ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &TRUYỀN THÔNG

HOÀNG XUÂN TRUNG

KHÔI PHỤC ẢNH BẰNG TỐI ƯU ĐỘ TƯƠNG TỰ CỤC BỘ

Chuyên ngành: Khoa học máy tính

Mã số: 60 48 01 01

LUẬN VĂN THẠC SĨ CHUYÊN NGÀNH KHOA HỌC MÁY TÍNH

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS.ĐÀO NAM ANH

THÁI NGUYÊN - 2015

LỜI CẢM ƠN

Trên thực tế không có thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ trong suốt thời gian từ khi bắt đầu học tập tại trường đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô Trường Đại học Công nghệ Thông tin và Truyền thông - Đại học Thái Nguyên cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường, và luôn luôn tạo mọi điều kiện tốt nhất cho chúng em trong suốt quá trình theo học tại trường. Em xin chân thành cảm ơn quý Thầy Cô và Ban lãnh đạo nhà trường!

Với lòng biết ơn sâu sắc nhất em xin gửi lời cảm ơn tới TS. Đào Nam Anh, là cán bộ trực tiếp hướng dẫn khoa học cho em. Thầy đã dành nhiều thời gian cho việc hướng dẫn em cách nghiên cứu, đọc tài liệu, cài đặt các thuật toán và giúp đỡ em trong việc xây dựng chương trình, em xin chân thành cảm ơn Thầy!

Và cuối cùng em xin bày tỏ lòng chân thành và biết ơn tới lãnh đạo khoa Công nghệ Thông tin trường Cao đẳng Hoan Châu Nghệ An cùng bạn bè đồng nghiệp đã luôn ở bên cạnh những lúc em khó khăn và tạo điều kiện thuận lợi giúp em hoàn thành luận văn.

Hoàng Xuân Trung

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn là kết quả nghiên cứu của tôi, không sao chép của ai. Nội dung luận văn có tham khảo và sử dụng các tài liệu liên quan, các thông tin trong tài liệu được đăng tải trên các tạp chí và các trang website theo danh mục tài liệu của luận văn.

Tác giả luận văn

Hoàng Xuân Trung

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

Các thuật ngữ	Ý nghĩa
Wavelet	Sóng nhỏ
Inpainting	Khôi phục ảnh
Các từ viết tắt	**Ý nghĩa**
XLA	Xử lý ảnh
PDE	Partial differential equation
MSE	Mean square error
PSNR	Peak Signal to Noise Ratio

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

MỞ ĐẦU

Khôi phục ảnh (inpainting) là quá trình xây dựng lại các bộ phận bị mất hoặc xuống cấp của ảnh và video. Trong trường hợp một bức tranh có giá trị, nhiệm vụ này sẽ được thực hiện bởi một nghệ sĩ có tay nghề cao phục hồi tranh. Trong thế giới công nghệ thông tin, khôi phục ảnh đề cập đến việc áp dụng các thuật toán phức tạp để thay thế các bộ phận dữ liệu ảnh bị mất hoặc bị hỏng.

Khôi phục ảnh có liên quan đến việc loại bỏ nhiễu, và đôi khi các thuật toán sử dụng các ý tưởng loại nhiễu, nhưng về cơ bản khôi phục ảnh là một vấn đề khác vấn đề loại nhiễu. Vùng nhiễu thường có một số thông tin của ảnh gốc nhưng trong khôi phục ảnh, một số vùng bị mất hoàn toàn dữ liệu ảnh gốc.

Trọng tâm của luận văn này là tìm hiểu các vấn đề liên quan đến việc khôi phục ảnh, nghiên cứu một số thuật toán khôi phục ảnh và tập trung tìm hiểu thuật toán tổng hợp để tạo ra các vùng ảnh lớn từ các kết cấu mẫu, và kỹ thuật lấp đầy những khoảng trống ảnh nhỏ. Trong đó các giá trị màu sắc được tính toán tổng hợp dựa trên mẫu. Thuật toán sẽ được thực nghiệm với chương trình sử dụng ngôn ngữ C++, MathLab.

Ngoài phần mở đầu và kết luận, luận văn được chia làm 3 chương, luận văn có các chương như sau:

Chương 1: Tổng quan về bài toán khôi phục ảnh

Trình bày các vấn đề đặt ra cần giải quyết trong bài toán khôi phục ảnh, các hướng tiếp cận chính xử lý nhiễu ảnh và khôi phục phần ảnh bị mất. Trình bày một số phương pháp. Khôi phục ảnh dựa vào hàm Gaussian, khôi phục ảnh nhanh, khôi phục ảnh Bertalmio. Một số phương pháp khôi phục ảnh khác. Một số tiêu chí dùng để đánh giá kết quả khôi phục ảnh

Chương 2: Khôi phục ảnh từ các bản vá và với điều kiện tối ưu địa phương. Các thuật toán khôi phục ảnh bằng bản vá sẽ được trình bày trong mối liên hệ với thuật toán khôi phục ảnh bằng bản vá với tối ưu địa phương

Các thuật toán khôi phục ảnh bằng bản vá có khả năng lấp khoảng trống ảnh bằng cách tổng hợp các vùng ảnh từ một vùng khác. Phương pháp này được gọi là vá, bởi vì trong mỗi lần điền thông tin, thuật toán điền một mảng các điểm ảnh, chứ không chỉ là một điểm ảnh duy nhất như trong phương pháp dùng biến phân từng phần.

Chương 3: Thực nghiệm và đánh giá

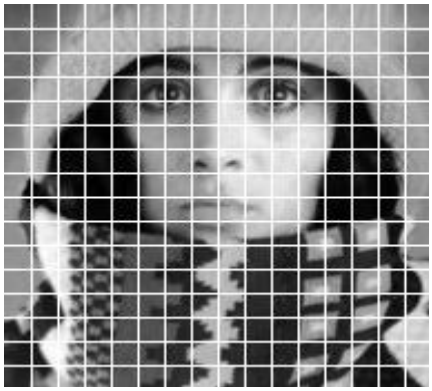
Trình bày về việc cài đặt chương trình, xây dựng dữ liệu thực nghiệm với thuật toán khôi phục ảnh có khả năng lấp khoảng trống ảnh bằng cách tổng hợp các vùng ảnh từ bản vá, dựa vào độ tương tự cục bộ, các quá trình thực nghiệm, kết quả.

CHƯƠNG 1:
TỔNG QUAN VỀ BÀI TOÁN KHÔI PHỤC ẢNH

1.1. Một số khái niệm cơ bản

1.1.1. *Phần tử ảnh (Picture Element)*

Ảnh trong thực tế là một ảnh liên tục về không gian và về giá trị độ sáng. Để có thể xử lý ảnh bằng máy tính cần thiết phải tiến hành số hoá ảnh. Trong quá trình số hoá, người ta biến đổi tín hiệu liên tục sang tín hiệu rời rạc thông qua quá trình lấy mẫu (rời rạc hóa về không gian) và lượng hoá thành phần giá trị mà thể về nguyên tắc bằng mắt thường không phân biệt được hai điểm kề nhau. Trong quá trình này, người ta sử dụng khái niệm Picture element mà ta quen gọi hay viết là Pixel - phần tử ảnh, ở đây cũng cần phân biệt khái niệm pixel hay đề cập đến trong các hệ thống đồ hoạ máy tính. Để tránh nhầm lẫn ta tạm gọi khái niệm pixel này là pixel thiết bị. Khái niệm pixel thiết bị có thể xem xét như sau: khi ta quan sát màn hình (trong chế độ đồ hoạ), màn hình không liên tục mà gồm nhiều điểm nhỏ, gọi là pixel. Mỗi pixel gồm một cặp toạ độ x, y và màu [1].



Hình 1: Mỗi điểm ảnh có một tọa độ x, y,
và một giá trị với ảnh xám

Cặp toạ độ x, y tạo nên độ phân giải (resolution). Như màn hình máy tính có nhiều loại với độ phân giải khác nhau: màn hình CGA có độ phân giải là 320 x 200; màn hình VGA là 640 x 350,…..

Như vậy, một ảnh là một tập hợp các điểm ảnh. Khi được số hoá, nó thường được biểu diễn bởi bảng hai chiều I(n,p): n dòng và p cột. Ta nói ảnh gồm n, x, p pixels. Người ta thường kí hiệu I(x,y) để chỉ một pixel. Thường giá trị của n chọn bằng p và bằng 256. Hình 1 cho ta thấy việc biểu diễn một ảnh với độ phân giải khác nhau. Một pixel có thể lưu trữ trên 1, 4, 8 hay 24 bit.

1.1.2. *Mức xám (Gray level)*

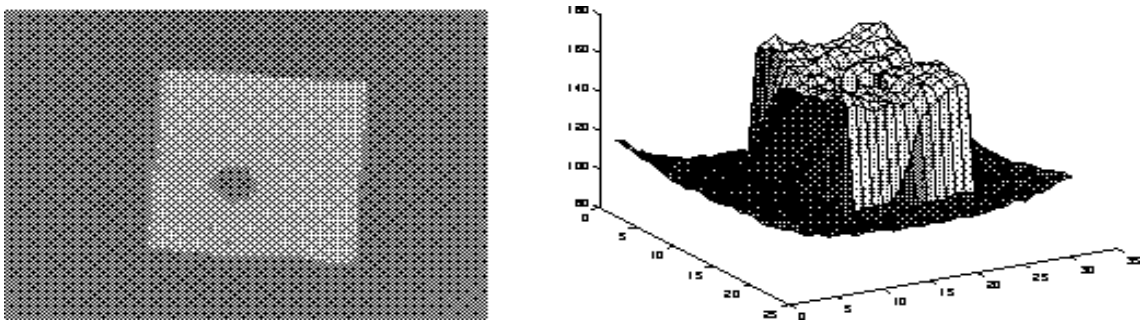
Mức xám là kết quả sự mã hoá tương ứng một cường độ sáng của mỗi điểm ảnh với một giá trị số kết quả của quá trình lượng hoá. Cách mã hoá kinh điển thường dùng 16, 32 hay 64 mức. Mã hoá 256 mức là phổ dụng nhất do lý do kỹ thuật. Vì 2^8 = 256 (0, 1,..., 255), nên với 256 mức, mỗi pixel sẽ được mã hoá bởi 8 bit.

Định nghĩa: Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại điểm đó.

Các thang giá trị mức xám thông thường: 16, 32, 642, 128, 256 (Mức 256 là mức phổ dụng. Lý do: từ kỹ thuật máy tính dùng 1byte (8 bit) để biểu diễn mức xám: Mức xám dùng 1 byte biểu diễn: 2^8=256 mức, tức là từ 0 đến 255).

Ảnh đen trắng: là ảnh có hai màu đen, trắng (không chứa màu khác) với mức xám ở các điểm ảnh có thể khác nhau.

Ảnh nhị phân: ảnh chỉ có 2 mức đen trắng phân biệt nhau tức dùng 1 bit mô tả 2^1 mức khác nhau. Nói cách khác: mỗi điểm ảnh của ảnh nhị phân chỉ có thể là 0 và 1.



Hình 2: Ảnh xám và đồ thị theo mức xám

Ảnh màu: trong khuôn khổ lý thuyết ba màu (Red, Blue, Green) để tạo nên thế giới màu, người ta thường dùng 3 byte để mô tả mức màu, khi đó các giá trị màu: $2^{8*3} = 2^{24}$ = 16,7 triệu màu.

1.1.3. *Quan hệ giữa ảnh, các điểm ảnh, mức xám*

Trong biểu diễn ảnh, người ta thường dùng các phần tử đặc trưng của ảnh là pixel. Nhìn chung có thể xem một hàm hai biến chứa các thông tin như biểu diễn của một ảnh. Các mô hình biểu diễn ảnh cho ta một mô tả lôgic hay định lượng các tính chất của hàm này. Trong biểu diễn ảnh cần chú ý đến tính trung thực của ảnh hoặc các tiêu chuẩn "thông minh" để đo chất lượng ảnh hoặc tính hiệu quả của các kỹ thuật xử lý.

Việc xử lý ảnh số yêu cầu ảnh phải được mẫu hoá và lượng tử hoá. Thí dụ một ảnh ma trận 512 dòng gồm khoảng 512 x 512 pixel. Việc lượng tử hoá ảnh là chuyển đổi tín hiệu tương tự sang tín hiệu số (Analog Digital Convert) của một ảnh đã lấy mẫu sang một số hữu hạn mức xám.

Một số mô hình thường được dùng trong biểu diễn ảnh: Mô hình toán, mô hình thống kê. Trong mô hình toán, ảnh hai chiều được biểu diễn

nhờ các hàm hai biến trực giao gọi là các hàm cơ sở. Các biến đổi này sẽ trình bày kỹ trong chương.

Với mô hình thống kê, một ảnh được coi như một phần tử của một tập hợp đặc trưng bởi các đại lượng như: kỳ vọng toán học, hiệp biến, phương sai, moment.

1.1.4. *Lân cận của điểm ảnh*

Giả sử một ảnh số được biểu diễn bằng hàm *f(x,y)*, *p* và *q* là cặp điểm ảnh có quan hệ với nhau, điểm ảnh *p* có tọa độ *(x,y)*. Định nghĩa các lân cận của điểm ảnh.

Lân cận 4 của *p* kí hiệu $N_4(p)$:

$$N_4(p)=\{(x-1,y);(x,y-1);(x,y+1);(x+1,y)\} \quad (1.1)$$

Lân cận chéo của *p* kí hiệu $N_p(p)$:

$$N_p(p)=\{(x+1,y+1);(x+1,y-1);(x-1,y+1);(x-1, y-1)\} \quad (1.1.2)$$

Lân cận 8 của *p* kí hiệu $N_8(p)$:

$$N_8(p) = N_4(p) + N_p(p) \quad (1.2)$$

1.1.5. *Mối liên kết điểm ảnh*

Các mối liên kết được sử dụng để xác định giới hạn của đối tượng hoặc xác định vùng trong một ảnh. Một liên kết được đặc trưng bởi tính liền kề giữa các điểm và mức xám của chúng.

Có ba loại liên kết:

Liên kết 4: Hai điểm ảnh *p* và *q* được gọi là liên kết 4 nếu *q* thuộc $N_4(p)$

Liên kết 8: Hai điểm ảnh *p* và *q* được gọi là liên kết 8 nếu *q* thuộc $N_8(p)$

Liên kết *m* (liên kết hỗn hợp): Hai điểm ảnh *p* và *q* được gọi là liên kết hỗn hợp nếu *q* thuộc $N_4(p)$ hoặc *q* thuộc $N_8(p)$

1.1.6. *Đo khoảng cách giữa các điểm ảnh*

Khoảng cách *D(p, q)* giữa hai điểm ảnh *p* có tọa độ *(x, y)*, *q* có tọa độ *(s, t)* là hàm khoảng cách (Distance) nếu:

$D(p, q) \geq 0$ (Với $D(p, q)=0$ khi và chỉ khi $p=q$)

$D(p, q) = D(q, p)$

$D(p, z) \leq D(p, q) + D(q, z)$; *z* là một điểm ảnh khác

Khoảng cách *Euclide* giữa hai điểm ảnh *p(x, y)* và *q(s, t)* được định nghĩa như sau:

$$D_e(p, q) = [(x - s)^2 + (y - t)^2]^{1/2} \quad (1.3)$$

1.2. Tổng quan khôi phục ảnh

Việc khôi phục ảnh có nguồn gốc từ việc phục hồi các tác phẩm hội họa, hay còn gọi là phục chế tranh, ở châu Âu các tác phẩm hội họa thời trung cổ bắt đầu được phục hồi vào thời kỳ Phục hưng. Các tác phẩm này sau vài trăm năm bị xuống cấp, có nhiều chỗ bị hỏng, bị mờ hoặc thậm chí không còn đường nét, màu sắc. Những khoảng trống của các tranh đó cần được vẽ lại.

a. hình bị hỏng **b. hình sau khi khôi phục**

Hình 3: Khôi phục tác phẩm hội họa

Sự cần thiết phải chỉnh sửa các tranh hội họa sau này được áp dụng cho việc khôi phục ảnh chụp và phim. Mục đích của việc khôi phục là việc khôi phục lại các phần đã bị mất. Phim nhựa để lâu có thể xuất hiện các vết xước hoặc các đốm trắng. Ảnh cũng vậy, giấy ảnh để lâu năm có thể bị trắng xóa vài chỗ. Các chỗ trống này cần được khôi phục lại.

Trong lĩnh vực ảnh kỹ thuật số, việc khôi phục ảnh đầu tiên xuất hiện nhằm mục đích loại bỏ lỗi, trong khi truyền dữ liệu ảnh số qua mạng, các ảnh này thường được nén cho nhẹ trước khi gửi đi. Tuy nhiên việc nén ảnh gây ra mất một số vùng của ảnh này. Khi ảnh nén được chuyển đến vị trí đích, ảnh được mở nén và các vùng ảnh bị mất thông tin do nén, cần được khôi phục lại. Như vậy, việc khôi phục ảnh này nhằm làm loại bỏ lỗi do nén ảnh khi truyền dữ liệu.

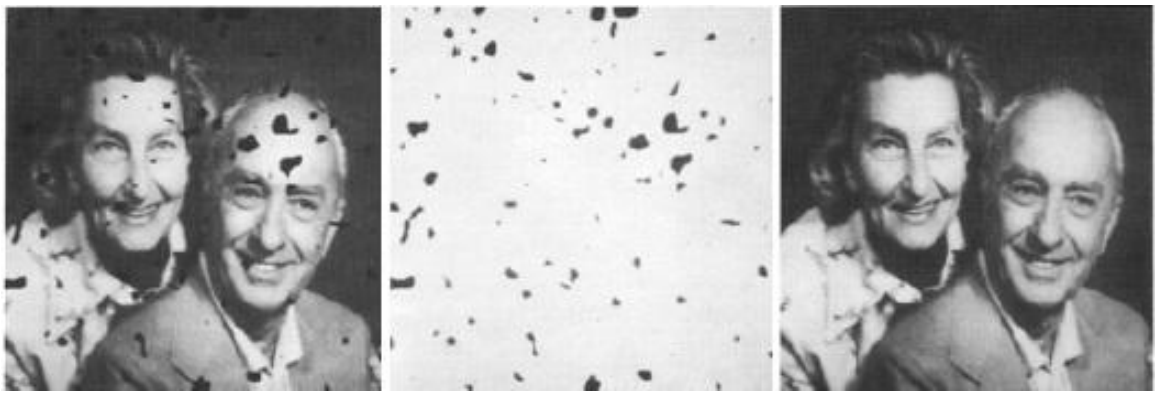
1.2.1. *Bài toán khôi phục ảnh*

Bài toán khôi phục ảnh có thể được nêu như sau:

Cho một ảnh ký hiệu là I và một Ω bên trong I bị mất hoàn toàn thông tin. Việc khôi phục ảnh I là tìm các giá trị phù hợp cho các điểm ảnh

trong vùng Ω, sao cho vùng này không khác biệt so với các vùng bao quanh nó.

Luận văn này chỉ đề cập đến việc khôi phục ảnh màu nơi có các vùng ảnh bị mất hoàn toàn thông tin. Việc khôi phục ảnh vẫn còn một phần thông tin như ảnh bị mờ, ảnh có nhiễu, là một bài toán khác mà luận văn sẽ không đề cập đến.



a. ảnh bị hỏng **b. mặt na, xác định vị trí cần khôi phục** **c. ảnh đã được khôi phục**

Hình 4: Khôi phục ảnh đen trắng

1.2.2. *Ứng dụng khôi phục ảnh*

Khôi phục ảnh có nhiều mục đích. Một là để khôi phục lại các phần bị hư hỏng của một ảnh. Ví dụ như một bức ảnh cũ có nếp gấp và vết xước đã để lại những khoảng trống ảnh. Hai là để loại bỏ các yếu tố không mong muốn hiện diện trong ảnh. Ví dụ trong một cảnh quay phim xuất hiện một microphone thu âm thanh của kỹ thuật viên âm thanh. Hình microphone này không được xuất hiện trong khung hình, do vậy, cần thiết phải xóa đi.

Vùng trống Ω do người sử dụng xác định. Bởi vậy, việc xác định vùng Ω trong ảnh, không phải là phần việc của vấn đề khôi phục ảnh. Đầu

vào của bài toán khôi phục ảnh phải có ảnh cần khôi phục và mặt nạ xác định vùng trống Ω.

Việc khôi phục một ảnh được áp dụng mở rộng cho trường hợp nhiều ảnh, như chuỗi hình ảnh trong video.

Sau đây là phần giới thiệu kỹ thuật khôi phục ảnh cơ bản, phương pháp dùng phương trình đạo hàm để đánh giá độ tương đồng giữ hai điểm ảnh.

1.3. Một số phương pháp khôi phục ảnh

Chương này tìm hiểu một số phương pháp khôi phục ảnh, ngoại trừ phương pháp khôi phục ảnh dùng bản vá và tối ưu địa phương sẽ được trình bày chi tiết tại chương 2.

1.3.1. Phương pháp khôi phục ảnh dùng kim tự tháp mờ Gaussian

Một phương pháp khôi phục ảnh dùng nội suy do Ogden, Adelson, Bergen, và Burt đề cập trong [1]. Từ một ảnh ban đầu, thuật toán liên tiếp áp dụng dùng bộ lọc làm mờ Gaussian với một nhân nhỏ. Kết quả của hai lần áp dụng bộ lọc Gausian tương tự như một lần lọc:

$$G^* (G^*I) = (G^*G)^*I \tag{1.4}$$

$$G_{\sigma 1} * G_{\sigma 2} = G_{\sigma} \quad \sigma^2 = \sigma_1^2 + \sigma_2^2 \tag{1.5}$$

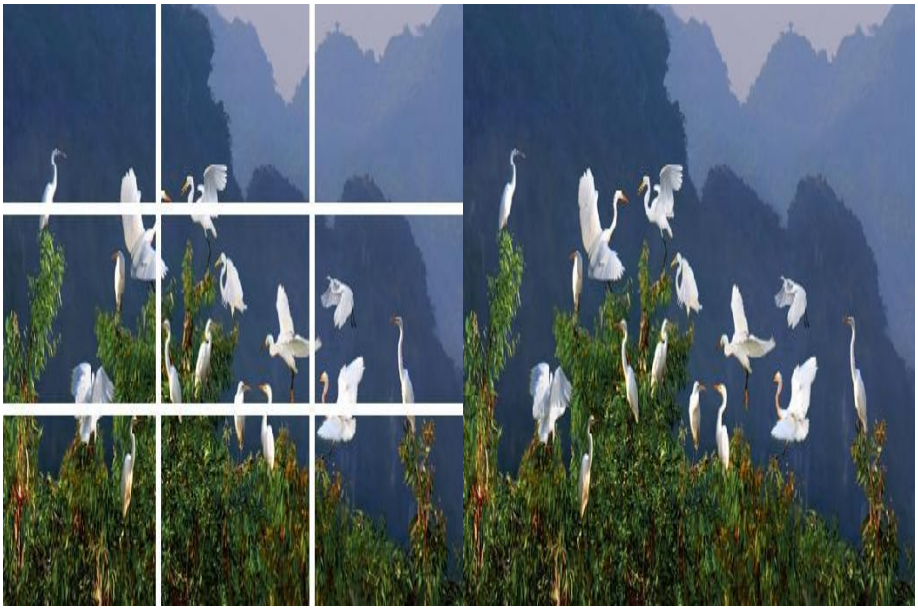
Bắt đầu từ ảnh gốc I_0, ảnh sẽ dần mờ đi do bộ lọc Gaussian (hình 5)

Thuật toán áp dụng bộ lọc Gaussian cho ảnh đầu vào nhiều lần, đồng thời thu nhỏ kích thước của ảnh, tỷ lệ ½. Trong quá trình mờ này, các vùng trống của ảnh sẽ được lấp dần từ các vùng lân cận. Quá trình này dừng lại khi tất cả các vùng trống đã được lấp đầy (hình 5).

Tiếp theo là việc sao chép các phần đã làm đầy từ I_{k+1} ngược lại cho I_k, đồng thời tăng dần kích thước (*2). Đây là quá trình ngược lại với quá

trình mờ bên trên. Cuối cùng thu được ảnh đã lấp đầy các khoảng trống (hình 5).

Hình 5: Khôi phục ảnh dùng kim tự tháp mờ Gaussian: từ trái sang phải: I_0, I_1, I_2, I_3

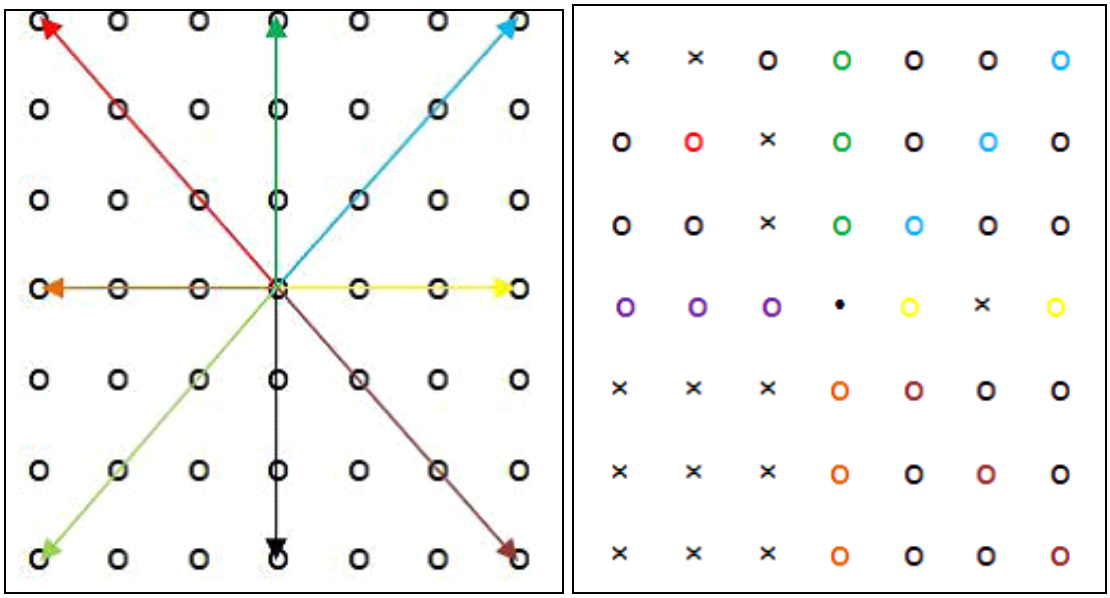


a. ảnh đầu vào b. ảnh đầu ra

Hình 6: Khôi phục ảnh dùng kim tự tháp mờ Gaussian

1.3.2. Phương pháp khôi phục ảnh dùng bộ lọc Median

Thuật toán có dùng nhiều vòng lặp nội suy với bộ lọc Median. Trong mỗi vòng lặp, đầu tiên ta tìm một điểm ảnh trong vùng trống, và có lân cận là điểm ảnh có đủ thông tin. Với mỗi điểm lân cận này xác định được một hướng. Với mỗi hướng, ta tìm giá trị median trên hướng đó. Sau đó ta lấy median của các median trên, và gán giá trị này cho điểm ảnh đang xét. Quá trình này dừng lại khi toàn bộ các điểm ảnh trống đã được lấp đầy.



a. là các hướng b. dấu nhân là cac vị trí có thông tin, dấu tròn mất thông tin

Hình 7: Khôi phục ảnh Median theo các hướng

1.3.3. Phương pháp khôi phục ảnh nhanh dựa vào bộ lọc

Đây là một thuật toán đơn giản. Thuật toán lặp nội suy. Trong mỗi vòng lặp cần tìm ra điểm ảnh trống, có lân cận là các điểm ảnh có đủ thông tin. Với các điểm ảnh trống này, ta dùng bộ lọc có dạng ma trận có số cạnh lẻ, và có nhân của ma trận có giá trị bằng không. Dưới đây là ví dụ 2 bộ lọc.

a	b	b
b	**0**	b
a	b	a

c	c	c
c	**0**	c
c	c	c

Trong đó a = 0.073235; b = 0.176765; c = 0.125.

a. ảnh đầu vào

b. ảnh đầu ra

Hình 8: Khôi phục ảnh nhanh dựa vào bộ lọc

1.3.4. Khôi phục ảnh dùng biến phân từng phần PDE

Phương pháp Partial Differential Equations (PDE) được dịch là Phương pháp khôi phục ảnh dùng hàm biến phân từng phần. Ở đây có thể dùng từ "vi" phân, tuy nhiên với từ "biến" phân thì có ý nghĩa sát hơn với

bài toán khôi phục ảnh: hàm này tìm sự biến đổi giá trị của một điểm ảnh so với giá trị tại điểm ảnh lân cận. Từ đó xét xem độ tương đồng vả giá trị tại hai điểm lân cận này. Như vậy hàm biến phân thể hiện mức độ biến đổi có ý nghĩa phù hợp với bài toán khôi phục ảnh.

Phương pháp khôi phục ảnh dùng hàm biến phân từng phần có đặc điểm như sau. Mỗi điểm ảnh của Ω nằm tại đường biên với phần I/Ω sẽ được gán cho giá trị, sao cho giá trị được gán này phải tương tự như các điểm ảnh trong vùng lân cận về mức độ biến phân. Tất cả các điểm ảnh trong vùng Ω sẽ được lần lượt gán giá trị, cho đến khi mọi điểm ảnh của Ω đều được gán giá trị.

Việc tìm độ tương tự với hàm biến phân ở đây có hai cách. Một là so sánh độ biến phân của điểm ảnh cần gán giá trị với độ biến phân của các điểm ảnh của vùng đã có thông tin. Hai là so sánh độ biến phân của điểm ảnh cần gán giá trị với độ biến phân của các điểm ảnh trong một ảnh mẫu khác: mục đích là dùng ảnh mẫu để điền vào phần trống của ảnh đầu vào.

Phương pháp khôi phục ảnh dùng hàm biến phân được Bertalmio, Sapiro, Caselles, Ballester đề cập như sau.

Thuật toán chạy lặp với các isophotes, được định nghĩa là các dòng của ảnh có cùng một mức xám. Để tìm các isophotes cần giải phương trình vi phân:

$$\frac{\partial I}{\partial t} = \nabla^{\perp} I . \nabla \Delta I \qquad (1.6)$$

Cho ảnh đầu vào I, bên trong vùng trống. Trong trạng thái ổn định phương trình có dạng

$$\nabla^{\perp} I . \nabla \Delta I = 0 \qquad (1.7)$$

a. ảnh đầu vào

b. ảnh kết quả

Hình 9: Khôi phục ảnh biến phân từng phần PDE

1.3.5. Khôi phục ảnh dùng phương trình Navier-Stokes

Một cách tượng tự với hàm biến thể từng phần PDE là áp dụng phương trình Navier-Stokes trong lý thuyết dòng chảy. Thuật toán chạy lặp với các điểm ảnh trống, mà có lân cận là các điểm ảnh có đủ thông tin.

Dùng phương trình dòng chảy Navier-Stokes mô phỏng xu hướng dòng chảy của phần có đủ thông tin để điền vào ô trống.

1.3.6. Khôi phục ảnh dùng tổng biến thể

Phương pháp tiếp cận tìm vị trí có thiểu tổng biến thể thấp nhất với điểm ảnh trống để điền thông tin.

Thuật toán của Chan và Shen chạy lặp với các điểm ảnh trống, mà có lân cận là các điểm ảnh có đủ thông tin. Dùng hàm tổng biến thể để xác định điểm ảnh có tổng biến thể tương tự điểm ảnh trống. Từ đó điền thông tin vào điểm ảnh trống.

Thuật toán dựa trên việc tối thiểu hóa giới hạn của bình phương độ tin cậy ở ngoài Ω và một tổng tiêu chí biến đổi trong Ω, ví dụ, giới hạn năng lượng.

$$\int_A |\nabla \mu| dx + \frac{\pi}{2} \int_\Omega |\mu - \mu_0|^2 \, dx \tag{1.8}$$

Với λ là một hệ số nhân Lagrange. Để thực hiện, Chan và Shen tìm kiếm điểm quan trọng, sử dụng một chương trình lặp Gauss-Jacobi cho hệ thống tuyến tính liên quan đến một kết quả gần đúng của phương trình Euler-Lagrange bằng những sự khác biệt hữu hạn.

a. ảnh đầu vào b. ảnh kết qủa

Hình 10: Khôi phục ảnh dùng tổng biến thể

1.4. Một số tiêu chí dùng để đánh giá chất khôi phục ảnh

1.4.1. Tổng quan về tiêu chí đánh giá chất lượng ảnh

Để kiểm tra kết quả các thuật toán khôi phục ảnh, cần có một số thống nhất về tiêu chí đánh giá chất lượng ảnh nói chung cũng như chất lượng ảnh trong lĩnh vực khôi phục ảnh nói riêng. Các tiêu chí nhằm đưa ra một đánh giá khách quan để đánh giá sự phát triển và cải thiện các phương pháp khôi phục ảnh.

Phần này sẽ giới thiệu một số tiêu chí cơ bản được dùng để đánh giá chất lượng ảnh. Tiêu chí có thể cho phép đánh giá kết quả khôi phục ảnh với các số liệu khách quan. Để đánh giá chất lượng của ảnh (hay khung ảnh video) ở đầu ra của thuật toán, người ta thường sử dụng hai tham số: Sai số bình phương trung bình - MSE (mean square error) và phương pháp đề xuất với tỷ lệ tín hiệu trên tín hiệu tạp PSNR (Peak Signal to Noise Ratio).

1.4.2. Sai số bình phương trung bình MSE

MSE giữa ảnh gốc và ảnh khôi phục được tính như sau:

$$MSE = \frac{1}{mn}\sum_{y=0}^{m-1}\sum_{x=0}^{n-1}[I_O(x,y) - I_R(x,y)]^2 \qquad (1.9)$$

Trong đó, I_0 là ảnh gốc, I_R là ảnh đầu ra. x và y chỉ vị trí của điểm ảnh, m và n chỉ độ số điểm ảnh theo chiều ngang và chiều dọc của ảnh. Chỉ số MSE cho giá trị càng nhỏ nghĩa là chất lượng của ảnh càng tốt.

1.4.3.Tỷ lệ tín hiệu trên tín hiệu tạp PSNR

PSNR, đơn vị: deciben (dB), thường được sử dụng trong nghiên cứu xử lý hình ảnh:

$$\mathbf{PSNR} = \mathbf{10} * \log_{10}(\frac{\mathbf{R}^2}{\mathbf{MSE}}) \qquad (1.10)$$

Trong đó, R là giá trị lớn nhất của điểm ảnh. Trong trưởng hợp ảnh được biểu diễn bằng số nguyên 8 bit, thì R=255.

Hình 11: Ví dụ PSNR

1.4.4. Ứng dụng của MSE và PSNR

MSE và PSNR được tính toán cho ảnh màu bằng cách chuyển đổi ảnh từ hệ RGB sang hệ YCbCr. Sự chuyển đổi này được đưa ra vì mắt người nhạy cảm với thông tin cường độ Y. Sau khi chuyển đổi sang hệ YCbCr, MSE và PSNR được tính trên kênh Y.

Thông thường, nếu PSNR ≥ 37 dB thì hệ thống mắt người gần như không phân biệt được giữa ảnh gốc và ảnh khôi phục. PSNR càng cao thì chất lượng ảnh khôi phục càng tốt. Khi hai hình ảnh giống hệt nhau, MSE sẽ bằng 0 và PSNR đi đến vô cực.

1.5. Kết luận chương 1

Khôi phục ảnh là một vấn đề rất khó và còn cần nhiều thời gian mới được giải quyết. Khôi phục ảnh có nhiều ứng dụng thực tiễn, áp dụng cho một ảnh đơn lẻ hoặc đồng thời cho một chuỗi các ảnh.

Chương tiếp theo tìm hiểu tổng quan một số phương pháp khôi phục ảnh.

Đã có nhiều kỹ thuật khôi phục ảnh được nghiên cứu ứng dụng, với dự đa dạng của các kỹ thuật. Các kỹ thuật có chi phí tính toán thấp có thể áp dụng thời gian thực cho video. Tuy nhiên hiện có ít kỹ thuật đáp ứng

yêu cầu thời gian thực. Đây có thể là chủ đề nghiên cứu ứng dụng trong thời gian tới.

Chương này tập trung vào phương pháp khôi phục ảnh dùng bản vá với điều kiện tối ưu hóa địa phương.

CHƯƠNG 2: KHÔI PHỤC ẢNH DÙNG BẢN VÁ VÀ TỐI ƯU ĐỊA PHƯƠNG

Chương này tập trung phân tích thuật toán khôi phục ảnh bằng bản vá với tối ưu địa phương. Các thuật toán khôi phục ảnh bằng bản vá sẽ được trình bày trong mối liên hệ với thuật toán khôi phục ảnh bằng bản vá với tối ưu địa phương

Các thuật toán khôi phục ảnh bằng bản vá có khả năng lấp khoảng trống ảnh bằng cách tổng hợp các vùng ảnh từ một vùng khác. Phương pháp này được gọi là vá, bởi vì trong mỗi lần điền thông tin, thuật toán điền một mảng các điểm ảnh, chứ không chỉ là một điểm ảnh duy nhất như trong phương pháp dùng biến phân từng phần.

2.1. Khôi phục ảnh dùng bản vá

Trong mục này, một số thuật toán khôi phục ảnh dùng bản vá được tổng hợp, từ đó đưa ra một thuật toán khôi phục ảnh dùng bản và với điều kiện tối ưu tại địa phương trong mục 2.2

2.1.1. Khôi phục ảnh dùng vùng mẫu

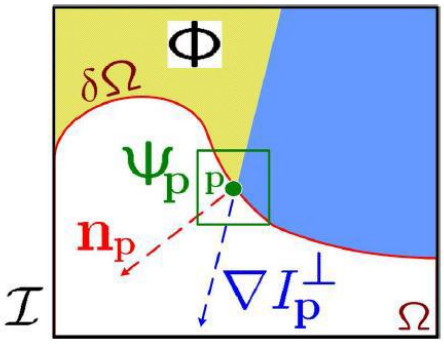
Thuật toán này giải quyết vấn đề khôi phục ảnh nhằm mục đích loại bỏ một số đối tượng của ảnh hoặc sửa chữa phần ảnh bị hỏng bằng cách thay thế các vùng trống bằng cách sử dụng thông tin trong phần còn lại của ảnh. Phương pháp khôi phục ảnh đề xuất ở đây được xây dựng trên quan điểm: dựa trên các mẫu có tính tương tự địa phương với vùng khôi phục.

Phương pháp này sử dụng các vùng mẫu. Thuật toán chạy lặp với các điểm ảnh trống nằm trên đường biên với phần ảnh có đủ thông tin. Với mỗi lần lặp, một bản mẫu được xác định, bản mẫu này sẽ được so sánh đối chiếu với các vùng khác, để tìm vùng có độ tương tự cao nhất. Hàm đánh giá độ tương tự này mang tính địa phương bởi nó chỉ áp dụng cho một phần của ảnh gần với điểm ảnh đang xét.

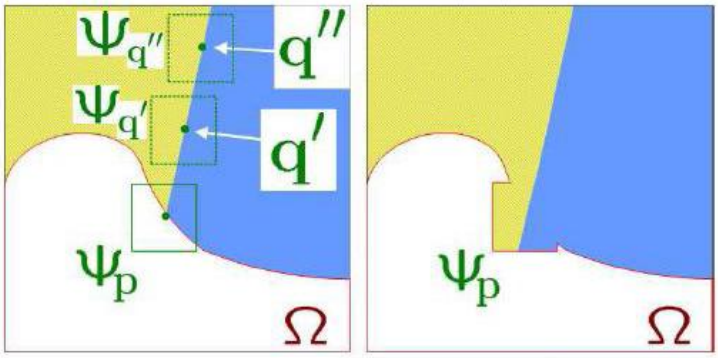
Khi đã tìm ra vùng có độ tương tự cục bộ cao nhất, vùng đó được dùng để vá vào phần trống của điểm ảnh đang xét.

Trong thuật toán này, kết quả và phụ thuộc nhiều vào trình tự vá. Với mỗi trình tự lặp tìm điểm trống cần vá, sẽ có một bản mẫu, khác với bản mẫu trong trình tự vá khác. Bởi vậy, ảnh sẽ được vá khác nhau khi trình tự duyệt các điểm trống khác nhau.

Trong hình 12 , điểm ảnh *p* của ảnh *I* bị trống. *p* nằm trên đường biên với vùng có đủ thông tin, của sổ nhỏ màu xanh lá cây tại *p* xác định ra mẫu. Mẫu này sẽ được đối sánh với vùng ảnh có đủ thông tin, ở gần *p*.



Hình 12: Khôi phục ảnh dùng vùng mẫu:
xác định mẫu tại *p*.



Hình 13: Khôi phục ảnh dùng vùng mẫu: tìm miếng vá thích hợp cho *p* tại *q'*, *q"*, và cuối cùng tiến hành vá q' cho *p*.

Việc tính toán tương tự tạo ra trọng số dựa trên cạnh và sự khác biệt cấu trúc giữa các mẫu ứng cử viên khôi phục. Phương pháp này cho phép chọn mẫu khôi phục dựa trên một số yếu tố.

a. ảnh đầu vào

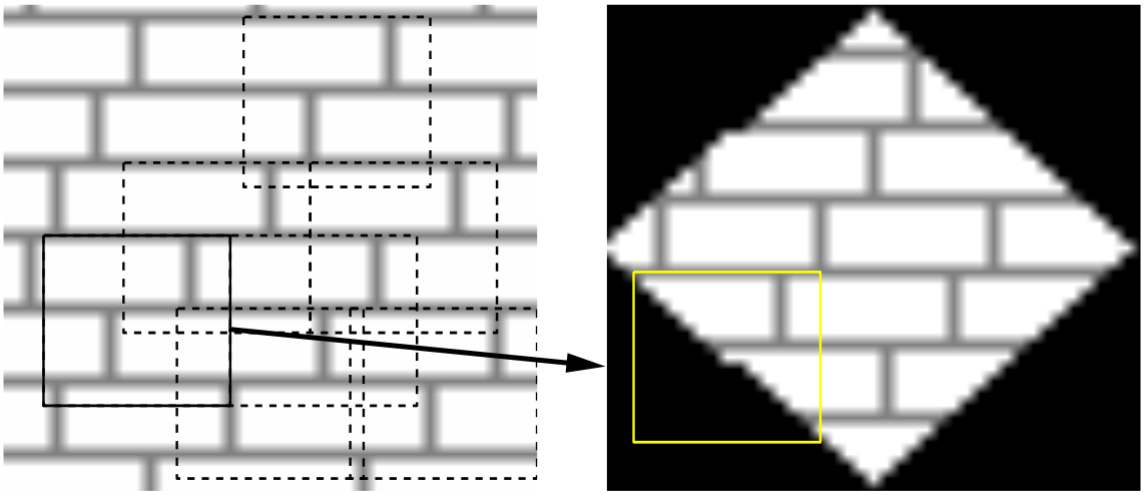
b. ảnh đầu ra

Hình 14: Khôi phục ảnh dùng vùng mẫu

2.1.2. Khôi phục ảnh dùng bản vá có kết cấu

Efros và Leung trong [12] đề xuất một phương pháp tổng hợp kết cấu từ một kết cấu mẫu. Phương pháp này được áp dụng khôi phục ảnh.

Thuật toán như sau: Khoảng trống ảnh được lấp đầy theo đệ quy. Tại mỗi điểm ảnh trống P nằm bên cạnh ranh giới với vùng đủ thông tin, vùng trống được vá bằng các giá trị của trong vùng đủ thông tin tại điểm ảnh Q sao cho vùng kề Ψ(Q) của Q (một miếng vá vuông có Q là trung tâm) có sự tương đồng nhất với vùng kề Ψ(P) của P.



a. ảnh kết cấu b. dùng kết cấu để vá

Hình 15: Khôi phục ảnh dùng bản vá có kết cấu: bên trái là ảnh kết cấu, bên phải: dùng kết cấu để vá.

Khoảng cách hình ảnh được lấp đầy một cách đệ quy, từ ranh giới vùng trắng và vùng có thông tin. Mỗi "ô trống" điểm ảnh P tại ranh giới được làm đầy với các giá trị của các điểm ảnh Q (nằm bên ngoài vùng trắng, ví dụ như Q là một điểm ảnh với thông tin hợp lệ). Như vậy các khu vực lân cận Ψ(Q) của Q (một miếng vá vuông có trung tâm tại Q) phần lớn tương tự như khu vực Ψ(P) của P. Điều này có thể được thể hiện như một vấn đề tối ưu hóa:

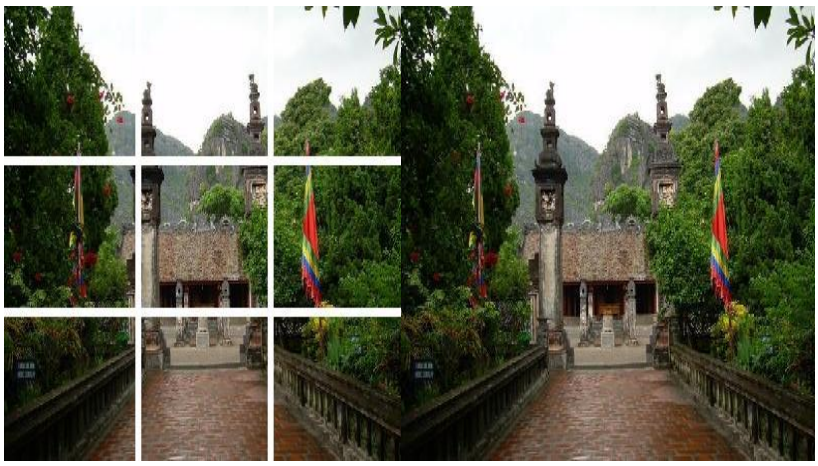
$$| Output(P) = Value(Q), P \in \Omega, Q \notin \Omega, Q = \arg\min d(\Psi(Q)) \qquad (1.11)$$

Trong đó d (Ψ (P), Ψ (Q)) là tổng của khoảng cách phương khác của hai bản vá Ψ(P) và Ψ(Q):

$$d(\Psi_1, \Psi_2) = \sum_i \sum_i |\Psi_1(i, j) - \Psi_2(i, j)|^2 \qquad (1.12)$$

Và các chỉ số i, j chạy theo chiều dài bản vá (ví dụ: nếu Ψ là bản vá có độ rộng dài là 11*11, thì $0 \le i, j \le 10$ khi. Khi P được lấp đầy, thuật toán đánh dấu P không còn là điểm trắng nữa, để không phải quay lại điểm P lần nữa.

Những thiếu sót chính của thuật toán này là chi phí tính toán cao, việc lựa chọn kích thước vùng kề (là một tham số toàn ảnh do người dùng lựa chọn, nhưng thay đổi tùy thuộc vào nội dung ảnh), thứ tự lấp đầy (có thể tạo ra ranh giới chưa được nối với một số đối tượng) và thực tế là không xử lý tốt với các cạnh ảnh. Ngoài ra, kết quả khá là kém nếu khoảng trống ảnh lớn và phân tán, ví dụ như một ảnh có 80% các điểm ảnh đã bị mất do nhiễu ngẫu nhiên.



a. ảnh đầu vào b. ảnh đầu ra

Hình 16: Khôi phục ảnh dùng bản vá có kết cấu theo Efros và Leung:

Criminisi và cộng sự [4] cải thiện thuật toán này ở hai khía cạnh. Thứ nhất, họ đã thay đổi thứ tự điền thành một phương pháp ưu tiên, trong đó điểm ảnh trống ở các cạnh có độ ưu tiên cao hơn so với các điểm ảnh trống trên vùng bằng phẳng. Do đó, có thể khôi phục chính xác cạnh mà có thể bị mất trong thuật toán đầu.

Thứ hai, họ sao chép toàn bộ các bản vá lỗi thay vì từng điểm đơn lẻ, vì vậy phương pháp này nhanh hơn đáng kể.

Vẫn có một số thiếu sót như không có khả năng xử lý với phối cảnh và cần phải tự chọn kích thước vùng kề (ở đây có hai kích thước để thiết

lập, một cho các bản vá để so sánh và một cho các bản vá để sao chép). Ngoài ra, các đối tượng có cạnh cong có thể không được khôi phục chính xác.

a. ảnh đầu vào b. ảnh đầu ra

Hình 17: Khôi phục ảnh dùng bản vá có kết cấu theo Criminisi và cộng sự: bên trái là ảnh đầu vào, bên phải là ảnh kết quả.

Ashikhmin [2] đóng góp cũng như cải thiện phương pháp ban đầu của Efros và Leung [12]. Với ý tưởng giảm chi phí tính toán của các thủ tục, Ashikhmin đề xuất tìm kiếm các ứng viên tốt nhất Q để sao chép giá trị của đó vào điểm ảnh trống P, không tìm kiếm toàn bộ ảnh mà chỉ tìm kiếm trong số các ứng cử viên của các điểm kề của P đã được khôi phục. Tốc độ đạt được với kỹ thuật đơn giản này là đáng kể, và cũng có một ảnh hưởng rất tích cực liên quan đến chất lượng ảnh đầu ra.

Trong khi hầu hết các phương pháp khôi phục ảnh cố gắng trở nên hoàn toàn tự động (ngoài việc thiết lập một số thông số theo cách thủ công), có những phương pháp trợ giúp người dùng cung cấp kết quả đáng kể chỉ với một ít đầu vào từ người sử dụng.

Trong nghiên cứu của Sun và cộng sự người sử dụng phải xác định các đường cong trong vùng trống, các đường cong tương ứng với biên của đối tượng có liên quan. Tổng hợp bản vá được thực hiện dọc theo những đường cong bên trong khoảng trống ảnh, bằng cách sao chép từ các bản vá lỗi nằm trên các phân đoạn của các đường cong nằm ngoài khoảng trống, trong vùng đã biết.

Một khi các đường cong được hoàn thành, các điểm ảnh trống còn lại được khôi phục sử dụng kỹ thuật của Ashikhmin [2] với các ưu tiên như trong Criminisi và cộng sự [4]. Barnes và và cộng sự [5] đẩy nhanh phương pháp này và làm cho đó tương tác, bằng cách sử dụng tìm kiếm ngẫu nhiên và kết hợp thành một bước tuyên truyền cấu trúc và tổng hợp kết cấu của Sun và cộng sự.

2.1.3. Khôi phục ảnh dùng bản vá với độ thưa

Trong các phương pháp khôi phục ảnh vá dùng kết cấu của Efros và Leung [12], và khôi phục ảnh bởi Criminisi và cộng sự [4], các bản vá của một ảnh tạo nên một từ điển tốt để biểu thị các phần khác của ảnh. Ý tưởng này đã được áp dụng thành công các lĩnh vực xử lý ảnh khác, ví dụ như giảm nhiễu và phân vùng ảnh.

Biểu diễn ảnh thưa có tính tổng quát hơn bằng cách sử dụng từ điển để khôi phục bối cảnh. Ví dụ, bằng cách sử dụng từ điển overcomplete thích hợp với việc biểu diễn ảnh hình học và kết cấu, Elad và cộng sự [15] đề xuất một mô hình phân tách ảnh với hệ số thưa dạng hình học và các

thành phần kết cấu của ảnh, và chỉ ra rằng mô hình có thể dễ dàng thích hợp với khôi phục ảnh.

Mô tả về mô hình này như sau.

Cho u là một ảnh đại diện là một vector trong R^N. Các ma trận D_g, D_t kích thước $N \times k_g$ và $N \times k_t$ biểu diễn cho bộ từ điển hình học và bộ từ điển kết cấu. Gọi $\alpha_g \in R^{kg}$ và $\alpha_t \in R^{kg}$ là hệ số hình học và hệ số kết cấu, vậy

$$u = D_g \alpha_g + D_t \alpha_t \qquad (1.13)$$

biểu diễn cho sự tách ảnh thành hai phần hình học và kết cấu, sử dụng từ điển thu thập trong D_g và D_t.

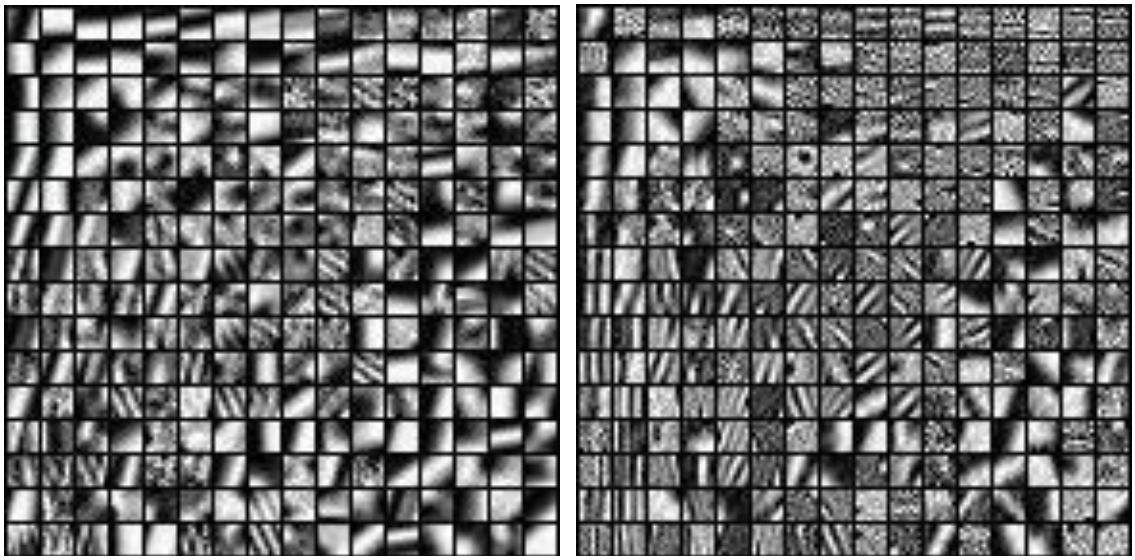
Elad và cộng sự [15] đề xuất mô hình biến phân với tổng biến thể để thể hiện các ràng buộc:

$$\min_{(\alpha_g,\alpha_t)} \|\alpha_g\|_1 + \|\alpha_{gt}\|_1 + \lambda \|u - D_g\alpha_g - D_t\alpha_t\|_2^2 + \gamma TV(D_g\alpha_g) \qquad (1.14)$$

Trong đó, TV biểu thị tổng số biến thể, λ, γ> 0. Mô hình này có thể dễ dàng thích ứng với một mô hình cho khôi phục ảnh.

Trong công thức trên u -D_g α_g -$D_t\alpha_t$ có thể được xem như phần nhiễu của ảnh và λ là một tham số mà phụ thuộc nghịch với độ nhiễu. Do đó, mặt nạ khôi phục có thể được xem như một vùng nơi độ nhiễu là rất lớn (vô hạn).

Các bản vá ảnh là căn hai kích thước của $n = \sqrt{n} \times \sqrt{n}$.



Hình 18: Khôi phục ảnh dùng độ thưa:
bộ từ điển hình học và bộ từ điển kết cấu

a. ảnh đầu vào b. ảnh đầu ra

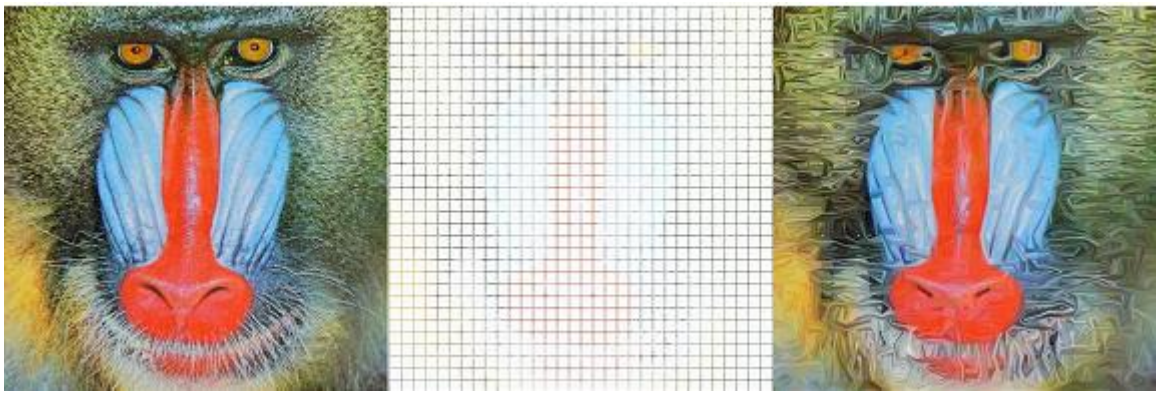
Hình 19: Khôi phục ảnh dùng độ thưa:

a. ảnh đầu vào bị nhiễu b. ảnh đầu ra

Hình 20: Khôi phục ảnh dùng độ thưa:
bên trái là ảnh đầu vào bị nhiễu, bên phải là ảnh kết quả

2.1.4. Khôi phục ảnh dùng kết hợp bản vá và biến phân từng phần PDE

Phương pháp khôi phục ảnh dùng biến phân từng phần PDE khôi phục ảnh tốt cho các vùng mỏng hoặc phân bố rải rác. Tuy nhiên, có nhược điểm chung: chúng không thể khôi phục đúng kết cấu, và điều này đặc biệt có thể nhìn thấy trên khôi phục một vùng lớn. Mặt khác, phương pháp vá không có khả năng để xử lý miền khôi phục thưa như trong hình 18, nơi độ thưa quá lớn. Ngược lại, hầu hết các mô hình PDE biến phân tiếp tục áp dụng tốt trong trường hợp này, ví dụ trong hình 21 theo mô hình đề xuất của Masnou và Morel mang lại kết quả khôi phục. Rõ ràng, một số thông tin hình học có thể được phục hồi, nhưng không có kết cấu.



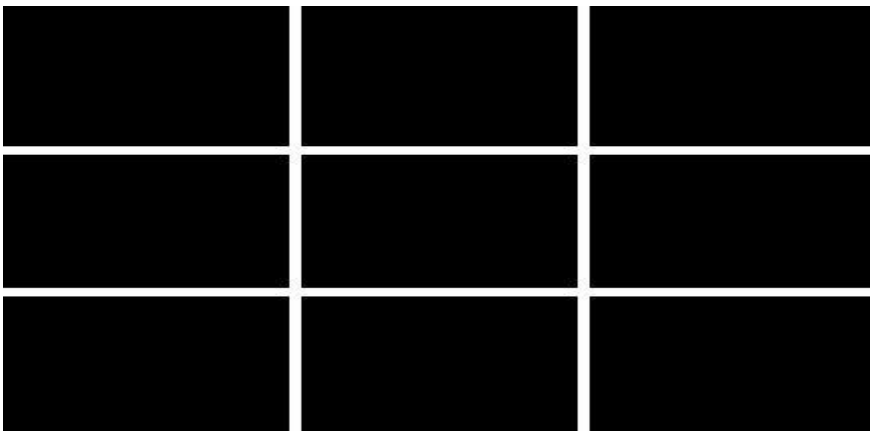
Hình 21: Một hình ảnh ban đầu, sau khi loại bỏ 15x15 hình vuông (hơn 87% các điểm ảnh được loại bỏ), và khôi phục lại với các phương pháp được giới thiệu bởi Masnou và Morel

Mặt khác, phương pháp khôi phục ảnh dùng bản vá không thể xử lý các vùng thưa, vì có thể không thể tìm thấy trong từ điển.

Đã có một số nghiên cứu kết hợp Khôi phục ảnh dùng kết hợp bản vá và biến phân từng phần PDE để xử lý cả kết cấu và hình học các cấu trúc.

Drori, Cohen, và Yeshurun trong [13] tiến hành tìm kiếm các vùng lân cận tương tự với hướng dẫn bởi một ước tính sơ bộ. Các giá trị khôi phục sử dụng trên nhiều cấp độ co ảnh (scale) và chiến lược chập. Ngoài ra, các bản vá lỗi hợp lệ được tăng cường bằng cách sử dụng phép quay, và phản xạ. Một ví dụ từ Drori và cộng sự [13] được thể hiện trong hình dưới.

a. ảnh đầu vào bị nhiễu



b. Mặt nạ

c. ảnh kết quả

Hình 22: Khôi phục dùng kết hợp bản vá và biến phân từng phần PDE

2.1.5. Một số dạng khôi phục ảnh bằng bản vá khác

Tất cả các phương pháp nêu trên chỉ liên quan đến một ảnh duy nhất. Đối với trường hợp nhiều ảnh, có cách để khôi phục ảnh: khôi phục chuỗi ảnh trong video, và khôi phục một ảnh nhưng sử dụng thông tin từ một số ảnh khác.

Video có thể cần được khôi phục để là "xóa lỗi quay phim" và khôi phục những khối ảnh bị mất và cho các ứng dụng phục hồi phim (xử lý những khoảng trống ảnh tạo ra bởi bụi, xước hay mài mòn)

Một khó khăn đặc biệt trong video khôi phục phục hồi phim là cho chất lượng ảnh đẹp của các đầu ra, phát hiện khoảng trống và làm đầy được giải quyết đồng thời, và trong giảm nhiễu.

a. video ban đầu

b. video kết quả

Hình 23: Khôi phục ảnh dùng bản vá cho video

Wexler và cộng sự đề xuất một thuật toán khôi phục video không gian - thời gian, phát triển từ kỹ thuật Efros và Leung [12] và kết hợp đó với ý tưởng gắn kết giữa các vùng láng giềng phát triển bởi Ashikhmin [2].

Đầu tiên, đối với mỗi sản phẩm điểm ảnh P ta xét một không gian - thời gian tập trung tại P, so sánh đó với tất cả các khối có thể trong đoạn video, tìm vị trí Q tương tự nhất, đó sẽ là lời giải của P.

Đối với mỗi khối thông tin ta xét và so sánh không chỉ màu sắc mà còn vector chuyển động. Sau đó, thay vì sao chép giá trị của Q vào P, ta chép vào P trung bình của các ứng viên những lân cận của P: ví dụ, nếu R là ở bên phải của P và S là phóng viên của R, sau đó các điểm ảnh bên trái của S sẽ được tham gia ở mức trung bình để điền vào P. Điều này được dựa trên ý tưởng của Ashikhmin [2]

Những thiếu sót của phương pháp khôi phục phim này là rằng các kết quả bị mờ đáng kể (do sử dụng hàm trung bình), chỉ được áp dụng giới hạn cho máy ảnh tĩnh (để đơn giản ước lượng chuyển động) và chuyển động định kỳ mà không có sự thay đổi về quy mô, và chi phí tính toán là khá cao (do sự so sánh của các khối 3D).

Shiratori và cộng sự thực hiện khôi phục phim với một bản vá dựa trên kỹ thuật tương tự như của Efros và Leung [12] và sau đó truyền các màu sắc dọc theo (khôi phục) quỹ đạo chuyển động. Phương pháp này giả định rằng thông tin chuyển động là đủ để điền vào các lỗ hổng trong video, tuy nhiên không phải luôn luôn như vậy. Kết quả có thể bị mờ, do bước truyền màu.

Patwardhan và cộng sự đề xuất một phương pháp hình khôi phục bao gồm ba bước. Trong bước đầu tiên phân tách các chuỗi video thành các lớp chuyển động nhị phân foreground và background, được sử dụng để xây dựng ba khuôn ảnh (một khuôn là tương đương với một ảnh toàn cảnh

được tạo ra bởi khâu cùng một số ảnh): một bức tranh cho nền trước, một cho nền và một phần cho các thông tin chuyển động. Hai bước khác của thuật toán khôi phục cho hai khuôn cuối cùng.

Các thuật toán được giới hạn cho trường hợp vị trí chuyển động máy ảnh gần như song song với mặt phẳng ảnh, và các đối tượng di chuyển một cách lặp và không thay đổi kích thước: đó là những hạn chế được áp đặt để thuật toán vá tổng hợp tương tự như của Efros và Leung [12] có thể được áp dụng được.

Hays và Efros [8] thực hiện khôi phục của một ảnh duy nhất sử dụng thông tin từ một cơ sở dữ liệu với một số hàng triệu ảnh. Họ sử dụng một cảnh - mô tả để giảm không gian tìm kiếm trong 2.000.000 ảnh khác để điền vào ảnh ban đầu.

Các thiếu sót chính của phương pháp này là đó dựa trên việc quản lý và điều hành một cơ sở dữ liệu ảnh lớn. Khi thuật toán không thành công, đó có thể là do thiếu cảnh tốt phù hợp (nếu ảnh mục tiêu là không điển hình), hoặc do hành vi vi phạm ngữ nghĩa (ví dụ như thất bại trong việc nhận ra một đối tượng do đó không tìm ra được bản vá phù hợp).

a. ảnh đầu vào

b. ảnh kết quả

Hình 24: Khôi phục ảnh dùng bản vá từ các ảnh khác:

2.2. Khôi phục ảnh dùng bản vá với điều kiện tối ưu địa phương

Trên cơ sở một số phương pháp khôi phục ảnh được giới thiệu tại chương 1, chương 2 chuyên sâu về phương pháp khôi phục ảnh dùng bộ vá.

2.2.1. Bước tiền xử lý ảnh màu: Tách ảnh

Trong bước tiền xử lý, ảnh màu đầu vào được tách theo bốn cách như sau:

A. Dùng mức xám:

Ảnh được tách thành 03 kênh RGB, sau đó mỗi kênh màu được nhân với tham số độ sáng cho từng kênh màu, tiếp theo cộng lại, ta được một ảnh xám. Các tham số đó như sau:

$$B' = B * 0.0114,$$
$$G' = G * 0.587, \quad (1.15)$$
$$R' = R * 0.299$$

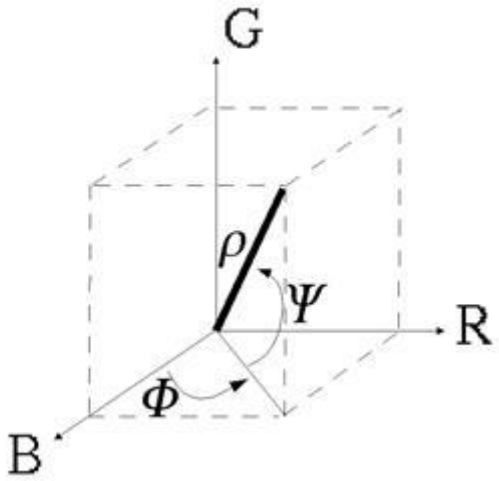
Với cách này, ảnh màu đầu vào sẽ cho một ảnh xám.

B. Tách RGB:

Mỗi kênh màu ảnh R, G và B của ảnh đầu vào được coi là một ảnh xám độc lập. Thuật toán khôi phục ảnh sẽ áp dụng cho từng kênh màu trên. Cuối cùng, các kết quả của từng kênh màu sẽ được nhóm lại để tạo ra ảnh màu kết quả.

C. Tách trong không gian Vector:

Ảnh màu đầu vào được tách thành ba kênh trong không gian vector màu RGB: Độ dài của vector p, và hai góc Ψ *và* Φ tạo bởi vector với hai mặt phẳng qui chiếu.



Hình 25: Tách trong không gian Vector

Sau khi tách ảnh bằng một trong bốn cách nêu trên, các ảnh xám sẽ được phân tích xung các vùng mất thông tin, từ đó tìm ra bản vá để đắp vào phần mất thông tin.

2.2.2. Phát biểu bài toán phôi phục ảnh bằng bản vá

Gọi Φ là vùng bị mất thông tin trong ảnh.Ψ_p là một điểm ảnh trong vùng này, đồng thời nằm trên đường biên với vùng có đủ thông tin. Gọi Ψ_p là miếng vá cho điểm *p*.

Bài toán có mục đích lựa chọn một bản vá x_p^* có độ tương tự địa phương cao nhất. Điều này có thể đạt được bằng cách quan tâm đến độ sắc nét ảnh và các chi tiết, trong khi xem xét mối quan hệ giữa các miếng vá.

2.2.3. Điều kiện tối ưu địa phương

Điền kiện cần thiết là: Bản vá x_p^* cho điểm *p* phải tương đồng với bản vá $x_{\bar{p}}^*$ của các điểm lân cận của *p*. Như vậy cần tìm tối thiểu sự khác biệt giữa x_p^* và $x_{\bar{p}}^*$. Đây chính là độ tương đồng địa phương mà thuật toán này quan tâm đến.

Biểu diễn một các hình thức quan hệ giữa các vùng như sau:

$$P(\Psi_p, x_p, x_{\bar{p}}^*) = P(\Psi_p, x_{\bar{p}}^* \mid x_p)P(x_p) \quad (1.16)$$

Trong đó X_p là tập hợp các bản vá cho điểm p, $X_{\bar{p}}$ là tập hợp các bản vá cho các lân cận của điểm p.

Bởi quá trình lựa chọn của $x_{\bar{p}}$. độc lập Ψ_p, phương trình trên có thể được viết lại như sau:

$$P(\Psi_{p,} x_p, x_{\bar{p}}) = P(\Psi_p \mid x_p)P\left(x_{\bar{p}} \mid x_p\right)P\left(x_p\right) \quad (1.17)$$

$$P(\Psi_{p,} x_p, x_{\bar{p}}^{\sim}) = P(\Psi_p \mid x_p)P\left(x_p \mid x_{\bar{p}}\right)P\left(x_{\bar{p}}\right) \quad (1.18)$$

Ta có thế xác định hàm số khoảng cách giữa x_p và Ψ_p

$$P(\Psi_p \mid x_p) = \exp(-dx_p, \Psi_p)^2) \quad (1.19)$$

Hàm này thể hiện độ tương đồng của các ứng cử vá với vùng mục tiêu.

Việc lựa chọn bản vá được thực hiện dựa trên tìm kiếm vùng.

Tuy nhiên, như Criminisi và cộng sự [3] cho thấy rằng, cấu trúc địa phương và kết cấu của ảnh có tác động lớn đến kết quả, các yếu tố này có thể được biểu diễn bởi một phần cường độ cạnh và thứ tự tiên điền trong vùng mục tiêu. Thứ tự ưu tiên này xác định trình tự khôi phục ảnh.

Ngoài ra, $P(x_p \mid x_{\bar{p}})$ tương ứng với sự khác biệt giữa các bản vá lỗi lân cận.

Tối ưu toàn ảnh của sự khác biệt đó có thể thu được thông qua các phương pháp tính toán. Ở đây, để giảm chi phí tính toán và đạt được một phương pháp khôi phục ảnh nhanh, ta tìm sự tối thiểu sự khác biệt của một bản vá ứng cử viên với vùng mục tiêu:

$$P(x_p \mid x_{\bar{p}}) = \max_{x_p \in X_p} \exp(-d(x_{\bar{p}}, x_p)^2) \quad (1.20)$$

2.2.4. Thuật toán

Thuật toán sau đây thực hiện việc tìm kiếm bản vá theo điều kiện tối ưu độ tương đồng địa phương. Thuật toán sử dụng các khái niệm sau:

Mặt nạ là ma trận xác định các vùng bị hỏng cần khôi phục.

Bản vá, như đã nêu trên, tìm một phần ảnh ở chỗ không hỏng để vá vào vùng bị hỏng.

Tối ưu độ tương tự địa phương: làm cho bản vá có độ tương đồng cao nhất với vùng lân cận xung quanh vùng cần vá.

THUẬT TOÁN KHÔI PHỤC ẢNH DÙNG BẢN VÁ TỐI ƯU ĐỘ TƯƠNG TỰ ĐỊA PHƯƠNG

Đầu vào: ảnh màu, mặt nạ xác định vùng cần vá.

Đầu ra: ảnh màu, đã được khôi phục.

Thực hiện:

1. Chọn các điểm ảnh trên ranh giới của các vùng bị mất với độ ưu tiên cao nhất theo điều kiện [15].
2. Trích xuất một danh mục các bản vá lỗi từ các điểm ảnh trong phần ảnh có đủ thông tin.
3. Lấy ra các bản vá lỗi cho các điểm ảnh lân cận tại đường biên
4. Tính giá trị nhất quán địa phương để chọn tối ưu vá lỗi cho điểm ở đường biên nêu trên.
5. Quay trở lại 1 cho đến khi toàn bộ các điểm trống đã được gán bản vá.

2.2.5. Đầu vào và đầu ra của thuật toán khôi phục ảnh dùng bản vá với điều kiện tối ưu địa phương.

Dưới đây là một số ví dụ khôi phục ảnh màu với ảnh đầu vào và ảnh đầu ra.

a. ảnh đầu vào bị nhiễu b. ảnh kết quả

Hình 26: Khôi phục ảnh dùng bản vá tối ưu địa phương

a. anh đầu vào bị nhiễu

b. ảnh kết quả

Hình 27: Khôi phục ảnh dùng bản vá tối ưu địa phương

2.3. Kết luận chương 2

Chương này phân tích một số thuật toán khôi phục ảnh dùng bả vá trong đó có phương pháp khôi phục giữ gìn ảnh chi tiết, kết cấu và sắc nét trong ảnh, bằng cách mở rộng các phương pháp mẫu dựa trên thiết lập tính nhất quán của địa phương. Kết hợp cạnh địa phương khi tìm sự giống nhau địa phương của các mẫu ứng cử viên gần ranh giới vùng khôi phục. Việc tính toán tương tự tạo ra trọng số cho mỗi miếng vá ứng cử viên, từ đó quyết định thứ tự khôi phục thông qua một danh mục các bản vá. Kết quả thực nghiệm cho thấy các ưu thế của phương pháp này so với các phương pháp khác

CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM

3.1. Môi trường cài đặt

Chương trình được cài đặt trên Môi trường Microsoft Windows Ultimate Edition 32-bit Service Pack 1, sử dụng Microsoft Visual Studio 2010 với máy tính có cấu hình như sau:

- CPU: Intel (R) Core (TM) i3 CPU - M 370 @ 2.40 GHz
- Memory Type: DDR3
- Memory Size: 4096 Mbytes (4 GB)
- HDD: 320 GB

Mã nguồn có từ Jun Zhou và Antonio Robles-Kelly, Canberra Research Lab, NICT, Australia và được chỉnh sửa theo yêu cầu của luận văn.

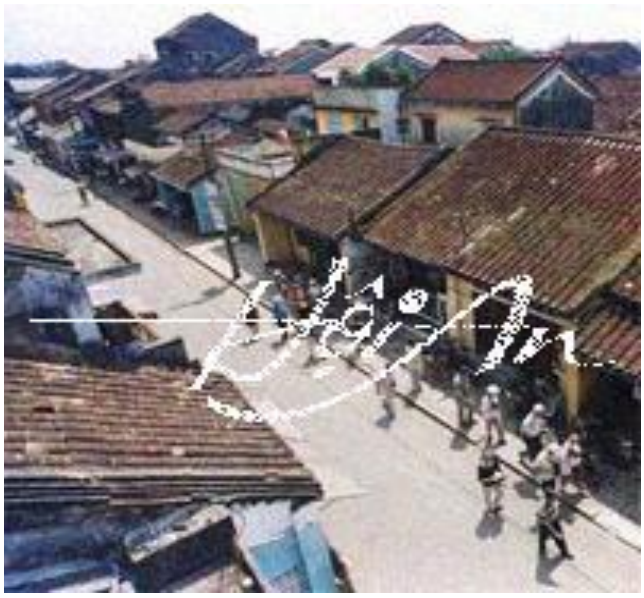
3.2. Kết quả thực nghiệm

Phần này báo cáo kết quả thử nghiệm sử dụng phương pháp khôi phục ảnh dùng bản vá tối ưu độ tương tự cục bộ.

Đầu vào của thuật toán là một ảnh gốc, một mặt nạ.

Đầu ra là ảnh được khôi phục ở các phần đánh dấu trong mặt nạ.

Mã nguồn có từ Jun Zhou và Antonio Robles-Kelly, Canberra Research Lab, NICT, Australia



a. ảnh gốc

b. Mặt nạ phần cần khôi phục

Hình 28: Ảnh gốc và mặt nạ

Các vị trí đường biên

Hình 29: Kiểm tra phần biên của mặt nạ

Thuật toán kiểm tra các điểm ảnh của mặt nạ, nằm trên đường biên với phần tốt của ảnh (hình 25)

ảnh kết quả

Phần được khôi phục

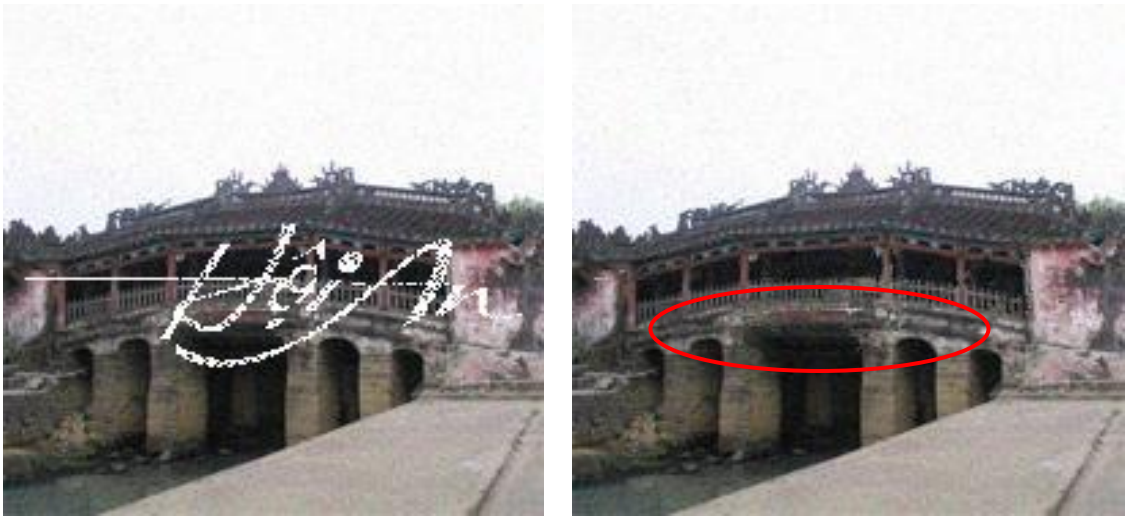
Hình 30: Kiểm tra phần biên của mặt nạ

Có thể nhận thấy rằng phần kết cấu (mái ngói) được khôi phục phù hợp với các vùng lân cận.

Một ví dụ khác khôi phục phần có kết cấu mạnh, ít lỗi:

MSE=0.018, PSNR=65.673

Một số ví dụ khác khôi phục phần có kết cấu mạnh, có lỗi:



Khôi phục kết cấu không đủ nét

MSE=0.015, PSNR=66.506

Hình 31: Khôi phục phần có kết cấu mạnh, có lỗi

MSE=0.012, PSNR=67.397

Hình 32: Lỗi khôi phục phần có kết cấu phức tạp

Trong hình 28 phần mái ngói có cấu trúc khá phức tạp,Việc khôi phục ảnh để lại lỗi khi phần mái cần liền, song bị phân thành 3 khúc.

Trong hình 29 phần mặt nạ cắt qua mặt nước, nơi có sự thay đổi màu yếu (kết cấu yếu). Tuy nhiên thuật toán khôi phục được tốt phần kết cấu yếu.

MSE=0.007, PSNR=69.755

Hình 33: Khôi phục kết cấu yếu, tốt

Phần màu trắng bị cắt khúc

Hình 34: Khôi phục kết cấu yếu, có lỗi

MSE=0.010, PSNR=68.090

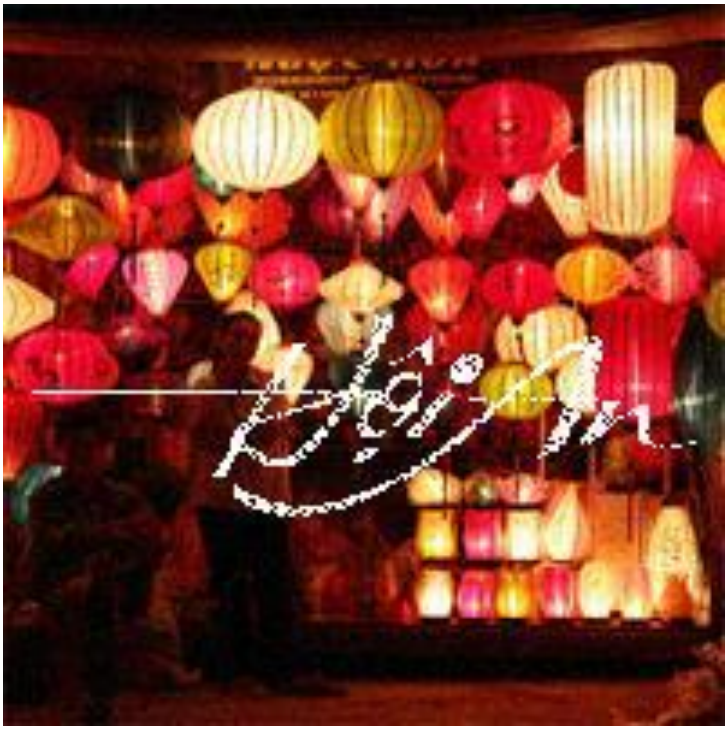
Hình 35: Khôi phục kết cấu yếu, ít lỗi

MSE=0.013, PSNR=66.898

MSE=0.007, PSNR=69.539

Hình 36: Khôi phục kết cấu phức tạp, ít lỗi

MSE=0.015, PSNR=66.428



Đường vai không được liền, do kết cấu yếu

MSE=0.018, PSNR=65.712

Hình 37: Khôi phục kết cấu phức tạp, ít lỗi

Đường trắng bị tách thành hai đoạn

MSE=0.011, PSNR=67.608

Hình 38: Khôi phục kết cấu phức tạp, ít lỗi

Lỗi đứt quãng nhỏ

MSE=0.007, PSNR=69.755

Hình 39: Khôi phục kết cấu phức tạp, ít lỗi

3.3. So sánh với một số phương pháp khác

So sánh phương pháp khôi phục ảnh dùng vá và kiểm tra độ tương tự cục bộ với các phương pháp khôi phục bằng vá khác được thực hiện với cùng một ảnh đầu vào dưới đây

a: Ảnh gốc 512 * 512 RGB Lena,

b: phương pháp khôi phục ảnh dùng vá và kiểm tra độ tương tự cục bộ (t = 783.79s, OSNR = 21.803 dB)

c: phuong pháp trung bình không cục bộ (t = 6696.7 s, PSNR = 16.750dB).

d: Mặt nạ xác định vùng hỏng

e: Phương pháp tổng biến thể (t = 10200 s, PSNR = 19.995 dB),

f : phương pháp cắt trên ngưỡng (t = 2015.1 s, PSNR = 17.468 dB)

Phương pháp khôi phục ảnh dùng vá và kiểm tra độ tương tự cục bộ (t = 783.79s, OSNR = 21.803 dB) cho phép giữ được sư tương đồng về kết cấu tốt nhất..

Tùy theo trường hợp cụ thể về kíc thước vùng mặt nạ và độ phức tạp của kết cấu, có thể có it lỗi khi khôi phục ảnh.

3.4. Kết luận chương 3

Chương này thử nghiệm phương pháp khôi phục ảnh giữ gìn ảnh chi tiết, kết cấu và sắc nét ảnh bằng cách mở rộng các phương pháp dựa vào mẫu dựa, trên một thiết lập tính nhất quán của địa phương. Thuật toán kết hợp cạnh địa phương trước khi có sự giống nhau của các mẫu ứng cử viên khôi phục ảnh tại ranh giới vùng khôi phục ảnh trên một vùng địa phương. Việc tính toán tương tự tạo ra trọng số cho mỗi miếng vá ứng cử viên, từ đó quyết định thứ tự khôi phục ảnh thông qua một số lần lặp.

KẾT LUẬN

Luận văn này giải quyết vấn đề khôi phục phần trong ảnh mà nhằm mục đích loại bỏ các đối tượng từ một ảnh hoặc sửa chữa ảnh bị hư hỏng bằng cách thay thế các vùng thiếu bằng cách sử dụng thông tin trong phần còn lại của trường. Phương pháp khôi phục ảnh đề xuất ở đây được xây dựng trên một quan điểm dựa trên các mẫu tăng cường tính nhất quán địa phương của vùng khôi phục.

Phương pháp này được thực hiện bằng cách chọn các bản vá tối ưu tối đa hoá tính nhất quán của địa phương đối với các bản vá lỗi tiếp giáp với ứng cử viên. Việc tính toán tương tự tạo ra trọng số dựa trên một cạnh trước và sự khác biệt cấu trúc giữa các mẫu ứng cử viên khôi phục mẫu. Phương pháp này cho phép các thế hệ của một chuỗi khôi phục dựa trên một danh mục các yếu tố. Các thực nghiệm cho thấy phương pháp đề xuất cung cấp một mức cải thiện so với các phương pháp khác.

TÀI LIỆU THAM KHẢO

[1] Lương Mạnh Bá, Nguyễn Thanh Thủy (1999). Nhập môn xử lý ảnh số, Nhà xuất bản Khoa học kỹ thuật, Hà Nội.

[2] Đỗ Năng Toàn, Phạm Việt Bình (2007). Giáo trình xử lý ảnh, Nhà xuất bản Đại học Hà nội.

[3] Võ Đức Khánh, Hoàng Kiếm (2007). Giáo trình xử lý ảnh. Nhà xuất bản Đại học Quốc Gia TP Hồ Chí Minh.

[4] Nguyễn Kim Sách (1977). Xử lý ảnh và video số, Nhà xuất bản Khoa học kỹ thuật, Hà Nội.

[5] Pablo Arias, Gabriele Facciolo, Vicent Caselles, Guillermo Sapiro, "A Variational Framework for Exemplar-Based Image Inpaiting", Springer Science and Business Media, 2011.

[6] Mahmoud Ghoniem, Youssef Chahir, Abderrahim Elmoataz, "Geometric And Texture Inpainting Based On Discrete Regularization On Graphs", ICIP 2009.

[7] Jun Zhou ; Canberra Res. Lab., Robles-Kelly, A., Image Inpainting Based on Local Optimisation, Pattern Recognition (ICPR), 2010.

[8] A. Criminisi, P. Perez, and K. Toyama, "Region Filling and Object Removal by Exemplar-Based Image Khôi phục," IEEE Trans. Image Processing, 13 (9), pp. 1200-1212, September 2004.

[9] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image Completion with Structure Propagation," SIGGRAPH, Vol. 24, pp. 861-868, 2005.

[10] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A Comparative Study of Energy Minimization Methods for Markov Random Fields," ECCV, volume 2, pages 16-29, Graz, Austria, May 2006.

[11] Alexandru Telea, “An Image Inpainting Technique Based on the Fast Marching Method”, Journal of graphics tools, 2004.

[12] A. Efros and T. Leung, “Texture synthesis by non-parametric sampling,” Proc. IEEE International Conference Computer Vision, pp. 1033-1038, Corfu, Greece, September 1999.

[13] E. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. 5th IEEE Int’l Conf. on Image Processing, Chicago, IL. Oct 4-7, 1998.

[14] S. Masnou and J.M. Morel. Level-lines based disocclusion. 5th IEEE Int’l Conf. on Image Processing, Chicago, IL. Oct 4-7, 1998.

[15] C. Kenney and J. Langan. A new image processing primitive: reconstructing images from modified flow fields. University of California Santa Barbara Preprint, 1999.

PHỤ LỤC: TRÍCH MÃ NGUỒN

Mục này trích một đoạn mã nguồn phân tích bản vá :

```
#include "Exemplar.h"
#include "Local.h"
#include "Parallel.h"

IplImage* exmpExtractPatch(IplImage *img, int r, int c, int size) {
  IplImage* ret = cvCreateImage(cvSize((size*2)+1,(size*2)+1),img->depth,img->nChannels);
  for(int a = -size; a <= size; a++) {
    for(int b = -size; b <= size; b++) {
      int row = r + a, col = c + b;
      if(row >= img->height) row = img->height-1;
      if(row < 0) row = 0;
      if(col >= img->width) col = img->width-1;
      if(col < 0) col = 0;
      int x = row - (r - size), y = col - (c - size);
      cvSet2D(ret, x, y, cvGet2D(img, row, col));
    }
  }
  return ret;
}

void exmpBorderFront(IplImage *msk, vector< pair<int, int> >& borderPixels) {
  if(!msk) return;
  borderPixels.clear();
```

```
        for(int a = 0; a < msk->height; a++) {
          for(int b = 0; b < msk->width; b++) {
            if( iPixel(msk, a, b) < 128 )  continue;
            if( iPixel(msk, a+1, b) < 128 ||
              iPixel(msk, a, b+1) < 128 ||
              iPixel(msk, a-1, b) < 128 ||
              iPixel(msk, a, b-1) < 128 ) {
              borderPixels.push_back(make_pair(a, b));
            }
          }
        }
        return;
      }

      void exmpListSourcePatches(IplImage *msk, vector< pair<int, int> >& sourcePatches,
                                 int patchSize, int stepSize) {
        if(!msk) return;
        sourcePatches.clear();
        bool foundUncertain = false;
        for(int a = patchSize; a < msk->height - patchSize; a+=stepSize) {
          for(int b = patchSize; b < msk->width - patchSize; b+=stepSize) {
            foundUncertain = false;
            for(int a1 = a - patchSize; a1 <= a + patchSize && !foundUncertain; a1++) {
              for(int b1 = b - patchSize; b1 <= b + patchSize && !foundUncertain; b1++) {
                if( iPixel(msk, a1, b1) < 128 ) {
                  foundUncertain = true;
```

```
                    break;
                }
            }
        }
        if( foundUncertain ) continue;
        sourcePatches.push_back(make_pair(a, b));
    }
    if( processEvents() ) {
        sourcePatches.clear();
        return;
    }
  }
}

double exmpGetPatchDistance( IplImage *img, IplImage *msk, int x1, int y1, int x2, int y2, int patchSize,
                             double earlyExit, double *result) {
  double dis = 0.0;
  for(int a = -patchSize; a <= patchSize; a++) {
    for(int b = -patchSize; b <= patchSize; b++) {
      int cx1 = x1 + a, cy1 = y1 + b;
      int cx2 = x2 + a, cy2 = y2 + b;
      if( cx1 < 0 || cx1 >= img->height ) continue;
      if( cy1 < 0 || cy1 >= img->width ) continue;
      if( cx2 < 0 || cx2 >= img->height ) continue;
      if( cy2 < 0 || cy2 >= img->width ) continue;
      if( iPixel(msk, cx1, cy1) > 128 ) continue;
      if( earlyExit >= 0.0 && dis > earlyExit ) return 1e50;
      for(int k = 0; k < img->nChannels; k++) {
```

```
                dis += iSq( iPixel( img, cx1, cy1, k ) - iPixel( img, cx2, cy2, k )
);
            }
          }
        }
        if( result != NULL ) (*result) = dis;
        return dis;
      }

      void exmpFillPatch( IplImage *img, IplImage *msk, int sx, int sy, int dx,
int dy, int patchSize ) {
        for(int a = -patchSize; a <= patchSize; a++) {
          for(int b = -patchSize; b <= patchSize; b++) {
            int cx1 = sx + a, cy1 = sy + b;
            int cx2 = dx + a, cy2 = dy + b;
            if( iPixel( msk, cx2, cy2 ) < 128.0 ) continue;
            if( cx1 < 0 || cy1 < 0 || cx2 < 0 || cy2 < 0 ) continue;
            if( cx1 >= img->height || cy1 >= img->width ||
              cx2 >= img->height || cy2 >= img->width ) continue;
            iSixel( msk, cx2, cy2, 0.0 );
            cvSet2D( img, cx2, cy2, cvGet2D( img, cx1, cy1 ) );
          }
        }
      }

      double exmpGetConfidence( IplImage *cp, int x, int y, int patchSize ) {
        double thelta = 0;
        for(int a = x - patchSize; a <= x + patchSize; a++) {
          for(int b = y - patchSize; b <= y + patchSize; b++) {
```

```
            thelta += iPixel(cp, a, b);
          }
        }
        int patchDiameter = (patchSize*2)+1;
        thelta /= (double)(patchDiameter*patchDiameter);
        if( thelta < 0.0 ) thelta = 0.0;
        if( thelta > 1.0 ) thelta = 1.0;
        return thelta;
      }

      void exmpUpdateConfidence( IplImage *img, IplImage *msk, IplImage
*cp, int x, int y, int patchSize ) {
        if( patchSize <= 0 || !img || !msk || !cp ) return;
        double thelta = exmpGetConfidence( cp, x, y, patchSize );
        for(int a = x - patchSize; a <= x + patchSize; a++) {
          for(int b = y - patchSize; b <= y + patchSize; b++) {
            if( iPixel(msk, a, b) < 128 ) continue;
            iSixel( cp, a, b, thelta );
            if( a == x && b == y ) {
            }
          }
        }
      }


      Vector2D exmpGradient( IplImage *image, IplImage *mask, int x, int y )
{
        double dx = 0.0, dy = 0.0, d = 0.0;
        int r = 1;
```

```
        while( d == 0.0 ) {
            double k = (1.0 / (2.0 * (double)(r)));
            double xpr = iPixel(image, x + r, y, -1);
            double xmr = iPixel(image, x - r, y, -1);
            double ypr = iPixel(image, x, y + r, -1);
            double ymr = iPixel(image, x, y - r, -1);
            double cdx = xpr - xmr;
            double cdy = ypr - ymr;
            if( iPixel(mask, x + r, y) > 128 || iPixel(mask, x - r, y) > 128 ) cdx =
0.0;
            if( iPixel(mask, x, y + r) > 128 || iPixel(mask, x, y - r) > 128 ) cdy =
0.0;
            dx = k * cdx; dy = k * cdy;
            d = sqrt(iSq(dx)+iSq(dy));
            r++;
            if( r > EXMP_GRADIENT_MAX_RADIUS ) return Vector2D(0.0,
0.0);
        }
        return Vector2D( dx , dy );
    }

    double exmpCalculateDatapoint( IplImage *img, IplImage *msk, int x, int
y,
                                   int windowSize, int patchSize ) {
        if( patchSize <= 0 || !img || !msk ) return 0.0;

        // Create image patches
        IplImage *mskWindow = exmpExtractPatch( msk, x, y, windowSize );
```

```
        IplImage *mskdx = cvCreateImage( cvGetSize(mskWindow),
IPL_DEPTH_32F, 1 );
        IplImage *mskdy = cvCreateImage( cvGetSize(mskWindow),
IPL_DEPTH_32F, 1 );

        //Sobel differentiate mask image
        cvSobel( mskWindow, mskdx, 0, 1, 3 );
        cvSobel( mskWindow, mskdy, 1, 0, 3 );

        // Setup for isotope
        Vector2D newIsotope(0.0, 0.0);
        Vector2D maxIsotope(0.0, 0.0);
        double maxIsotopeLength = 0.0;

        // Loop through to find maximum isotopes
        for(int a = -patchSize; a <= patchSize; a++) {
           for(int b = -patchSize; b <= patchSize; b++) {
              if( iPixel( msk, x + a, y + b ) > 128 ) continue;
              newIsotope = exmpGradient( img, msk, x + a, y + b );
              double newLength = newIsotope.length();
              if( newLength > maxIsotopeLength ) {
                 maxIsotopeLength = newLength;
                 maxIsotope = newIsotope;
              }
           }
        }
        // Turn vector 90 degrees
        Vector2D isotope = Vector2D( -maxIsotope.y, maxIsotope.x );
```

```
        // Get normal and normalize it
        Vector2D normal( iPixel( mskdx, (mskdx->height/2)+1, (mskdx->width/2)+1 ),
                         iPixel( mskdy, (mskdy->height/2)+1, (mskdy->width/2)+1 ) );
        normal.normalize();

        double dataPointValue = fabs(isotope * normal) / EXMP_DATAPOINT_ALPHA;
        if( dataPointValue < 0.0 ) dataPointValue = 0.0;

        cvReleaseImage( &mskWindow );
        cvReleaseImage( &mskdx );
        cvReleaseImage( &mskdy );

        return dataPointValue;
    }

    double exmpCalculatePriority( IplImage *img, IplImage *msk, IplImage *cp, int x, int y, int patchSize,
                                  double *result) {
        if( patchSize <= 0 || !img || !msk || !cp) return 0.0;
        double ret = exmpGetConfidence(cp, x, y, patchSize) *
          exmpCalculateDatapoint( img, msk, x, y,
          EXMP_DATAPOINT_NORMAL_WINDOWSIZE, patchSize );
        if( result ) (*result) = ret;
        return ret;
    }
```

```
void exmpInitializePriority( IplImage *img, IplImage *msk, IplImage *cp
) {
    if( !img || !msk || !cp ) return;
    for(int a = 0; a < msk->height; a++) {
        for(int b = 0; b < msk->width; b++) {
            if( iPixel(msk, a, b) < 128 ) iSixel( cp, a, b, 1.0 );
            else iSixel( cp, a, b, 0.0 );
        }
    }
}

void exmpSourceMask( IplImage *imsk, IplImage *smsk, int dis ) {
    cvDilate(smsk, smsk, 0, dis);
    for(int a = 0; a < smsk->height; a++) {
        for(int b = 0; b < smsk->width; b++) {
            if( iPixel(imsk, a, b) > 128 ) {
                iSixel(smsk, a, b, 0.0);
            }
        }
    }
}

void exmpInPaint(IplImage *img, IplImage *msk, int iter, int patchSize,
int boundary, int skip, int method, bool GPU) {
    if(!img || !msk || iter<=0 || patchSize<=1) return;

    IplImage *iterImg = iCloneFloat( img, 1.0 );
    IplImage *iterMask = cvCloneImage( msk );
    IplImage *sourceMask = cvCloneImage( msk );
```

```
IplImage *confidence = iCloneFloat( msk, 1.0 );

exmpInitializePriority( iterImg, iterMask, confidence );

vector< pair<int, int> > borderPixels;
vector< pair<int, int> > sourcePatches;
vector<double> borderPriority;

exmpSourceMask( iterMask, sourceMask, boundary );
exmpListSourcePatches( sourceMask, sourcePatches, patchSize, skip );
if( sourcePatches.size() <= 0 ) {
   cvReleaseImage( &iterImg );
   cvReleaseImage( &iterMask );
   cvReleaseImage( &sourceMask );
   cvReleaseImage( &confidence );
   return;
}

for(int i = 0; i < iter; i++) {
   if( processEvents() ) { break; }
   // Determine border front pixels
   exmpBorderFront( iterMask, borderPixels );
   if( borderPixels.size() <= 0 ) break;

   // Calculate priorities and find first priority pixel
   double pr = 0.0, maxpr = -1.0;
   int px = -1, py = -1;
   printl("Calculating border pixel priorities...\n");
   borderPriority.clear();
```

```
            for(int a = 0; a < borderPixels.size(); a++) {
               int x = borderPixels[a].first;
               int y = borderPixels[a].second;
               exmpCalculatePriority( iterImg, iterMask, confidence, x, y,
patchSize, &pr );
               if( method != EXMP_METHOD ) borderPriority.push_back( pr );
               if( pr > maxpr ) {
                  maxpr = pr;
                  px = x;
                  py = y;
               }
            }
            // If we're out of confidence, pick a random
            if( maxpr <= 0.00000001 ) {
               int r = rand()%borderPixels.size();
               px = borderPixels[r].first;
               py = borderPixels[r].second;
            }
            if( px < 0 || py < 0 ) break;

            // Find exemplar image patch in source image
            int sx = -1, sy = -1;
            if( method == EXMP_METHOD ) {
               double dist = 0.0, mindist = -1.0;
               if( gpuEnabled && GPU ) {
                  #ifdef _GPU_
                     printl("Calculating exemplar (GPU) ...\n");
                     float *result = gpuComputeDistanceList( iterImg, iterMask,
patchSize, sourcePatches, px, py );
```

```
                    if( !result ) {
                        printl("Error: Result NULL!\n");
                    } else {
                        for(int a = 0; a < sourcePatches.size(); a++) {
                            if( result[a] < mindist || mindist < -0.5 ) {
                                mindist = result[a];
                                sx = sourcePatches[a].first;
                                sy = sourcePatches[a].second;
                            }
                        }
                        delete[] result;
                    }
                #endif
            } else {
                printl("Calculating exemplar...\n");
                dist = 0.0, mindist = -1.0;
                for(int a = 0; a < sourcePatches.size(); a++) {
                    int x = sourcePatches[a].first;
                    int y = sourcePatches[a].second;
                    exmpGetPatchDistance( iterImg, iterMask, px, py, x, y,
patchSize, mindist, &dist );
                    if( dist < mindist || mindist < -0.5) {
                        mindist = dist;
                        sx = x;
                        sy = y;
                    }
                }
            }
        } else {
```

```
                localGetExemplarPatch( iterImg, iterMask, confidence, px, py,
patchSize,
                    sourcePatches, borderPriority, &sx, &sy );
            }
            if( sx < 0 || sy < 0 ) break;

            // Update confidence values
            printl("Updating confidence...\n");
            exmpUpdateConfidence( iterImg, iterMask, confidence, px, py,
patchSize );

            // Fill in the gaps in the target patch using the exemplar
            printl("Filling in information...\n");
            exmpFillPatch( iterImg, iterMask, sx, sy, px, py, patchSize );
            printl("Confidence %lf\n", iPixel(confidence, px, py) );

            cvCvtScale( iterImg, img, 1.0 );
            windowUpdate();

        }

        cvCvtScale( iterImg, img, 1.0 );
        cvCvtScale( iterMask, msk, 1.0 );
        cvThreshold(msk, msk, 128.0, 255.0, CV_THRESH_BINARY);


        cvReleaseImage( &iterImg );
        cvReleaseImage( &iterMask );
        cvReleaseImage( &sourceMask );
        cvReleaseImage( &confidence );
}
```