bản cũng như tài liệu cũng liên quan đến phân đoạn ảnh. Để thiết kế được mô hình có khả năng hiểu nội dung bức ảnh cụ thể hơn thì ta sẽ cần đến phân đoạn ảnh.

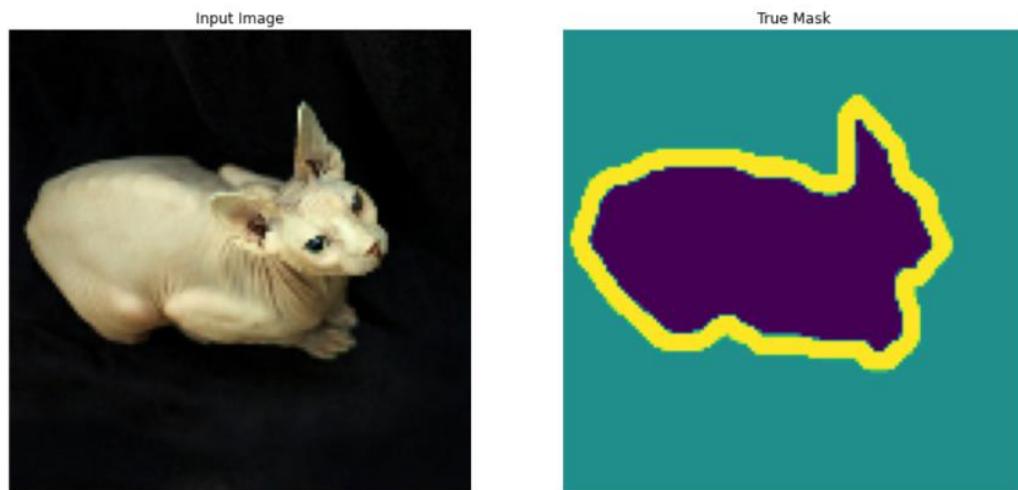


Hình 2-23 Phân biệt giữa thuật toán phát hiện vật thể và phân đoạn ảnh

Đây là kỹ thuật được xem là nâng cao hơn khi chúng ta không muốn xác định vật thể chỉ bằng khung viền (bounding box) tốt hơn để bao sát vào vật thể. Tác vụ này được gọi với một cái tên: Phân đoạn hình ảnh (Image Segmentation).

2.3.1 Đầu vào và đầu ra của phân đoạn ảnh

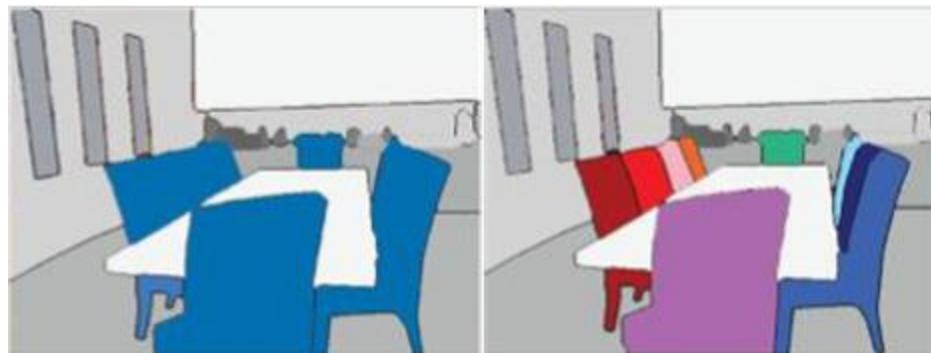
Input của bài toán là một bức ảnh và output là một ma trận mask mà giá trị của từng pixel đã được gán nhãn trên đó. Những Pixel sẽ được gán nhãn bằng một màu nhất định(hoặc một giá trị nhất định).



Hình 2-24 Input (bên trái) và Output (bên phải) của mô hình Image Segmentation

2.3.2 Các dạng phân đoạn ảnh

Có hai dạng chính trong bài toán phân đoạn ảnh:



Hình 2-25 Semantic segmentation và Instance segmentation

2.3.2.1 Semantic segmentation

Phân đoạn các vùng ảnh theo những nhãn khác nhau mà không phân biệt sự khác nhau giữa các đối tượng trong từng nhãn. Ví dụ trong hình ảnh bên trái chúng ta phân biệt được pixel nào thuộc về ghế và pixel nào thuộc về background. Tuy nhiên trong bức ảnh xuất hiện nhiều ghế, mức độ phân chia sẽ không xác định từng pixel thuộc về ghế nào.

2.3.2.2 Instance segmentation

Chúng ta phân đoạn các vùng ảnh chi tiết đến từng đối tượng trong mỗi nhãn. Ví dụ: ở hình ảnh bên phải đối với nhãn mỗi ghế sẽ được phân chia chi tiết tới từng ghế 1, ghế 2, ..., ghế 9.

2.3.3 Đánh giá bài toán Segmentation

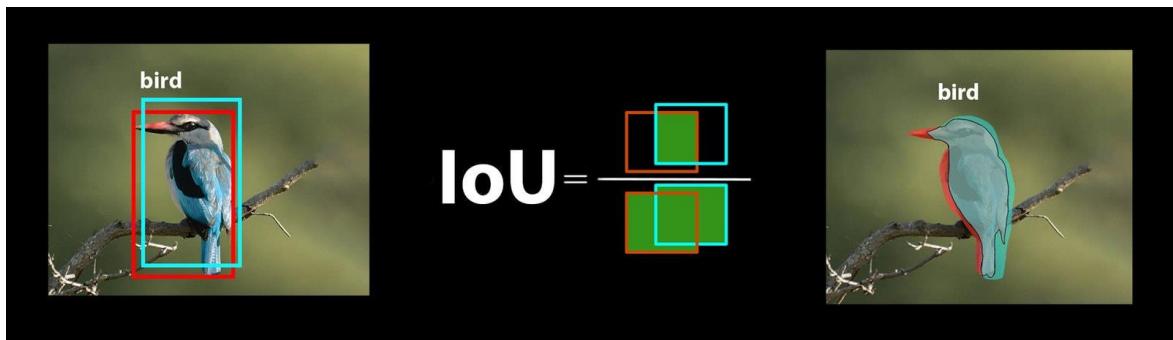
2.3.3.1 IoU (Intersection over Union)

IoU là chỉ số đánh giá được sử dụng để đo độ chính xác của phát hiện đối tượng trên tập dữ liệu cụ thể. Chỉ số này thường được gấp trong các thử thách phát hiện đối tượng (Object Detection Challenge). IoU thường được đánh giá hiệu năng của các bộ

phát hiện đối tượng như HOG + Linear SVM và neural networks tích chập (R-CNN, FastR-CNN, YOLO,...).

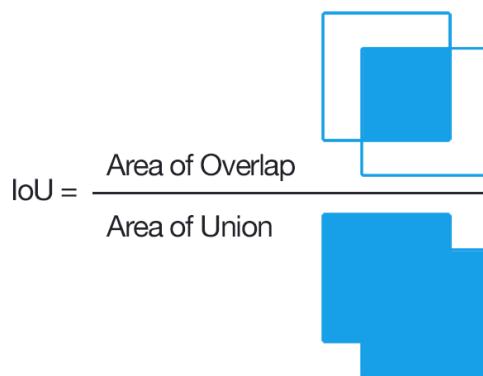
- Đường bao thực (ground-truth bounding box): là đường bao mà chúng ta gán cho vật thể bằng các phần mềm gán nhãn.
- Đường bao dự đoán (predicted bounding box): là đường bao chúng ta sử dụng tệp trọng số (file Weights) sau khi đào tạo để nhận dạng.

Dưới đây là ví dụ về đường bao thực và đường bao được dự đoán. Đường bao được dự đoán được vẽ bằng màu xanh, trong khi đó đường bao thực được vẽ bằng màu đỏ. Mục tiêu ta là tính toán IoU giữa hai đường bao.



Hình 2-26 Minh họa cách tính IoU

Tỷ lệ này là IoU là tỉ lệ giữa diện tích giao nhau giữa hai đường bao (thường là đường bao dự đoán và đường bao thực) để nhằm xác định hai khung hình có bị đè chèn lên nhau không. Tỷ lệ này được tính dựa trên phần diện tích giao nhau giữa 2 đường bao với phần tổng diện tích giao nhau và không giao nhau giữa chúng.



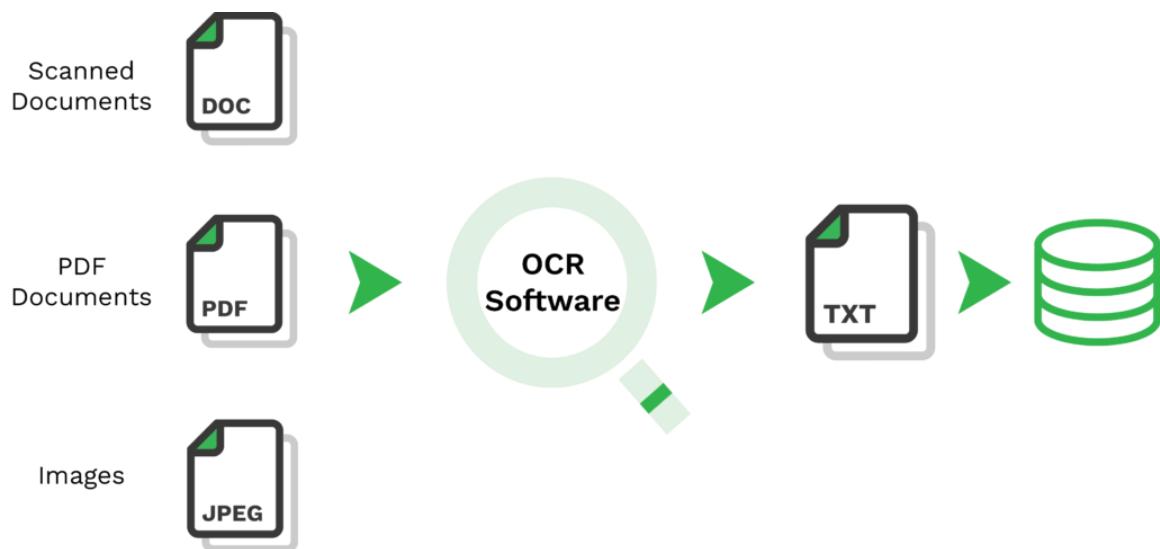
Hình 2-27 Minh họa công thức tính IoU

Các tiêu chí được dùng để đánh giá dựa trên việc xác định ngưỡng cho bài toán, giả sử bài toán nhận dạng được xem là chính xác với ngưỡng là 0.5, ta được:

- Đối tượng được nhận dạng đúng với tỉ lệ IoU > 0.5 (True positive: TP).
- Đối tượng được nhận dạng sai với tỉ lệ IoU < 0.5 (False positive: FP).

2.4 NHẬN DẠNG KÝ TỰ QUANG HỌC (OCR)

2.4.1 Giới thiệu OCR



Hình 2-28 Quy trình xử lý OCR

OCR là viết tắt của Optical Character Recognition, có nghĩa là nhận dạng ký tự quang học. Đây là một công nghệ máy tính cho phép chuyển đổi hình ảnh văn bản thành định dạng văn bản mà máy có thể đọc được.

Quy trình OCR bao gồm các bước sau:

Chuẩn bị hình ảnh: Hình ảnh văn bản cần được chuẩn bị trước khi có thể được nhận dạng bởi OCR. Điều này có thể bao gồm việc loại bỏ các yếu tố không cần thiết, chẳng hạn như đường viền, nền và các ký hiệu.

Phân đoạn: Hình ảnh văn bản được phân đoạn thành các ký tự riêng lẻ. Điều này được thực hiện bằng cách sử dụng các thuật toán phân tích hình ảnh để xác định các cạnh và góc của các ký tự.

Phân loại: Mỗi ký tự được phân loại thành một ký tự cụ thể. Điều này được thực hiện bằng cách sử dụng một cơ sở dữ liệu các ký tự mẫu.

2.4.2 *Phân loại OCR*

Có hai loại chính của OCR:

OCR đánh máy: Loại OCR này được sử dụng để nhận dạng văn bản được đánh máy. Văn bản đánh máy thường có độ chính xác cao hơn do các ký tự nhận dạng phù hợp cao hơn trong cơ sở dữ liệu các mẫu kí tự.

OCR viết tay: Loại OCR này được sử dụng để nhận dạng văn bản viết tay. Văn bản viết tay thường có độ chính xác thấp hơn vì không thể giới hạn được các vấn đề liên quan đến font chữ.

2.4.3 *Ứng dụng OCR*

OCR được sử dụng trong nhiều ứng dụng khác nhau, bao gồm:

Số hóa tài liệu: OCR được sử dụng để chuyển đổi các tài liệu giấy thành định dạng kỹ thuật số. Điều này giúp dễ dàng lưu trữ, truy cập và chia sẻ tài liệu.

Tự động hóa quy trình: OCR có thể được sử dụng để tự động hóa các quy trình, chẳng hạn như xử lý hóa đơn và nhập dữ liệu.

Tìm kiếm thông tin: OCR có thể được sử dụng để tìm kiếm thông tin trong các tài liệu văn bản.

Tăng khả năng tiếp cận: OCR có thể được sử dụng để tăng khả năng tiếp cận của thông tin cho người khiếm thị hoặc người có vấn đề về thị lực.

2.4.4 *Đánh giá bài toán OCR*

Nhận dạng ký tự quang học OCR là công nghệ hỗ trợ nhận dạng và trích xuất thông tin từ ảnh được sử dụng rất phổ biến hiện nay. Sau quá trình phát triển mô hình, các nhà nghiên cứu cần đánh giá xem mô hình có đạt được hiệu quả hay không dựa trên các chỉ số tiêu chuẩn. Trong các chỉ số đó, có hai chỉ số được sử dụng phổ biến hiện nay gồm: Tỷ lệ lỗi ký tự (Character Error Rate – CER) và Tỷ lệ lỗi từ (Word Error Rate – WER).

2.4.4.1 Các nhóm lỗi cơ bản

Cả hai chỉ số trên có ba loại lỗi cơ bản được xem xét như sau: Lỗi thay thế (Substitution), Lỗi xóa (Deletion) và Lỗi chèn (Insertion)



Hình 2-29 Ba lỗi cơ bản trong đánh giá bài toán OCR

Lỗi thay thế là lỗi làm cho ký tự hoặc từ bị sai chính tả, được đánh dấu màu xanh lá như trên hình. Lỗi xóa là lỗi mà các ký tự hoặc từ bị mất hay thiếu, được đánh dấu màu đỏ như trên hình. Còn Lỗi từ là các ký tự nhận dạng không chính xác, được đánh dấu màu xanh dương như trên hình.

2.4.4.2 Tỷ lệ lỗi ký tự (Character Error Rate – CER)

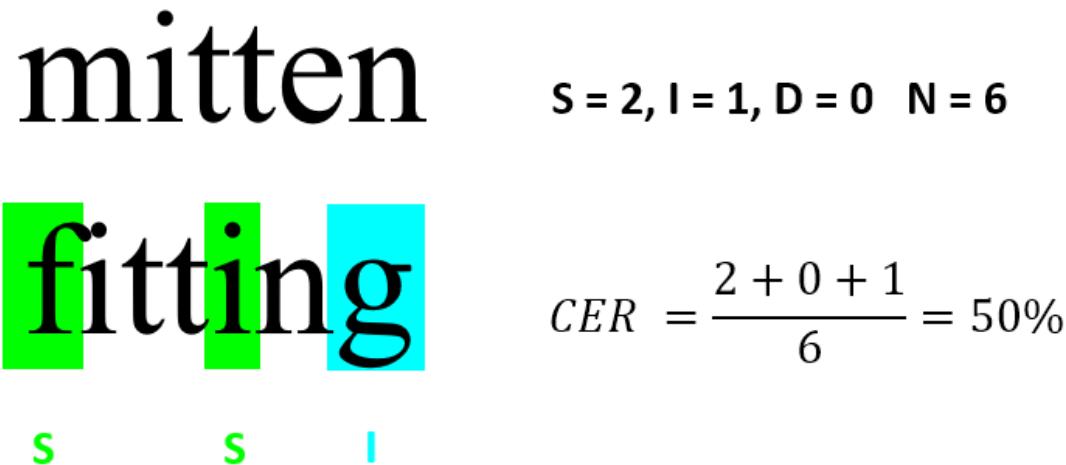
Tính toán chỉ số CER dựa trên kỹ thuật khoảng cách Levenshtein bằng cách đếm số lượng tối thiểu các hoạt động cấp ký tự cần thiết để chuyển đổi văn bản tham chiếu đầu vào văn bản đầu ra OCR.

$$CER = \frac{S + D + I}{N} \quad (2.22)$$

Trong đó:

- S: số ký tự lỗi thay thế.
- D: số ký tự lỗi xóa.
- I: số ký tự lỗi thêm.
- N: Tổng số ký tự trong văn bản tham chiếu. ($N = S + D + C$, trong đó C là số ký tự đúng)

Ví dụ: Thực hiện tính chỉ số CER giữa kí tự đầu vào là “mitten” và kí tự đầu ra của xử lý OCR là “fitting”.



Hình 2-30 Ví dụ tính Tỷ lệ lỗi ký tự CER

Tỷ lệ lỗi ký tự CER đại diện cho tỷ lệ số ký từ đầu ra OCR không chính xác so với văn bản tham chiếu đầu vào. Giá trị CER càng thấp (mô hình hoàn hảo khi CER=0), hiệu suất của mô hình OCR càng tốt. Chúng ta có thể tham khảo đánh giá độ chính xác của một số tổ chức uy tín tại Úc như sau (đối với văn bản in):

- Độ chính xác OCR tốt: CER 1-2% (tức là chính xác 98–99%)
- Độ chính xác OCR trung bình: CER 2-10%
- Độ chính xác OCR kém: CER > 10% (tức là độ chính xác dưới 90%)

Đối với các trường hợp phức tạp liên quan đến văn bản viết tay có nội dung không đồng nhất và xa rời từ vựng (ví dụ: đơn đăng ký), giá trị CER cao khoảng 20% có thể được coi là đạt yêu cầu.

2.4.4.3 Tỷ lệ lỗi từ (Word Error Rate – WER)

Nếu CER thường được sử dụng trong việc phát hiện và trích xuất các tài liệu, chuỗi ký tự có trình tự cụ thể (ví dụ: biển số xe, số điện thoại...) thì WER thường được áp dụng khi liên quan đến việc phiên âm các đoạn văn và câu chứa các từ có nghĩa (ví dụ: các trang sách, báo).

Công thức của WER giống với công thức của CER, nhưng thay vào đó, WER hoạt động ở cấp độ từ. Nó thể hiện số lượng từ thay thế, xóa hoặc chèn cần thiết để

chuyển một câu thành câu khác. WER thường có mối liên quan mật thiết với CER (miễn là tỷ lệ lỗi không quá cao), mặc dù giá trị WER luôn ghi nhận cao hơn CER.

$$WER = \frac{S_w + D_w + I_w}{N_w} \quad (2.23)$$

Ví dụ: Tính chỉ số WER với văn bản đầu vào là “My name is Kenneth” và văn bản đầu ra của xử lý OCR là “Myy nime iz Kenneth”

My name is Kenneth **S = 3, I = 0, D = 0 N = 4**

Myy nime iz Kenneth $WER = \frac{3 + 0 + 0}{4} = 75\%$

S S S

Hình 2-31 Ví dụ tính Tỷ lệ lỗi từ WER

Mặc dù chỉ số CER và WER rất tiện dụng trong việc đánh giá hiệu quả mô hình OCR nhưng đây không phải là chỉ số duy nhất cho việc đánh giá mô hình này. Độ chính xác của kĩ thuật OCR còn phụ thuộc nhiều vào yếu tố chất lượng của ảnh đầu vào và tình trạng của tài liệu gốc như DPI hình ảnh, mức độ dễ đọc của chữ viết tay. Thế nên chỉ số chỉ mang tính chất tương đối trong những trường hợp nhất định.

2.5 THƯ VIỆN SỬ DỤNG

2.5.1 WinUI 3 – .NET Framework 4.8

.NET là nền tảng phát triển phần mềm được phát triển bởi Microsoft. Cung cấp một tập hợp các công cụ, thư viện và framework để giúp các nhà phát triển tạo ra các ứng dụng web, ứng dụng máy tính, ứng dụng di động và các loại ứng dụng khác.

.NET – A unified platform



Hình 2-32 .NET Hỗ trợ đa nền tảng

.NET được thiết kế để phát triển ứng dụng đa nền tảng, .NET có thể chạy trên nhiều nền tảng khác nhau, bao gồm Windows, macOS, Linux và Android.

Ngoài ra, .NET được xây dựng hỗ trợ đa dạng cho lập trình viên từ ngôn ngữ lập trình bao gồm C#, Visual Basic, F#, C++ và Python. Đến việc hỗ trợ rất nhiều bộ thư viện hỗ trợ trong việc triển khai các thuật toán vào ứng dụng qua trình quản lý gói Nuget. .NET cung cấp các giải pháp và quy trình giúp các nhà phát triển ứng dụng triển khai từ đầu đến khi ra được sản phẩm.

Thiết kế giao diện, .NET hỗ trợ nhiều nền tảng thiết kế giao diện người dùng: Windows desktop, UWP(Universal Windows Platforms), WinUI, MAUI... Trong đó WinUI là thư viện giúp xây dựng giao diện có sự kết hợp và kế thừa của Windows desktop và UWP. Cùng với sự kết hợp với Fluent Design System mang lại trải nghiệm sâu sắc bằng cách sử dụng các kỹ thuật thiết kế đa chiều mới. Vì vậy, WinUI 3 là lựa chọn phù hợp nhất.

Với sự phổ biến là hệ điều hành Windows vô cùng phổ biến hiện nay, để tài lựa chọn sử dụng .NET Framework 4.8 để xây dựng ứng dụng trên nền tảng Windows. Giúp tạo sự quen thuộc trong sử dụng để có được hiệu quả sử dụng phần mềm nhanh nhất. Và chọn WinUI 3 cho thiết kế giao diện của ứng dụng.

2.5.1.1 Ưu điểm:

Tham vọng của Microsoft phát triển .NET trong việc cung cấp cho các nhà phát triển một nền tảng để giải quyết mọi loại vấn đề. .NET mang trong mình các ưu điểm nổi bật như:

Hỗ trợ đa ngôn ngữ: .NET là một nền tảng hỗ trợ đa ngôn ngữ nhờ vào CLR giúp biên dịch các ngôn ngữ khác nhau về mã máy. Từ đó giúp phát triển kết hợp nhiều hơn một ngôn ngữ trong một ứng dụng.

Đa nền tảng: .NET có cung cấp một thành phần quan trọng là .NET Core. Thành phần này giúp tạo một môi trường để chạy các chương trình được xây dựng bằng .NET trên các nền tảng khác như Windows, macOS, Linux. Giúp tiết kiệm chi phí phát triển, tăng khả năng tiếp cận của ứng dụng và tăng tính linh hoạt.

Trình quản lý gói Nuget: NuGet là trình quản lý gói chính thức cho nhà phát triển .NET. NuGet lưu trữ một hệ sinh thái gói không lồ tại nuget.org và cung cấp các công cụ để giúp các nhà phát triển tạo, xuất bản và sử dụng các thư viện .NET.

Mã nguồn mở: .NET sử dụng giấy phép bản quyền sử dụng các loại phần mềm mã nguồn mở là MIT. Điều này giúp .NET không ngừng được phát triển nhờ sự đóng góp của cộng đồng lập trình viên lớn.

Giao diện hiện đại: Sử dụng WinUI 3 với ngôn ngữ thiết kế Fluent Design tạo nên tính sáng tạo, linh hoạt và hiện đại trong thiết kế giao diện người dùng. Làm nâng cao tối đa trải nghiệm người dùng khi sử dụng ứng dụng.

2.5.1.2 Ứng dụng chính:

Các ứng dụng chính của WinUI 3 và .NET Framework 4.8:

- Sử dụng .NET Framework để thực hiện xây dựng kiến trúc phần mềm. Lập trình hướng đối tượng, mô hình sự kiện, thiết kế các hàm sự kiện và các hàm xử lý, xử lý đa luồng.
- Sử dụng WinUI thiết kế giao diện người dùng đầu vào, giúp nâng cao trải nghiệm người dùng. Thiết kế với các thành phần giao diện đơn giản, hiện đại và dễ sử dụng.
- Hiện thị kết quả xử lý và các hướng xử lý khác nhau từ ảnh, văn bản, và có thể copy vào bộ nhớ đệm.
- Xuất các dạng hình ảnh và thông tin trực tiếp trên máy tính cá nhân.
- Dễ dàng cài đặt và sử dụng.

2.5.2 *OpenCV Sharp*

OpenCV là một thư viện được xây dựng bằng ngôn ngữ lập trình C và C++. Ví thế để sử dụng trong C#, hay trong .NET thì chúng ta cần phải đóng gói thư viện này và viết lại một wrapper. Điều này giúp chúng ta có thể sử dụng các tính năng thi giác máy của OpenCV dưới dạng các APIs.

2.5.2.1 Ưu điểm:

Các ưu điểm cơ bản của OpenCV Sharp:

Tương tự OpenCV: Các chức năng của OpenCV tương đối hoàn thiện trong bản wrapper này. Các tính năng hoạt động tương đối ổn định.

Sử dụng ngôn ngữ C#: Có thể lập trình bằng ngôn ngữ C#, giúp phát triển các ứng dụng thi giác máy tính trên Windows được dễ dàng hơn, đặt biệt trong .NET Framework.

2.5.2.2 Ứng dụng chính:

Các ứng dụng chính của OpenCV Sharp:

- Thực hiện cá thao tác xử lý ảnh chính: Inpaint ảnh, thực hiện vẽ các mặt nạ cho việc Inpaint.
- Trích xuất ảnh dưới dạng Bitmap: Có thể thử xử lý ảnh một cách tự nhiên trong hệ điều hành dưới dạng ảnh Bitmap.
- Xử lý Inpaint cho ảnh với các mask được chọn.

2.5.3 *Tesseract Python* và *IronOCR*

Nhận dạng ký tự quang học OCR là quá trình chuyển đổi một hình ảnh văn bản thành định dạng văn bản mà máy có thể đọc được. OCR hỗ trợ nhận dạng các dạng văn bản viết tay hoặc đánh máy. Trong đó văn bản viết tay có độ chính xác thấp hơn, khó nhận dạng hơn so với văn bản đánh máy.

Có rất nhiều thư viện hỗ trợ kỹ thuật OCR, được phát triển bởi các nhà phát triển độc lập, các tổ chức nghiên cứu và các công ty phần mềm. Một số thư viện phổ biến giúp nhận dạng các ký tự quang học như:

- Tesseract: Tesseract là một thư viện OCR mã nguồn mở được phát triển bởi Google. Tesseract là một thư viện mạnh mẽ và chính xác, có thể nhận dạng nhiều loại văn bản khác nhau.
- OpenCV: OpenCV là một thư viện xử lý hình ảnh mã nguồn mở do Intel nghiên cứu cung cấp các công cụ xử lý ảnh mạnh mẽ. OpenCV có thể được sử dụng để xây dựng các hệ thống OCR.
- Google Cloud Vision API: Google Cloud Vision API là một dịch vụ đám mây của Google cung cấp các chức năng nhận dạng hình ảnh, bao gồm cả OCR. Google Cloud Vision API có thể được sử dụng để nhận dạng văn bản từ hình ảnh trực tuyến hoặc hình ảnh được lưu trữ trên đám mây.

Trong phạm vi đề tài của đồ án, chúng tôi sử dụng thư viện pytesseract cho python, và IronOCR cho C# để thử nghiệm và xây dựng các tính năng của ứng dụng của đề tài.

2.5.3.1 Ưu điểm:

Ưu điểm chung của hai thư viện pytesseract và IronOCR:

Hỗ trợ đa dạng các chuẩn ảnh: Hỗ trợ các chuẩn ảnh phổ biến như .JPEG, .PNG, .BMP, .TIFF và các chuẩn ảnh khác.

Dễ dàng cài đặt: Đối với IronOCR không cần phải cài Tesseract Engine vào hệ thống để sử dụng mà chỉ cần cài gói thư viện từ Nuget là có thể sử dụng được.

Cú pháp đơn giản: Cú pháp của cả hai thư viện này tương đối ngắn gọn là có thể đưa vào xử lý với nhu cầu cơ bản.

2.5.3.2 Ứng dụng chính:

Các ứng dụng chính của Tesseract Python và IronOCR:

- Pytesseract: dùng để thực hiện thử nghiệm các tính năng cơ bản trên python, sau đó tiến hành xây dựng các luồng xử lý cơ bản trên đây.
- IronOCR: sử dụng để xây dựng trích xuất thông tin từ ảnh cho ứng dụng trong C Sharp.
- Hỗ trợ đa dạng định dạng ảnh: Thực hiện hỗ trợ xử lý các chuẩn ảnh khác nhau tương ứng với từng nhu cầu người dùng.

- Hỗ trợ nhiều ngôn ngữ (Thông tin): IronOCR được xây dựng riêng cho các ngôn ngữ cụ thể để tăng hiệu quả nhận dạng. Trong đồ án này thực hiện xây dựng cho ba ngôn ngữ chính là Việt, Anh, Trung.

2.5.4 *Pilgram Python*

Pilgram là một thư viện Python được sử dụng để làm việc với ảnh. Thư viện này cho phép bạn thực hiện các thao tác filter – thay đổi màu ảnh bằng cách sử dụng mã nguồn Python. Pilgram dựa trên thư viện CSS gram, một thư viện mã nguồn mở cho việc chỉnh sửa ảnh trên Instagram.

Với Pilgram, bạn có thể thực hiện các tác vụ như thay đổi màu cho ảnh với nhiều định dạng được xây dựng sẵn như: _1977, aden, brannan, brooklyn, clarendon, earlybird, gingham, hudson, inkwell, kelvin, lark, lofi, maven, mayfair, moon, nashville, perpetua, reyes, rise, slumber, stinson, toaster, valencia, walden, willow, xpro2.

2.5.4.1 Ứng dụng chính:

- Xử lý bộ lọc màu cho ảnh trong Python và Ứng dụng Windows.

2.5.5 *Biểu thức chính quy – Regex*

Biểu thức chính quy – (Regular expressions hay Regex) là một chuỗi miêu tả một bộ các chuỗi khác, theo những quy tắc cú pháp nhất định. Biểu thức chính quy thường được dùng trong các trình biên tập văn bản và các tiện ích tìm kiếm và xử lý văn bản dựa trên các mẫu được quy định. Nhiều ngôn ngữ lập trình cũng hỗ trợ biểu thức chính quy trong việc xử lý chuỗi, chẳng hạn như Perl có bộ máy mạnh mẽ để xử lý biểu thức chính quy được xây dựng trực tiếp trong cú pháp của chúng.

Các thao tác có thể thực hiện được của Regex trong xử lý văn bản như:

So khớp chuỗi (Matching): So khớp là quá trình tìm kiếm một chuỗi theo một mẫu cụ thể. Ví dụ, một biểu thức chính quy có thể được sử dụng để kiểm tra xem một chuỗi có chứa một từ cụ thể không. Ví dụ: ‘/[0-9]+/’ sẽ so khớp với bất kỳ chuỗi nào có ít nhất một chữ số.

Tìm kiếm và thay thế (Search and Replace): Regex cho phép tìm kiếm chuỗi theo mẫu cụ thể và thay thế chúng bằng chuỗi khác. Ví dụ: '/apple/' sẽ tìm kiếm từ "apple" trong chuỗi và có thể thay thế nó bằng "orange".

Thao tác với ký tự đặc biệt (Special Characters): Regex sử dụng các ký tự đặc biệt để biểu thị các mẫu như ký tự đầu dòng, ký tự kết thúc dòng, các nhóm ký tự, ký tự đại diện cho ký tự trắng, ký tự số,. Ví dụ: '^' biểu thị ký tự đầu dòng, '\$' biểu thị ký tự kết thúc dòng, '.' biểu thị bất kỳ ký tự nào, '\d' biểu thị ký tự số.

Nhóm và phân đoạn (Grouping and Capturing): Regex cho phép nhóm các mẫu để xử lý chúng cùng nhau và thực hiện việc ghi nhớ các phần của chuỗi phù hợp với mẫu. Ví dụ: '/(abc)+/' sẽ khớp với chuỗi "abcababc" hoặc "abc", '/(\d{3})-(\d{4})/' sẽ khớp với số điện thoại trong định dạng "123-4567".

Bắt đầu và kết thúc (Anchors): Regex cung cấp các ký tự để chỉ ra vị trí bắt đầu và kết thúc của chuỗi hoặc dòng. Ví dụ: '^regex' sẽ khớp với chuỗi bắt đầu bằng "regex", 'regex\$' sẽ khớp với chuỗi kết thúc bằng "regex".

Quy tắc về lựa chọn (Quantifiers): Regex cho phép xác định số lượng xuất hiện của một mẫu. Ví dụ: '/a*/' sẽ khớp với chuỗi "aaaa" hoặc không có "a" nào, '/a+/' sẽ khớp với ít nhất một "a".

2.5.5.1 Ứng dụng chính:

- Hỗ trợ trích xuất thông tin theo các mẫu thông dụng như: email, số điện thoại, số tiền, ngày tháng năm, một dãy số bất kì, một chuỗi bắt đầu bằng một vài kí tự cụ thể.

CHƯƠNG 3. KẾT QUẢ VÀ PHÂN TÍCH

3.1 PHƯƠNG PHÁP TIẾP CẬN

3.1.1 *Thiên nghiệm trên Python*

3.1.1.1 Inpaint Image

Quá trình thực hiện triển khai thuật toán Inpaint trên Python có 2 hình thức tiếp cận chính dùng OpenCV và Inpaint Anything, thực hiện các bước sau:

Bước 1: Tiến hành thu thập dữ liệu ảnh thử nghiệm cho thuật toán Inpaint phân làm 3 cấp độ khác nhau:

- Cơ bản (Simple): Đối tượng thực hiện Inpaint đơn giản và kích thước nhỏ, nền có cấu trúc đơn giản tương đối đồng nhất.
- Trung bình (Intermediate): Đối tượng thực hiện Inpaint kích thước trung bình, nền có cấu trúc tương đối đồng nhất.
- Phức tạp (Complex): Đối tượng thực hiện Inpaint kích thước trung bình hoặc lớn, nền có cấu trúc phức tạp.

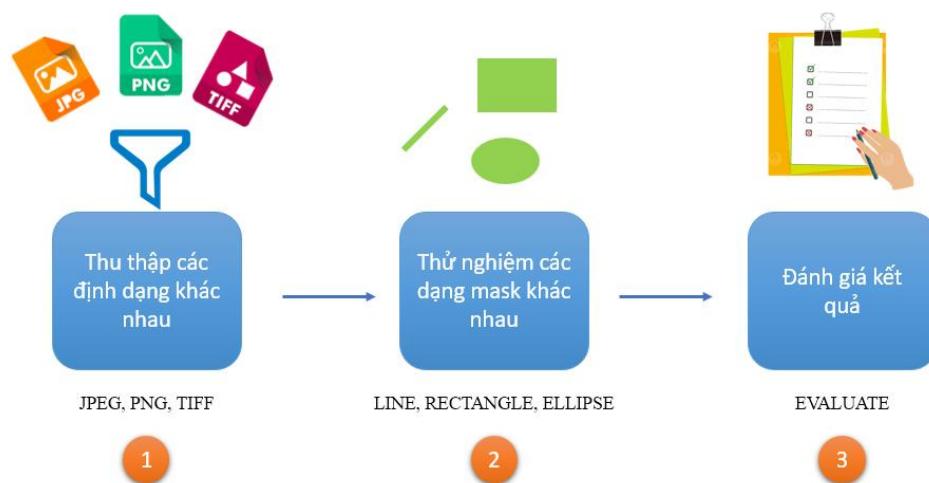
Bước 2: Thực hiện lần lượt 2 quá trình tiếp cận khác nhau:

- OpenCV: Thuật toán Inpaint trong OpenCV quy trình sau:
 - Thực hiện tải ảnh xử lý vào.
 - Tạo mặt nạ (mask) bằng các hình dạng khác nhau gồm: hình chữ nhật, hình elip, và các điểm liên tục trên hình ảnh có bán kính nhỏ.
 - Thực hiện quá trình Inpaint ảnh từ ảnh đầu vào và mặt nạ vừa tạo.
 - Lưu kết quả vừa thực hiện.
- Inpaint Anything: Thuật toán Inpaint Anything quy trình sau:
 - Thực hiện tải ảnh xử lý vào.
 - Thực hiện đánh dấu điểm chọn vào đối tượng so với ảnh đầu vào để xác định đối tượng trong hình ảnh.

- Thực hiện Inpaint Anything để xóa vật thể khỏi bức ảnh một cách hiệu quả với ngũ cành.
- Lưu kết quả vừa thực hiện.

Bước 3: Thực hiện so sánh kết quả, đánh giá và nhận xét hai thuật toán xử lý trên.

Bước 4: Thực hiện viết chương trình Python cho thuật toán, sử dụng cho việc gọi chương trình Python thông qua Terminal trong C Sharp.



Hình 3-1 Quy trình xử lý Inpaint trong Python

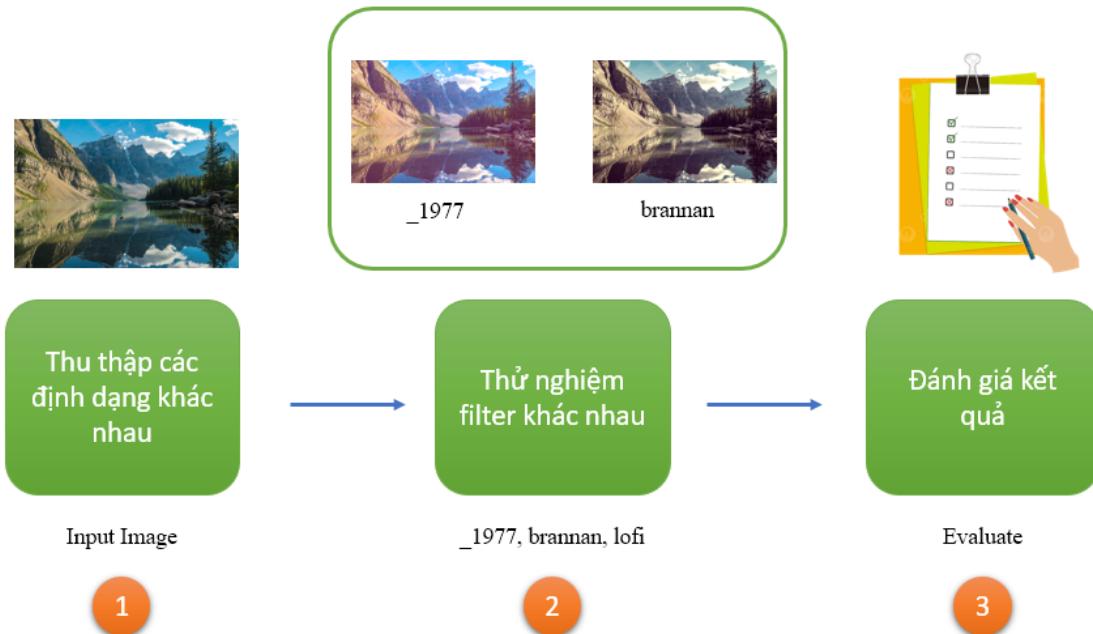
3.1.1.2 Filter – Chính sửa màu ảnh

Tiến hành sử dụng thư viện Pilgram trong Python để thực hiện thay đổi màu ảnh

Bước 1: Thu thập dữ liệu từ các định dạng ảnh khác nhau như: JPEG, PNG, TIFF, WEBP, BMP, SVG, GIFF. Thử nghiệm mức độ hỗ trợ của thư viện Pilgram.

Bước 2: Thử nghiệm các dạng Filter khác nhau có các ảnh.

Bước 3: Đánh giá thời gian xử lý và kết quả của ảnh trước và sau khi xử lý.



Hình 3-2 Quy trình chỉnh màu ảnh trong Python

3.1.1.3 OCR

Tiến hành trích xuất văn bản dùng kĩ thuật OCR trong Python thực hiện theo các bước sau:

Bước 1: Tiến hành thu thập dữ liệu ảnh thử nghiệm cho thuật toán OCR phân làm 3 cấp độ khác nhau:

- Cơ bản (Simple): Chữ đen và nền trắng (có độ tương phản tốt), không có ảnh hưởng nhiều bởi ngữ cảnh trong ảnh.
- Trung bình (Intermediate): Chữ và nền có độ tương phản tốt, tuy nhiên có ảnh hưởng với yếu tố ngữ cảnh
- Phức tạp (Complex): Chữ và nền không phân biệt rõ, số lượng chi tiết nhiều và không theo cấu trúc cố định, có yếu tố ngữ cảnh.

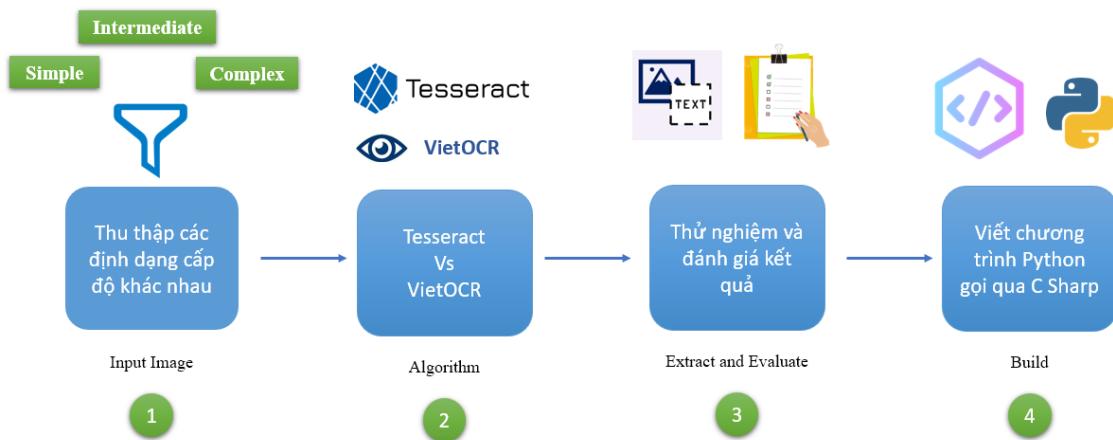
Bước 2: Thực hiện lần lượt 2 quá trình tiếp cận khác nhau:

- Pytesseract: Thuật toán OCR trong Tesseract Python quy trình sau: [Link](#), [Install Pytesseract](#) ; [Tesseract Wraper](#)
 - Thực hiện tải ảnh xử lý vào.

- Thực hiện quá trình trích xuất thông tin từ ảnh và lưu trữ. Hình thức sử dụng là toàn bộ ảnh và từng vùng ảnh.
 - Thực hiện với các ngôn ngữ khác nhau: tiếng Việt, tiếng Anh, tiếng Trung.
- VietOCR: Thuật toán OCR được thực hiện với Model transformer của VietOCR quy trình sau:
 - Thực hiện tải ảnh xử lý vào.
 - Thực hiện quá trình trích xuất thông tin từ ảnh. Hình thức sử dụng là toàn bộ ảnh và từng vùng ảnh.
 - Thực hiện với các ngôn ngữ: tiếng Việt, tiếng Anh

Bước 3: Thực hiện so sánh kết quả, đánh giá và nhận xét hai thuật toán xử lý trên.

Bước 4: Thực hiện viết chương trình Python cho thuật toán, sử dụng cho việc gọi chương trình Python thông qua Terminal trong C Sharp.



Hình 3-3 Quy trình trích xuất thông tin từ ảnh trong Python

3.1.2 *Thử nghiệm trên C#*

Tiến hành thực hiện xây dựng ứng dụng trên C# bằng .NET Framework 4.8 và thiết kế giao diện bằng WinUI 3, thực hiện các bước xử lý sau.

3.1.2.1 Thực hiện Inpaint Image trên C#:

Qua trình thực hiện Inpaint Image trong C# tiến hành các bước sau:

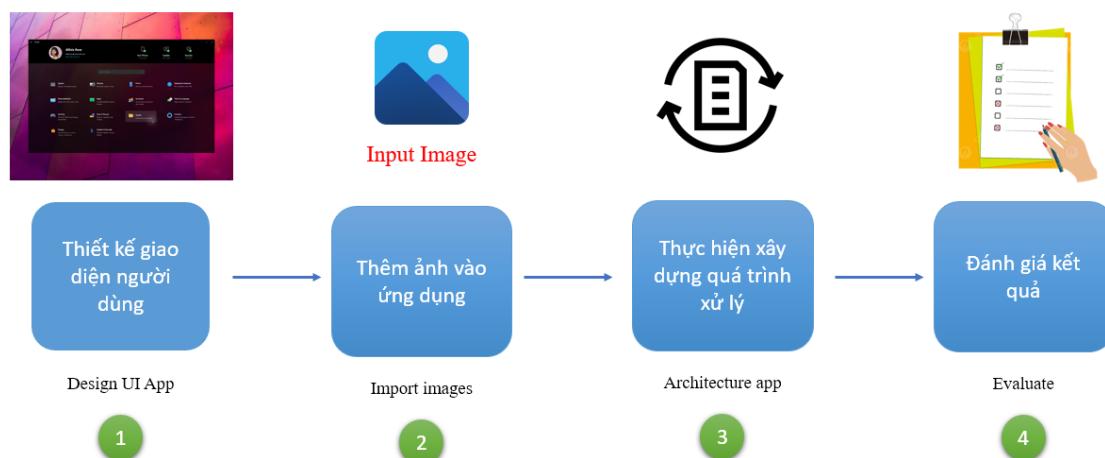
Bước 1: Thực hiện thiết kế giao diện người dùng sử dụng bao gồm: Danh sách ảnh sử dụng List View, từng ảnh đơn lẻ được hiển thị bằng Canvas, Thiết kế các bảng chọn mặt nạ thông dụng đường thẳng có độ dày, hình chữ nhật, hình elip.

Bước 2: Xây dựng quá trình thêm ảnh vào ứng dụng kết hợp với xử lý đa luồng – Multi Threads. Giúp việc thêm ảnh vào ứng dụng không bị khóa giao diện người dùng, tránh được tình trạng ứng dụng không thể tương tác trong khoảng thời gian nhất định

Bước 3: Thực hiện phần xử lý lỗi:

- Thực hiện nhận mặt nạ xử lý Inpaint vào trước bảng thao tác trên màn hình, sau đó chuyển đổi và lưu trữ dưới dạng bitmap Image mà OpenCV Sharp hỗ trợ riêng bằng cách lấy toạ độ của các điểm được vẽ trên Canvas.
- Thực hiện Inpaint Image ngay sau khi vẽ xong và tiến hành lưu trữ các ảnh được xử lý dưới dạng một danh sách các ảnh thuận tiện cho việc thực hiện các thao tác như quay lại (Undo), tiến tới (Redo) và lưu ảnh với kích thước khác nhau về sau:
- Xây dựng tính năng Undo, Redo: Thực hiện các thao tác quay lại, thuận tiện hơn cho việc chỉnh sửa ảnh, giúp các thao tác có thể tùy chỉnh dễ dàng.
- Thực hiện xây dựng phần lưu trữ ảnh xuống dưới dạng file khác, lúc này sẽ thuận tiện hơn cho việc lưu trữ ảnh sau khi xử lý.

Bước 4: Tiến hành đánh giá các bước xử lý và cải thiện thực hiện Inpaint ảnh.



Hình 3-4 Quy trình xử lý Inpaint trong ứng dụng Windows

3.1.2.2 Filter – Cảnh sửa màu ảnh

Tiến hành sử dụng thư viện Pilgram trong Python để thực hiện thay đổi màu ảnh, sau đó chúng ta thực hiện gọi các lệnh này qua Command trong C#. Thực hiện các bước tiến hành sau:

Bước 1: Thực hiện lấy dữ liệu ảnh bằng đường dẫn ảnh được thêm vào ứng dụng

Bước 2: Thực hiện xây dựng bảng chọn với các dạng filter chỉnh sửa màu khác nhau của ảnh.

Bước 3: Sau khi nhận được filter được chọn, tiến hành gọi lệnh thực thi qua code Python trong dự án để tiến hành thực hiện bước điều chỉnh màu cho ảnh.

Bước 4: Tiến hành lưu ảnh dưới dạng tạm, lưu các thông tin của ảnh gốc.

Bước 5: Sau đó tiến hành load lại ảnh vào trong ứng dụng cho người dùng xem.

Bước 6: Tiến hành đánh giá kết quả xử lý của ảnh.

3.1.2.3 OCR – IronOCR

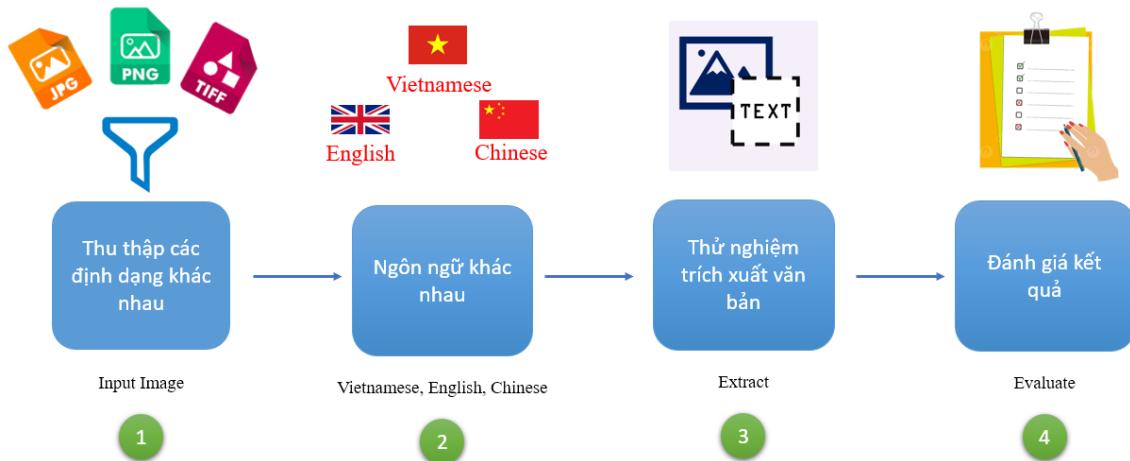
Tiến hành trích xuất văn bản dùng kỹ thuật OCR trong Python qua thư viện pytesseract thực hiện theo các bước sau:

Bước 1: Thực hiện lấy dữ liệu ảnh bằng đường dẫn ảnh được thêm vào ứng dụng

Bước 2: Thực hiện lựa chọn ngôn ngữ được thực hiện: tiếng Việt, tiếng Anh, tiếng Trung.

Bước 3: Thủ nghiệm trích xuất văn bản từ ảnh.

Bước 4: Đánh giá kết quả của ảnh trước và sau khi xử lý.



Hình 3-5 Quy trình trích xuất thông tin từ ảnh trong ứng dụng Windows

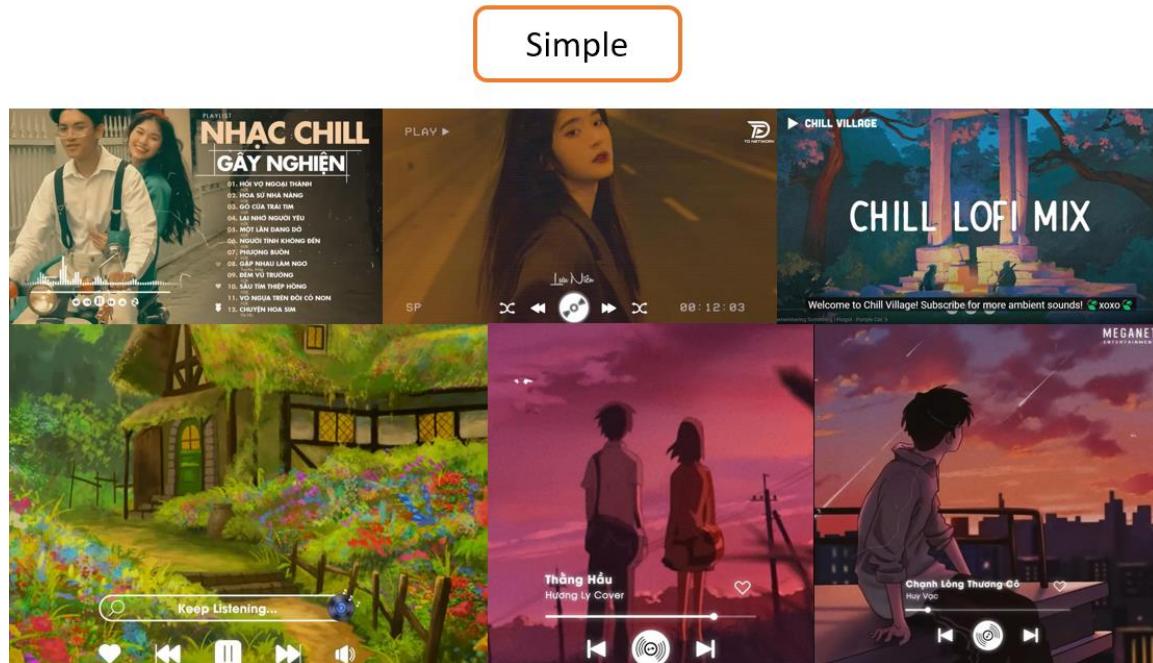
3.2 KẾT QUẢ VÀ PHÂN TÍCH

3.2.1 Thực hiện Inpaint trong Python

Quá trình thực hiện triển khai thuật toán Inpaint trên Python có 2 hình thức tiếp cận chính dùng OpenCV và Inpaint Anything đạt được kết quả sau:

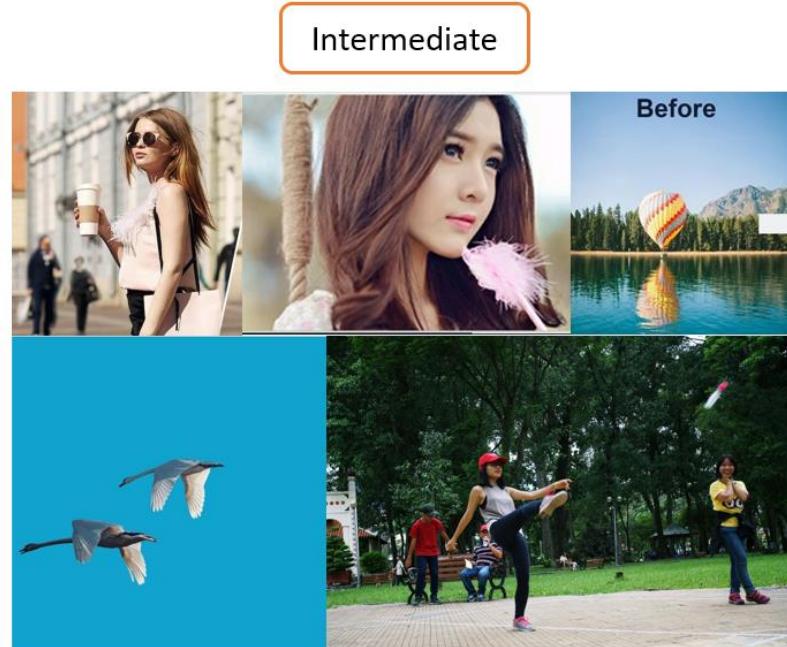
Bước 1: Tiến hành thu thập dữ liệu ảnh thử nghiệm cho thuật toán Inpaint phân làm 3 cấp độ khác nhau:

- Cơ bản (Simple): Đổi tượng thực hiện Inpaint đơn giản và kích thước nhỏ, nền có cấu trúc đơn giản tương đối đồng nhất.



Hình 3-6 Tập ảnh đơn giản trong xử lý Inpaint trong Python

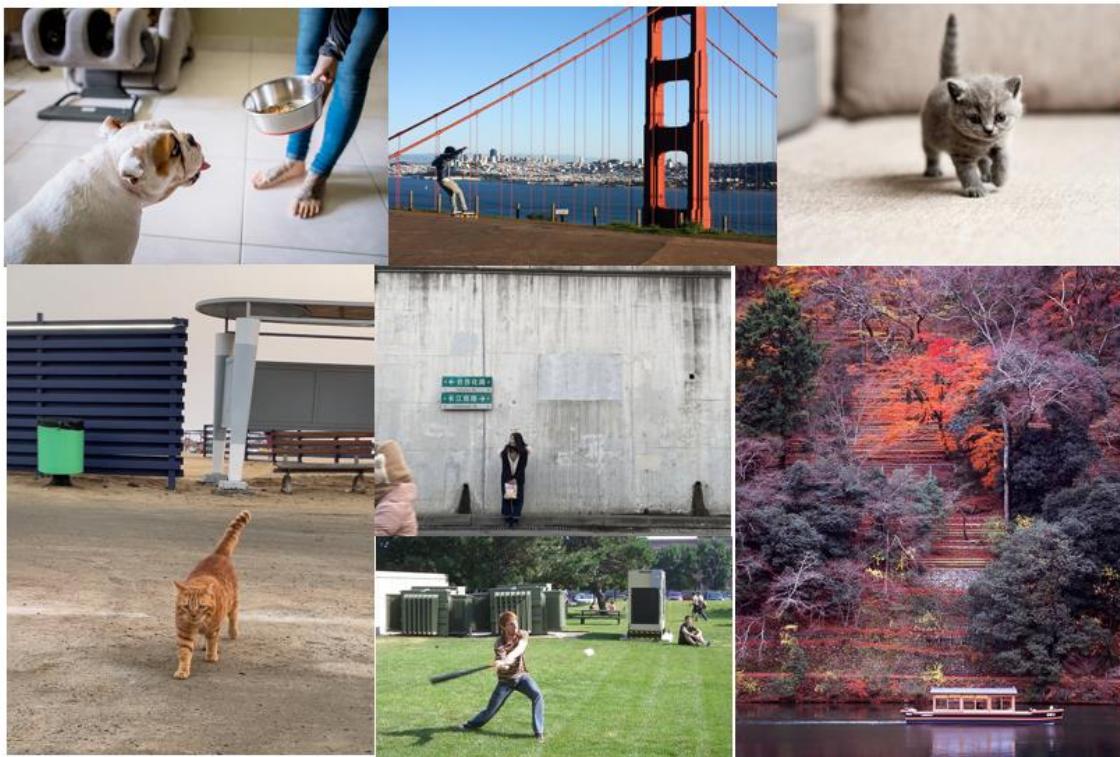
- Trung bình (Intermediate): Đối tượng thực hiện Inpaint kích thước trung bình, nền có cấu trúc tương đối đồng nhất.



Hình 3-7 Tập ảnh trung bình trong xử lý Inpaint trong Python

- Phức tạp (Complex): Đối tượng thực hiện Inpaint kích thước trung bình hoặc lớn, nền có cấu trúc phức tạp.

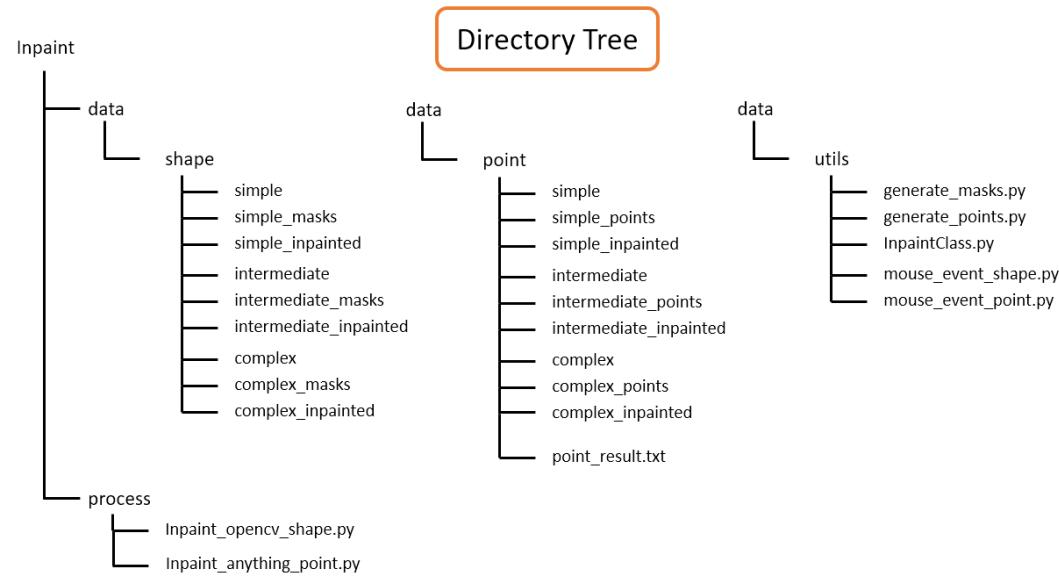
Complex



Hình 3-8 Tập ảnh phức tạp trong xử lý Inpaint trong Python

Bước 2: Thực hiện lần lược 2 quá trình tiếp cận khác nhau:

Trước hết tổ chức cây thư mục sau:



Hình 3-9 Tô chức thư mục Inpaint trong Python

Trong đó:

- ‘data’: Chứa dữ liệu đầu vào xử lý và kết quả xử lý bao gồm masks, point, inpainted theo các mức độ phức tạp khác nhau. Trong hình thức tiếp cận point, ‘point_result.txt’ chứa dữ liệu cho phần xử lý inpaint có cấu trúc như sau:

'inputPath | outputFolder | coordinate (x, y)'

Với x là toạ độ trục đứng, y là toạ độ trục ngang có đơn vị nhỏ nhất là 1 pixel.

```

1  point\simple\simple_0.jpg|point\simple_inpainted|(280, 108)
2  point\simple\simple_1.jpg|point\simple_inpainted|(410, 778)
3  point\simple\simple_2.jpg|point\simple_inpainted|(522, 716)
4  point\simple\simple_3.jpg|point\simple_inpainted|(352, 623)
5  point\simple\simple_4.jpg|point\simple_inpainted|(45, 48)
6  point\simple\simple_5.jpg|point\simple_inpainted|(691, 214)
  
```

Hình 3-10 Kết quả lý ảnh đơn giản Inpaint trong Python

- ‘utils’: Chứa các chương trình được Python hỗ trợ quá trình xử lý Inpaint
 - ‘generate_masks.py’: Thực hiện tạo mask cho tương ứng từng ảnh, sau đó kết quả sẽ được lưu vào folder tương ứng với cú pháp sau:

f'shape\{level\}_masks\{level\}_{index}.jpg'

Với:

- ‘level’: bao gồm các giá trị {‘simple’, ‘intermediate’, ‘complex’}
- ‘index’: Số thứ tự của ảnh tương ứng với ảnh đầu vào.
- o ‘generate_points.py’: Thực hiện xác định point, vị trí của đối tượng thực hiện inpaint cho tương ứng từng ảnh, sau đó kết quả sẽ được ghi vào ‘point_result.txt’
- o ‘inpaintClass.py’: Để thuận tiện cho quá trình xử lý, đề tài thực hiện tạo các class để tổ chức có thê thống và xử lý rõ ràng theo lập trình hướng đối tượng như sau:
 - o “Mode”: Đây là kiểu dữ liệu Enum, gồm ba chế độ: RECTANGLE, ELIP, POLYGON
 - o “Point”: Một điểm trên ảnh với toạ độ là x, y.
 - o “Shape”: gồm hai tham số, mode lưu trữ dạng hình của mask, param lưu trữ tập hợp các điểm dùng để vẽ một hình.
 - o “Mask”: Mỗi đối tượng lưu trữ một mask, gồm rất nhiều hình dạng(Shape) khác nhau. Có hàm thực hiện tạo mask sau khi tạo đối tượng, giúp tạo và lưu mask ra ngoài.

Sau đó chúng ta thực hiện 2 bước tiếp cận như sau:

- OpenCV: Thuật toán Inpaint trong OpenCV quy trình sau, thực hiện chạy chương trình của file ‘generate_masks.py’:
 - o Thực hiện tải ảnh xử lý vào.

```
images = [
    r'shape\simple\simple_0.jpg',
    r'shape\simple\simple_1.jpg',
    r'shape\simple\simple_2.jpg',
    r'shape\simple\simple_3.jpg',
    r'shape\simple\simple_4.jpg',
    r'shape\simple\simple_5.jpg',
    r'shape\intermediate\intermediate_0.jpg',
    r'shape\intermediate\intermediate_1.jpg',
    r'shape\intermediate\intermediate_2.jpg',
    r'shape\intermediate\intermediate_3.jpg',
    r'shape\intermediate\intermediate_4.jpg',
    r'shape\intermediate\intermediate_5.jpg',
    r'shape\intermediate\intermediate_6.jpg',
    r'shape\intermediate\intermediate_7.jpg',
    r'shape\intermediate\intermediate_8.jpg',
    r'shape\complex\complex_0.jpg',
    r'shape\complex\complex_1.jpg',
    r'shape\complex\complex_2.jpg',
    r'shape\complex\complex_3.jpg',
    r'shape\complex\complex_4.jpg',
    r'shape\complex\complex_5.jpg',
    r'shape\complex\complex_6.jpg',
]
```

Hình 3-11 Danh sách ảnh đầu vào xử lý Inpaint Python

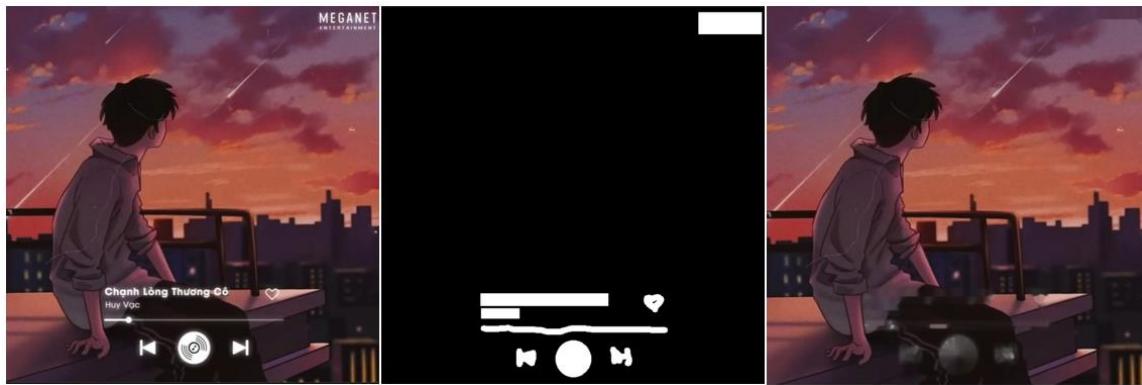
- Tạo mặt nạ (mask) bằng các hình dạng khác nhau gồm: hình chữ nhật, hình elip, và các điểm liên tục trên hình ảnh có bán kính nhỏ.



Hình 3-12 Kết quả tạo mặt nạ cho xử lý Inpaint Python

Quá trình thực hiện tạo mặt nạ (mask) được thực hiện bằng tương tác người dùng, với các phím sau: phím ‘r’ thực hiện vẽ hình chữ nhật với 2 điểm là góc trái trên và góc phải dưới ; phím ‘e’ thực hiện vẽ hình elip với 2 điểm là góc trái trên và góc phải dưới sau đó tự động tính các thông số còn lại của elip; phím ‘p’ thực hiện vẽ các điểm hình tròn liên tục với đường kính nhỏ; phím ‘n’ thực hiện nhảy đến hình ảnh kế tiếp, và phím ‘q’ dùng để kết thúc quá trình tạo mask.

- Thực hiện quá trình Inpaint ảnh từ ảnh đầu vào và mặt nạ vừa tạo



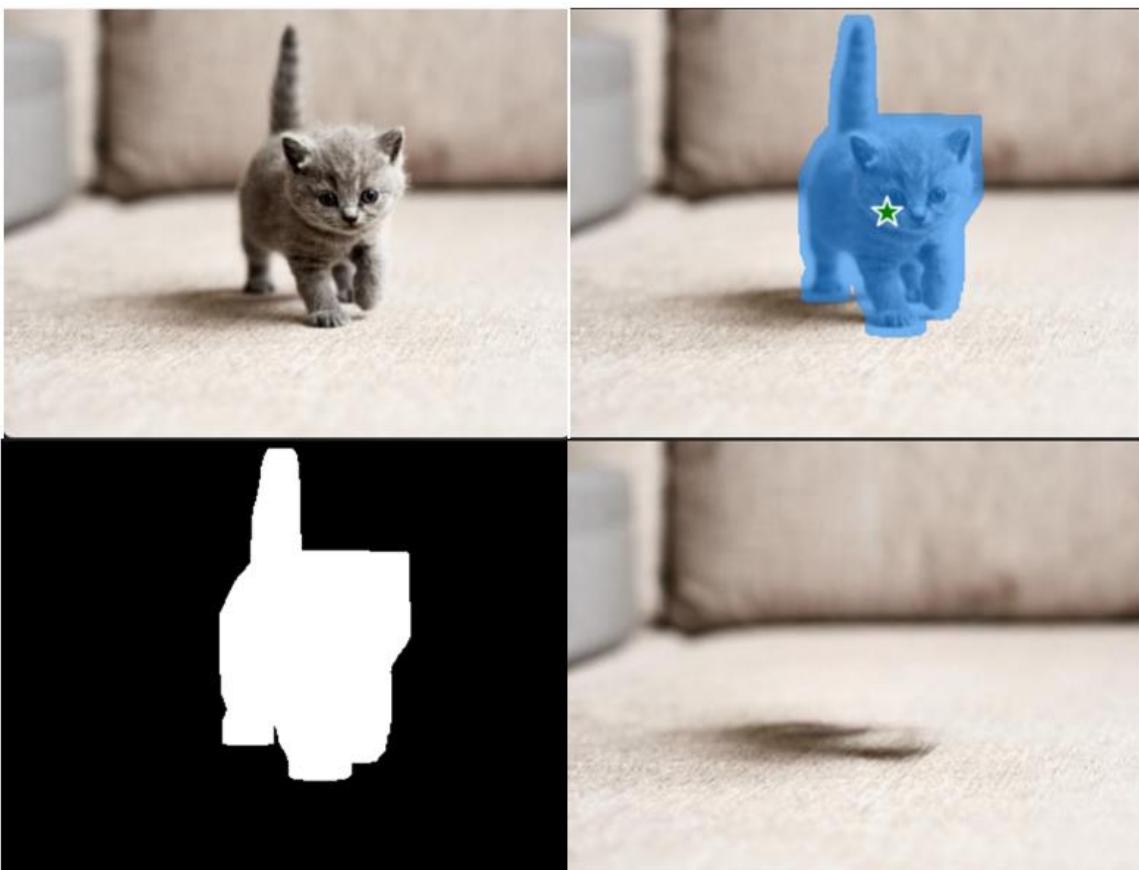
Hình 3-13 Minh họa kết quả xử lý Inpaint Python

- Kết quả trên là thực hiện với cấp độ đơn giản nhất
- Lưu kết quả vừa thực hiện.
- Inpaint Anything: Thuật toán Inpaint Anything quy trình sau:
 - Thực hiện tải ảnh xử lý vào như trên.
 - Thực hiện đánh dấu điểm chọn vào đối tượng so với ảnh đầu vào để xác định đối tượng trong hình ảnh.



Hình 3-14 Xác định vị trí vật thể trong Inpaint Anything Python

- Thực hiện Inpaint Anything để remove vật thể khỏi bức ảnh một cách hiệu quả với ngữ cảnh.



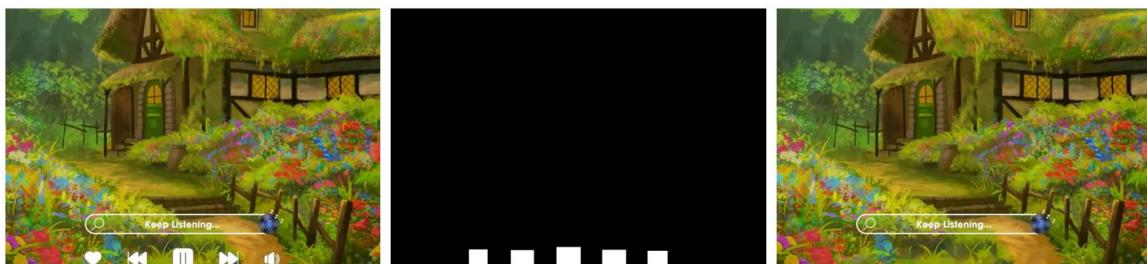
Hình 3-15 Kết quả xử lý Inpaint Anything Python

- Lưu kết quả vừa thực hiện.

Bước 3: So sánh kết quả, đánh giá và nhận xét hai thuật toán xử lý trên.

Thực hiện so sánh với mức độ phức tạp khác nhau để đánh giá thuật toán:

- Cơ bản (Simple)



Hình 3-16 Kết quả xử lý ảnh đơn giản Inpaint Python



Hình 3-17 Kết quả xử lý ảnh đơn giản Inpaint Anything Python

Cả hai thuật toán thực hiện quá trình Inpaint đều mang lại hiệu quả nhất định. Đối với thuật toán dùng OpenCV thì thực hiện xóa vật thể quá tốt trong tình huống đơn giản. Trong khi đó thuật toán Inpaint Anything cũng có thể làm điều tương tự, tuy nhiên lại chỉ có thể thực hiện chọn một đối tượng trong một phần xử lý mà chưa thực hiện nhận dạng được các đối tượng lân cận tốt.

Đánh giá: Thuật toán dùng OpenCV phù hợp hơn trong việc xử lý các chi tiết nhỏ, có độ tương phản tốt với nền và nền có cấu trúc đồng nhất. Còn thuật toán Inpaint Anything thì chưa phát huy tốt trong trường hợp các chi tiết đối tượng là nhỏ và thiết liên kết với nhau.

- Trung bình (Intermediate)



Hình 3-18 Kết quả xử lý ảnh trung bình Inpaint Python



Hình 3-19 Kết quả xử lý ảnh trung bình Inpaint Anything Python

Cả hai thuật toán thực hiện quá trình Inpaint đều mang lại hiệu quả nhất định. Đối với thuật toán dùng OpenCV thì thực hiện xóa vật không quá tốt trong mức độ trung bình. Trong khi đó thuật toán Inpaint Anything xử lý rất tốt trong trường hợp đơn vật thể, thực hiện phân đoạn ảnh để tạo mask hiệu quả, và kết quả của thuật toán Inpaint Anything là vượt trội hơn hẳn.

Đánh giá: Thuật toán dùng OpenCV gặp khó khăn trong việc xử lý các chi tiết trung bình, có độ tương kén với nền và nền có cấu trúc không đồng nhất. Còn thuật toán Inpaint Anything thì phát huy tốt trong trường hợp các chi tiết đối tượng là trung bình và phân biệt rõ với ngữ cảnh. Bên cạnh đó chất lượng xử lý Inpaint Anything là vô cùng vượt trội.

- Phức tạp (Complex)



Hình 3-20 Kết quả xử lý ảnh phức tạp Inpaint Python



Hình 3-21 Kết quả xử lý ảnh phức tạp Inpaint Anything Python

Cả hai thuật toán thực hiện quá trình Inpaint để mang lại hiệu quả nhất định. Đối với thuật toán dùng OpenCV thì thực hiện xóa vật không quá tốt trong mức độ phức tạp. Trong khi đó thuật toán Inpaint Anything xử lý rất tốt trong trường hợp đơn vật thể, thực hiện phân đoạn ảnh để tạo mask hiệu quả, và kết quả của thuật toán Inpaint Anything là vượt trội hơn hẳn.

Đánh giá: Thuận toán dùng OpenCV gặp khó khăn trong việc xử lý các chi tiết phức tạp, có độ tương kém với nền và nền có cấu trúc không đồng nhất. Còn thuật toán Inpaint Anything thì phát huy tốt trong trường hợp các chi tiết đối tượng là trung bình và phân biệt rõ với ngữ cảnh. Bên cạnh đó chất lượng xử lý Inpaint Anything là vô cùng vượt trội.

Và cuối cùng hai thuật toán để có các ưu nhược điểm khác nhau, sau đây là đánh giá về 2 thuật toán;

- Inpaint – OpenCV:

- **Ưu điểm:** Thuận tiện cho việc xử lý nhiều đối tượng cùng một lúc, có thể thực hiện được các dạng mask khác nhau giúp quá trình xử lý vô cùng linh hoạt, bên cạnh đó kết quả xử lý nhanh. Đạt hiệu quả nhất với các ảnh ở

cấp độ cơ bản (Simple) với chi tiết nhỏ, độ tương phản cao và cấu trúc nền là đồng nhất

- Nhược điểm: Cần phải thực hiện xác định mask một cách chính xác và vừa đủ, không thể thực hiện Inpaint với các chi tiết quá phức tạp và nền có cấu trúc không đồng nhất. Kết quả thực hiện có thể ảnh hưởng nhiều đến ảnh sau xử lý khi chi tiết là lớn so với bức ảnh.

- Inpaint Anything:

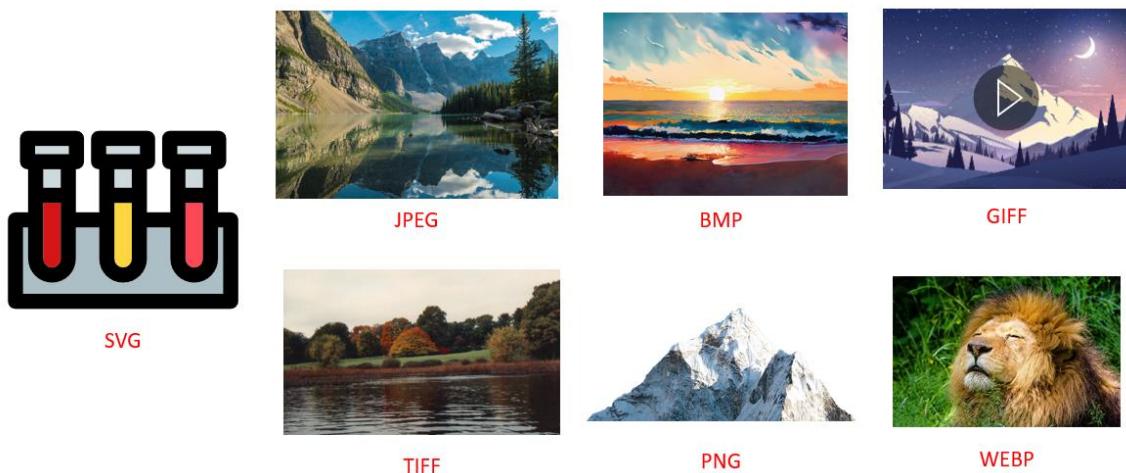
- Ưu điểm: Xử lý có độ chính xác cao đối với cấp độ khác nhau từ cơ bản đến phức tạp. Hiệu quả ảnh sau xử lý là rất tốt so với thuật toán Inpaint OpenCV. Thực hiện nhận dạng được các đối tượng có tích chất tương đồng và tạo mask phù hợp với ngữ cảnh nhất.
- Nhược điểm: Thực hiện chỉ một đối tượng ở một thời điểm, quá trình thực hiện tương đối lâu hơn. Tránh các đối tượng xung quanh có tính chất khá tương đồng với đối tượng được chọn có thể khiến đối tượng bị nhầm lẫn

Bước 4: Thực hiện viết chương trình Python cho thuật toán, sử dụng cho việc gọi chương trình Python thông qua Terminal trong C Sharp.

3.2.2 Thực hiện chỉnh màu ảnh - Filter trong Python

Thực hiện chỉnh sửa màu ảnh trong Python thực hiện các bước sau:

Bước 1: Tiến hành thu thập dữ liệu ảnh ở các định dạng khác nhau:



Hình 3-22 Các định dạng ảnh khác nhau chỉnh sửa màu ảnh Python

Thực hiện filter trên các chuẩn ảnh khác nhau để xem thư viện hỗ trợ chuẩn ảnh nào, Dùng filter Kelvin để thực hiện lọc màu ảnh để xem kết quả mang lại.



Hình 3-23 Kết quả xử lý các định dạng ảnh có thể chỉnh sửa màu trong Python

Dựa vào kết quả trên, thư viện Pilgram thực hiện tốt đói với các chuẩn ảnh như: JPEG, PNG, TIFF, WEBP, BMP. Và chưa thực hiện tốt đói với chuẩn ảnh GIFF khi không đảm bảo được chi tiết ảnh động của chuẩn ảnh này. Cuối cùng thư viện không hỗ trợ chuẩn ảnh SVG.

Bước 2: Thực hiện xử lý với các ảnh đâu vào khác nhau với các filter sau: _1977, aden, brannan, brooklyn, clarendon, earlybird, gingham, hudson, inkwell, kelvin, lark,

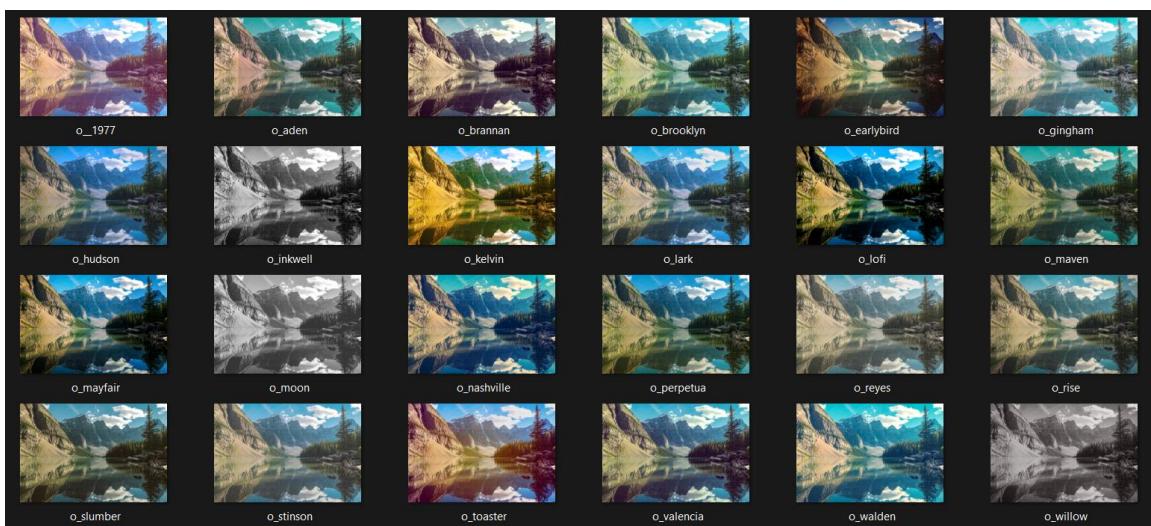
lofi, maven, mayfair, moon, nashville, perpetua, reyes, rise, slumber, stinson, toaster, valencia, walden, willow, xpro2.

Thực hiện các thuật toán Filter với ảnh đầu vào là các chuẩn ảnh trên với đầu vào là ảnh JPEG, một chuẩn ảnh phổ biến hiện nay.



Hình 3-24 Ảnh đầu vào chỉnh sửa màu ảnh Python

Sau quá trình thực hiện chạy code tự động, kết quả thử nghiệm chúng ta được kết quả sau:

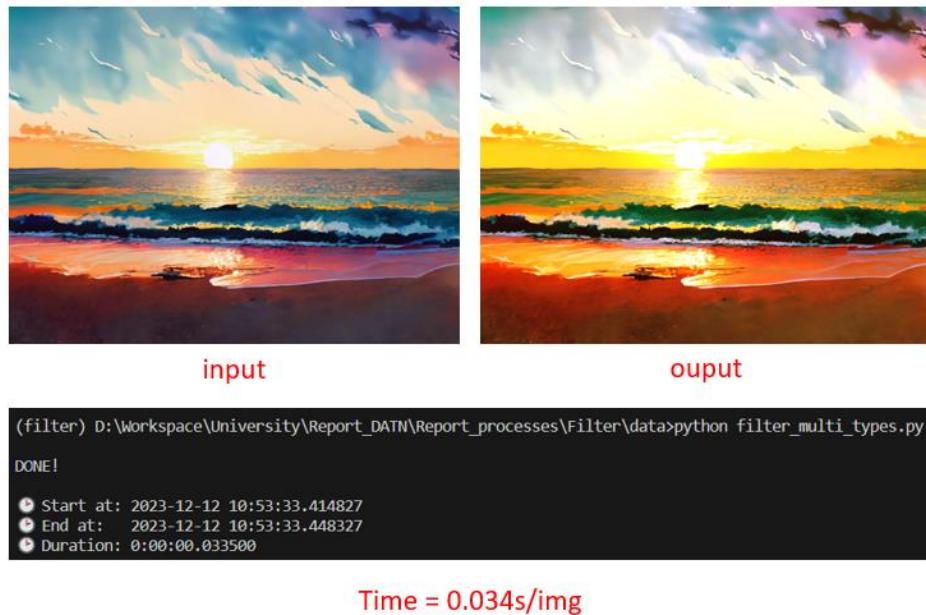


Hình 3-25 Kết quả chỉnh sửa màu ảnh hàng loạt Python

Việc thực hiện filter tất cả các dạng cho ảnh thực hiện tốt, và đạt được những màu sắc đang.

Bước 3: Tiến hành đánh giá thời gian xử lý và kết quả thuật toán.

Thực hiện xử lý ảnh với chuẩn ảnh BMP, đạt kết quả như sau:



Hình 3-26 Kết quả chỉnh sửa màu ảnh bitmap

Thời gian đo đạt được cho việc xử lý filter của một bức ảnh rơi vào khoảng $0.034s/image$ cho một bức ảnh. Với thời gian xử lý và kết quả như trên là lý tưởng cho việc đáp ứng thời gian thực của ứng dụng.

3.2.3 Thực hiện OCR trong Python

Tiến hành trích xuất văn bản dùng kỹ thuật OCR trong Python thực hiện theo các bước sau:

Bước 1: Tiến hành thu thập dữ liệu ảnh thử nghiệm cho thuật toán OCR phân làm 3 cấp độ khác nhau:

- ### o Cơ bản (Simple):

[1] Computer Organization and Architecture 8th Edition, William Stallings,2011
 [2] Bài giảng kiến trúc máy tính – Viện CNTT-ĐH BKHN
 [3] Andrew S. Tanenbaum, Structured Computer Organization, 5rd Edition, Prentice- Hall International Edition, 2006.
 [4] Các nguồn tham khảo khác từ internet

Kiến trúc tập lệnh (Instruction Set Architecture): bao gồm: tập lệnh, biểu diễn dữ liệu, các cơ chế vào ra, kỹ thuật đánh địa chỉ
 • Tập lệnh: tập hợp các chuỗi số nhị phân mã hóa cho các thao tác mà máy tính có thể thực hiện
 • Các kiểu dữ liệu: các kiểu dữ liệu mà máy tính có thể xử lý

¹Machine Learning Department, Carnegie Mellon University
²Department of Computer Science, Princeton University
 agu@cs.cmu.edu, tri@tridao.me

simple_0

simple_1

simple_2

Every table is broken up into smaller entities called fields. The fields in the Customers table consist of CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country. A field is a column in a table that is designed to maintain specific information about every record in the table.

A record, also called a row, is each individual entry that exists in a table. For example, there are 91 records in the above Customers table. A record is a horizontal entity in a table.

simple_3

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

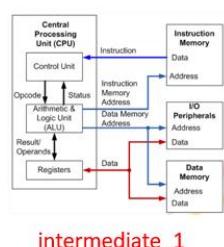
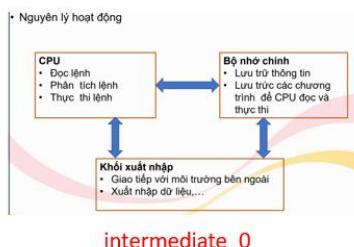
Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

In this tutorial, we will use semicolon at the end of each SQL statement.

simple_4

Hình 3-27 Tập dữ liệu cơ bản trong xử lý OCR Python

- Trung bình (Intermediate):



Loại lệnh	A	B	C
CPI theo loại lệnh	1	2	3
IC trong dãy lệnh 1	2	1	2
IC trong dãy lệnh 2	4	1	1

- Dãy lệnh 1: $IC = 2+1+2=5 \Rightarrow N1 = 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
- Dãy lệnh 2: $IC = 4+1+1=6 \Rightarrow N2 = 4 \times 1 + 1 \times 2 + 1 \times 3 = 9$

$$CPI_1 = \frac{10}{5} = 2$$

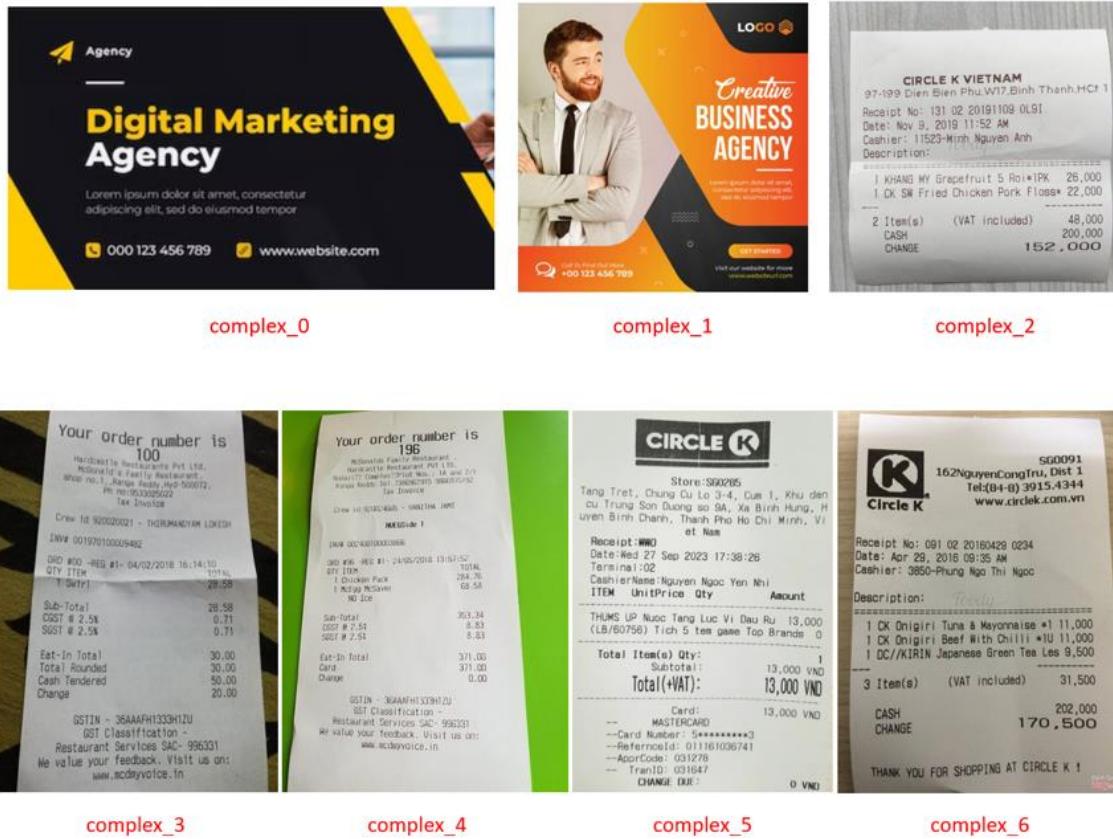
$$CPI_2 = \frac{9}{6} = 1.5$$

intermediate_2



Hình 3-28 Tập dữ liệu trung bình trong xử lý OCR Python

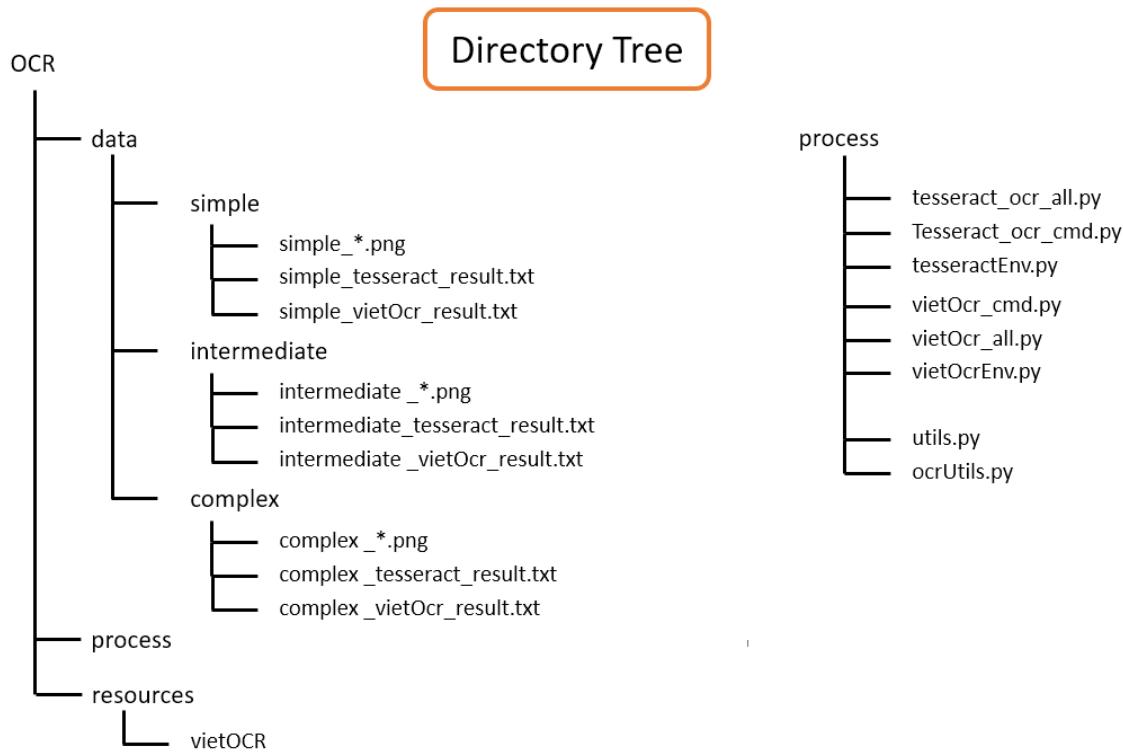
- Phức tạp (Complex):



Hình 3-29 Tập dữ liệu phức tạp trong xử lý OCR Python

Bước 2: Thực hiện lần lượt 2 quá trình tiếp cận khác nhau:

Trước hết tổ chức cây thư mục sau:



Hình 3-30 Tô chúc thư mục xử lý OCR Python

Trong đó:

- ‘data’: Là thư mục chứa dữ liệu phân làm 3 cấp độ {‘simple’, ‘intermediate’, ‘complex’}. Trong mỗi thư mục có chứa ảnh, kết quả xử lý của các ảnh bằng thư viện tesseract và VietOCR theo cú pháp sau:

$f'{}{{level}}_{{library}}_result.txt'$

- ‘process’: Chứa các code xử lý chính cho việc trích xuất thông tin từ ảnh. Mỗi thư viện sử dụng tiến hành viết ba file xử lý gồm:

$f'{}{{library}}_all.py'$

$f'{}{{library}}_cmd.py'$

$f'{}{{library}}_Env.py'$

Với $f'{}{{library}}_all.py'$, thực hiện chức năng xử lý toàn bộ ảnh trong các thư mục chứa dữ liệu; $f'{}{{library}}_cmd.py'$ thực hiện chức năng gọi và truyền tham số bằng command line, giúp việc kích hoạt môi trường và gọi chương trình qua C Sharp

được thuận tiện. Cuối cùng là $f'\{library\} Env.py$ thực hiện gọi quá trình xử lý ở bất cứ đâu trong máy tính, giúp việc gọi chương trình Python được linh hoạt hơn.

- ‘resources’: Chứa phần xử lý chính gồm model pretrained và trọng số của model sử dụng cho việc trích xuất văn bản từ ảnh.
 - Pytesseract: Thuật toán OCR trong Tesseract Python quy trình sau: [Link](#), [Install Pytesseract](#) ; [Tesseract Wraper](#).
 - Cơ bản (Simple):

[1] Computer Organization and Architecture 8th Edition, William Stallings, 2011

[2] Bài giảng kiến trúc máy tính – Viện CNTT-ĐH BK HN

[3] Andrew S. Tanenbaum, Structured Computer Organization, 5rd Edition, Prentice- Hall International Edition, 2006.

[4] Các nguồn tham khảo khác từ internet

simple_0

```
data > simple > simple_tesseract_result.txt
1  #1
2  simple_0.png
3  |
4  [1] Computer Organization and Architecture 8th Edition, William
5  Stallings, 2011
6
7  [2] Bài giảng kiến trúc máy tính - Viện CNTT-ĐH BK HN
8
9  [3] Andrew S. Tanenbaum, Structured Computer Organization, 5rd
10 Edition, Prentice- Hall International Edition, 2006.
11
12 [4] Các nguồn tham khảo khác từ internet
13
14 |
15 #1
```

simple_0_result

100%

Hình 3-31 Kết quả xử lý ảnh cơ bản Tesseract OCR Python lần 1

¹Machine Learning Department, Carnegie Mellon University
²Department of Computer Science, Princeton University
 agu@cs.cmu.edu, tri@tridao.me

simple_3

```

data > simple > simple_tesseract_result.txt
43  #4
44  simple_3.png
45  |
46  "Machine Learning Department, Carnegie Mellon University
47  "Department of Computer Science, Princeton University
48  agu@cs.cmu.edu, tri@tridao.me
49
50  |
51  #4
  
```

simple_3_result

98%

Hình 3-32 Kết quả xử lý ảnh cơ bản Tesseract OCR Python lần 2

Nhận xét: Thư viện Tesseract thực hiện trích xuất ảnh trong điều kiện lý tưởng, cấp độ cơ bản, thực hiện rất tốt và đạt được kết quả chính xác cao. Tuy nhiên thư viện khó có thể nhận dạng các chi tiết quá nhỏ, vậy cần thực hiện phóng to vùng ảnh trước khi đưa vào xử lý để tăng hiệu quả trích xuất thông tin.

- Trung bình (Intermediate):



intermediate_3

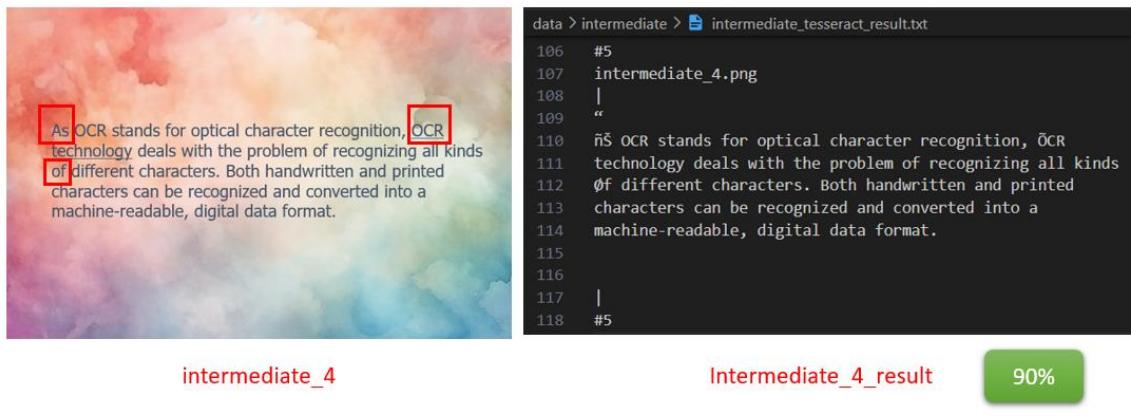
```

data > intermediate > intermediate_tesseract_result.txt
94  #4
95  intermediate_3.png
96  |
97  Giá trị cốt lõi
98
99  Đặt khách hàng làm trọng tâm, VinFast Sản phẩm đẳng cấp, giá
100  không ngừng sáng tạo để tạo ra các sản | tốt, hậu mãi vượt trội.
101  phẩm đẳng cấp và trải nghiệm xuất sắc
102  cho mọi người.
103
104  |
105  #4
  
```

intermediate_3_result

80%

Hình 3-33 Kết quả lý ảnh trung bình Tesseract OCR Python lần 1



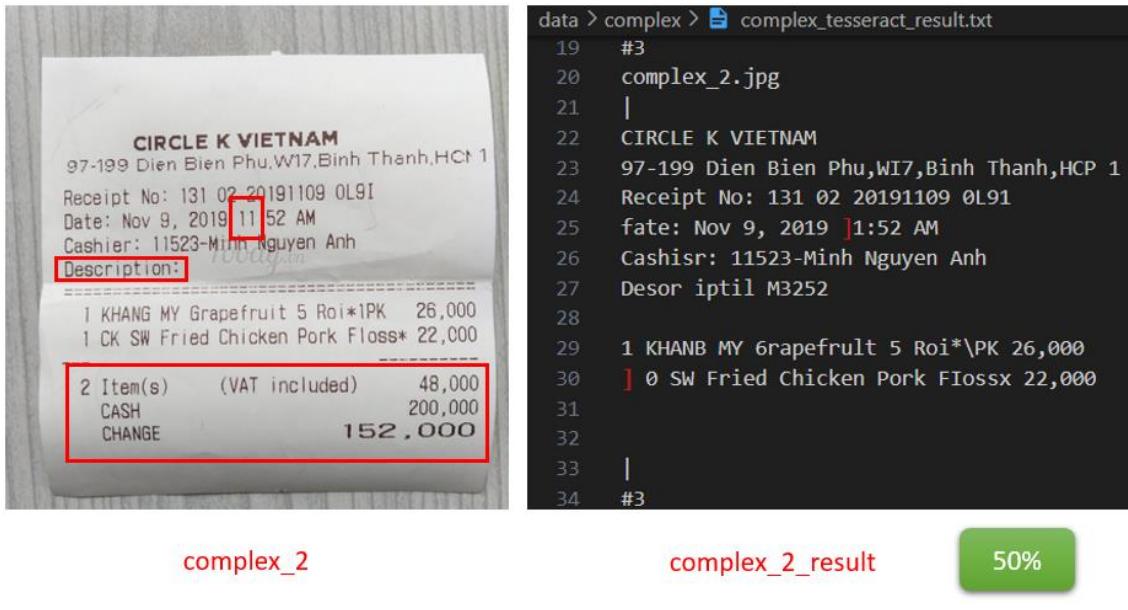
Hình 3-34 Kết quả xử lý ảnh trung bình Tesseract OCR Python lần 2

Nhận xét: Thư viện Tesseract thực hiện trích xuất ảnh trong điều kiện bị ảnh hưởng bởi ngữ cảnh như các hàng không liền kề nhau và không tương phản tốt, cấp độ trung bình. Thự hiện nhận dạng thiếu và sai ký tự.

- Phức tạp (Complex):



Hình 3-35 Kết quả xử lý ảnh phức tạp Tesseract OCR Python lần 1



Hình 3-36 Kết quả xử lý ảnh phức tạp Tesseract OCR Python lần 2

Nhận xét: Thư viện Tesseract thực hiện trích xuất ảnh trong điều kiện bị ảnh hưởng bởi ngữ cảnh, màu sắc không đồng đều, các dòng chữ không ngay ngắn, các kí tự nhỏ, cấp độ phức tạp. Thực hiện nhận dạng thiếu và sai ký tự xảy ra nhiều hơn.

- VietOCR: Thuật toán OCR được thực hiện với Model transformer của VietOCR quy trình sau:

- Cơ bản (Simple):

[1] Computer Organization and Architecture 8th Edition, William Stallings,2011
[2] Bài giảng kiến trúc máy tính – Viện CNTT-ĐH BKHN
[3] Andrew S. Tanenbaum, Structured Computer Organization, 5rd Edition, Prentice- Hall International Edition, 2006.
[4] Các nguồn tham khảo khác từ internet

simple_0

data > simple > simple_vietOcr_result.txt

```

1  #1
2  D:\Workspace\University\Report_DATN\Report_processes\OCR\data\simple\simple_0.png
3  |
4  036000000095
5  |
6  #1

```

simple_0_result

0%

Hình 3-37 Kết quả xử lý ảnh đơn giản VietOCR Python lần 1

¹Machine Learning Department, Carnegie Mellon University
²Department of Computer Science, Princeton University
 agu@cs.cmu.edu, tri@tridao.me

simple_3

```
data > simple > simple_vietOcr_result.txt
19  #4
20  D:\Workspace\University\Report_DATN\Report_processes\OCR\data\simple\simple_3.png
21  |
22  "Mightine Learing The ""Communice Comple Assition Stationary"""
23  |
24  #4
```

simple_3_result

0%

Hình 3-38 Kết quả xử lý ảnh đơn giản VietOCR Python lần 2

Nhận xét: Thư viện VietOCR thực hiện trích xuất ảnh trong điều kiện lý tưởng, cấp độ cơ bản, không thể trích xuất văn bản từ hình ảnh. Vậy thư viện VietOCR không thể xử lý ảnh với thông tin dạng nhiều dòng.

- Trung bình (Intermediate):

intermediate_3

```
data > intermediate > intermediate_vietOcr_result.txt
19  #4
20  D:\Workspace\University\Report_DATN\Report_processes\OCR\data\intermediate\intermediate_3.png
21  |
22  0
23  |
24  #4
```

intermediate_3_result

0%

Hình 3-39 Kết quả xử lý ảnh trung bình VietOCR lần 1

Nhận xét: Thư viện VietOCR thực hiện trích xuất ảnh trong điều kiện trung bình không thể trích xuất văn bản từ hình ảnh.

- Phức tạp (Complex):

complex_0

```
data > complex > complex_vietOcr_result.txt
1  #1
2  D:\Workspace\University\Report_DATN\Report_processes\OCR\data\complex\complex_0.png
3  |
4  Expection
5  |
6  #1
```

Complex_0_result 0%

Hình 3-40 Kết quả xử lý ảnh phức tạp VietOCR Python lần 1

complex_2

```
data > complex > complex_vietOcr_result.txt
13  #3
14  D:\Workspace\University\Report_DATN\Report_processes\OCR\data\complex\complex_2.jpg
15  |
16  03800000099
17  |
18  #3
19  #4
```

complex_2_result 0%

Hình 3-41 Kết quả xử lý ảnh phức tạp VietOCR Python lần 2

Nhận xét: Thư viện VietOCR thực hiện trích xuất ảnh trong điều kiện bị ảnh hưởng bởi ngữ cảnh, màu sắc không đồng đều, các dòng chữ không ngay ngắn, các kí tự nhỏ, cấp độ phức tạp. Không thể trích xuất văn bản từ hình ảnh.

Bước 3: Thực hiện so sánh kết quả, đánh giá và nhận xét hai thuật toán xử lý trên

Đánh giá:

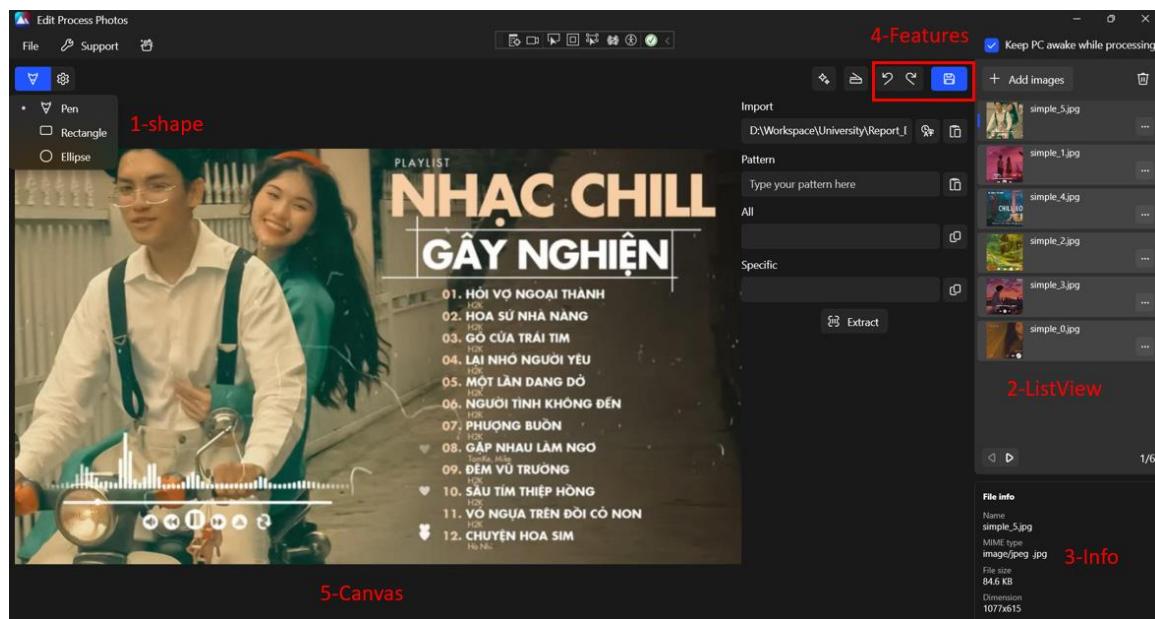
- Tesseract: Là thư viện thực hiện tốt với xử lý đơn ảnh, trong điều kiện lý tưởng. Tuy nhiên với các điều kiện trung bình và phức tạp thì thư viện thực hiện trích xuất thông tin có sai và thiếu các kí tự. Đặc biệt là văn bản bị có màu kém tương phản so với nền và kí tự không theo các hàng dạng thẳng và liền mạch. Vậy thư viện Tesseract thực hiện tốt trong điều kiện lý tưởng, có thể xử lý toàn bộ bức ảnh và từng vùng ảnh khác nhau.
- VietOCR: Thực hiện không thể thực hiện trích xuất văn bản trong tình huống ảnh gồm nhiều dòng kí tự. Vì vậy VietOCR chỉ thích hợp với việc trích xuất các dòng kí tự riêng lẻ. Thuật toán này sẽ phát huy hiệu quả khi xử lý với việc phân đoạn vùng chứa văn bản sau đó tiến hành trích xuất thông tin từ các vùng đó. Vì vậy cần tiền xử lý trước khi sử dụng thuật toán này.

Bước 4: Thực hiện viết chương trình Python cho thuật toán, sử dụng cho việc gọi chương trình Python thông qua Terminal trong C Sharp.

3.2.4 Thực hiện Inpaint trong C Sharp

Qua trình thực hiện Inpaint Image trong C# tiến hành các bước sau:

Bước 1: Thực hiện thiết kế giao diện người dùng sử dụng bao gồm: Danh sách ảnh sử dụng List View, từng ảnh đơn lẻ được hiển thị bằng Canvas, Thiết kế các bảng chọn mặt nạ thông dụng đường thẳng có độ dày, hình chữ nhật, hình elip.

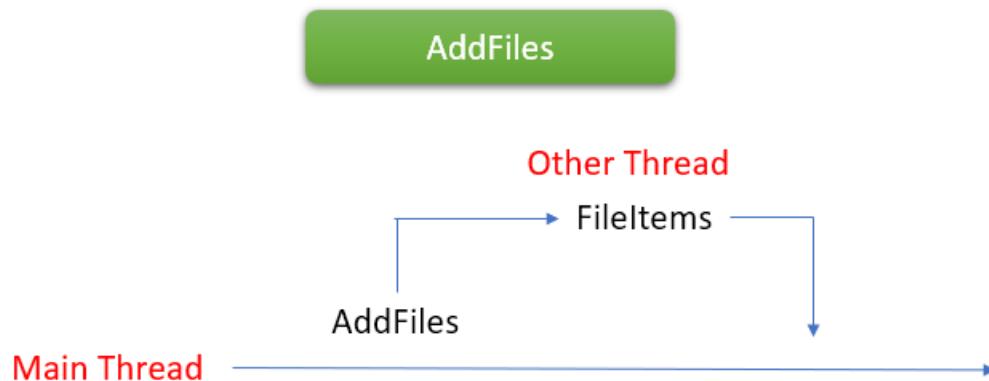


Hình 3-42 Kết quả xây dựng Inpaint trong ứng dụng Windows

Quá trình Inpaint trong C Sharp với hình thức tiếp cận là tạo mask bằng các hình dạng khác nhau, trong đó:

1. Shape: Một bảng chọn dạng mask sẽ được tạo là dạng RECTANGLE, ELIP, POLYGON. Trong đó POLYGON, là chuỗi các điểm nối lại với nhau.
2. List View: Vùng hiển thị danh sách các ảnh được lưu trữ trong một phiên xử lý.
3. Info: Hiện các thông tin chi tiết về ảnh như: tên file, kích thước file, và độ phân giải của file.
4. Features:
 - a. Undo: Khôi phục quá trình Inpaint vừa thực hiện.
 - b. Redo: Thực hiện lại quá trình Inpaint.
 - c. Save: Thực hiện lưu ảnh với kích thước nhất định sau khi xử lý.

Bước 2: Xây dựng quá trình thêm ảnh vào ứng dụng kết hợp với xử lý đa luồng – Multi Threads. Giúp việc thêm ảnh vào ứng dụng không bị khóa giao diện người dùng, tránh được tình trạng ứng dụng không thể tương tác trong khoảng thời gian nhất định



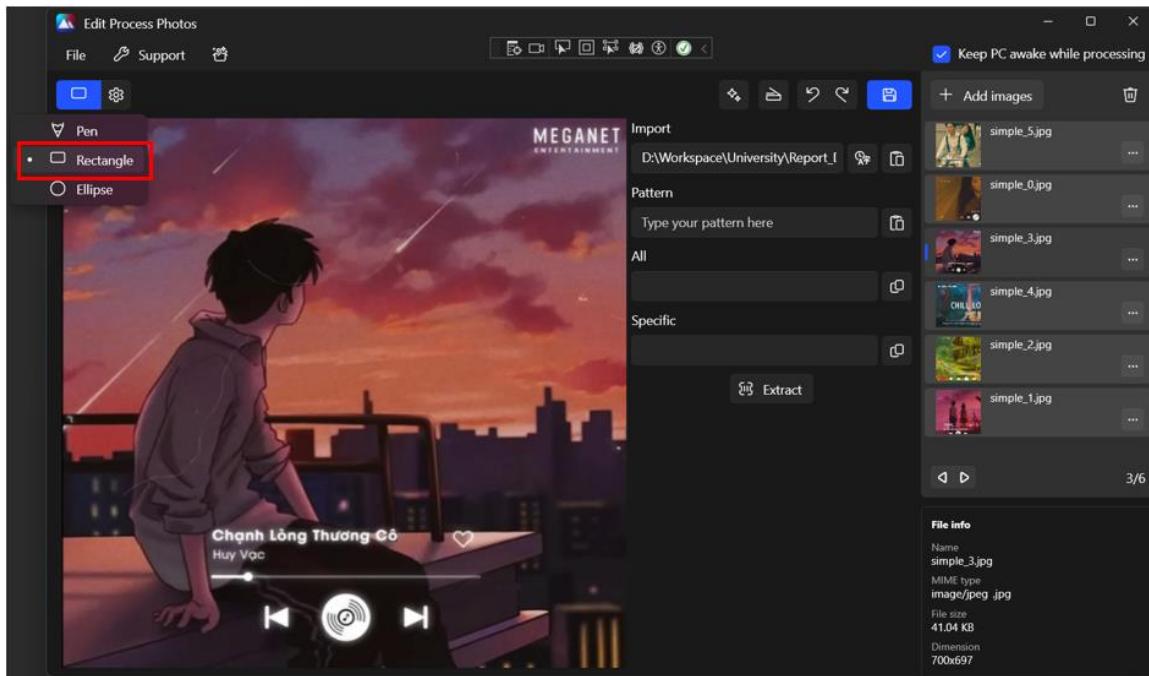
Hình 3-43 Minh họat xử lý đa luồng (multi threads) trong ứng dụng Windows

Bước 3: Thực hiện phần xử lý lỗi:

- Thực hiện nhận mặt nạ xử lý Inpaint vào trước bằng thao tác trên màn hình, sau đó chuyển đổi và lưu trữ dưới dạng bitmap Image mà OpenCV Sharp hỗ trợ riêng bằng cách lấy toạ độ của các điểm được vẽ trên Canvas.
- Thực hiện Inpaint Image ngay sau khi vẽ xong và tiến hành lưu trữ các ảnh được xử lý dưới dạng một danh sách các ảnh thuận tiện cho việc thực hiện các thao tác như quay lại (Undo), tiến tới (Redo) và lưu ảnh với kích thước khác nhau về sau:
- Xây dựng tính năng Undo, Redo: Thực hiện các thao tác quay lại, thuận tiện hơn cho việc chỉnh sửa ảnh, giúp các thao tác có thể tùy chỉnh dễ dàng.
- Thực hiện xây dựng phần lưu trữ ảnh xuống dưới dạng file khác, lúc này sẽ thuận tiện hơn cho việc lưu trữ ảnh sau khi xử lý.

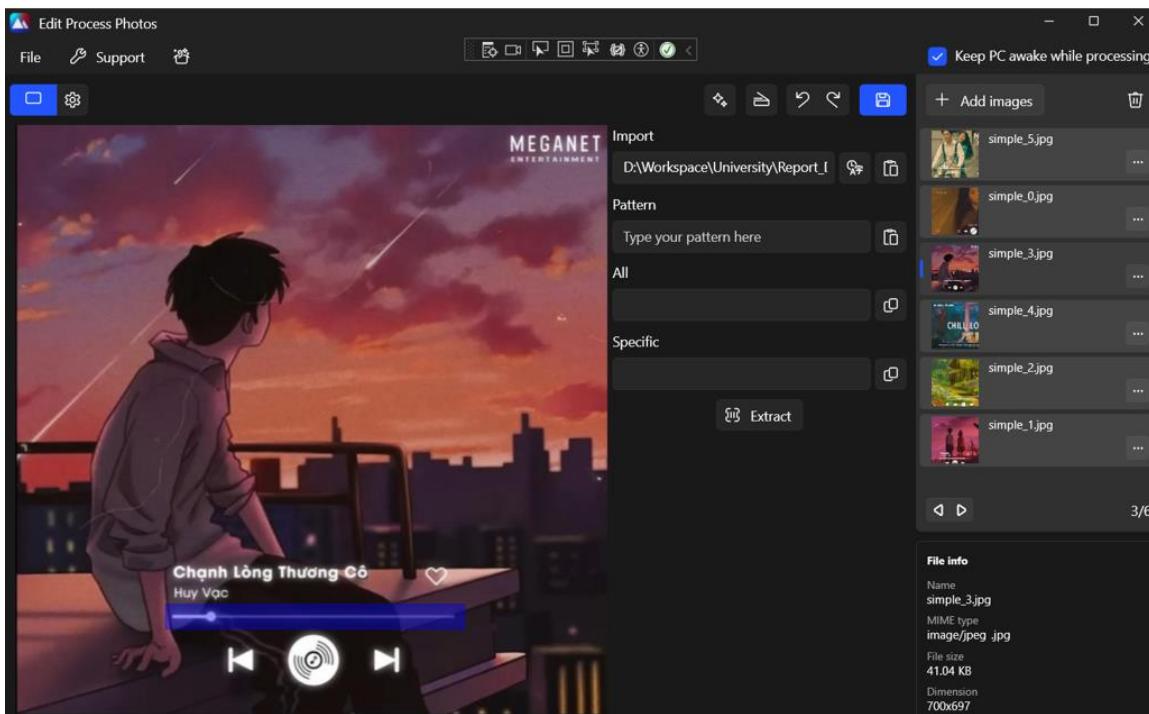
Bước 4: Tiến hành đánh giá các bước xử lý và cải thiện thực hiện Inpaint ảnh.

Thực hiện chọn danh sách ảnh đưa vào ứng dụng, sau đó tiến hành Inpaint theo quy trình sau:



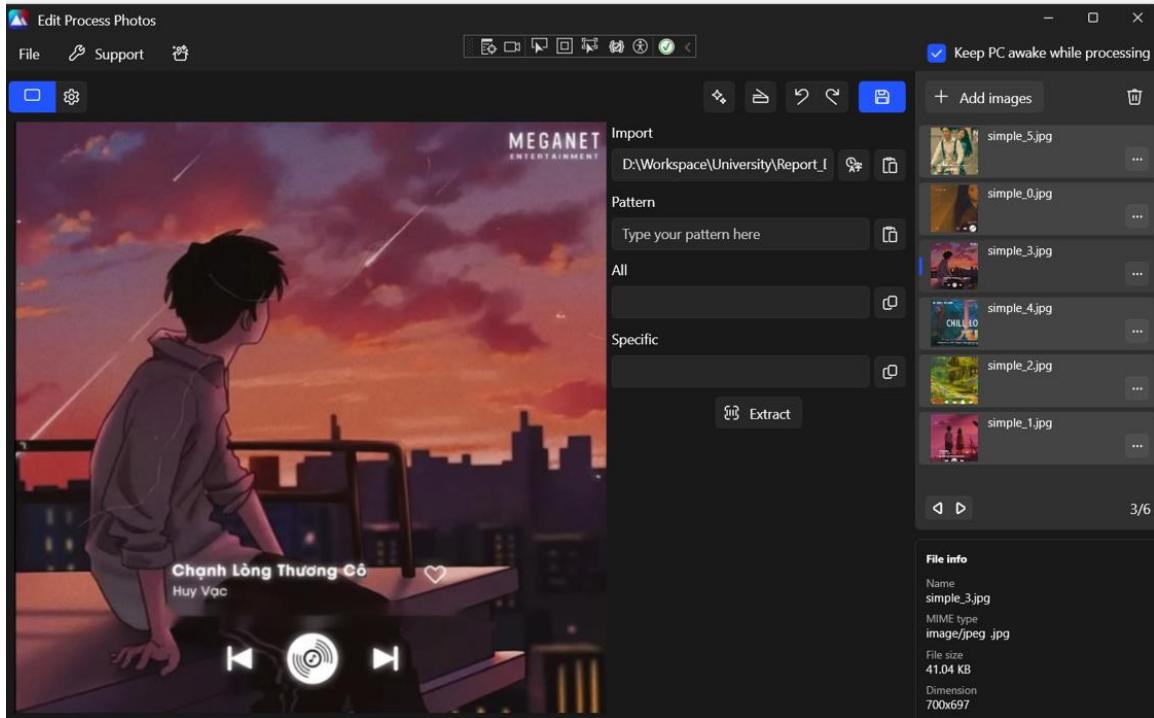
Hình 3-44 Hướng dẫn chọn dạng hình cho việc tạo Mask

Thực hiện Click vào nút Shape Menu trên màn hình, ứng dụng sẽ hiện ra bảng chọn. Chọn Rectangle để sử dụng tạo mask là hình chữ nhật.



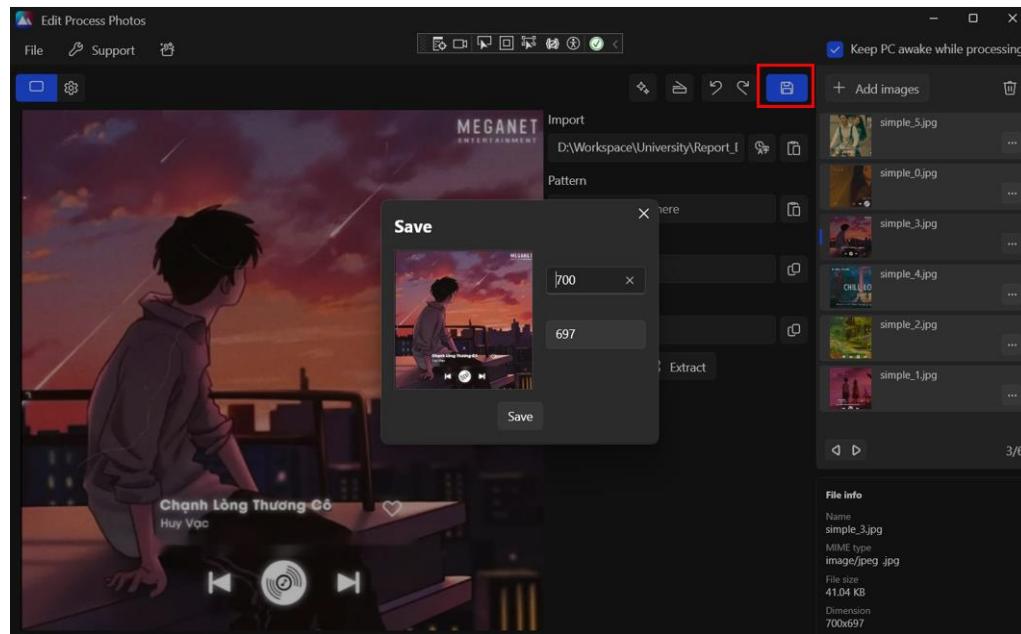
Hình 3-45 Minh họa việc tạo Mask trong ứng dụng Windows

Sau đó thực hiện chọn vùng muốn xử lý Inpaint trong ảnh, bằng cách clickdown vào góc trái trên của hình chữ nhật, và di chuyển tới góc phải dưới và click up. Quá trình này đã tạo được mask, Quá trình Inpaint sẽ được diễn ra ngay lúc đó.



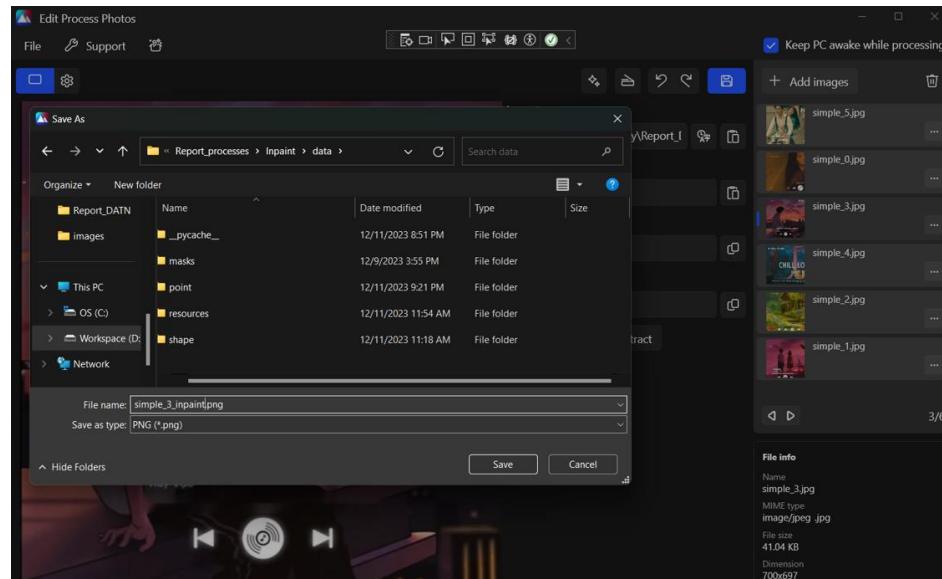
Hình 3-46 Kết quả xử lý Inpaint trong ứng dụng Windows

Và ảnh trên là kết quả sau khi xử lý, cho ra kết quả tốt với chi tiết đơn giản.



Hình 3-47 Lưu ảnh sau khi xử lý Inpaint trong ứng dụng Windows

Và có thể tiếp tục thực hiện Inpaint đến khi nào quá trình xử ảnh trên kết thúc. Tiến hành lưu sau xử lý, thực hiện Click vào Save Button. Một Dialog sẽ hiện ra cho việc điều chỉnh kích thước của hình ảnh, chức năng này có thể đảm bảo được tỷ lệ của ảnh. Sau đó click vào Save Button trên Dialog để thực hiện chọn nơi lưu ảnh.



Hình 3-48 Minh họa đặt tên và chọn nơi lưu sau khi xử lý Inpaint Windows

Thực hiện chọn nơi lưu ảnh và đặt tên file muốn lưu. Sau quá trình trên việc xử lý Inpaint, hoàn thành và app sẽ tự động mở thư mục người dùng lưu ảnh để tiện cho quá trình kiểm tra như trên.

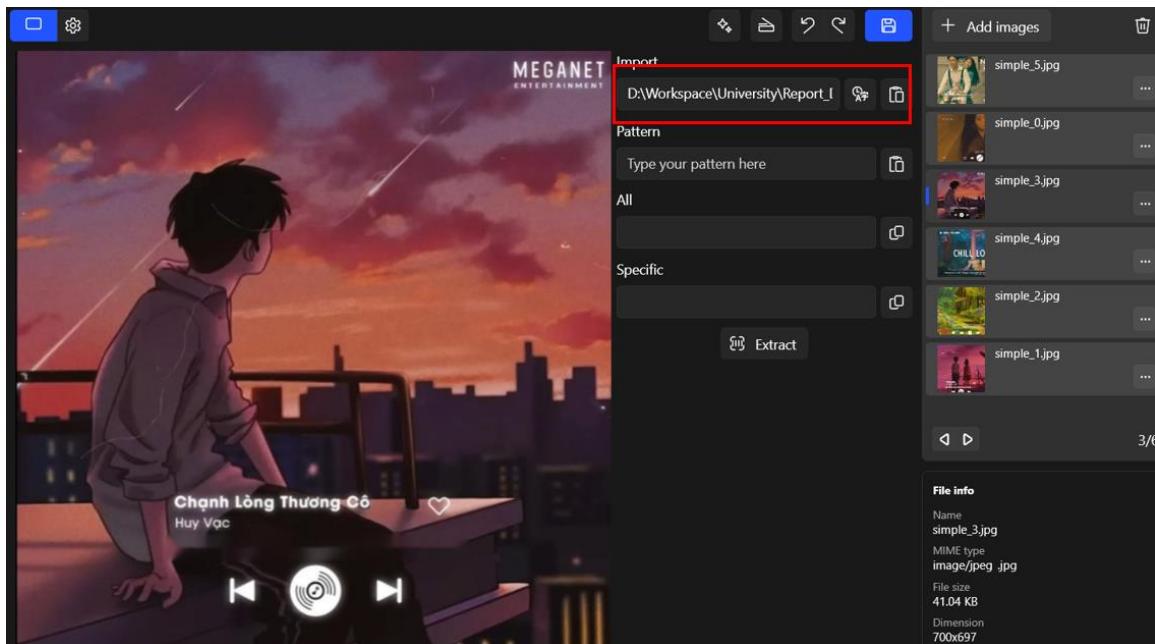
Nhận xét:

Thuật toán Inpaint được xử lý trong C Sharp dùng OpenCV là rất tốt, đạt được những yêu cầu ban đầu đề ra. Ứng dụng được độ chính xác và thời gian đáp ứng nhanh cho người dùng.

3.2.5 Thực hiện Filter trong C Sharp

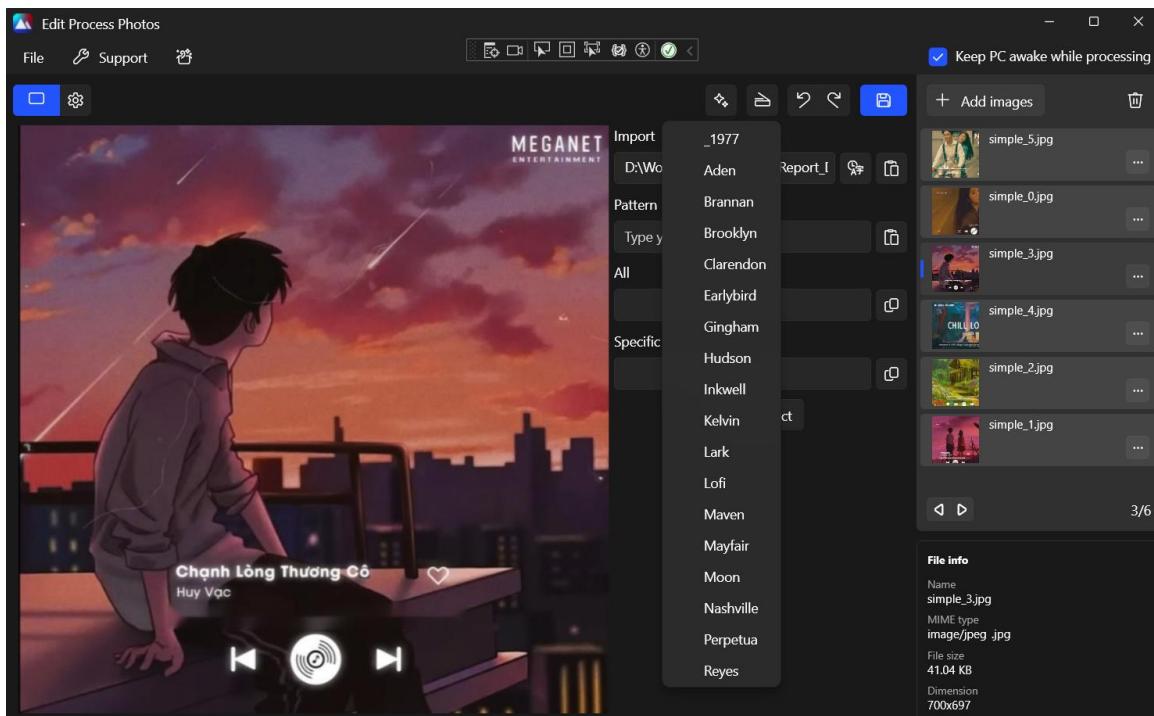
Tiến hành sử dụng thư viện Pilgram trong Python để thực hiện thay đổi màu ảnh, sau đó chúng ta thực hiện gọi các lệnh này qua Command trong C#. Thực hiện các bước tiến hành sau:

Bước 1: Thực hiện lấy dữ liệu ảnh bằng đường dẫn ảnh được thêm vào ứng dụng.



Hình 3-49 Lấy đường dẫn tới ảnh xử lý chỉnh màu trong Windows

Bước 2: Thực hiện xây dựng bảng chọn với các dạng filter chỉnh sửa màu khác nhau của ảnh.

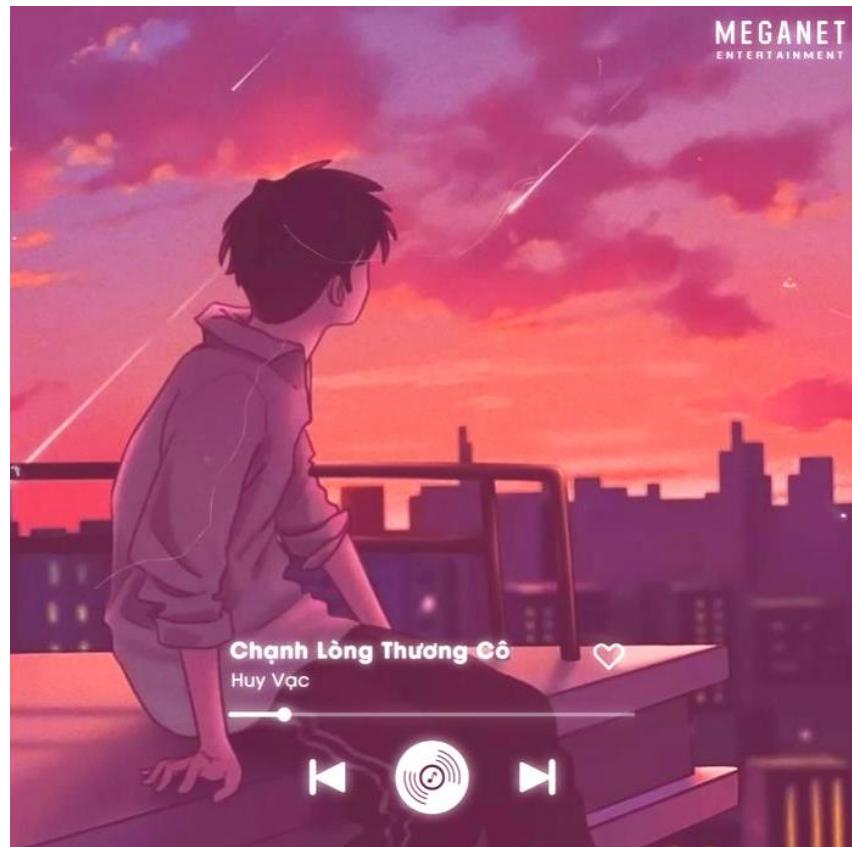


Hình 3-50 Danh sách các Filter trong ứng dụng Windows

Bước 3: Sau khi nhận được filter được chọn, tiến hành gọi lệnh thực thi qua code Python trong dự án để tiến hành thực hiện bước điều chỉnh màu cho ảnh.

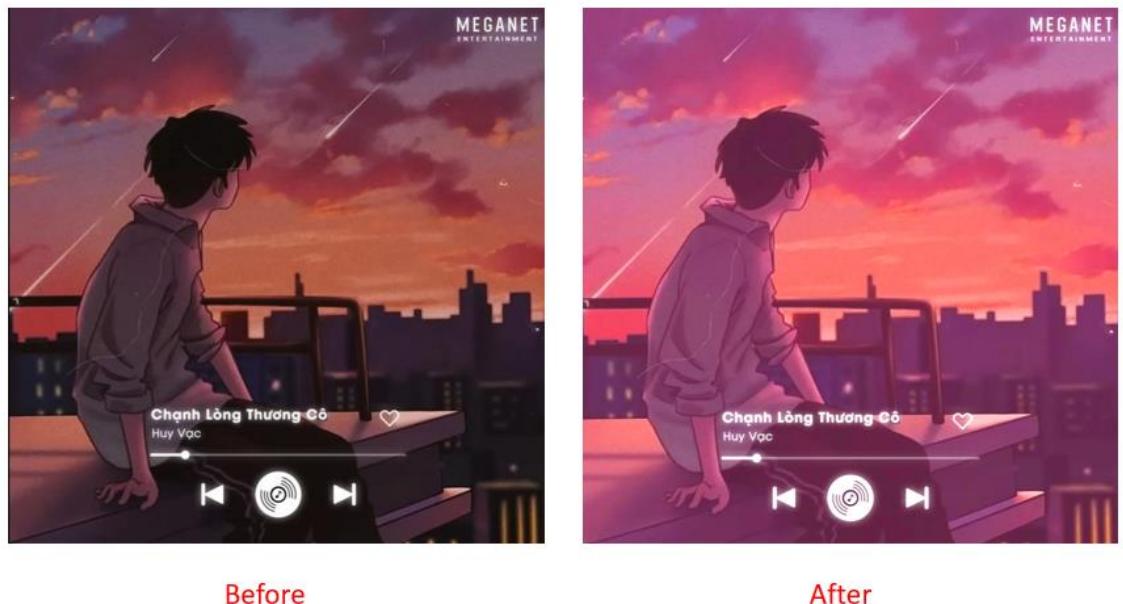
Bước 4: Tiến hành lưu ảnh dưới dạng tạm, lưu các thông tin của ảnh gốc.

Bước 5: Sau đó tiến hành load lại ảnh vào trong ứng dụng cho người dùng xem.



Hình 3-51 Kết quả sau khi xử lý chỉnh màu trong Windows

Bước 6: Tiến hành đánh giá kết quả xử lí của ảnh.



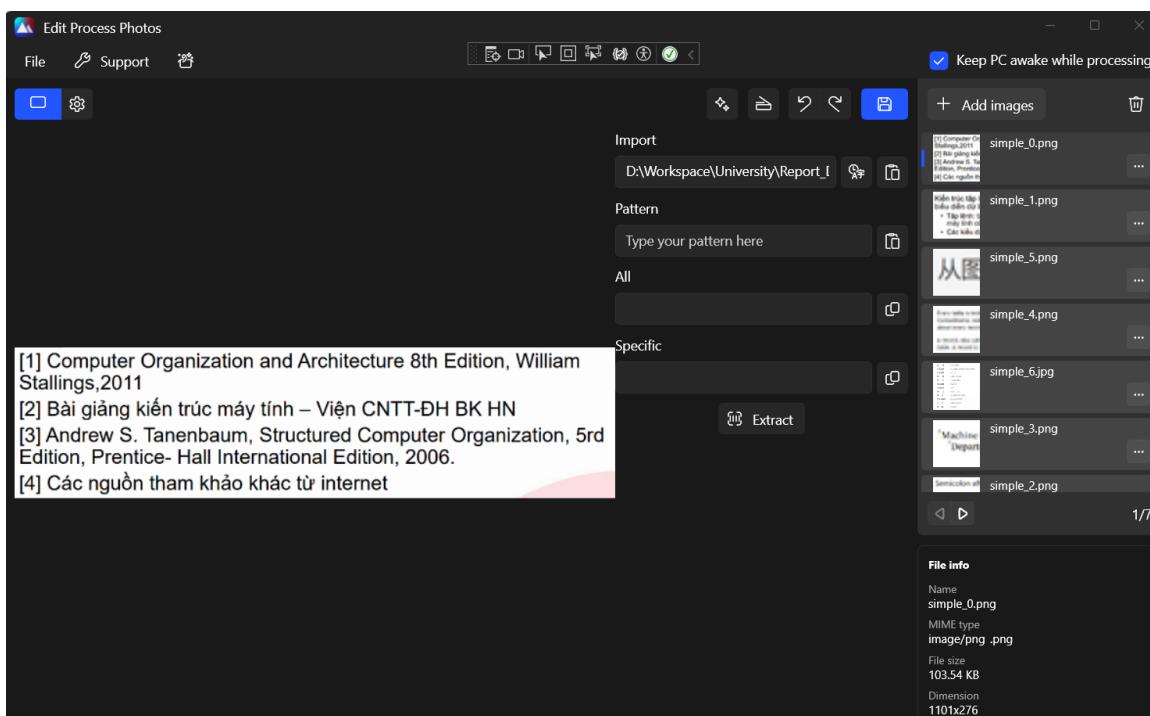
Hình 3-52 So sánh kết quả chỉnh màu ảnh trong Windows

Nhận xét: Vậy việc thực hiện lọc màu cho ảnh trong ứng dụng đạt được kết quả như mong đợi, thời gian xử lý nhanh và đa dạng các bộ lọc màu khác nhau.

3.2.6 Thực hiện OCR trong C Sharp

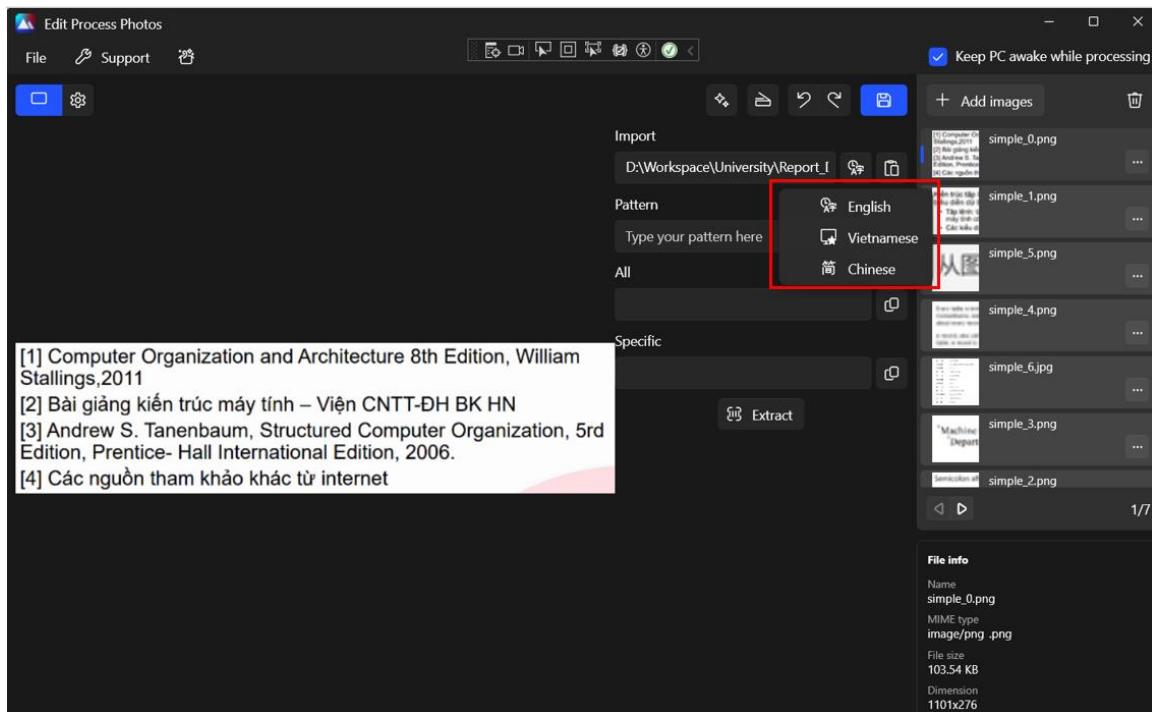
Tiến hành trích xuất văn bản dùng kỹ thuật OCR trong C Sharp qua thư viện IronOCR, có xử lý lỗi dựa trên Tesseract 5.0, thực hiện theo các bước sau:

Bước 1: Thực hiện lấy dữ liệu ảnh bằng đường dẫn ảnh được thêm vào ứng dụng.



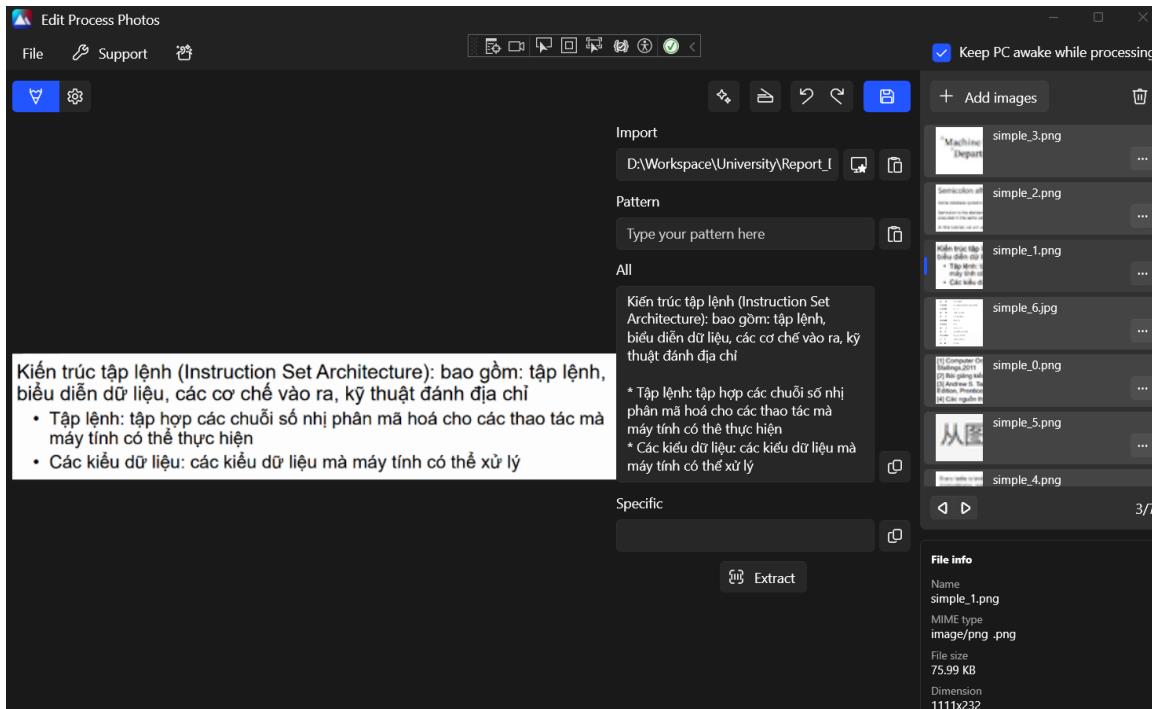
Hình 3-53 Thêm ảnh trích xuất thông tin vào ứng dụng Windows

Bước 2: Thực hiện lựa chọn ngôn ngữ tiếng Việt để thử nghiệm, người dùng có thể lựa chọn tiếng Anh và tiếng Trung.



Hình 3-54 Minh họa chọn ngôn ngữ OCR trong Windows

Bước 3: Thủ nghiệm trích xuất văn bản từ ảnh.



Hình 3-55 Kết quả trích xuất thông tin trong ứng dụng Windows

Mặc định ứng dụng sẽ trích xuất toàn bộ chữ từ ảnh và lưu vào TextBox ‘All’. Còn nếu người dùng sử dụng Pattern Regex thì có thể trích xuất đúng dạng chuỗi kí tự mà mình muốn từ thông tin trích xuất từ ảnh.

Bước 4: Đánh giá kết quả của ảnh trước và sau khi xử lý.

Nhận xét: Việc trích xuất thông tin từ ảnh trong ứng dụng hoạt động tốt trong điều kiện lý tưởng, vẫn đáp ứng được nhu cầu cơ bản của người dùng. Tuy nhiên với các trường hợp phức tạp hợp việc trích xuất thông tin bị biết hoặc bị sai giá trị.

3.3 KẾT LUẬN CHƯƠNG

Sau quá trình thực hiện nghiên cứu và triển khai các hình thức tiếp cận khác nhau với các kĩ thuật xử lý của ứng dụng như: Inpaint, Filter, OCR. Đạt được các kết quả như sau:

Thuật toán Inpaint xử lý trong Python: OpenCV, Inpaint Anything thì thực hiện xóa vật thể quá tốt trong tình huống đơn giản. Thuật toán dùng OpenCV phù hợp hơn trong việc xử lý các chi tiết nhỏ, có độ tương phản tốt với nền và nền có cấu trúc đồng nhất. Còn thuật toán Inpaint Anything thì chưa phát huy tốt trong trường hợp các chi tiết đối tượng là nhỏ và thiết liên kết với nhau. Để đáp ứng nhu cầu của đề tài tiến hành sử dụng Inpaint OpenCV trong ứng dụng Windows.

Inpaint OpenCV trong ứng dụng Windows đã đáp ứng được tính đơn giản, dễ dùng, thời gian đáp ứng nhanh như nhu cầu đã đề ra.

Filter Python đáp ứng được các yêu cầu đề ra. Thời gian đo đạt được cho việc xử lý filter của một bức ảnh rơi vào khoảng $0.034s/image$ cho một bức ảnh.

Filter trong ứng dụng Windows có thể đáp ứng được nhu cầu tương tự, nhưng việc xử lý File còn nhiều vấn đề xoay quanh như: Việc xử lý lưu file tạm, xử lý các threads khác nhau giúp không bị khóa UI, và thực hiện tải lại ảnh vào ứng dụng.

Thuật toán OCR được triển khai trong Python và C Sharp để đáp ứng được nhu cầu đề ra. Việc xử lý OCR trong ứng dụng Windows còn gặp khó khăn cho việc xử lý các ngữ cảnh phức tạp.

CHƯƠNG 4. KẾT LUẬN

4.1 TÓM TẮT VÀ KẾT LUẬN CHUNG

Đề tài xây dựng ứng dụng chỉnh sửa và trích xuất thông tin từ ảnh trên Windows đã đáp ứng được các câu hỏi đề ra. Nghiên cứu và tìm hiểu OpenCV trong việc thực hiện các thao tác xử lý ảnh, Inpaint. Thiết kế giao diện dùng WinUI 3 Framework, tổ chức thuận tiện cho việc phát triển và bảo trì xây dựng trên nền tảng Windows. Thực hiện nghiên cứu các thành phần hỗ trợ xử lý ảnh OpenCV và Pilgram, filter. Nghiên cứu trích xuất thông tin thông qua OCR. Và ứng dụng cũng đáp ứng được các nhu cầu đề ra của đề tài: Độ chính xác, thời gian đáp ứng nhanh, đơn giản cho việc sử dụng.

Tuy nhiên đề tài chưa thể xử lý tốt trong các trường hợp tương đối phức tạp. Thuật toán Inpaint chưa thể xóa các vật thể có chi tiết nhiều và nền không có cấu trúc đồng nhất. Chỉnh sửa ảnh còn chưa đa dạng thao tác xử lý như: thay đổi độ tương phản, độ sáng, ... Thực hiện trích xuất thông tin ảnh còn chưa thực hiện tốt trong tình huống các văn bản có độ tương phản kém và bị lồng vào ngữ cảnh nhiều.

4.2 HƯỚNG PHÁT TRIỂN

Đề tài xây dựng ứng dụng chỉnh sửa và trích xuất thông tin từ ảnh trên Windows. Đã đáp ứng những câu hỏi nghiên cứu đề ra của đề tài. Tuy nhiên cần có sự cải tiến để đáp ứng độ chính xác cao hơn xong các bước xử lý.

Xử lý Inpaint có thể sử dụng thuật toán Inpaint Anything để có thể xóa các vật thể phức tạp hơn trong nhiều ngữ cảnh đa dạng hơn. Việc xử lý này giúp tăng tính đáp ứng của ứng dụng.

Xử lý chỉnh sửa ảnh có thể bổ sung cho việc xử lý Filter bằng các hoạt động xử lý thông dụng như: Điều chỉnh độ sáng, cắt ảnh, xoay ảnh.

Trích xuất thông tin dùng thư viện Tesseract có thể đạt độ chính xác tốt hơn nếu thực hiện thêm tính năng khoanh vùng các ô chữ nhật cụ thể để có thể trích xuất thông tin chính xác hơn.

PHỤ LỤC A

(Ghi chú: Code chương trình có thể để vào phụ lục A và các chứng minh toán học hỗ trợ có thể để vào phụ lục B)

A.1 Code chương trình Filter Python

```
import os
import sys
import PIL.Image
import pilgram

from enum import Enum
from utils.utils import Utils

class FilterType(Enum):
    _1977 = 1
    aden = 2
    brannan = 3
    brooklyn = 4
    clarendon = 5
    earlybird = 6
    gingham = 7
    hudson = 8
    inkwell = 9
    kelvin = 10
    lark = 11
    lofi = 12
    maven = 13
    mayfair = 14
    moon = 15
    nashville = 16
    perpetua = 17
    reyes = 18
    rise = 19
    slumber = 20
    stinson = 21
    toaster = 22
    valencia = 23
    walden = 24
```

```
willow = 25
xpro2 = 26

def FilterImage(filterType: FilterType, src, dest):
    img = PIL.Image.open(src)

    if filterType == FilterType._1977:
        pilgram._1977(img).save(dest)
    elif filterType == FilterType.aden:
        pilgram.aden(img).save(dest)
    elif filterType == FilterType.brannan:
        pilgram.brannan(img).save(dest)
    elif filterType == FilterType.brooklyn:
        pilgram.brooklyn(img).save(dest)
    elif filterType == FilterType.clarendon:
        pilgram.clarendon(img).save(dest)
    elif filterType == FilterType.earlybird:
        pilgram.earlybird(img).save(dest)
    elif filterType == FilterType.gingham:
        pilgram.gingham(img).save(dest)
    elif filterType == FilterType.hudson:
        pilgram.hudson(img).save(dest)
    elif filterType == FilterType.inkwell:
        pilgram.inkwell(img).save(dest)
    elif filterType == FilterType.kelvin:
        pilgram.kelvin(img).save(dest)
    elif filterType == FilterType.lark:
        pilgram.lark(img).save(dest)
    elif filterType == FilterType.lofi:
        pilgram.lofi(img).save(dest)
    elif filterType == FilterType.maven:
        pilgram.maven(img).save(dest)
    elif filterType == FilterType.mayfair:
        pilgram.mayfair(img).save(dest)
    elif filterType == FilterType.moon:
        pilgram.moon(img).save(dest)
    elif filterType == FilterType.nashville:
        pilgram.nashville(img).save(dest)
    elif filterType == FilterType.perpetua:
        pilgram.perpetua(img).save(dest)
    elif filterType == FilterType.reyes:
        pilgram.reyes(img).save(dest)
    elif filterType == FilterType.rise:
        pilgram.rise(img).save(dest)
    elif filterType == FilterType.slumber:
```

```

        pilgram.slumber(img).save(dest)
    elif filterType == FilterType.stinson:
        pilgram.stinson(img).save(dest)
    elif filterType == FilterType.toaster:
        pilgram.toaster(img).save(dest)
    elif filterType == FilterType.valencia:
        pilgram.valencia(img).save(dest)
    elif filterType == FilterType.walden:
        pilgram.walden(img).save(dest)
    elif filterType == FilterType.willow:
        pilgram.willow(img).save(dest)
    elif filterType == FilterType.xpro2:
        pilgram.xpro2(img).save(dest)

def main():
    for filter in FilterType:
        FilterImage(filterType= filter, src= r'assets\mountain.jpg', dest=
r'outputs\o' + '_' + filter.name + r'.jpg')

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("Interrupted")
        try:
            sys.exit(0)
        except SystemExit:
            os._exit(0)

```

A.2 Code chương trình xử lý C Sharp

Thiết kế Filter:

```

public enum FilterType
{
    _1977,
    Aden,
    Brannan,
    Brooklyn,
    Clarendon,

```

```
    Earlybird,
    Gingham,
    Hudson,
    Inkwell,
    Kelvin,
    Lark,
    Lofi,
    Maven,
    Mayfair,
    Moon,
    Nashville,
    Perpetua,
    Reyes,
    Rise,
    Slumber,
    Stinson,
    Toaster,
    Valencia,
    Walden,
    Willow,
    Xpro2,
}

public Dictionary<FilterType, RadioMenuFlyoutItem> FILTERS;

public enum StatusType
{
    Ready,
    Loading,
    Loaded,
    LoadFailed,
    Processing,
    Processed,
    ProcessFailed,
}
```

Xử lý Input File ảnh vào trong ứng dụng:

```
public void AddFiles(IReadOnlyList<string> paths, Action startAction =
null, Action<StatusType> endAction = null, Action<List<ThumbnailItem>>
itemAction = null)
{
    if (Utils.Any(_status, StatusType.Loading, StatusType.Processing))
return;
```

```
    Status = StatusType.Loading;
    startAction?.Invoke();

    var hasCorrupted = false;
    var hasEpsAndCannotRead = false;

    List<ThumbnailItem> items = new();
    var coreCount = Environment.ProcessorCount;
    object locked = new();

    var start = FileItems.Count == 0;

    IProgress<StatusType> progress = new Progress<StatusType>(status =>
    {
        if (status == StatusType.Loaded)
        {
            if (hasEpsAndCannotRead)
                MainWindow.Inst.ShowMissingLibDialog();
            else if (hasCorrupted)
                MainWindow.Inst.ShowMessageTeachingTip(null, string.Empty,
                _resourceLoader.GetString("LoadCorruptedSomeFiles"));
        }

        Status = status;
        endAction?.Invoke(status);

        if (start && FileItems.Count > 0)
            FileListView.SelectedIndex = 0;
    });

    IProgress<List<ThumbnailItem>> itemProgress = new
    Progress<List<ThumbnailItem>>(items =>
    {
        lock (locked)
        {
            FileItems.AddRange(items);
            itemAction?.Invoke(items);
        }
    });

    _ = Task.Run(() =>
    {
        try
        {
            var packages = paths.Chunk(256);
```

```
foreach (var package in packages)
{
    var pathChunksPerPackage = package.Chunk(coreCount);

    lock (locked)
    {
        items.Clear();
    }

    foreach (var pathChunks in pathChunksPerPackage)
    {
        Parallel.ForEach(pathChunks, path =>
        {
            try
            {
                var isFilePath = Utils.IsFilePath(path);
                if (isFilePath.GetValueOrDefault(false) &&
!FileItems.Any(i => i.InputFilePath == path))
                {
                    var imageMeta =
ImageMagickUtils.GetMagickImageMeta(path);

                    if
(Profile.IsAcceptedInputFormat(imageMeta?.Format))
                    {
                        lock (locked)
                        {
                            items.Add(new() { InputInfo =
new(path) });
                        }
                    }
                    else throw new();
                }
            }
            catch (MissingLibException mle)
            {
                if (mle.Message == "eps")
                    hasEpsAndCannotRead = true;
                else
                    hasCorrupted = true;
            }
            catch (Exception)
            {
                hasCorrupted = true;
            }
        });
    }
}
```

```
        }
    });

    itemProgress.Report(items);
}

progress.Report(StatusType.Loaded);
}
catch (Exception)
{
    progress.Report(StatusType.LoadFailed);
}
});
}
```

Chương trình xử lý phần trích xuất văn bản ra khỏi ảnh:

```
private void OCRButton_Click(object sender, RoutedEventArgs e)
{
    if (sender is not Control control) return;
    if (InputPath.Text == "") return;

    string pattern = InputPattern.Text;
    string inputPath = InputPath.Text;

    IronTesseract IronOcr = new();
    IronOcr.Language = (LangType)LanguageButton.Tag switch
    {
        LangType.English     => OcrLanguage.English,
        LangType.Vietnamese => OcrLanguage.Vietnamese,
        LangType.Chinese     => OcrLanguage.ChineseSimplifiedFast,
        _                     => OcrLanguage.English,
    };

    OCROutput.Text = IronOcr.Read(inputPath).Text;

    if (OCROutput.Text != "" && pattern != "")
    {
        Regex regex = new Regex(pattern);
        Match match = regex.Match(OCROutput.Text);
        if (match.Success)
            OCRSpecificOutput.Text = match.Value;
    }
}
```

TÀI LIỆU THAM KHẢO

- [1] Statista Research Department, "Market share held by the leading computer (desktop/tablet/console) operating systems worldwide from January 2012 to June 2023", 2023. Tại: <https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/>.
- [2] Tao Yu, Rungseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, Zhibo Chen. "Inpaint Anything: Segment Anything Meets Image Inpainting", 2023. Tại: <https://arxiv.org/abs/2304.06790>.
- [3] R. C. Gonzalez, R. E. Woods, "Digital Image Processing, Addison Wesley, 3rd edition" (2008).
- [4] Telea, Alexandru. "An image inpainting technique based on the fast marching method." Journal of graphics tools 9.1, 23 -24, 2004. Tại: <https://www.olivier-augereau.com/docs/2004JGraphToolsTelea.pdf>
- [5] Bertalmio, Marcelo, Andrea L. Bertozzi, and Guillermo Sapiro. "Navier-stokes, fluid dynamics, and image and video inpainting." In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, pp. I-355. IEEE, 2001. Tại: <https://www.math.ucla.edu/~bertozzi/papers/cvpr01.pdf>
- [6] Stephen V. Rice, Frank R. Jenkins, and Thomas A. Nartker. "The Fourth Annual Test of OCR Accuracy", 1995. Tại: <https://tesseract-ocr.github.io/docs/AT-1995.pdf>
- [7] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, Ross Girshick, "Segment Anything ". Tại <https://arxiv.org/pdf/2304.02643.pdf>
- [8] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, Victor Lempitsky, "Resolution-robust Large Mask Inpainting with Fourier Convolutions ". Tại : <https://arxiv.org/pdf/2109.07161.pdf>
- [9] Microsoft, "Windows UI Library in the Windows App SDK (WinUI 3)". Tại: <https://learn.microsoft.com/en-us/windows/apps/winui/winui3/>
- [10] Microsoft, "Build a C# .NET app with WinUI 3 and Win32 interop". Tại: <https://learn.microsoft.com/en-us/windows/apps/winui/winui3/desktop-winui3-app-with-basic-interop>