Mục lục

I – Các kiểu ảnh , các thao tác ảnh cơ bản trong Toolbox

1 . Ảnh được định chỉ số (Indexed Images)

- Một ảnh chỉ số bao gồm một ma trận dữ liệu X và ma trận bản đồ màu map . Ma trận dữ liệu có thể có kiểu thuộc lớp uint8,uint16 hoặc kiểu double . Ma trận bản đồ màu là một mảng mx3 kiểu double bao gồm các giá trị dấu phẩy động nằm giữa 0 và 1 . Mỗi hàng của bản đồ chỉ ra các giá trị mà : red , green và blue của một màu đơn . Một ảnh chỉ số sử dụng ánh xạ trực tiếp giữa giá trị của pixel ảnh tới giá trị trong bản đồ màu . Màu sắc của mỗi pixel ảnh được tính toán bằng cách sử dụng giá trị tương ứng của X ánh xạ tới một giá trị chỉ số của map . Giá trị 1 chỉ ra hàng đầu tiên , giá trị 2 chỉ ra hàng thứ hai trong bản đồ màu …
- Một bản đồ màu thường được chứa cùng với ảnh chỉ số và được tự động nạp cùng với ảnh khi sử dụng hàm **imread** để đọc ảnh .Tuy nhiên , ta không bị giới hạn khi sử dụng bản đồ màu mặc định , ta có thể sử dụng mất kì bản đồ màu nào . Hình sau đây minh hoạ cấu trúc của một ảnh chỉ số . Các pixel trong ảnh được đại diện bởi một số nguyên ánh xạ tới một giá trị tương ứng trong bản đồ màu .

(ẢNH)

Lớp và độ lệch của bản đồ màu (Colormap Offsets)

- Quan hệ giữa giá trị trong ma trận ảnh và giá trị trong bản đồ màu phụ thuộc vào kiểu giá trị của các phần tử ma trận ảnh . Nếu các phần tử ma trận ảnh thuộc kiểu double , giá trị 1 sẽ tương ứng với giá trị trong hàng thứ nhất của bản đồ màu , giá trị 2 sẽ tương ứng

với giá trị trong hàng thứ 2 của bản đồ màu … Nếu các phần tử của ma trận ảnh thuộc kiểu uint8 hay uint16 sẽ có một độ lệch (offset) – giá trị 0 trong ma trận ảnh sẽ tương ứng với giá trị trong hàng đầu tiên của bản đồ màu , giá trị 1 sẽ tương ứng với giá trị trong hàng thứ 2 của bản đồ màu ….

- Độ lệch cũng được sử dụng trong việc định dạng file ảnh đồ hoạ để tăng tối đa số lượng màu sắc có thể được trợ giúp .

Giới hạn trong việc trợ giúp ảnh thuộc lớp unit16

- Toolbox xử lý ảnh của Matlab trợ giúp có giới hạn ảnh chỉ số thuộc lớp uint16 . Ta có thể đọc những ảnh đó và hiển thị chúng trong Matlab nhưng trước khi xử lý chúng , ta phải chuyển đổi chúng sang kiểu uint8 hoặc double . Để chuyển đổi (convert) tới kiểu double ta dùng hàm **im2double** . Để giảm số lượng màu của ảnh xuống 256 màu (uint8) sử dụng hàm **imapprox .**

2. Ảnh cường độ (Intensity Images)

- Một ảnh cường độ là một ma trận dữ liệu ảnh I mà giá trị của nó đại diện cho cường độ trong một số vùng nào đó của ảnh . Matlab chứa một ảnh cường độ như một ma trận đơn , với mỗi phần tử của ma trận tương ứng với một pixel của ảnh . Ma trận có thể thuộc lớp double , uint8 hay uint16 . Trong khi ảnh cường độ hiếm khi được lưu với bản đồ màu , Matlab sử dụng bản đồ màu để hiển thị chúng .

- Những phần tử trong ma trận cường độ đại diện cho các cường độ khác nhau hoặc độ xám . Những điểm có cường độ bằng 0 thường được đại diện bằng màu đen và cường độ 1,255 hoặc 65535 thường đại diện cho cường độ cao nhất hay màu trắng .

3. Ảnh nhị phân (Binary Images)

-Trong một ảnh nhị phân , mỗi pixel chỉ có thể chứa một trong hai giá trị nhị phân 0 hoặc 1 . Hai giá trị này tương ứng với bật hoặ tắt (on hoặc off) . Một ảnh nhị phân được lưu trữ như một mảng lôgíc của 0 và 1 .

4. Ảnh RGB (RGB Images)

- Một ảnh RGB - thường được gọi là *true-color* , được lưu trữ trong Matlab dưới dạng một mảng dữ liệu có kích thước 3 chiều mxnx3 định nghĩa các giá trị màu red, green và blue cho mỗi pixel riêng biệt . Ảnh RGB không sử dụng palette . Màu của mỗi pixel được quyết định bởi sự kết hợp giữa các giá trị R,G,B (Red, Green , Blue) được lưu trữ trong một mặt phẳng màu tại vị trí của pixel . Định dạng file đồ hoạ lưu trữ ảnh RGB giống

như một ảnh 24 bít trong đó R,G,B chiếm tương ứng 8 bít một . Điều này cho phép nhận được 16 triệu màu khác nhau .

- Một mảng RGB có thể thuộc lớp double , uint8 hoặc uint16 . Trong một mảng RGB thuộc lớp double , mỗi thành phần màu có giá trị giữa 0 và 1 . Một pixel mà thành phần màu của nó là (0,0,0) được hiển thị với màu đen và một pixel mà thành phần màu là (1,1,1) được hiển thị với màu trắng . Ba thành phần màu của mỗi pixel được lưu trữ cùng với chiều thứ 3 của mảng dữ liệu . Chẳng hạn , giá trị màu R,G,B của pixel (10,5) được lưu trữ trong RGB(10,5,1) , RGB(10,5,2) và RGB(10,5,3) tương ứng .
- Để tính toán màu sắc của pixel tại hàng 2 và cột 3 chẳng hạn , ta nhìn vào bộ ba giá trị được lưu trữ trong (2,3,1:3) . Giả sử (2,3,1) chứa giá trị 0.5176 ; (2,3,2) chứa giá trị 0.1608 và (2,3,3) chứa giá trị 0.0627 thì màu sắc của pixel tại (2,3) sẽ là (0.5176,0.1608,0.0627)
- Để minh hoạ xa hơn khái niệm ba mặt phẳng màu riêng biệt được sử dụng trong một ảnh RGB , đoạn mã sau đây tạo một ảnh RGB đơn giản chứa các vùng liên tục của R,G,B và sau đó tạo một ảnh cho mỗi mặt phẳng riêng của nó (R,G,B) . Nó hiển thị mỗi mặt phẳng màu riêng rẽ và cũng hiển thị ảnh gốc .


```
RGB=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
R=RGB(:,:,1);
G=RGB(:,:,2);
B=RGB(:,:,3);
imshow(R)
figure, imshow(G)
figure, imshow(B)
figure, imshow(RGB)
```


Các mặt phẳng màu riêng rẽ của một ảnh RGB

- Chú ý rằng mỗi mặt phẳng màu riêng rẽ chứa một khoảng trắng . Khoảng trắng tương ứng với giá trị cao nhất của mỗi màu riêng rẽ . Chẳng hạn trong ảnh mặt phẳng R , vùng trắng đại diện cho sự tập trung cao nhất của màu đỏ thuần khiết . Nếu R được trộn với G hoặc B ta sẽ có màu xám . Vùng màu đen trong ảnh chỉ ra giá trị của pixel mà không chứa màu đỏ R=0. Tương tự cho các mặt phẳng màu G và B .

5. Mảng ảnh nhiều khung hình (Multiframe Image Arrays)

- Với một vài ứng dụng , ta có thể cần làm việc với một tập hợp các ảnh quan hệ với thời gian hoặc khung nhìn như MRI hay khung hình phim .

- Toolbox xử lý ảnh trong Matlab cung cấp sự trợ giúp cho việc lưu trữ nhiều ảnh trong cùng một mảng . Mỗi ảnh được gọi là một khung hình (Frame) . Nếu một mảng giữ nhiều frame , chúng được nối theo 4 chiều . Chẳng hạn , một mảng với năm ảnh có kích thước 400x300 sẽ là một mảng có kích thước 400x300x3x5 . Một ảnh chỉ số hoặc ảnh cường độ nhiều khung tương tự sẽ là 400x300x1x5 .

- Sử dụng lệnh **cat** để chứa các ảnh riêng rẽ trong một mảng nhiều khung hình . Chẳng hạn , nếu ta có một nhóm các ảnh A1,A2,A3,A4 và A5 , ta có thể chứa chúng trong một mảng duy nhất sử dụng

A=cat(4,A1,A2,A3,A4,A5)

- Ta cũng có thể trích các khung hình từ một ảnh nhiều khung hình . Chẳng hạn , nếu ta có một ảnh nhiều khung hình MULTI , lệnh sau đây sẽ trích ra khung hình thứ 3

FRM3=MULTI(: , : , : , 3)

- Ghi nhớ rằng , trong một mảng ảnh nhiều khung hình , mỗi ảnh phải có cùng kích thước và có cùng số mặt phẳng . Trong một ảnh chỉ số nhiều khung , mỗi ảnh phải sử dụng cùng một bản đồ màu

Sự trợ giúp giới hạn với ảnh nhiều khung

- Nhiều hàm trong toolbox hoạt động chỉ trên 2 hoặc 3 chiều đầu tiên . Ta có thể sử dụng chiều thứ 4 với những hàm này nhưng ta phải xử lý mỗi khung hình một cách độc lập . Chẳng hạn , lời gọi hàm sau sẽ hiển thị khung hình thứ 7 trong một mảng MULTI

imshow(MULTI(: , : , : , 7))

- Nếu ta truyền một mảng vào hàm và mảng có nhiều chiều hơn số chiều mà hàm đã được thiết kế để hoạt động , kết quả có thể không đoán trước được . Trong một số trường hợp , hàm đơn giản chỉ xử lý khung hình đầu tiên nhưng trong các trường hợp khác , sự hoạt động không tạo ra kết quả nào có ý nghĩa .

🗁- Các hàm chuyển đổi kiểu ảnh

- Với các thao tác nhất định , sẽ thật hữu ích khi có thể chuyển đổi ảnh từ dạng này sang dạng khác . Chẳng hạn , nếu ta muốn lọc một màu ảnh được lưu trữ dưới dạng ảnh chỉ số , đầu tiên , ta nên chuyển đổi nó thành dạng ảnh RGB . Khi ta áp dụng phép lọc tới ảnh RGB , Matlab sẽ lọc giá trị cường độ trong ảnh tương ứng . Nếu ta cố gắng lọc ảnh chỉ số

, Matlab đơn giản chỉ áp đặt phép lọc tới ma trận ảnh chỉ số và kết quả sẽ không có ý nghĩa

Chú ý : Khi convert một ảnh từ dạng này sang dạng khác , ảnh kết quả có thể khác ảnh ban đầu . Chẳng hạn , nếu ta convert một ảnh màu chỉ số sang một ảnh cường độ , kết quả ta sẽ thu được một ảnh đen trắng .

- Danh sách sau đây sẽ liệt kê các hàm được sử dụng trong việc convert ảnh :

+ **dither :** Tạo một ảnh nhị phân từ một ảnh cường độ đen trắng bằng cách trộn , tạo một ảnh chỉ số từ một ảnh RGB bằng cách trộng (dither)

+ **gray2id :** Tạo một ảnh chỉ số từ một ảnh cường độ đen trắng .

+ **grayslice :** Tạo một ảnh chỉ số từ một ảnh cường độ đen trắng bằng cách đặt ngưỡng

+ **im2bw :** Tạo một ảnh nhị phân từ một ảnh cường độ , ảnh chỉ số hay ảnh RGB trên cơ sở của ngưỡng ánh sáng .

+ **ind2gray :** Tạo một ảnh cường độ đen trắng từ một ảnh chỉ số

+ **ind2rgb :** Tạo một ảnh RGB từ một ảnh chỉ số

+ **mat2gray :** Tạo một ảnh cường độ đen trắng từ dữ liệu trong một ma trận bằng cách lấy tỉ lệ giữ liệu

+ **rgb2gray :** Tạo một ảnh cường độ đen trắng từ một ảnh RGB

+ **rgb2ind :** Tạo một ảnh chỉ số từ một ảnh RGB

- Ta cũng có thể thực hiện các phép chuyển đổi kiểu chỉ sử dụng cú pháp của Matlab . Chẳng hạn , ta có thể convert một ảnh cường độ sang ảnh RGB bằng cách ghép nối 3 phần copy của ma trận ảnh gốc giữa 3 chiều :

RGB=cat(3,I,I,I);

- Ảnh RGB thu được có các ma trận đồng nhất cho các mặt phẳng R,G,B vì vậy ảnh hiển thị giống như bóng xám .

- Thêm vào những công cụ chuyển đổi chuẩn đã nói ở trên , cũng có một số hàm mà trả lại kiểu ảnh khác như một phần trong thao tác mà chúng thực hiện . Xem thêm Help Online

Chuyển đổi không gian màu

- Toolbox xử lý ảnh biểu diễn màu sắc như các giá trị RGB (trực tiếp trong ảnh RGB hoặc gián tiếp trong ảnh chỉ số) . Tuy nhiên , có các phương pháp khác cho việc biểu diễn màu sắc . Chẳng hạn , một màu có thể được đại diện bởi các giá trị **hue , saturation**

và các giá trị thành phần (HSV) . Các phương pháp khác cho việc biểu diễn màu được gọi là không gian màu .

- Toolbox cung cấp một tập các thủ tục để chuyển đổi giữa các không gian màu . Các hàm xử lý ảnh tự chúng coi dữ liệu màu sắc dưới dạng RGB tuy nhiên , ta có thể xử lý một ảnh mà sử dụng các không gian màu khác nhau bằng cách chuyển đổi nó sang RGB sau đó chuyển đổi ảnh đã được xử lý trở lại không gian màu ban đầu .

🗁- Đọc và ghi dữ liệu ảnh

- Phần này sẽ giới thiệu cách đọc và ghi dữ liệu ảnh

1. Đọc một ảnh đồ hoạ

- Hàm imread đọc một ảnh từ bất kì định dạng nào được trợ giúp trong bất kì chiều sâu bit nào được trợ giúp . Hầu hết các file ảnh sử dụng 8 bít để chứa giá trị của pixel . Khi chúng được đọc vào bộ nhớ , Matlab chứa chúng dưới dạng uint8 . Với các file trợ giúp 16 bít dữ liệu , PNG và TIFF , Matlab chứa chúng dưới dạng uint16

Chú ý : Với ảnh chỉ số , imread luôn luôn đọc bản đồ màu vào trong một chuỗi thuộc lớp double , thậm chí mảng ảnh tự nó thuộc lớp uint8 hay uint16

- Chẳng hạn , đoạn mã sau sẽ đọc một ảnh RGB vào không gian làm việc của Matlab lưu trong biến RGB

RGB=imread('football.jpg');

- Trong ví dụ này , imread sẽ nhận ra định dạng file để sử dụng từ tên file . Ta cũng có thể chỉ ra định dạng file như một tham số trong hàm imread . Matlab trợ giúp rất nhiều định dạng đồ hoạ thông dụng chẳng hạn : BMP , GIF , JPEG , PNG , TIFF … Để biết thêm các kiểu gọi hàm và tham số truyền vào , xem trợ giúp online của Matlab .

Đọc nhiều ảnh từ một file đồ hoạ

- Matlab trợ giúp một số định dạng file đồ hoạ chẳng hạn như : HDF và TIFF , chúng chứa nhiều ảnh . Theo mặc định , imread chỉ trợ giúp ảnh đầu tiên trong file . Để nhập thêm các ảnh từ file , sử dụng cú pháp được trợ giúp bởi định dạng file . Chẳng hạn , khi được sử dụng với TIFF , ta có thể sử dụng một giá trị chỉ số với imread để chỉ ra ảnh mà ta muốn nhập vào . Ví dụ sau đây đọc một chuỗi 27 ảnh từ một file TIFF và lưu những ảnh anỳ trong một mảng 4 chiều . Ta có thể sử dụng hàm **iminfo** để xem bao nhiêu ảnh đã được lưu trữ trong file :

mri = uint8(zeros(128,128,1,27)); % preallocate 4-D array

```
for frame=1:27
   [mri(:,:,:,frame),map] = imread('mri.tif',frame);
end
```

- Khi file chứa nhiều ảnh theo một số kiểu nhất định chẳng hạn theo thứ tự thời gian , ta có thể lưu ảnh trong Matlab dưới dạng mảng 4 chiều . Tất cả các ảnh phải có cùng kích thước .

2. Ghi một ảnh đồ hoạ

- Hàm imwrite sẽ ghi một ảnh tới một file đồ hoạ dưới một trong các định dạng được trợ giúp . Cấu trúc cơ bản nhất của imwrite sẽ yêu cầu một biến ảnh và tên file . Nếu ta gộp một phần mở rộng trong tên file , Matlab sẽ nhận ra định dạng mong muốn từ nó . Ví dụ sau tải một ảnh chỉ số X từ một file Mat với bản đồ màu kết hợp với nó map sau đó ghi ảnh xuống một file bitmap .

```
load clown
whos
  Name        Size            Bytes  Class
```



```
  X           200x320        512000  double array
  caption       2x1              4  char array
  map          81x3             1944  double array

Grand total is 64245 elements using 513948 bytes

imwrite(X,map,'clown.bmp')
```

Chỉ ra định dạng phụ - Tham số đặc biệt

- Khi sử dụng imwrite với một số định dạng đồ hoạ , ta có thể chỉ ra các tham số phụ . Chẳng hạn , với định dạng PNG ta có thể chỉ ra độ sâu bít như một tham số phụ . Ví dụ sau sẽ chi một ảnh cường độ I với một file ảnh 4 bít PNG


```
imwrite(I,'clown.png','BitDepth',4 );
```

- Để biết thêm các cấu trúc khác của hàm xem phần trợ giúp trực tuyến của Matlab .

Đọc và ghi ảnh nhị phân theo định dạng 1 bít

- Trong một số định dạng file , một ảnh nhị phân có thể được lưu trong một định dạng 1 bít . Nếu định dạng file trợ giúp nó ,Matlab ghi ảnh nhị phân như ảnh 1 bít theo mặc định

. Khi ta đọc một ảnh nhị phân với định dạng 1 bít , Matlab đại diện nó trong không gian làm việc như một mảng lôgíc .

- Ví dụ sau đọc một ảnh nhị phân và ghi nó dưới dạng file TIFF . Bởi vì định dạng TIFF trợ giúp ảnh 1 bít , file được ghi lên đĩa theo định dạng 1 bít :

BW = imread('text.png');

imwrite(BW,'test.tif');

Để kiểm tra chiều sâu bít của file test.tif , gọi hàm iminfo và kiểm tra trường BitDepth của nó :

info = imfinfo('test.tif');

info.BitDepth

ans =

1

Chú ý : Khi gi file nhị phân , Matlab thiết lập trường ColorType thành 'grayscale'

Xem lớp lưu trữ của file

- Hàm imwrite sử dụng luật sau đây để quyết định lớp lưu trữ được sử dụng trong ảnh kết quả :

+ logical : Nếu định dạng ảnh ra (Output Image) được chỉ rõ là trợ giúp ảnh 1 bít , hàm imwrite tạo một file ảnh 1 bít . Nếu định dạng ảnh ra được chỉ rõ là không trợ giúp ảnh 1 bít (như JPEG) , hàm imwrite chuyển ảnh tới một ảnh thuộc lớp uint8

+ uint8 : Nếu định dạng ảnh ra được chỉ rõ là trợ giúp ảnh 8 bít , hàm imwrite tạo một ảnh 8 bít

+uint16 : Nếu định dạng ảnh ra được chỉ rõ trợ giúp ảnh 16 bít (PNG hoặc TIFF) , hàm imwrite tạo một ảnh 16 bít . Nếu định dạng ảnh ra không trợ giúp ảnh 16 bít , hàm chuyển đổi dữ liệu ảnh tới lớp uint8 và tạo một ảnh 8 bít .

+double : Matlab chuyển dữ liệu ảnh tới dạng uint8 và tạo một ảnh 8 bít bởi vì hầu hết các file ảnh sử dụng định dạng 8 bít .

2. Truy vấn một file đồ hoạ

- Hàm iminfo cho phép ta có thể nhận được thông tin về một file ảnh được trợ giúp bởi toolbox . Thông tin mà ta nhận được phụ thuộc vào kiểu của file nhưng nó luôn bao gồm những thông tin sau :

+ Tên của file ảnh

+ Định dạng file ảnh

+ Số version của định dạng file

+ Ngày sửa đổi file gần nhất

+ Kích thước file tính theo byte

+ Chiều rộng ảnh tính theo pixel

+ Chiều cao ảnh tính theo pixel

+ Số lượng bít trên một pixel

+ Kiểu ảnh : RGB, chỉ số …

🗁**- Chuyển đổi định dạng các file ảnh**

- Để thay đổi định dạng đồ hoạ của một ảnh , sử dụng hàm imread để đọc một ảnh và sau đó lưu nó với hàm imwrite đồng thời chỉ ra định dạng tương ứng .

- Để minh hoạ , ví dụ sau đây sử dụng hàm imread để đọc một file BMP vào không gian làm việc .Sau đó , hàm imwrite lưu ảnh này dưới định dạng PNG

```
bitmap = imread('mybitmap.bmp','bmp');
imwrite(bitmap,'mybitmap.png','png');
```


🗁**- Đọc và ghi ảnh DICOM**

- Toolbox xử lý ảnh bao gồm trợ giúp cho việc thao tác với ảnh số (Digital Imaging) và ảnh y học (Communication in Medicine) .

1. Đọc dữ liệu ảnh từ một file DICOM

- Để đọc một dữ liệu ảnh từ một file DICOM , sử dụng hàm đicomread . Hàm này đọc các file tuân theo đặc trưng DICOM nhưng có thể đọc được các file không theo chuẩn chung nào

- Ví dụ sau đây đọc một ảnh từ một file DICOM mẫu đi kèm với toolbox .

```
I = dicomread('CT-MONO2-16-ankle.dcm');
```


Để xem dữ liệu ảnh , sử dụng hàm hiển thị của toolbox – imshow hoặc imview (Do dữ liệu ảnh là số 16 bít có dấu , ta phải sử dụng cấu trúc tự chuyển đổi với mỗi hàm hiển thị)

```
imview(I,[])
```

2. Đọc Metadata từ một file DICOM

- Các file DICOM bao gồm các thông tin được gọi là Metadata . Những thông tin này mô tả tặc tính của dữ liệu ảnh nó nắm giữ như : kích thước , chiều , chiều sâu bít . Thêm vào đó , đặc trưng DICOM định nghĩa nhiều các trường metadata khác để mô tả các đặc tính

khác của dữ liệu như : cách thức được sử dụng để tạo dữ liệu , thiết lập thiết bị dùng để chụp ảnh , thông tin về việc nghiên cứu …Hàm dicomread có thể xử lý hầu hết tất cả các trường metadata được định nghĩa bởi đặc trưng DICOM (hay chuẩn DICOM)

- Để đọc metadata từ một file DICOM , sử dụng hàm dicominfo . Hàm này trả về một cấu trúc metadata mà mọi trường trong cấu trúc là một phần đặc trưng của metadata trong file DICOM đó .

```
info = dicominfo('CT-MONO2-16-ankle.dcm');

info =

                Filename: [1x47 char]
            FileModDate: '24-Dec-2000 19:54:47'
                FileSize: 525436
                  Format: 'DICOM'
           FormatVersion: 3
                   Width: 512
                  Height: 512
                BitDepth: 16
               ColorType: 'grayscale'
          SelectedFrames: []
              FileStruct: [1x1 struct]
         StartOfPixelData: 1140
    MetaElementGroupLength: 192
FileMetaInformationVersion: [2x1 double]
  MediaStorageSOPClassUID: '1.2.840.10008.5.1.4.1.1.7'
MediaStorageSOPInstanceUID: [1x50 char]
      TransferSyntaxUID: '1.2.840.10008.1.2'
  ImplementationClassUID: '1.2.840.113619.6.5'
                       .
                       .
                       .
```

Ta có thể sử dụng cấu trúc metadata được trả lại bởi hàm dicominfo để chỉ định file DICOM ta muốn đọc sử dụng hàm dicomread . Chẳng hạn , ta có thể sử dụng đoạn mã sau đây để đọc metadata từ một file DICOM mẫu và sau đó truyền metadata đó tới hàm dicomread để đọc ảnh từ file :

```
info = dicominfo('CT-MONO2-16-ankle.dcm');
I = dicomread(info);
```

3. Ghi dữ liệu lên một file DICOM

- Để ghi dữ liệu lên một file DICOM , sử dụng hàm dicomwrite . Ví dụ sau ghi một ảnh I tới file DICOM ankle.dcm

```
dicomwrite(I,'h:\matlab\tmp\ankle.dcm')
```

Ghi metadata lên một file DICOM

Khi ta ghi dữ liệu ảnh lên một file DICOM , hàm dicomwrite bao gồm một tập hợp nhỏ nhất của các trường trong metadata được yêu cầu bởi kiểu của đối tượng thông tin DICOM (IOD) mà ta đang tạo . dicomwrite trợ giúp 3 kiểu DICOM IOD :

+ Secondary capture (mặc định)
+ Magnetic resonance
+ Computed tomography

- Ta có thể chỉ rõ matadata nào ta muốn ghi lên file bằng cách truyền tới hàm dicomwrite một cấu trúc metadata nhận được từ hàm dicominfo

```
info = dicominfo('CT-MONO2-16-ankle.dcm');
I = dicomread(info);
dicomwrite(I,'h:\matlab\tmp\ankle.dcm',info)
```

- Trong trường hợp này , hàm dicomwrite ghi thông tin trong cấu trúc metadata info lên một file DICOM mới . Khi ghi dữ liệu tới file , có một số trường mà dicomwrite phải cập nhật . Chẳng hạn , dicomwrite phải cập nhật ngày tháng sửa đổi của file mới . Để minh hoạ , so sánh ngày sửa đổi của metadata gốc với ngày sửa đổi file trong file mới :

```
info.FileModDate

ans =

24-Dec-2000 19:54:47
```

Sử dụng dicominfo , đọc metadata từ một file mới được ghi , kiểm tra ngày sửa đổi file :

info2 = dicominfo('h:\matlab\tmp\ankle.dcm');

info2.FileModDate

ans =

16-Mar-2003 15:32:43

- Số học ảnh

- Số học ảnh sự ứng dụng của các phép toán số học chuẩn như : cộng , trừ , nhân , chia lên ảnh . Số học ảnh được sử dụng nhiều trong xử lý ảnh trong cả các bước ban đầu lẫn các thao tác phức tạp hơn . Chẳng hạn , trừ ảnh có thể được sử dụng để phát hiện sự khác nhau giữa hai hoặc nhiều ảnh của cùng một cảnh hoặc một vật .
- Ta có thể thực hiện số học ảnh sử dụng các toán tử số học của Matlab . Toolbox xử lý ảnh bao gồm một tập hợp các hàm ứng dụng các phép toán số học trên tất cả các con số không lấp đầy . Hàm số học của toolbox chấp nhận bất kì kiểu dữ liệu số nào bao gồm uint8 , uint16 hay double và trả lại ảnh kết quả trong cùng định dạng . Các hàm thực hiện các phép toán với độ chính xác kép trên từng phần tử nhưng không chuyển đổi ảnh tới giá trị chính xác kép trong không gian làm việc của Matlab . Sự tràn số được điều khiển tự động . Hàm sẽ cắt bỏ giá trị trả về để vừa với kiểu dữ liệu .

1 . Luật cắt bỏ trong số học ảnh

- Kết quả của số học nguyên có thể dễ dàng tràn số dùng cho lưu trữ . Chẳng hạn , giá trị cực đại ta có thể lưu trữ trong uint8 là 255 . Các phép toán số học có thể trả về giá trị phân số - không được biểu diễn bởi một chuỗi số nguyên .
- Các hàm số học ảnh sử dụng những luật này cho số học nguyên :

+ Giá trị vượt quá khoảng của kiểu số nguyên bị cắt bỏ tới khoảng đó

+ Giá trị phân số được làm tròn

Chẳng hạn , nếu dữ liệu có kiểu uint8 , kết quả trả về nếu lớn hơn 255 (bao gồm Inf) thì được gán là 255 .

2. Lời gọi lồng nhau tới hàm số học ảnh

- Ta có thể sử dụng các hàm số học ảnh kết hợp để thực hiện một chuỗi các phép toán . Chẳng hạn để tính giá trị trung bình của hai ảnh :

C=(A+B) /2

Ta có thể nhập vào như sau :

```
I = imread('rice.png');
I2 = imread('cameraman.tif');
K = imdivide(imadd(I,I2), 2); % not recommended
```

- Khi được sử dụng với kiểu uint8 hay uint16 , mỗi hàm số học cắt kết quả của nó trước khi truyền nó cho hàm thiếp theo . Sự cắt bỏ này có thể giảm đáng kể lượng thông tin trong ảnh cuối cùng . Một cách làm tốt hơn để thực hiện một chuỗi các tính toán là sử dụng hàm **imlincomb** . Hàm này thi hành tất cả các phép toán số học trong sự kết hợp tuyến tính của độ chính xác kép và chỉ cắt bỏ kết quả cuối cùng :

```
K = imlincomb(.5,I,.5,I2); % recommended
```

🗁- Hệ thống toạ độ

- Vị trí trong một ảnh có thể được biểu diễn trong các hệ thống toạ độ khác nhau phụ thuộc vào ngữ cảnh . Có hai hệ thống toạ độ trong Matlab là : hệ thống toạ độ pixel và hệ thống toạ độ không gian .

1. Toạ độ pixel

- Nhìn chung , phương pháp thuận tiện nhất cho việc biểu diễn vị trí trong một ảnh là sử dụng toạ độ pixel . Trong hệ toạ độ này , ảnh được xử lý như một lưới của các phần tử riêng biệt được đánh thứ tự từ đỉnh tới đáy và từ trái sang phải .
- Với toạ độ pixel , thành phần đầu tiên r (hàng) được tăng từ khi đi từ trên xuống dưới trong khi c (cột) được tăng khi đi từ trái sang phải . Hệ toạ độ pixel là giá trị nguyên và khoảng giữa 1 và chiều dài của hàng hay cột
- Có một tương ứng 1-1 giữa toạ độ pixel và toạ độ Matlab sử dụng để mô tả ma trận . Sự tương ứng này tạo một quan hệ gữa ma trận dữ liệu ảnh và cách ảnh được hiển thị . Chẳng hạn , dữ liệu cho pixel trên hàng thứ 5 , cột thứ 2 được lưu trữ tại phần từ (5,2) của ma trận .

2. Toạ độ không gian

- Trong hệ toạ độ pixel , một pixel được xử lý như một đơn vị riêng rẽ được phân biệt duy nhất bởi một cặp toạ độ chẳng hạn (5,2) . Từ quan điểm này , một vị trí chẳng hạn (5,2) không có ý nghĩa . Tuy nhiên , sẽ hữu ích khi nghĩ đến một pixel như một miếng vá hình vuông . Từ quan điểm này , một vị trí chẳng hạn (5.3,2.2) là có nghĩa và được phân biệt với (5,2) . Trong toạ độ không gian này , vị trí trong một ảnh được định vị trên một

mặt phẳng và chúng được mô tả bằng một cặp x và y (không phải r và c như toạ độ pixel) .

- Hệ toạ độ không gian này gần tương ứng với hệ toạ độ pixel trong một chừng mực nào đó . Chẳng hạn , toạ độ không gian của điểm gữa của bất kì pixel nào được phân biệt với toạ độ pixel của pixel đó . Cũng có một vài khác biệt , tuy nhiên , trong toạ độ pixel , góc trên trái của một ảnh là (1,1) trong khi trong toạ độ không gian , vị trí này mặc định là (0.5,0.5) . Sự khác nhau này là do hệ toạ độ pixel là rời rạc trong khi toạ độ không gian là liên tục . Cũng vậy , góc trên trái luôn là (1,1) trong hệ pixel , nhưng ta có thể chỉ ra một điểm gốc không chính quy cho hệ toạ độ không gian . Một sự khác biệt dễ gây nhầm lẫn nữa là quy ước : thứ tự của các thành phần nằm ngang và thẳng đứng được phục vụ cho kí hiệu của hai hệ thống . Như đã đề cập trước đây , toạ độ pixel được đại diện bởi một cặp (r,c) trong khi toạ độ không gian được biểu diễn bởi (x,y) . Khi cú pháp cho một hàm sử dụng r và c , nó tham chiếu đến hệ toạ độ pixel . Khi cú pháp sử dụng x, y nó đang ngầm định sử dụng hệ toạ độ không gian .

Sử dụng hệ toạ độ không gian không chính quy

- Theo mặc định , toạ độ không gian của một ảnh tương ứng với toạ độ pixel . Chẳng hạn , điểm giữa của pixel tại (5,3) có một toạ độ không gian là x=3, y=5 (nhớ rằng thứ tự của toạ độ bị đảo ngược) . Sự tương ứng này làm đơn giản nhiều hàm trong toolbox . Một vài hàm ban đầu làm việc với toạ độ không gian hơn là toạ độ pixel nhưng khi ta đang sử dụng toạ độ không gian theo mặc định , ta có thể chỉ ra vị trí trong toạ độ pixel
- Trong một số tình huống , tuy nhiên , ta có thể muốn sử dụng toạ độ không gian không chính quy (không mặc định) . Chẳng hạn , ta có thể chỉ ra góc trên trái của một ảnh tại điểm (19.0,7.5) thay cho (0.5,0,5) . Nếu ta gọi một hàm mà trả về toạ độ cho ảnh này , toạ độ được trả lại sẽ là giá trị trong hệ toạ độ không chính quy .
- Để thành lập toạ độ không chính quy , ta có thể chỉ ra Xdata và Ydata của một ảnh khi hiển thị nó . Những thuộc tính này là véc tơ 2 phần tử để điều khiển khoảng của góc quay của một ảnh . Theo mặc định , một ảnh A , Xdata là [1 size(A,2)] , và Ydata là [1 size(A,1)] . Chẳng hạn , nếu A là 100 hàng x 200 cột , giá trị Xdata mặc định là [1 200] và Ydata là [1 100] . Những giá trị trong những véc tơ này thực là toạ độ của điểm giữa của pixel đầu tiên và cuối cùng vì vậy , khoảng toạ độ thực được quay là lớn hơn , chẳng hạn , nếu Xdata là [1 200] thì khoảng của x là [0.5 200.5] . Những lệnh sau hiển thị một ảnh sử dụng các giá trị không mặc định Xdata và Ydata :

```
A = magic(5);
x = [19.5 23.5];
y = [8.0 12.0];
image(A,'XData',x,'YData',y), axis image, colormap(jet(25))
```

🗁- Hiển thị ảnh

1. Dùng hàm imview

- Để hiển thị một ảnh sử dụng hàm imview , dùng hàm imview , chỉ rõ ảnh mà ta muốn hiển thị . Ta có thể sử dụng imview để hiển thị một ảnh mà đã được nhập vào trong không gian làm việc của Matlab

```
moonfig = imread('moon.tif');
imview(moonfig);
```

Ta cũng có thể chỉ định tên của file ảnh như trong ví dụ sau :

```
imview('moon.tif');
```

- File ảnh phải có mặt trong thư mục hiện tại hoặc trong đường dẫn của Matlab . Cấu trúc này có thể hữu ích cho việc quét qua nhiều ảnh . Tuy nhiên , lưu ý , khi sử dụng cấu trúc này , dữ liệu ảnh không được lưu trong không gian làm việc của Matlab .
- Nếu ta gọi hàm imview mà không chỉ ra mất kì tham số nào , nó sẽ hiển thị một hộp chọn file cho phép ta chỉ ra tên file muốn hiển thị .

Xem nhiều ảnh

- Nếu ta chỉ ra một file mà chứa nhiều ảnh , hàm imview chỉ hiển thị ảnh đầu tiên trong file đó . Để xem tất cả các ảnh trong file , sử dụng hàm imread để nhập mỗi ảnh vào trong không gian làm việc của Matlab sau đó gọi hàm imview nhiều lần để hiển thị mỗi ảnh riêng biệt

2. Dùng hàm imshow

- Để xem ảnh , ta có thể sử dụng hàm imshow thay cho imview . Ta sử dụng imshow để hiển thị một ảnh đã được nhập vào trong không gian làm việc như ví dụ sau :

```
moon = imread('moon.tif');
imshow(moon);
```

Ta cũng có thể chỉ ra tên của file ảnh như một tham số truyền vào cho hàm như ví dụ sau :

```
imshow('moon.tif');
```

Khi sử dụng cấu trúc này thì dữ liệu ảnh không được nhập vào trong không gian làm việc . Tuy nhiên ,ta có thể mang ảnh vào trong không gian làm việc bằng cách sử dụng hàm getimage . Hàm này sẽ nhận dữ liệu ảnh từ handle của một đối tượng ảnh hiện tại . Chẳng hạn :

moon = getimage;

Sẽ gán dữ liệu ảnh từ moon.tif vào biến moon .

II . Xử lý trên cơ sở vùng chọn (Region – Based Processing)

- Trong phần này , ta sẽ xem xét những khía cạnh sau :

+ Bảng thuật ngữ : Cung cấp các thuật ngữ được sử dụng trong các phép xử lý

+ Chỉ định rõ một vùng ta quan tâm : Mô tả làm sao để chỉ ra một vùng quan tâm sử dụng hàm **roipoly**

+ Lọc một vùng : Diễn tả làm sao để áp đặt một phép lọc lên mộg vùng nhất định của ảnh sử dụng hàm **roifilt2**

+ Tô đầy một vùng : Sử dụng hàm roifill để tô đầy một vùng đã chọn

1. Bảng các thuật ngữ :

Tên thuật ngữ	Diễn tả
Binary mask	Ảnh nhị phân với cùng kích thước như ảnh ta muốn xử lý . Mặt nạ chứa giá trị 1 cho tất cả các pixel thuộc trong vùng ta quan tâm và chứa giá trị 0 cho các vùng khác
Filling a region	Là quá trình xử lý điền đầy (hay tô màu) một vùng nhất định bằng cách nội suy giá trị pixel từ viền của vùng . Quá trình xử lý này có thể được sử dụng để tạo một đối tượng trong một ảnh dường như biến mất khi chúng được thay thế với giá trị được trộn với vùng nền
Filtering a region	Áp đặt một phép lọc lên một vùng nhất định . Chẳng hạn , ta có thể áp đặt một sự

	phép lọc điều chỉnh cường độ lên một vùng của ảnh
Nội suy	Phương pháp được sử dụng để ước lượng một giá trị ảnh ở một vị trí nhất định giữa các pixel của ảnh
Masked filtering	Thao tác chỉ áp đặt một phép lọc lên một vùng quan tâm trong một ảnh được phân biệt bằng mặt nạ nhị phân . Giá trị được lọc được trả lại cho các pixel mà mặt nạ nhị phân chứa giá trị 1 , giá trị không được lọc được trả về cho các pixel mà mặt nạ nhị phân chứa giá trị 0

2. Chỉ định một vùng quan tâm trên ảnh

- Một vùng quan tâm là một phần của ảnh mà ta muốn lọc hoặc thi hành các thao tác khác trên nó . Ta định nghĩa một vùng quan tâm bằng cách tạo ra một mặt nạ nhị phân , đó là một ảnh nhị phân có cùng kích thước với ảnh ta muốn xử lý . Mặt nạ chứa giá trị 1 cho tất cả các pixel nằm trong vùng quan tâm và chứa giá trị 0 cho các pixel ở những vùng khác .

a - Chọn một hình đa giác

- Ta có thể sử dụng hàm **roipoly** để chỉ ra một vùng hình đa giác quan tâm . Nếu ta gọi hàm roipoly không có tham số , con trỏ thay đổi thành hình chữ thập khi nó đi qua ảnh đang được hiển thị trên trục hiện tại . Sau đó , ta có thể chỉ ra đỉnh của đa giác bằng cách kích trỏ chuột trên ảnh . Khi thực hiện xong việc chọn các đỉnh , nhấn phím Enter để kết thúc . hàm roipoly trả về một ảnh nhị phân có cùng kích thước với ảnh gốc chứa giá trị 1 trong vùng được chọn và 0 ở phần còn lại .

```
I = imread('pout.tif');
imshow(I)
BW = roipoly;
```

- Ta cũng có thể sử dụng hàm roipoly mà không tương tác . Cú pháp của hàm này như sau :

```
BW = roipoly(I,c,r)
BW = roipoly(I)
```

BW = roipoly(x,y,I,xi,yi)

[BW,xi,yi] = roipoly(...)

[x,y,BW,xi,yi] = roipoly(...)

Diễn giải

+ BW=roipoly(I,c,r) trả lại một vùng quan tâm được lựa chọn bởi hình đa giác được mô tả bởi véc tơ c và r . BW là một ảnh nhị phân có cùng kích thước với ảnh ban đầu .

+ BW=roipoly(I) : Hiển thị ảnh I trên màn hình và để ta chỉ ra vùng chọn với trỏ chuột . Nếu bỏ qua I , roipoly hoạt động trên ảnh ở trục hiện tại . Sử dụng click chuột để thêm các đỉnh tới đa giác . Bằng cách nhấn Backspace hoặc Delete để xoá các đỉnh đã chọn trước đó . Khi chọn xong , nhấn Enter để kết thúc việc chọn

+BW=roipoly(x,y,I,xi,yi) : Sử dụng véc tơ x và y để tạo lập hệ toạ độ không gian không mặc định . xi ,yi là véc tơ có cùng chiều dài chỉ ra các đỉnh của đa giác như các vị trí trong hệ toạ độ này .

+ [BW , xi,yi] = roipoly(…) trả lại toạ độ của đa giác trong xi , yi . Chú ý rằng roipoly luôn luôn tạo ra một đa giác kín .

+ [x,y,BW,xi,yi]=roipoly(…) trả lại XData và Ydata trong x và y , mặt nạ ảnh trong BW và đỉnh của đa giác trong xi và yi

- Nếu roipoly được gọi không có tham số ra , ảnh kết quả sẽ được hiển thị trên một hình mới .

Lớp trợ giúp

- Ảnh đầu vào I có thể thuộc lớp uint8 , uint16 hoặc double . Ảnh ra BW thuộc lớp logical . Tất cả các đầu vào và ra khác thuộc lớp double

Ví dụ

```
I = imread('eight.tif');
c = [222 272 300 270 221 194];
r = [21 21 75 121 121 75];
BW = roipoly(I,c,r);
imshow(I)
figure, imshow(BW)
```

b – Các phương pháp lựa chọn khác

- Hàm roipoly cung cấp một cách dễ dàng để tạo một mặt nạ nhị phân . Tuy nhiên , ta có thể sử dụng bất cứ ảnh nhị phân nào làm mặt nạ miễn là ảnh đó có cùng kích thước với

ảnh đang được lọc . Chẳng hạn , giả sử ta muốn lọc ảnh cường độ I , chỉ lọc những pixel mà giá trị của nó lớn hơn 0.5 . Ta có thể tạo một mặt nạ tương ứng với lệnh :

BW = (I > 0.5);

- Ta cũng có thể sử dụng hàm poly2mask để tạo một mặt nạ nhị phân . Không giống hàm roipoly , poly2mask không yêu cầu một ảnh vào . Ngoài ra , ta còn có thể sử dụng hàm roicolor để định nghĩa một vùng quan tâm trên cơ sở một màu hoặc một vùng cường độ nào đó . Cú pháp của hàm này như sau :

BW = roicolor(A,low,high)

BW = roicolor(A,v)

Diễn giải

- Hàm roicolor lựa chọn một vùng quan tâm trong một ảnh chỉ số hoặc ảnh cường độ và trả về một ảnh nhị phân

+ BW=roicolor(A,low,high) : trả về một vùng quan tâm được lựa chọn với những pixel nằm trong khoảng giữa low và high trong bản đồ màu sắc

BW = (A >= low) & (A <= high)

BW là một ảnh nhị phân

+BW = roicolor(A,v): Trả về một vùng quan tâm được lựa chọn với những pixel trong A mà hợp với các giá trị trong véc tơ v . BW là một ảnh nhị phân .

Lớp trợ giúp

- Ảnh vào A phải thuộc lớp numeric . Ảnh ra BW thuộc lớp logical

Ví dụ

```
I = imread('rice.png');
BW = roicolor(I,128,255);
imshow(I);
figure, imshow(BW)
```


2. Lọc một vùng

- Ta có thể sử dụng hàm roifilt2 để xử lý một vùng quan tâm . Khi ta gọi hàm roifilt2 , ta chỉ ra một ảnh cường độ , một mặt nạ nhị phân và một bộ lọc . Hàm roifilt2 lọc ảnh vào và trả về một ảnh mà chứa các giá trị đã được lọc cho các pixel mà mặt nạ nhị phân chứa 1 và các giá trị cho các pixel mà mặt nạ nhị phân chứa 0 . Kiểu lọc này được gọi là lọc có mặt nạ . Cú pháp của hàm roifilt2 như sau :

J = roifilt2(h,I,BW)

J = roifilt2(I,BW,fun)

J = roifilt2(I,BW,fun,P1,P2,...)

Diễn giải

+ J=roifilt2(h,I,BW) : Lọc dữ liệu trong I với bộ lọc tuyến tính hai chiều h . BW là một ảnh nhị phân có cùng kích thước với ảnh gốc và được sử dụng như một mặt nạ cho việc lọc . Hàm roifilt2 trả về một ảnh chứa các giá trị được lọc ở trong vùng chọn (hay các pixel mà BW có giá trị 1)

+ J=roifilt2(I,BW,fun) : Xử lý dữ liệu trong I sử dụng hàm fun . Kết quả , J chứa các giá trị đã được tính toán cho các pixel mà tại đó BW chứa 1 và giá trị thực trong I cho các pixel mà tại đó BW chứa giá trị 0 .

+ J=roifilt2(I,BW,fun,P1,P2…) Truyền thêm các tham số P1,P2 cho hàm fun

Lớp trợ giúp

- Với cấu trúc mà có chứa bộ lọc h , ảnh vào I có thể thuộc lợp uint ,uint16 hoặc double và ảnh ra J có cùng lớp với ảnh vào .

Ví dụ


```
I = imread('eight.tif');
c = [222 272 300 270 221 194];
r = [21 21 75 121 121 75];
BW = roipoly(I,c,r);
h = fspecial('unsharp');
J = roifilt2(h,I,BW);
imshow(J), figure, imshow(J)
```

Để hiểu rõ hơn về áp dụng bộ lọc cho một vùng , ta hãy xem xét cụ thể một ví dụ sau :

1. Đọc một ảnh :

```
I = imread('pout.tif');
```


2. Tạo mặt nạ : Sử dụng chuột để tạo vùng chọn và lấy mặt nạ trả về qua BW

3. Tạo bộ lọc

```
h = fspecial('unsharp');
```

4. Gọi hàm roifilt2 , chỉ ra ảnh cần lọc , mặt nạ và bộ lọc

```
I2 = roifilt2(h,I,BW);
imshow(I)
figure, imshow(I2)
```

Chỉ định thao tác lọc

- Hàm roifilt2 cũng cho phép ta chỉ định một hàm riêng để tao tác trên vùng quan tâm . Ví dụ sau sử dụng hàm imadjust để làm sáng một phần của ảnh

1. Đọc một ảnh

I = imread('cameraman.tif');

2. Tạo mặt nạ : Trong ví dụ này , mặt nạ là một ảnh nhị phân chứa chữ . Ảnh mặt nạ phải được cắt để có cùng kích thước với ảnh được lọc

BW = imread('text.png');

mask = BW(1:256,1:256);

3. Tạo bộ lọc

f = inline('imadjust(x,[],[],0.3)');

4. Gọi hàm roifilt2 , chỉ ra ảnh được lọc , mặt nạ và bộ lọc . Ảnh kết quả I2 có chữ như bị khắc trên nó

I2 = roifilt2(I,mask,f);

imshow(I2)

3. Điền đầy một vùng

- Ta có thể sử dụng hàm roifill để điền đầy một vùng quan tâm , tuyến tính hóa từ biên của vùng . Hàm này sẽ hữu ích cho việc chỉnh sửa ảnh , bao gồm xoá các chi tiết ngoài hoặc giả tạo

- Hàm roifill thực thi việc điền đầy sử dụng một phương pháp tuyến tính hoá trên cơ sở của phương trình Laplace . Phương pháp này dẫn đến vùng được điền mượt nhất có thể .

- Với roifill , ta lựa chọn một vùng quan tâm bằng trỏ chuột . Khi lựa chọn xong , hàm roifill trả lại một ảnh với vùng được chọn đã bị điền đầy

- Ví dụ sau sử dụng hàm roifill để sửa ảnh . Đường viền của vùng được chọn được hiển thị là màu đỏ trên ảnh gốc

load trees

I = ind2gray(X,map);

imshow(I)

I2 = roifill;

imshow(I2)

Cú pháp của hàm roifill

J = roifill(I,c,r)

J = roifill(I)

J = roifill(I,BW)

[J,BW] = roifill(...)

J = roifill(x,y,I,xi,yi)

[x,y,J,BW,xi,yi] = roifill(...)

Diễn giải

- Hàm roifill điền đầy trong một vùng đa giác trong một ảnh cường độ . Nó tuyến tính một cách mượt mà các giá trị pixel từ phía viền của đa giác vào phía trong đa giác bằng cách giải phương trình Laplace . Hàm roifill có thể được sử dụng , chẳng hạn để xoá một đối tượng nhỏ trong một ảnh

+ J=roifill(I,c,r) : Điền đầy một đa giác được chỉ ra bởi các véc tơ có cùng chiều dài c và r . Chúng chứa toạ độ hàng - cột của các pixel trên các đỉnh của đa giác .

+ J=roifill(I) : Hiển thị ảnh I trên màn hình và để ta lựa chọn vùng đa giác bằng trỏ chuột . Nếu bỏ qua I , hàm thao tác trên ảnh đang chọn . Sử dụng phím Backspace hoặc Delete để xoá các đỉnh trước đó đã chọn . Khi chọn xong , dùng phím Enter để kết thúc chọn

+ J=roifill(I,BW) : Sử dụng BW (một ảnh nhị phân cùng kích thước với I) như một mặt nạ . Hàm roifill sẽ điền đầy vùng trong I tương ứng với các pixel khác 0 trong BW . Nếu có nhiều vùng , roifill thi hành tuyến tính hoá trên mỗi vùng độc lập

+[J , BW]=roifill(…) : trả về mặt nạ nhị phân được sử dụng để tính toán pixel nào I sẽ điền đầy . BW là một ảnh nhị phân có cùng kích thước với I

+ J=roifill(x,y,I,xi,yi) : Sử dụng véc tơ x và y để thành lập một hệ toạ độ không gian không mặc định . xi , yi có cùng độ dài chỉ ra đỉnh của đa giác

+ [x,y,J,BW,xi,yi]=roifill(…) : trả lại Xdata và Ydata trong x và y , ảnh ra J , mặt nạ ảnh BW và đỉnh đa giác trong hai véc tơ xi , yi .

Lớp trợ giúp

- Ảnh vào I – uint8 , uint16 hoặc double . BW có thể là bất cứ kiểu số hoặc logical .

Ví dụ

I = imread('eight.tif');

c = [222 272 300 270 221 194];

r = [21 21 75 121 121 75];

J = roifill(I,c,r);

imshow(I)

figure, imshow(J)

III – Xử lý ảnh mờ

1. Bảng thuật ngữ

Tên thuật ngữ	Diễn giải
deconvolution	Xử lý ngược lại với hiệu ứng xoáy , cuốn
Distortion operator	Toán tử mô tả một quá trình ảnh nhận được khác với ảnh ban đầu . Distortion được gây ra bởi một hàm PSF chỉ là một trong những kiểu distortion
Optical transfer function(OTF)	Trong vùng tần số , OTF mô tả đáp ứng của một hệ thống tuyến tính , vị trí không biến đổi với một xung vào . OTF là một biến đổi Fourier của hàm PSF
Point spread function (PSF)	Trong miền không gian , PSF diễn tả cấp bậc mà một hệ thống quang học làm mờ một điểm sáng . PSF là biến đổi Fourier ngược của OTF

2. Thế nào là làm mờ ?

a - Nguyên nhân của sự mờ

- Sự làm mờ hay sự phai nhạt của một ảnh có thể gây ra bởi nhiều tác nhân :

+ Chuyển động trong khi capture ảnh - bởi camera hoặc khi thời gian lộ sáng nhiều được sử dụng - bởi vật

+ Ngoài vùng tiêu cự của ống kính , sử dụng một ống kính có góc mở rộng , sự hỗn loạn của môi trường , thời gian lộ sáng ngắn … sẽ làm giảm số lượng phôtôn được bắt giữ (captured)

b - Các chế độ chống mờ ảnh

- Một ảnh bị mờ hay bị phai nhạt có thể được mô tả vắn tắt bởi phương trình g=Hf+n trong đó

+ g : Ảnh bị mờ

+ H : Tác nhân làm méo cũng được gọi là PSF .

+ f : Ảnh gốc

+ n : Nhiễu phụ , được tạo ra trong quá trình nhận ảnh , nó làm hỏng ảnh

Chú ý : Ảnh f thực tế không tồn tại . Ảnh này đại diện cho bức ảnh mà ta có nếu tình trạng thu nhận ảnh là hoàn hảo

Tầm quan trọng của PSF

- Dựa trên chế độ này , tác vụ chính của việc chống làm mờ là ***Deconvolve*** ảnh bị mờ với PSF . Để minh hoạ , ví dụ này sẽ lấy 3 ảnh không bị mờ và cố ý làm mờ chúng bằng cách ***convolve*** nó với PSF . Ví dụ sử dụng hàm fspecial để tạo một PSF mô phỏng một chuyển động mờ , chỉ ra chiều dài của mờ tính bằng pixel (LEN=31) và góc mờ tính theo độ (THETA=11) . Một khi PSF được tạo , ví dụ sử dụng hàm imfilter để ***convolve*** PSF với ảnh gốc I để tạo ảnh bị làm mờ Blurred .

```
I = imread('peppers.png');
I = I(60+[1:256],222+[1:256],:); % crop the image
figure; imshow(I); title('Original Image');
LEN = 31;
THETA = 11;
PSF = fspecial('motion',LEN,THETA); % create PSF
Blurred = imfilter(I,PSF,'circular','conv');
figure; imshow(Blurred); title('Blurred Image');
```

3. Sử dụng các hàm khôi phục ảnh mờ

- Toolbox xử lý ảnh của Matlab gồm có 4 hàm khôi phục ảnh mờ bao gồm :

+deconvwnr : Sủ dụng bộ lọc Wiener

+ deconvreg : Sử dụng bộ lọc được quy tắc hoá

+ deconvlucy : Sử dụng giải thuật Lucy-Richardson

+ đeconvblind : Sử dụng giải thuật ***blind deconvolution***

- Tất cả những hàm này chấp nhận một PSF và một ảnh bị mờ như là các tham số chính của nó .Với hai hàm đầu tiên , ta cung cấp một số thông tin về nhiễu để giảm sự khuếch đại nhiễu đến mức có thể trong quá trình khôi phục .

- Hàm deconvlucy thi hành một cách nhanh chóng giải thuật Lucy-Richardson . Hàm này thực hiện nhiều vòng lặp , sử dụng kĩ thuật tối ưu và thống kê Poisson . Với hàm này , ta không cần phải cung cấp thông tin về nhiễu phụ trong ảnh bị "bẩn"

- Hàm deconvblind thi hành giải thuật blind deconvolution mà không cần nhận ra PSF . Khi ta gọi hàm deconvblind , ta truyền một tham số như là gía trị đoán biết ban đầu ở PSF . Hàm deconvblind trả lại một PSF đã được khôi phục để khôi phục ảnh . Sự thi hành sử dụng cùng chế độ suy giảm và lặp như hàm deconvlucy .

Chú ý : Ta có thể cần phải thi hành nhiều quá trình khử mờ lặp đi lặp lại , mỗi lần thay đổi tham số truyền vào hàm khử mờ cho tới khi thu được một ảnh gần xấp xỉ với ảnh gốc .

- Để tránh bị rung động trong ảnh được khử nhiễu , ta có thể sử dụng hàm edgetaper để tiền xử lý ảnh trước khi truyền nó cho hàm khử mờ .

a - Khử mờ với bộ lọc Wiener

- Sử dụng hàm deconvwnr để khử mờ một ảnh sử dụng bộ lọc Wiener . Bộ lọc này có thể được sử dụng rất hiệu quả khi đặc tính tần số của ảnh và nhiễu phụ là đã biết ít nhất là vài bậc . Trong trường hợp không có nhiễu , bộ lọc Wiener giảm tới bộ lọc đảo lý tưởng .

- Ví dụ sau khử nhiễu trong một ảnh bị mờ được tạo trước đây , chỉ ra cùng một hàm PSF được sử dụng để tạo mờ . Ví dụ này cũng minh hoạ tầm quan trọng của việc biết về PSF – hàm gây ra mờ . Khi chúng ta biết chính xác về PSF , kết quả của việc khử mờ có thể khá hiệu quả .

1. Đọc một ảnh vào không gian làm việc (để tăng tốc quá trình khử mờ , ví dụ này cũng cắt ảnh)

```
I = imread('peppers.png');
I = I(10+[1:256],222+[1:256],:);
figure;imshow(I);title('Original Image');
```

2. Tạo hàm PSF

```
LEN = 31;
THETA = 11;
PSF = fspecial('motion',LEN,THETA);
```

3. Tạo sự mờ giả trên ảnh

```
Blurred = imfilter(I,PSF,'circular','conv');
figure; imshow(Blurred);title('Blurred Image');
```

4. Khử mờ cho ảnh

```
wnr1 = deconvwnr(Blurred,PSF);
figure;imshow(wnr1);
title('Restored, True PSF');
```

Tinh chế kết quả

- Ta có thể tác động lên kết quả bằng cách cung cấp các giá trị cho các tham số tuỳ chọn được trợ giúp bởi hàm deconvwnr .

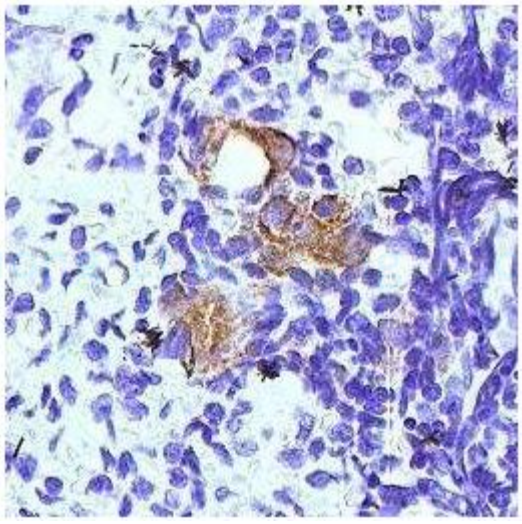
b - Khử mờ với bộ lọc được quy tắc hoá

- Sử dụng hàm deconvreg để khử mờ một ảnh sử dụng bộ lọc được quy tắc hoá . Một bộ lọc kiểu này có thể được sử dụng hiệu quả khi thông tin về nhiễu phụ được biết một cách hạn chế

- Để minh hoạ , ví dụ này mô phỏng việc làm mờ một ảnh sử dụng một hàm PSF lọc Gaussian với một ảnh (sử dụng imfilter) . Nhiễu phụ trong ảnh được mô phỏng bằng cách thêm vào một nhiễu Gaussian của biến V vào ảnh bị mờ (sử dụng hàm imnoise)

1. Đọc một ảnh vào trong không gian làm việc . Ví dụ này sẽ cắt ảnh để giảm kích thước của ảnh .

```
I = imread('tissue.png');
I = I(125+[1:256],1:256,:);
figure; imshow(I); title('Original Image');
```

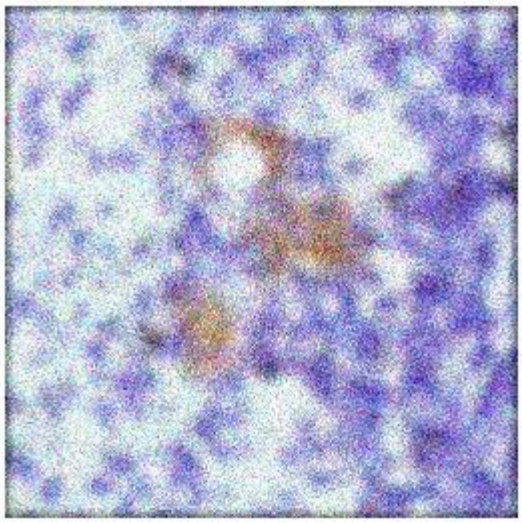


2. Tạo hàm PSF để làm mờ ảnh

PSF = fspecial('gaussian',11,5);

3. Làm mờ ảnh và thêm nhiễu vào ảnh

Blurred = imfilter(I,PSF,'conv');



V = .02;

BlurredNoisy = imnoise(Blurred,'gaussian',0,V);

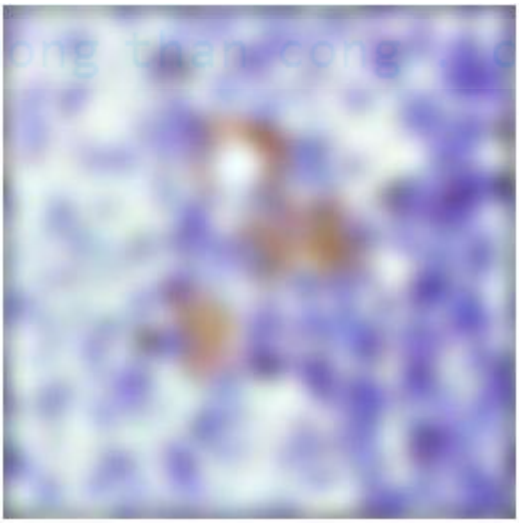
figure;imshow(BlurredNoisy);title('Blurred and Noisy Image');

4. Sử dụng hàm deconvreg để khử mờ ảnh , chỉ ra hàm PSF được sử dụng để làm mờ ảnh và nhiễu NP

NP = V*prod(size(I));

[reg1 LAGRA] = deconvreg(BlurredNoisy,PSF,NP);

figure,imshow(reg1),title('Restored Image');



C - Khử mờ với giải thuật Lucy- Richardson

- Sử dụng hàm deconvlucy để khử mờ một ảnh bằng cách sử dụng giải thuật Lucy-Richardson . Hàm này có thể được sử dụng hiệu quả khi biết được hàm PSF nhưng biết ít về nhiễu tác động phụ lên ảnh

- Hàm deconvlucy thi hành vài sự điều hợp tới giải thuật Lucy-Richardson . Sử dụng những điều hợp này ta có thể :

+ Giảm tác động của sự mở rộng nhiễu trên một ảnh khôi phục

+ Giải thích được tính không đồng nhất của chất lượng ảnh

+ Điều khiển camera đọc hết nhiễu nền

+ Cải thiện độ phân giải của ảnh phục hồi bằng cách lấy mẫu phụ

Giảm tác động của sự mở rộng nhiễu

- Sự mở rộng nhiễu là một vấn đề thường gặp của phương pháp giống cực đại (maximum likelihood) cố gắng lấp đầy dữ liệu gần nhất có thể . Sau một số vòng lặp , ảnh được khôi phục có thể có hình lốm đốm , đặc biệt với một đối tượng phẳng được quan sát tại tỉ số tín hiệu / nhiễu nhỏ . Những đốm này không đại diện cho bất kì một cấu trúc nào trong ảnh thực nhưng là giả tạo của việc làm khớp nhiễu trong ảnh quá gần .

- Để điều khiển sự mở rộng nhiễu , hàm deconvlucy sử dụng một tham số gọi là DAMPAR . Tham số này chỉ ra mức ngưỡng cho độ lệch của ảnh kết quả so với ảnh gốc . Với các pixel mà lệch khỏi vùng lân cận của các giá trị gốc của chúng , vòng lặp bị treo .

- ***Damping*** cũng được sử dụng để giảm rung (ringing) – hình dạng của cấu trúc tần số cao trong ảnh khôi phục . Ringing không cần thiết đến kết quả của mở rộng nhiễu .

Tính không đồng nhất của chất lượng ảnh

- Bất kì sự biến thể nào của ảnh khôi phục thực là do dữ liệu có thể chứa những pixel hỏng hoặc chất lượng của pixel nhận được biến đổi theo thời gian và vị trí . Bằng cách chỉ ra tham số mảng WEIGHT với hàm deconvlucy , ta có thể chỉ ra rằng những pixel nào đó trong ảnh bị bỏ qua . Để bỏ qua một pixel , gán một trọng lượng 0 tới một phần từ trong mảng WEIGHT tương ứng với pixel trong ảnh .

- Giải thuật hội tụ trên các giá trị đã đoán biết được với các pixel hỏng trên cơ sở của thông tin từ các pixel lân cận . Sự biến đổi trong đáp ứng phát hiện từ pixel tới pixel có thể được điều tiết bởi mảng WEIGHT . Thay cho việc gán một giá trị trọng lượng 1 tới các pixel tốt , ta có thể chỉ ra các giá trị và trọng lượng mà các pixel tuỳ thuộc vào lượng các flat-field correction

Điều khiển camera đọc hết nhiễu

- Nhiễu trong thiết bị CCD có hai thành phần chính :

+ Phôton kể đến nhiễu với một phân bố Poisson

+ Đọc toàn bộ nhiễu với phân bố Gauss

- Vòng lặp Lucy-Richardson thực chất nhận ra kiểu nhiễu đầu tiên . Ta phải nhận ra kiểu nhiễu thứ hai , nếu không nó có thể là nguyên nhân các pixel với các phôton tới ở mức thấp sẽ nhận giá trị âm .

- Hàm deconvlucy sử dụng tham số vào READOUT để điều khiển camera đọc toàn bộ nhiễu . Giá trị của tham số này điển hình là tổng của nhiễu tổng cộng và nhiễu nền . Giá trị của tham số READOUT chỉ ra một độ lệch đảm bảo rằng tất cả các giá trị đều dương .

Ví dụ : Sử dụng hàm deconvlucy để khử mờ một ảnh

- Để minh hoạ , ví dụ này sử dụng hàm deconvlucy , nó mô phỏng một ảnh bị mờ và nhiễu bằng cách convolve một bộ lọc Gauss với một ảnh (sử dụng hàm imfilter) và sau đó thêm nhiễu Gauss (sử dụng hàm imnoise) .

1. Đọc một ảnh vào không gian làm việc .

```
I = imread('board.tif');
I = I(50+[1:256],2+[1:256],:);
figure;imshow(I);title('Original Image');
```

2. Tạo hàm PSF để làm mờ ảnh

```
PSF = fspecial('gaussian',5,5);
```

3. Tạo mờ ảnh và thêm nhiễu vào ảnh

```
Blurred = imfilter(I,PSF,'symmetric','conv');

V = .002;
BlurredNoisy = imnoise(Blurred,'gaussian',0,V);
figure;imshow(BlurredNoisy);title('Blurred and Noisy Image');
```

4. Sử dụng hàm devconlucy để khôi phục ảnh gốc , chỉ ra hàm PSF được sử dụng để tạo mờ và giới hạn số vòng lặp tới 5 (mặc định là 10)

```
luc1 = deconvlucy(BlurredNoisy,PSF,5);
```

figure; imshow(luc1);
title('Restored Image');

d - Sử dụng giải thuật trộn deconvolution

- Sử dụng hàm deconvblind để khử mờ một ảnh . Giải thuật này có thể được sử dụng hiệu quả khi không có thông tin về mờ hoặc nhiễu được biết . Hàm deconvblind khôi phục ảnh và PSF đồng thời , sử dụng một quá trình lặp tương tự như giải thuật Lucy-Richardson

- Hàm deconvblind chỉ giống như hàm deconvlucy , thi hành một vài sự điều hợp tới giải thuật Lucy-Richardson .

Ví dụ sau đây minh hoạ việc sử dụng hàm deconvblind . Nó sẽ tạo một ảnh giả mờ và sau đó dùng hàm này để khử mờ .

1. Đọc một ảnh vào không gian làm việc

I = imread('cameraman.tif');

figure; imshow(I); title('Original Image');

2. Tạo hàm PSF để làm mờ ảnh

```
PSF = fspecial('motion',13,45);
figure; imshow(PSF,[],'notruesize'); title('Original PSF');
```

3. Tạo mờ trên ảnh

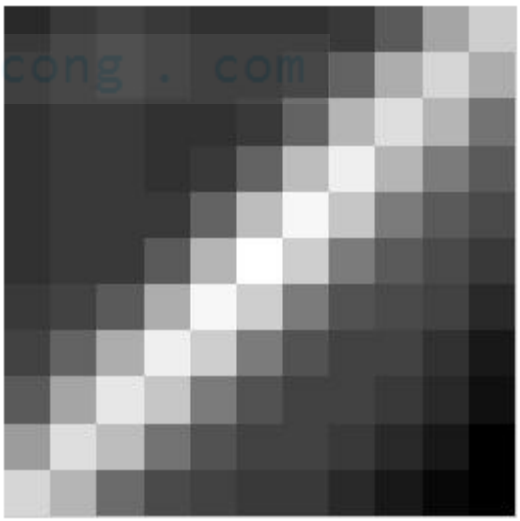
```
Blurred = imfilter(I,PSF,'circ','conv');
figure; imshow(Blurred); title('Blurred Image');
```

4. Khử mờ ảnh , tạo một sự ước lượng ban đầu cho kích thước của PSF

- Để quyết định kích thước của PSF , kiểm tra ảnh mờ và đo chiều rộng của mờ (theo pixel) xung quanh một vật có hình dạng nhất định . Trong ảnh mờ mẫu , ta có thể đo mờ gần với đường viền tà áo của người đàn ông . Bởi vì kích thước của PSF là quan trọng hơn giá trị của nó , ta có thể chỉ ra một mảng toàn số 1 như là PSF ban đầu .
- Hình sau đây chỉ ra một sự khôi phục mà sự phỏng đoán ban đầu về kích thước của PSF là giống với kích thước của PSF đã gây ra mờ . Trong ứng dụng thực , ta có thể cần chạy lại hàm deconvblind , kiểm tra PSF với các kích thước khác nhau cho tới khi nhận được một kết quả ổn định . PSF được khôi phục trả lại bởi mỗi lần deconvolution có thể cung cấp một chỉ dẫn có giá trị cho kích thước tối ưu của PSF .

```
INITPSF = ones(size(PSF));
[J P]= deconvblind(Blurred,INITPSF,30);
figure; imshow(J); title('Restored Image');
figure; imshow(P,[],'notruesize');
title('Restored PSF');
```

Restored Image



Restored PSF

- Mặc dù hàm deconvblind có thể khử nhiễu ảnh tới một khoảng rất rộng , hiện tượng rung xung quanh vùng tương phản cường độ trong ảnh phục hồi là không ổn định . Bước tiếp theo trong ví dụ lặp lại quá trình khử mờ , cố gắng đạt tới một kết quả tốt hơn bằng cách :

+ Tránh vùng tương phản cao từ sự xử lý

+ Chỉ ra một giá trị PSF tốt hơn

5. Tạo một mảng WEIGHT để loại trừ vùng tương phản cao từ thao tác khử mờ . Điều này có thể làm giảm rung do tương phản như trong kết quả .

Để loại trừ một pixel từ sự xử lý , ta tạo một mảng có cùng kích thước với ảnh gốc và gán giá trị 0 cho các pixel trong mảng tương ứng với pixel trong ảnh gốc mà ta muốn loại bỏ . Để tạo một mảng WEIGHT , ví dụ sử dụng một sự kết hợp của việc phát hiện cạnh và xử lý hình thái học để phát hiện vùng có độ tương phản cao trong ảnh . Do mờ trong ảnh là tuyến tính , ví dụ mở rộng ảnh ra 2 lần . Để loại trừ các pixel thuộc vùng biên của ảnh (một vùng có độ tương phản cao) từ quá trình xử lý , ví dụ này sử dụng padarray để gán giá trị 0 tới tất cả các pixel trên biên ảnh :

```
WEIGHT = edge(I,'sobel',.28);
se1 = strel('disk',1);
se2 = strel('line',13,45);
WEIGHT = ~imdilate(WEIGHT,[se1 se2]);
WEIGHT = padarray(WEIGHT(2:end-1,2:end-1),[2 2]);
figure; imshow(WEIGHT); title('Weight Array');
```

Weight Array

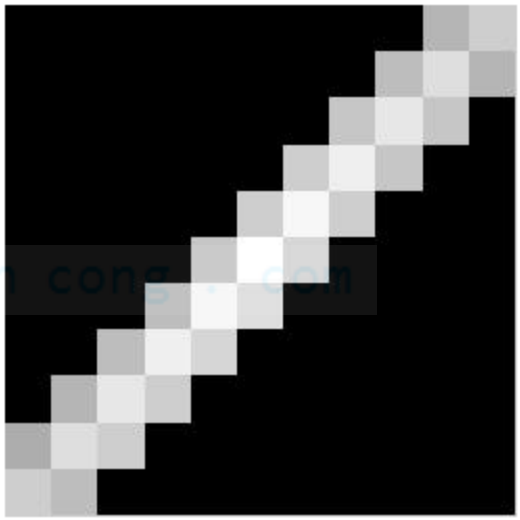
6. Tinh chế giá trị đã ước lượng cho PSF . PSF được xây dựng lại p được trả về từ lần truyền đầu tiên mà kết quả đạt được một độ tuyến tính khá tốt . Với lần truyền thứ hai , ví dụ này sử dụng một giá trị PSF mới p1 giống như p nhưng với biên độ nhỏ pixel thiết lập bằng 0

```
P1 = P;
P1(find(P1 < 0.01))=0;
```

7. Trả lại deconvolution , chỉ ra mảng WEIGHT và sửa giá trị PSF . Chú ý , ảnh phục hồi có ít rung xung quanh vùng có cường độ tương phản lớn hơn kết quả trước đây :

```
[J2 P2] = deconvblind(Blurred,P1,50,[],WEIGHT);
figure; imshow(J2);
title('Newly Deblurred Image');
figure; imshow(P2,[],'notruesize');
title('Newly Reconstructed PSF');
```

Newly Deblurred Image



Newly Reconstructed PSF

IV – Màu sắc

1. Bảng thuật ngữ

Tên thuật ngữ	Diễn giải
Approximation	Phương pháp nhờ đó phần mềm lựa chọn sự thay thế màu sắc trong một sự kiện mà trực tiếp thích hợp không thể tìm thấy .Phương pháp xấp xỉ được thảo luận ở đây là ánh xạ bản đồ màu , lượng tử hoá đồng nhất và lượng tử hoá với biến động nhỏ nhất
Indexed image	Ảnh mà các giá trị pixel của nó được đánh số trực tiếp trong một bản đồ màu . Trong Matlab , một ảnh chỉ số được đại diện bởi một mảng thuộc lớp uint8 , uint16 hoặc double . Bản đồ màu luôn là một mảng mx3 thuộc lớp double .
Intensity image	Ảnh bao gồm các giá trị cường độ . Trong Matlab , ảnh cường độ được đại diện bởi một mảng thuộc lớp uint8 ,uint16 hoặc double . Trong khi ảnh cường độ không được lưu trữ với bản đồ màu , Matlab sử dụng bản đồ màu hệ thống để hiển thị chúng .
RGB image	Ảnh mà trong đó mỗi pixel được chỉ định rõ 3 màu R,G,B . Trong Matlab , một ảnh RGB được đại diện bởi một ma trận mxnx3 thuộc lớp uint8 , uint16 hoặc double .
Screen bit depth	Số lượng bít trên một pixel màn hình
Screen color resolution	Số lượng các màu riêng biệt có thể được

	tạo ra bởi màn hình

2. Làm việc với các chiều sâu bít màn hình khác nhau

- Hầu hết các màn hình máy tính sử dụng 8,16,24 bít trên một pixel màn hình . Số lượng bít trên một pixel màn hình quyết định độ sâu bít màn hình . Độ sâu bít màn hình quyết định đến độ phân giải màu sắc của màn hình đó là bao nhiêu màu sắc riêng biệt màn hình có thể tạo ra .

- Bất chấp số lượng màu sắc hệ thống có thể hiển thị , Matlab có thể lưu trữ và xử lý các ảnh với độ sâu bít rất cao . Những ảnh này được hiển thị tốt nhất trên hệ thống 24 bít màu nhưng thường khá tốt trên hệ thống 16 bít màu .

a – Tính độ sau bít màn hình

- Để tính độ sâu bít của màn hình , sử dụng lệnh sau đây :

```
get(0,'ScreenDepth');
```

Giá trị nguyên trả về chính là số bít trên một pixel màn hình

+ 8 : 8 bít hiển thị trợ giúp 256 màu . Một màn hình 8 bít có thể sản sinh ra bất kì màu sắc nào có trên màn hình 24 bít nhưng chỉ 256 màu có thể xuất hiện cùng lúc .

+ 16 :Chế độ hiển thị 16 bít thường sử dụng 5 bít cho mỗi thành phần màu dẫn đến 32 mức cho mỗi màu R,G,B . Do đó , nó trợ giúp 32768 màu riêng biệt . Một vài hệ thống sử dụng các bít phụ để tăng số lượng mức của màu G . Trong trường hợp này , số lượng màu khác nhau được trợ giúp là 64536 .

+ 24 : Chế độ hiển thị 24 bít sử dụng 8 bít cho mỗi màu R, G,B dẫn đến 256 mức cho mỗi màu này , do đó , nó trợ giúp 16777216 màu khác nhau .

+ 32 : Sử dụng 24 bít để lưu thông tin màu và sử dụng 8 bít còn lại để lưu dữ liệu trong suốt (kênh anpha)

b - Chọn độ sâu bít màn hình

- Phụ thuộc vào hệ thống , ta có thể chọn độ sâu bít màn hình ta muốn . Nhìn chung , chế độ hiển thị 24 bít tạo ra kết quả tốt nhất . Nếu ta muốn sử dụng một độ sâu thấp hơn 16 bít thường được sử dụng hơn là 8 bít . Tuy nhiên , hiển thị 16 bít có một số giới hạn (Xem thêm help online)

3. Giảm số lượng màu trong một ảnh

- Ta sẽ xem xét làm cách nào để giảm số lượng màu trong một ảnh chỉ số hoặc ảnh RGB . Ta cũng xem xét đến kĩ thuật trộn (dithering) . Dithering được sử dụng để tăng số lượng màu của một ảnh

- Toolbox cung cấp cho ta một số hàm sau đây để giảm màu :

+ imapprox : Giảm số lượng màu được sử dụng bởi ảnh chỉ số , cho phép ta chỉ ra số lượng màu trong bản đồ màu mới .

+ rgb2ind : Convert một ảnh RGB thành một ảnh chỉ số , cho phép ta chỉ ra số lượng màu chứa trong bản đồ màu mới .

- Tên các hệ thống với hiển thị 24 bít màu , ảnh RGB (true-color) có thể hiển thị tới 16777216 màu khác nhau . Trên các hệ thống với độ sâu bít màn hình thấp hơn , ảnh RGB vẫn được hiển thị khá tốt bởi vì Matlab tự động sử dụng phối trộn và xấp xỉ màu nếu thấy cần thiết .

- Ảnh chỉ số tuy nhiên , có thể gây ra vấn đề nếu chúng có một số lượng màu lớn . Nhìn chung , ta nên giới hạn ảnh chỉ số tới 256 màu vì các lý do sau đây :

+ Trên hệ thống với chế độ hiển thị 8 bít , ảnh chỉ số với nhiều hơn 256 màu sẽ cần được phối trộn (dithered) hoặc ánh xạ (mapped) và do đó có thể không hiển thị tốt .

+ Trên một số hệ điều hành (platform) , bản đồ màu không thể vượt quá 256 màu

+ Nếu một ảnh chỉ số có nhiều hơn 256 màu , Matlab không thể lưu dữ liệu ảnh trong một mảng thuộc lớp uint8 tuy nhiên , nhìn chung sử dụng một mảng thuộc lớp double thay thế làm cho kích thước của ảnh lớn hơn .

+ Hầu hết các định dạng file ảnh giới hạn ảnh chỉ số tới 256 màu . Nếu ta viết một ảnh chỉ số với nhiều hơn 256 màu (sử dụng hàm imwrite) tới một định dạng không trợ giúp nhiều hơn 256 màu , ta sẽ nhận được một thông báo lỗi

Hàm imapprox

- Cú pháp của nó như sau :

```
[Y,newmap] = imapprox(X,map,n)
[Y,newmap] = imapprox(X,map,tol)
Y = imapprox(X,map,newmap)
[...] = imapprox(...,dither_option)
```

Diễn giải

+ [Y,newmap]=imapprox(X,map,n) : Xấp xỉ các màu trong một ảnh chỉ số X được kết hợp với bản đồ màu map bằng cách sử dụng lượng tử biến đổi nhỏ nhất (minumum

variance quantization) . Hàm imapprox trả về ảnh chỉ số Y với bản đồ màu newmap có nhiều nhất n màu .

+ [Y,newmap]=imapprox(X,map,tol) : Xấp xỉ các màu trong X và map thông qua lượng tử đều (uniform quantization) . newmap chứa nhiều nhất (floor (1/tol)+1)^3 màu . tol phải nằm giữa 0 và 1

+ Y=imapprox(X,map,newmap) : Xấp xỉ màu trong map bằng cách sử dụng ánh xạ bản đồ màu để tìm các màu trong newmap tương hợp tốt nhất với các màu trong map .

+ Y=imapprox(…,dither_option) : cho phép hoặc cấm trộn . dither_option là một chuỗi có một hoặc nhiều giá trị . Giá trị mặc định được đặt trong dấu ({}). Nếu dither_option là {‘dither’} - sẽ trộn nếu cần thiết để thu được một độ phân giải màu tốt hơn . Nếu dither_option là (‘nodither’) – ánh xạ mỗi màu trong ảnh gốc tới màu gần nhất trong bản đồ màu mới . Không có phép trộn nào được thực hiện .

Giải thuật

- Hàm imapprox sử dụng hàm rgb2ind để tạo một bản đồ màu mới sử dụng ít màu hơn

Ví dụ

- Xấp xỉ ảnh chỉ số trees.tif bằng một ảnh chỉ số khác chỉ chứa 16 màu

```
[X, map] = imread('trees.tif');
[Y, newmap] = imapprox(X, map, 16);
imview(Y, newmap)
```

Hàm rgb2ind

- Hàm này dùng để convert một ảnh RGB thành một ảnh chỉ số . Cú pháp của nó như sau :

```
[X,map] = rgb2ind(RGB,tol)
[X,map] = rgb2ind(RGB,n)
X = rgb2ind(RGB,map)
[...] = rgb2ind(...,dither_option)
```

Diễn giải

- Hàm rgb2ind convert ảnh RGB thành ảnh chỉ số sử dụng một trong 3 cách khác nhau : lượng tử đều , lượng tử biến đổi cực tiểu và ánh xạ bản đồ màu . Với tất cả các phương pháp này , rgb2ind cũng trộn ảnh trừ khi ta chỉ ra nodither trong dither_option .

+ [X,map]=rgb2ind(RGB,tol) : Convert ảnh RGB thành một ảnh chỉ số X sử dụng lượng tử đều . map chứa nhiều nhất (floor(1/tol)+1)^3 màu . tol có giá trị từ 0 đến 1

+ [X,map]=rgb2ind(RGB,n) : Convert ảnh RGB thành ảnh chỉ số X sử dụng lượng tử biến đổi cực tiểu , map chứa nhiều nhất n màu , n phải nhỏ hơn hoặc bằng 65536

+ X=rgb2ind(RGB,map) : Convert ảnh RGB thành ảnh chỉ số X với bản đồ màu map bằng cách hợp các màu trong RGB với các màu gần nhất trong bản đồ màu map , size(map,1) phải nhỏ hơn hoặc bằng 65536

+[…]=rgb2ind(…,dither_option) : Cho phép hoặc cấm trộn , dither_option là một chuỗi . Các giá trị của dither_option giống như trong hàm imapprox .

Chú ý : Nếu ta chỉ ra tol , hàm rgb2ind sử dụng lượng tử đều để convert ảnh . Phương pháp này bao gồm việc cắt hình lập phương màu RGB thành các hình lập phương nhỏ hơn có chiều dài tol . Chẳng hạn , nếu tol=0.1 , cạnh của các hình lập phương mới sẽ là 1/10 cạnh của hình lập phương RGB . Tổng số hình lập phương nhỏ sẽ là :

n=(floor(1/tol)+1)^3)

Mỗi hình lập phương đại diện cho một màu đơn trong ảnh kết quả . Vì vậy , chiều dài cực đại của bản đồ màu là n . Hàm rgb2ind bỏ đi bất kì màu nào không xuất hiện trong ảnh vào vì vậy bản đồ màu thực có thể nhỏ hơn n .

- Nếu ta chỉ ra n , hàm rgb2ind sử dụng lượng tử biến đổi cực tiểu . Phương pháp này bao gồm việc cắt hình lập phương RGB thành các hình hộp nhỏ hơn (không cần hình lập phương) với các kích thước khác nhau phụ thuộc vào màu sắc được phân phố như thế nào trong ảnh . Nếu ảnh vào sử dụng ít màu hơn số lượng màu ta chỉ ra (n) , bản đồ màu ra sẽ nhỏ hơn .

- Nếu ta chỉ ra map , hàm rgb2ind sử dụng phương pháp ánh xạ bản đồ màu , nó sẽ tìm những màu trong map thích hợp tốt nhất với các màu trong ảnh RGB

Ví dụ

```
RGB = imread('peppers.png');
[X,map] = rgb2ind(RGB,128);
imshow(X,map)
```

4. Dithering

- Khi sử dụng hàm rgb2ind hoặc imapprox để giảm số lượng màu trong một ảnh , ảnh kết quả trông xấu hơn ảnh gốc bởi vì một số màu bị mất . Hàm rgb2ind và imapprox cả hai đều thực hiện dithering để tăng số lượng màu trong ảnh kết quả . Dithering thay đổi màu sắc của các pixel ở một vùng lân cận vì vậy , màu sắc trung bình của mỗi vùng xấp xỉ màu gốc .

- Chẳng hạn một ví dụ để xem dithering làm việc ra sao , xem xét một ảnh chứa một số pixel màu cam tối mà không có một sự tương hợp chính xác trong bản đồ màu . Để tạo ra sắc thái cam này , toolbox lựa chọn sự kết hợp màu từ bản đồ màu . Nó sẽ trộn 6 pixel lại với nhau thành một nhóm , xấp xỉ sắc thái hồng . Từ một khía cạnh nào đó , các pixel xuất hiện như những sắc thái chính xác nhưng nếu ta nhìn gần ảnh , ta có thể thấy sự phối trộn của các sắc thái . Ví dụ này nạp một ảnh 24 bít sau đó , sử dụng hàm rgb2ind để tạo ra hai ảnh chỉ số chỉ với 8 màu cho mỗi ảnh :

1. Đọc một ảnh và hiển thị nó :

```
rgb=imread('onion.png');
imshow(rgb);
```

2. Tạo một ảnh chỉ số với 8 màu mà không trộn

```
[X_no_dither,map]=rgb2ind(rgb,8,'nodither');
figure, imshow(X_no_dither,map);
```

3. Tạo một ảnh chỉ số 8 màu có trộn

```
[X_dither,map]=rgb2ind(rgb,8,'dither');
figure, imshow(X_dither,map);
```

Để ý rằng ảnh được dithering có số lượng màu nhiều hơn nhưng trông có vẻ hơi nhạt . Ảnh không được dithering có ít màu hơn nhưng tăng độ phân giải không gian so với ảnh

dithering . Một sự nguy hiểm khi giảm màu mà không dither là ảnh mới có thể chứa các đường viền sai .

5. Chuyển đổi màu sắc giữa các không gian màu

- Toolbox biểu diễn màu sắc như là các giá trị RGB trực tiếp (trong ảnh RGB) hoặc gián tiếp (trong ảnh chỉ số - bản đồ màu được lưu theo định dạng RGB) . Tuy nhiên , có những chế độ khác bên cạnh RGB để biểu diễn màu . Các chế độ khác nhau được xem như là các không gian màu (color spaces) bởi vì hầu hết chúng có thể được ánh xạ tới một hệ toạ độ 2D, 3D hoặc 4D . Vì vậy , một đặc trưng màu được tạo thành từ một toạ độ 2D, 3D hoặc 4D .
- Các không gian màu khác nhau tồn tại bởi vì chúng đại diện cho thông tin về màu sắc theo các cách để tạo ra sự tính toán thuận tiện hơn hoặc bởi vì chúng cung cấp một cách để phân biệt các màu hiệu quả hơn . Chẳng hạn , không gian màu RGB định nghĩa một màu như là tổ hợp của 3 màu R,G,B . Các chế độ màu khác mô tả màu sắc bởi các giá trị hue (green) , saturation (dark green) và ánh sáng hay cường độ .
- Toolbox trợ giúp những không gian màu này bằng cách cung cấp một cách cho việc chuyển đổi dữ liệu màu từ không gian màu này đến không gian màu khác thông qua các phép biến đổi toán học .
- Trong phần này ta sẽ :

+ Chỉ ra làm cách nào để chuyển dữ liệu màu giữa các không gian màu
+ Xem xét cách thức để chuyển đổi không gian màu sử dụng profile ICC
+ Xem xét một số hàm cho việc chuyển đổi giữa không gian màu RGB và 3 không gian màu thông dụng khác là : YIQ , HSV và YCbCr .

a - Chuyển đổi giữa các không gian màu độc lập thiết bị

- Các thuật ngữ chuẩn được dùng để mô tả màu sắc như : hue , độ sáng và cường độ sáng là chủ quan và làm cho sự so sánh trở nên khó khăn .
- Năm 1931 , uỷ ban quốc tế về độ rọi CIE đã nghiên cứu sự tiếp nhận màu sắc của con người và đã phát triển một chuẩn được gọi là chuẩn CIE XYZ . Chuẩn này định nghĩa 1 không gian 3 chiều với 3 giá trị được gọi là 3 giá trị tác nhân (trisimulus values) để định nghĩa một màu sắc . Chuẩn này vẫn còn được sử dụng rộng rãi cho đến ngày nay .
- Trong những thập kỉ của đặc trưng đầu tiên đó , CIE đã phát triển một vài đặc trưng không gian màu phụ để cố gắng cung cấp một biểu diễn màu tương tự nhưng thích hợp

tốt hơn so với một số mục đích của XYZ . Chẳng hạn , năm 1976 , một nỗ lực để có được một không gian màu đồng nhất có thể tương quan với thể hiện trực quan của màu sắc – CIE đã tạo ra không gian màu L*a*b .

- Toolbox cung cấp trợ giúp cho việc chuyển đổi giữa các không gian màu không phụ thuộc thiết bị trong số các chuẩn của CIE . Thêm vào đó , toolbox cũng trợ giúp chuyển đổi giữa các không gian màu CIE và không gian màu RGB . Không gian màu này được định nghĩa bởi một nhóm các nhà công nghiệp để mô tả đặc tính của một màn hình PC tiêu biểu .

Những chuyển đổi được trợ giúp

Không gian màu	Diễn giải	Chuyển đổi được trợ giúp
XYZ	Chuẩn gốc được CIE định nghĩa năm 1931	xyY , uvl , u'v'L và L*a*b
xyY	Đặc trưng CIE cung cấp các giá trị màu được tiêu chuẩn hoá . Giá trị Y đại diện cho độ sáng và giống như trong XYZ	XYZ
uvL	Đặc trưng CIE mà cố gắng tạo ra một mặt phẳng màu đồng nhất hơn . l là ánh sáng	XYZ
u'v'L	Đặc trưng CIE trong đó u và v được định lại tỉ lệ để cải thiện tính đồng nhất	XYZ
L*a*b	Đặc trưng CIE nhằm cố gắng tạo ra sự định tỉ lệ màu sắc đồng nhất hơn . L* là tỉ lệ không tuyến tính của L được tiêu chuẩn hoá tới một điểm tham chiếu trắng	XYZ
L*ch	Đặc trưng CIE trong đó c là	L*a*b

	chroma và h là hue . Những giá trị này là kết quả của sự chuyển đổi toạ độ cực của a* và b* trong L*a*b	
sRGB	Chuẩn được ban hành bởi các nhà sản xuất lớn để đặc tính hoá mà hình PC	XYZ và L*a*b

Ví dụ : Thực hiện sự chuyển đổi giữa các không gian màu

- Để minh hoạ một sự chuyển đổi giữa các không gian màu độc lập thiết bị , ví dụ này đọc một ảnh RGB vào không gian làm việc và chuyển đổi dữ liệu màu sang không gian màu XYZ :

1. Nhập dữ liệu không gian màu . Ví dụ sẽ đọc một ảnh RGB vào không gian làm việc

I_rgb = imread('peppers.png');

2. Tạo một cấu trúc chuyển đổi màu . Một cấu trúc chuyển đổi màu định nghĩa sự chuyển đổi giữa hai không gian màu . Ta sử dụng hàm makecform để tạo cấu trúc , chỉ ra một chuỗi kiểu cấu trúc làm tham số :

C = makecform('srgb2xyz'); - Tạo một cấu trúc chuyển đổi màu định nghĩa sự chuyển đổi từ RGB sang XYZ

3. Thực hiện chuyển đổi . Ta sử dụng hàm applycform để thi hành chuyển đổi , chỉ ra dữ liệu màu ta muốn chuyển đổi và cấu trúc chuyển đổi màu định nghĩa sự chuyển đổi đó . Hàm applycform trả về dữ liệu đã được chuyển đổi :

I_xyz = applycform(I_rgb,C);

whos

Name	Size	Bytes	Class
C	1x1	7744	struct array
I_xyz	384x512x3	1179648	uint16 array
I_rgb	384x512x3	589824	uint8 array

b- Chuyển đổi giữa các không gian màu phụ thuộc thiết bị

- Toolbox bao gồm các hàm ta có thể sử dụng để chuyển đổi dữ liệu ảnh RGB sang một số không gian màu phụ thuộc thiết bị và ngược lại :

+ YIQ

+ YCbCr

+ Hue , saturation , value (HSV)

Không gian màu YIQ

- Uỷ ban hệ thống truyền hình quốc gia Mỹ (NTSC) định nghĩa một không gian màu gọi là YIQ . Không gian màu này được sử dụng trong các máy thu hình ở Mỹ . Một trong những ưu điểm chính của định dạng này là thông tin gam màu xám (grayscale) được tách ra khỏi dữ liệu màu vì vậy cùng một tín hiệu có thể được sử dụng cho cả tivi đen trắng và tivi màu .

- Trong không gian màu NTSC , dữ liệu ảnh bao gồm 3 thành phần là : ánh sáng (Y) , hue (I) và saturation (Q) . Thành phần đầu tiên , ánh sáng đại diện cho thông tin gam màu xám , trong khi hai thành phần cuối cùng đại diện cho thông tin về màu sắc .

- Hàm rgb2ntsc convert bản đồ màu hoặc ảnh RGB thành không gian màu NTSC . Hàm ntcs2rgb chuyển đổi ngược lại .

- Chẳng hạn , những lệnh sau đây chuyển đổi một ảnh RGB sang ảnh NTSC .

```
RGB = imread('peppers.png');
YIQ = rgb2ntsc(RGB);
```

- Do ánh sáng là một thành phần của định dạng NTSC , sự chuyển đổi từ RGB sang NTSC là hữu ích cho việc cách ly thông tin độ xám trong một ảnh . Trong thực tế , các hàm rgb2gray và ind2gray sử dụng hàm rgb2ntsc để tách thông tin gam màu xám từ một ảnh màu .

Chẳng hạn , những lệnh sau đây là tương đương với việc gọi hàm rgb2gray :

```
YIQ = rgb2ntsc(RGB);
I = YIQ(:,:,1);
```


Không gian màu YCbCr

- Không gian màu YCbCr được sử dụng rộng rãi trong video số . Trong định dạng này , thông tin về ánh sáng được lưu trữ trong một thành phần riêng Y và thông tin màu sắc được lưu trữ như hai thành phần màu độc lập (Cb và Cr) . Cb đại diện cho sự khác nhau giữa giá trị blue và một giá trị tham chiếu trong khi đó Cr đại diện cho sự khác nhau giữa thành phần đỏ và một giá trị tham chiếu .

- Dữ liệu YCbCr có thể là dạng chính xác kép nhưng không gian màu thường thích hợp tốt nhất với dữ liệu dạng uint8 . Với các ảnh uint8 , vùng dữ liệu mà Y có thể nhận là [16,255] và vùng cho Cb , Cr là [16,240] . Hàm rgb2ycbcr convert bản đồ màu hoặc ảnh RGB sang không gian màu YCbCr , hàm ycbcr2rgb chuyển đổi ngược lại .

Chẳng hạn , những lệnh sau sẽ chuyển đổi một ảnh RGB sang định dạng YCbCr .

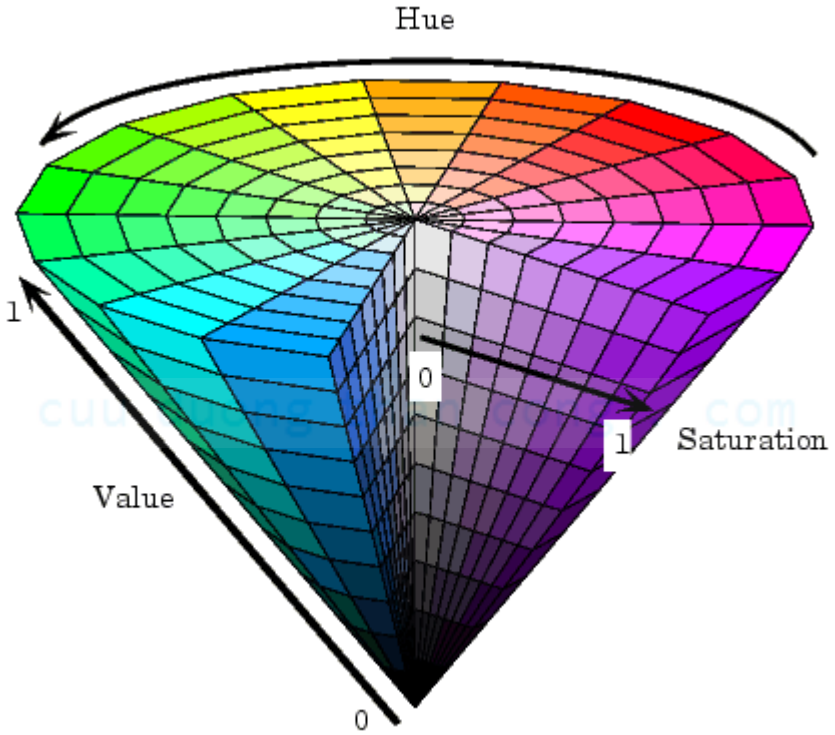
```
RGB = imread('peppers.png');
YCBCR = rgb2ycbcr(RGB);
```

Không gian màu HSV

- Không gian màu HSV (Hue , Saturation , Value) thường được sử dụng bởi những người cần phải lựa chọn màu sắc (chẳng hạn để in hoặc vẽ) từ một đĩa màu do nó tương ứng tốt hơn với kinh nghiệm màu sắc của những người này hơn là không gian màu RGB đem lại . Hàm rgb2hsv và hsv2rgb chuyển đổi ảnh giữa các không gian màu RGB và HSV .

- Khi giá trị Hue nằm trong khoảng 0 và 1 thì màu tương ứng sẽ biến đổi từ red , yellow , green , cyan , blue , magenta và black sang red . Vì vậy , giá trị thực của red là từ 0 đến 1 . Cũng vậy , khi saturation biến thiên từ 0 đến 1 , màu tương ứng (hues) biến đổi từ chưa bão hoà đến hoàn toàn bão hoà (unsaturated to fully saturated) . Cuối cùng , nếu value (hay độ sáng) biến đổi từ 0 đến 1 , màu tương ứng sẽ càng sáng hơn .

- Hình sau đây minh hoạ không gian màu HSV

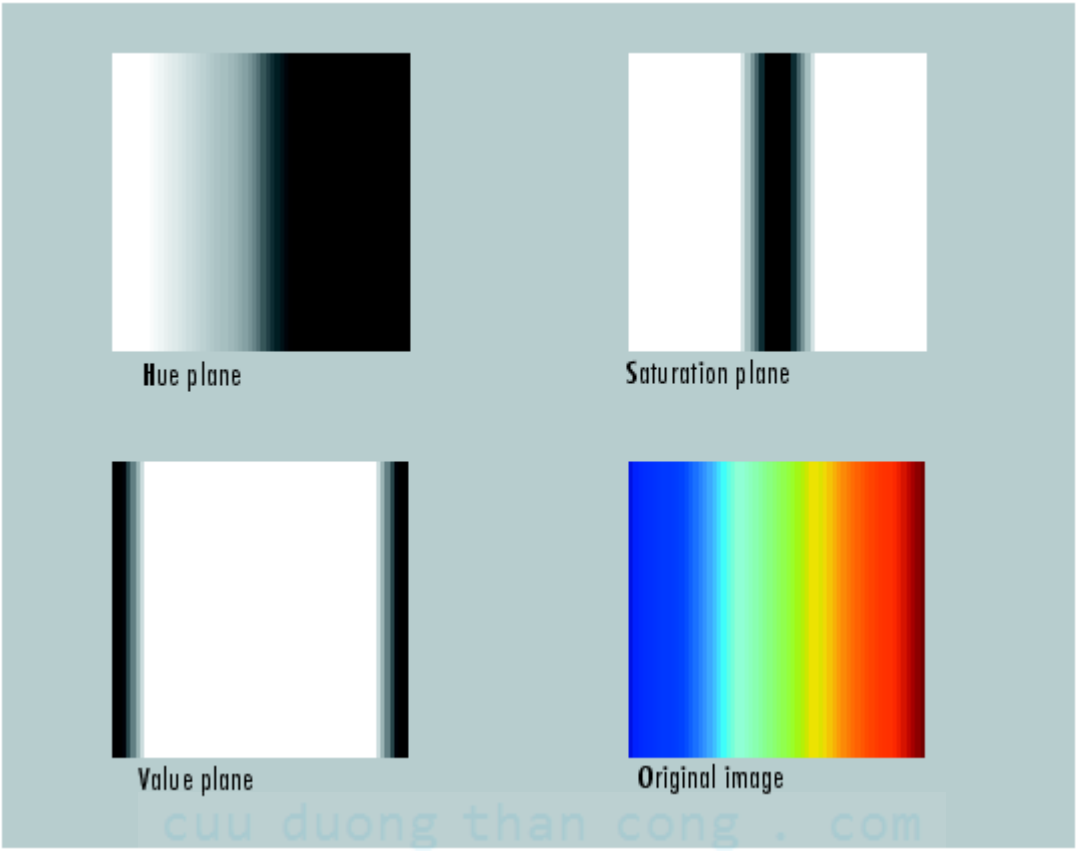


- Hàm rgb2hsv chuyển đổi bản đồ màu hoặc ảnh RGB sang không gian màu HSV . Hàm hsv2rgb thi hành thao tác ngược lại . Những lệnh sau đây sẽ convert một ảnh RGB sang không gian màu HSV .

```
RGB = imread('peppers.png');
HSV = rgb2hsv(RGB);
```

- Để xem xét gần hơn không gian màu HSV , khối lệnh sau đây sẽ chia mặt phẳng màu (hue , saturation và value) của một ảnh HSV :

```
RGB=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
HSV=rgb2hsv(RGB);
H=HSV(:,:,1);
S=HSV(:,:,2);
V=HSV(:,:,3);
imshow(H)
figure, imshow(S);
figure, imshow(V);
figure, imshow(RGB);
```



V - Biến đổi ảnh

- Sự biểu diễn toán học cho một ảnh thường là một hàm với hai biến không gian f(x,y) . Giá trị của hàm tại một vị trí (x,y) đại diện cho cường độ của ảnh tại vị trí đó . Thuật ngữ biến đổi (transform) nhằm nói đến một sự biểu diễn toán học tương tự của ảnh .

1. Bảng thuật ngữ

Tên thuật ngữ	Diễn giải
Discrete transform	Là biến đổi mà giá trị đầu vào và ra của nó là các mẫu rời rác , biến đổi này thường thuận lợi cho các thao tác của máy tính . Biến đổi rời rác được ứng dụng bởi Matlab và toolbox xử lý ảnh bao gồm biến đổi Fourier rời rạc (DFT) và biến đổi cô-sin rời rạc (DCT)

Frequency domain	Là một vùng trong đó ảnh được đại diện bởi tổng của các tín hiệu tuần hoàn với tần số thay đổi
Inverse transform	Biến đổi ngược để tạo thành ảnh ban đầu
Spatial domain	Vùng không gian mà trong đó ảnh được biểu diễn bởi cường độ (độ sáng) của một điểm trong không gian . Đây là biểu diễn phổ biến nhất của dữ liệu ảnh
Transform	Biểu diễn toán học tương tự của ảnh . Chẳng hạn , biến đổi Fourier là biểu diễn của ảnh như là tổng các hàm phức của tần số , pha và biên độ thay đổi . Biến đổi là hữu ích với nhiều mục đích bao gồm convolution , làm giàu ảnh , nhận dạng đặc điểm và nén ảnh .

2. Phép biến đổi Fourier

- Phép biến đổi Fourier là một biểu diễn của ảnh như là tổng của các hàm mũ phức của biên độ , tần số và pha biến đổi . Biến đổi Fourier chiếm một vai trò quan trọng trong các ứng dụng xử lý ảnh bao gồm : làm giàu ảnh (hay cải thiện chất lượng ảnh – enhancement) , phân tích , phục hồi và nén ảnh .

a - Định nghĩa phép biến đổi Fourier

- Nếu f(m,n) là một hàm với hai biến không gian độc lập m và n , thì biến đổi Fourier hai chiều của hàm f(m,n) được định nghĩa bởi quan hệ :

$$F(\omega_1,\omega_2)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(m,n)e^{-j\omega_1 m}e^{-j\omega_2 n}$$

Biến ω_1 và ω_2 là các biến tần số . Hàm $F(\omega_1,\omega_2)$ được gọi là biểu diễn trong miền tần số của hàm f(m,n) . Nó là một hàm phức tuần hoàn với chu kì 2π . Do tính tuần hoàn , nên ω_1 và ω_2 thường được chọn trong khoảng $-\pi$ đến π . Chú ý rằng F(0,0) là tổng của tất cả các giá trị của f(m,n) .Vì lý do này F(0,0) thường được gọi là thành phần không đổi hoặc thành phần một chiều DC của biến đổi Fourier .

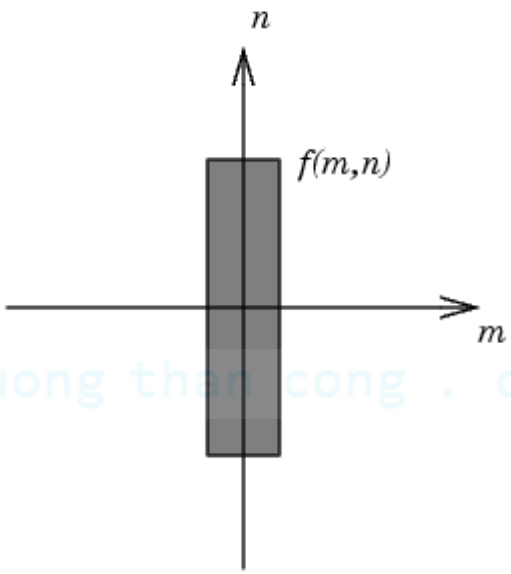
- Biến đổi Fourier ngược được cho bởi công thức :

$$f(m,n)=\frac{1}{4\pi^2}\int_{\omega_1=-\pi}^{\pi}\int_{\omega_2=-\pi}^{\pi} F(\omega_1,\omega_2)e^{j\omega_1 m}e^{j\omega_2 n}d\omega_1\omega_2$$

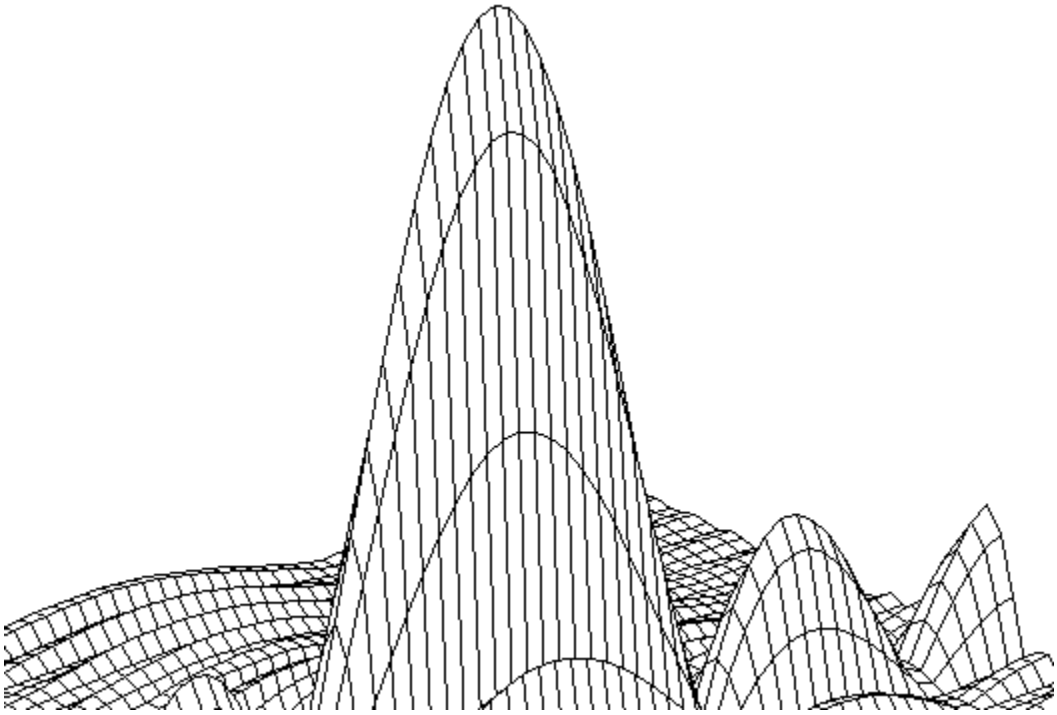
Nói chung , phương trình này có nghĩa rằng f(m,n) có thể được đại diện như là tổng vô hạn của các hàm mũ phức với các tần số khác nhau . Biên độ và pha của thành phần ở tần số (ω_1, ω_2) được lưu trong F (ω_1, ω_2)

Biểu diễn trực quan biến đổi Fourier

- Để minh hoạ , ta hãy xem hàm f(m,n) nhận giá trị bằng 1 trong khoảng hình chữ nhật (xem hình) và bằng 0 tại mọi điểm khác . Để đơn giản sơ đồ f(m,n) được coi như một hàm liên tục mặc dù m, n là các biến rời rạc.

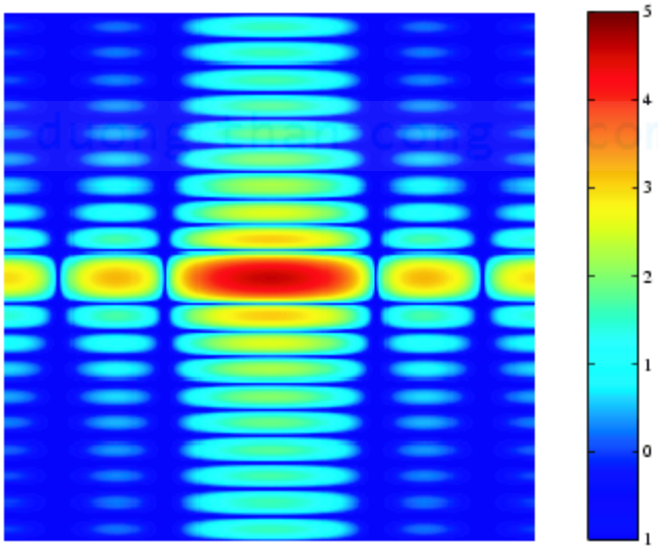


- Hình biểu diễn sau đây biểu diễn biên độ của biến đổi Fourier $|F(\omega_1, \omega_2)|$ của hàm chữ nhật như trên .

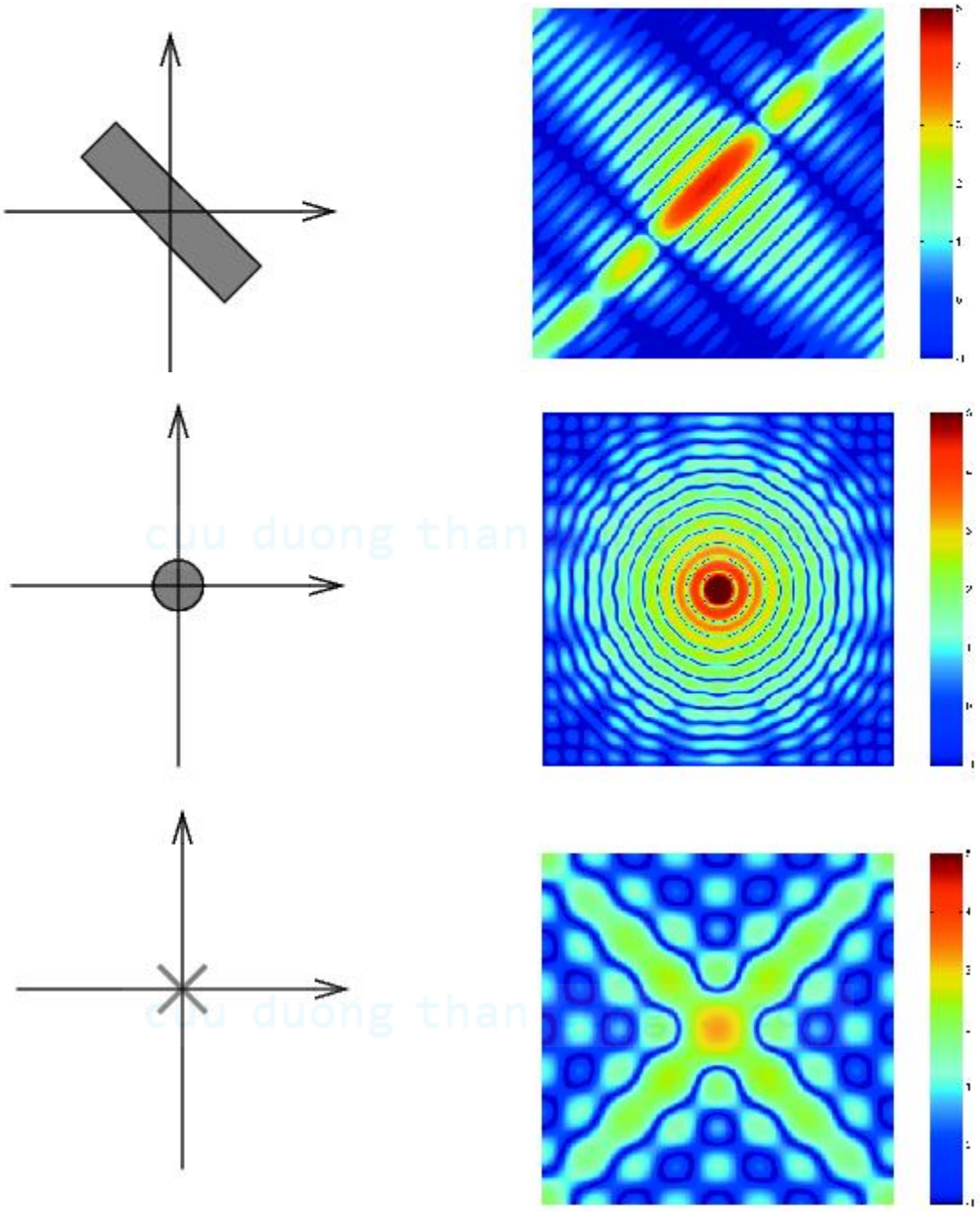


- Giá trị đỉnh ở tâm của đồ thị là F(0,0) , đó là tổng của tất cả các giá trị của hàm f(m,n) . Đồ thị cũng chỉ ra rằng $F(\omega_1, \omega_2)$ có nhiều năng lượng hơn ở vùng tần số ngang so với tần số đứng . Điều này phản ánh sự thực rằng các vùng giao ngang của hàm f(m,n) (horizontal cross sections) là các xung hẹp trong khi các vùng giao đứng của hàm này là các xung rộng . Xung hẹp mang nhiều nội dung tần số hơn xung hẹp .

Một cách khác để biểu diễn trực quan biến đổi Fourier là hiển thị hàm $\log|F(\omega_1, \omega_2)|$ như một ảnh :



- Sử dụng hàm lôgarit giúp cho việc nhận rõ đặc tính của biến đổi Fourier ở các vùng mà giá trị $F(\omega_1, \omega_2)$ gần bằng 0 . Để minh hoạ , hãy xem biểu diễn trực quan các biến đổi Fourier sau :



b- Biến đổi Fourier rời rạc

-Biến đổi Fourier trên máy tính là biến đổi Fourier rời rạc (DFT) . Có hai lý do chính để sử dụng loại biến đổi Fourier này :

+ Hàm vào và ra của biến đổi Fourier là các hàm rời rạc , điều này thích hợp cho các thao tác biến đổi trên máy tính .

+ Có một giải thuật nhanh cho việc tính toán DFT được gọi là biến đổi Fourier nhanh (FFT)

- DFT thường được định nghĩa cho các hàm rời rạc f(m,n) khác 0 và m , n lần lượt nhận các giá trị $0 \le m \le M-1$, $0 \le n \le N-1$

$$F(p,q) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{-j(2\pi/M)pm}e^{-j(2\pi/N)qn} \qquad \begin{matrix} p = 0, 1, \ldots, M-1 \\ q = 0, 1, \ldots, N-1 \end{matrix}$$

$$f(m,n) = \frac{1}{MN}\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F(p,q)e^{j(2\pi/M)pm}e^{j(2\pi/N)qn} \qquad \begin{matrix} m = 0, 1, \ldots, M-1 \\ n = 0, 1, \ldots, N-1 \end{matrix}$$

- Matlab sử dụng các hàm fft , fft2 và fftn sử dụng thuật toán biến đổi Fourier nhanh cho việc tính toán DFT một chiều , hai chiều và N chiều tương ứng . Các hàm ifft, ifft2 và ifftn tính toán DFT ngược .

Quan hệ với biến đổi Fourier

- Ta có quan hệ biểu diễn bằng công thức sau :

$$F(p,q) = F(\omega_1, \omega_2)\Big|_{\substack{\omega_1 = 2\pi p/M \\ \omega_2 = 2\pi q/N}} \qquad \begin{matrix} p = 0, 1, \ldots, M-1 \\ q = 0, 1, \ldots, N-1 \end{matrix}$$

Ví dụ :

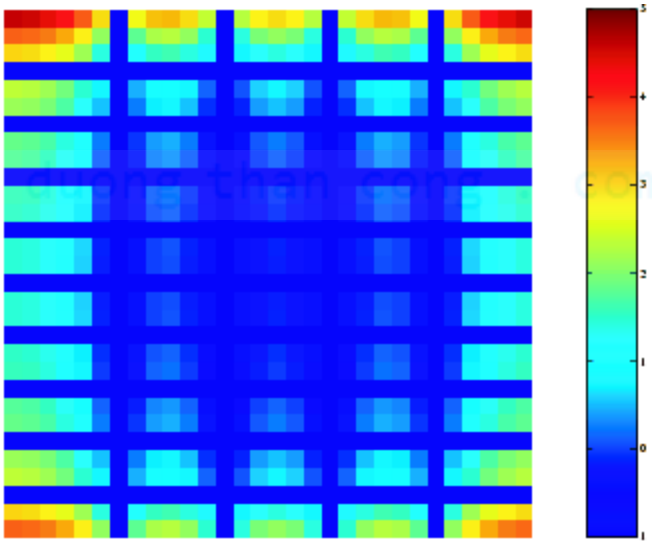
1. Tạo ma trận f tương tự như hàm f(m,n) chữ nhật đã xét trước đây . Hàm f(m,n) bằng 1 trong vùng chữ nhật và bằng 0 trong vùng khác . Sử dụng một ảnh nhị phân để thay thế cho f(m,n)

```
f = zeros(30,30);
```

```
f(5:24,13:17) = 1;
imshow(f,'notruesize')
```

2. Tính toán và biểu diễn trực quan biến đổi Fourier bởi những lệnh sau đây :

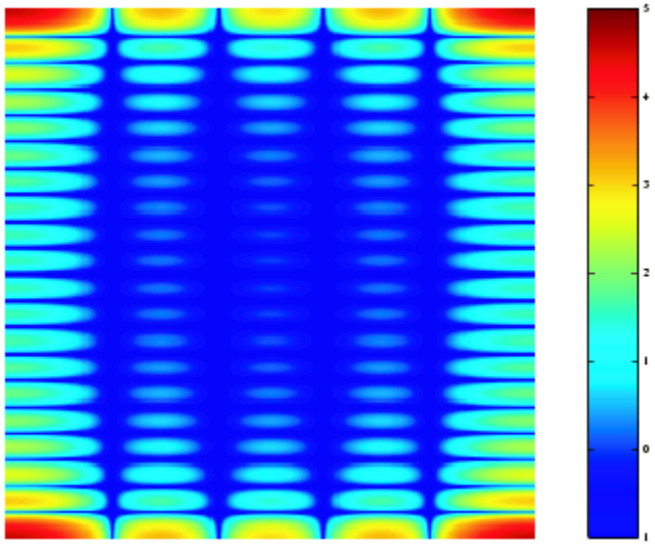
```
F = fft2(f);
F2 = log(abs(F));
imshow(F2,[-1 5],'notruesize'); colormap(jet); colorbar
```



3 . Để thu được kết quả tốt hơn , ta sử dụng các lệnh sau đây :

F = fft2(f,256,256);

imshow(log(abs(F)),[-1 5]); colormap(jet); colorbar



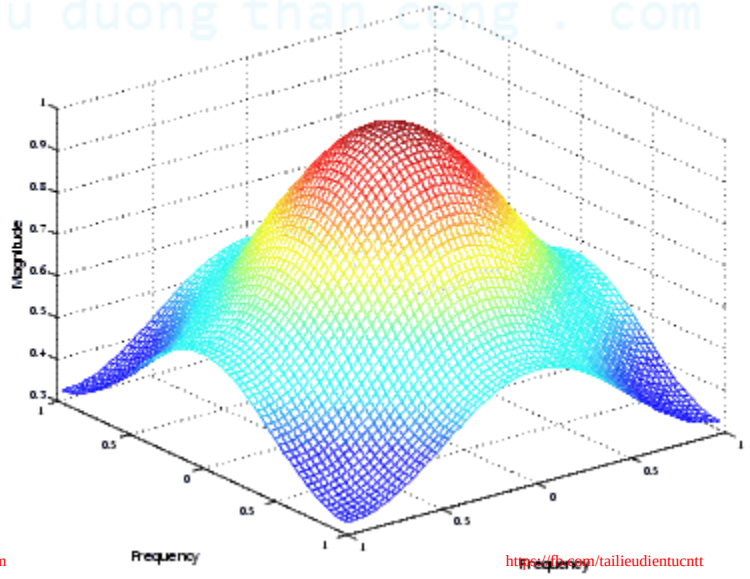
c- Các ứng dụng sử dụng phép biến đổi Fourier

Đáp ứng tần số của bộ lọc tuyến tính

- Biến đổi Fourier của đáp ứng xung của bộ lọc tuyến tính cho phép nhận được đáp ứng tần số của bộ lọc . Hàm freqz2 tính toán và hiển thị đáp ứng tần số của một bộ lọc . Đáp ứng tần số của bộ lọc Gauss chỉ ra rằng bộ lọc này là một bộ lọc thông thấp .

h = fspecial('gaussian');

freqz2(h)



Nhân chập nhanh

- Một tính chất quan trọng của biến đổi Fourier là nhân hai biến đổi Fourier tương ứng với nhân chập (convolution) của hai hàm kết hợp trong không gian . Tính chất này kết hợp với biến đổi Fourier nhanh làm cơ sở cho giải thuật nhân chập nhanh .

Chú ý : FFT dựa trên phương pháp nhân chập thường được sử dụng với đầu vào lớn . Với các đầu vào nhỏ , nhìn chung sử dụng hàm imfiler sẽ nhanh hơn .

- Để minh hoạ , ví dụ sau thực hiện nhân chập của A và B trong đó A là một ma trận MxN và B là ma trận PxQ

1. Tạo ra hai ma trận

```
A = magic(3);
B = ones(3);
```

2. Thêm các giá trị 0 vào A và B để chúng có chiều ít nhất là (M+P-1)x(N+Q-1) (chú ý rằng : hàm fft2 sẽ nhanh hơn nếu các kích thước của A và B là luỹ thừa của 2) .Ví dụ sau chèn thêm các phần tử 0 để hai ma trận có kích thước 8x8

```
A(8,8)=0;
B(8,8)=0;
```


3. Tính biến đổi Fourier hai chiều của A và B sử dụng hàm fft2

4. Nhân hai biến đổi Fourier lại với nhau

5. Tính biến đổi Fourier ngược của kết quả trên bằng hàm ifft2

```
C = ifft2(fft2(A).*fft2(B));
```

6. Trích ra thành phần khác 0 của kết quả và loại bỏ phần ảo do sai số làm tròn

```
C = C(1:5,1:5);
C = real(C)
```



```
C =

   8.0000    9.0000   15.0000    7.0000    6.0000
  11.0000   17.0000   30.0000   19.0000   13.0000
  15.0000   30.0000   45.0000   30.0000   15.0000
   7.0000   21.0000   30.0000   23.0000    9.0000
   4.0000   13.0000   15.0000   11.0000    2.000
```

Phát hiện chi tiết của ảnh

- Biến đổi Fourier có thể được sử dụng để thực thi tương quan – có liên hệ gần với nhân chập (convolution) . Tương quan có thể được sử dụng để phát hiện chi tiết trong một ảnh , trong trường hợp này , tương quan thường được gọi là *hợp mẫu* (template matching)

- Ví dụ sau minh hoạ việc sử dụng tương quan để phát hiện sự có mặt của kí tự "a" trong một ảnh có chữ :

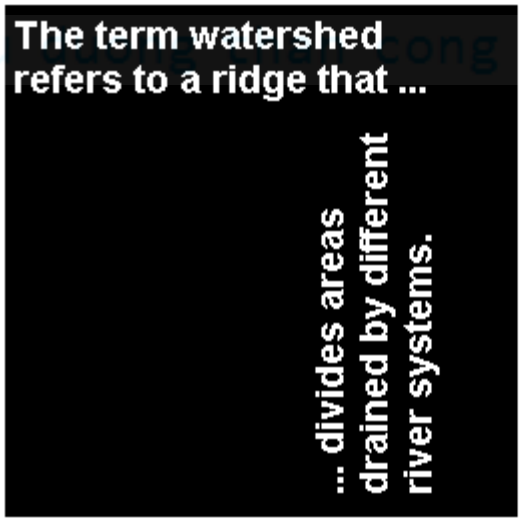
1. Đọc vào ảnh mẫu :

bw=imread('text.png');

2. Tạo một mẫu (template) để làm hợp bằng cách trích kí tự "a" từ ảnh :

a=bw(32:45,88:98);

Ta có thể tạo ảnh mẫu bằng cách sử dụng phiên bản tương tác của hàm **imcrop** , sử dụng hàm **pixval** để tính toạ độ của một chi tiết trong ảnh . Hình sau đây chỉ ra ảnh gốc và ảnh mẫu :



3. Tính toán tương quan của ảnh mẫu a với ảnh gốc bw bằng cách quay ảnh mẫu 180 độ và sau đó dùng biến đổi Fourier nhanh trên cơ sở phép nhân chập như đã đề cập trước đây (Nhân chập sẽ tương đương với tương quan nếu ta quay nhân chập 180 độ) . Để hợp với ảnh mẫu , dùng hàm fft2 và ifft2 .

C = real(ifft2(fft2(bw) .* fft2(rot90(a,2),256,256)));

Ảnh sau đây minh hoạ kết quả của tương quan . Đốm sáng trong ảnh tương ứng với sự xuất hiện của kí tự :

figure, imshow(C,[]) % Scale image to appropriate display range.

4. Để quan sát vị trí của mẫu trong ảnh , tìm giá trị pixel lớn nhất sau đó định nghĩa một giá trị ngưỡng nhỏ hơn giá trị này . Vị trí của đỉnh được chỉ ra bằng một điểm trắng trong ảnh tương quan mẫu (để xác định chúng dễ hơn , ảnh ngưỡng đã được mở rộng kích thước điểm)

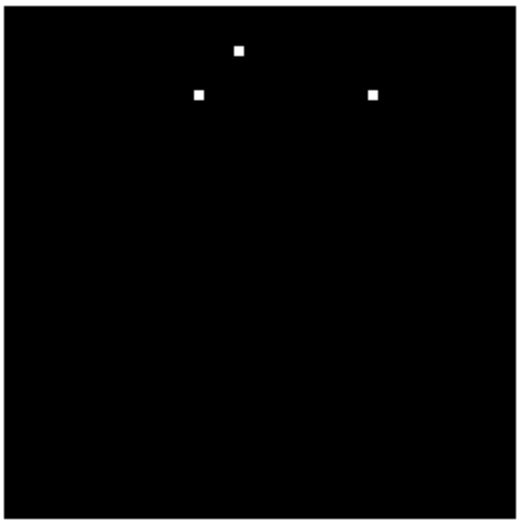
max(C(:))

ans =

68.0000

thresh = 60; % Use a threshold that's a little less than max.

figure, imshow(C > thresh)% Display showing pixels over threshold.



3. Biến đổi cô-sin rời rạc

- Biến đổi cô-sin rời rạc (DCT) biểu diễn một ảnh như là tổng của các hàm sin với biên độ và tần số biến đổi . Hàm dct2 tính DCT hai chiều của một ảnh . DCT có tính chất mà với các ảnh điển hình , hầu hết các thông tin có ý nghĩa về ảnh được tập trung vào các hệ số của DCT . Vì lý do này , DCT thường được sử dụng trong các ứng dụng nén ảnh . Chẳng hạn ,DCT là trung tâm của giải thuật nén ảnh theo chuẩn quốc tế thường được biết với tên JPEG (tên này do nhóm phát triển đặt ra : Joint Photographic Experts Group)
- DCT hai chiều của ma trận A có kích thước MxN được định nghĩa như sau :

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

- Các giá trị B_{pq} được gọi là hệ số DCT của A . DCT có thể biến đổi ngược được và biến đổi ngược của nó cho bởi công thức :

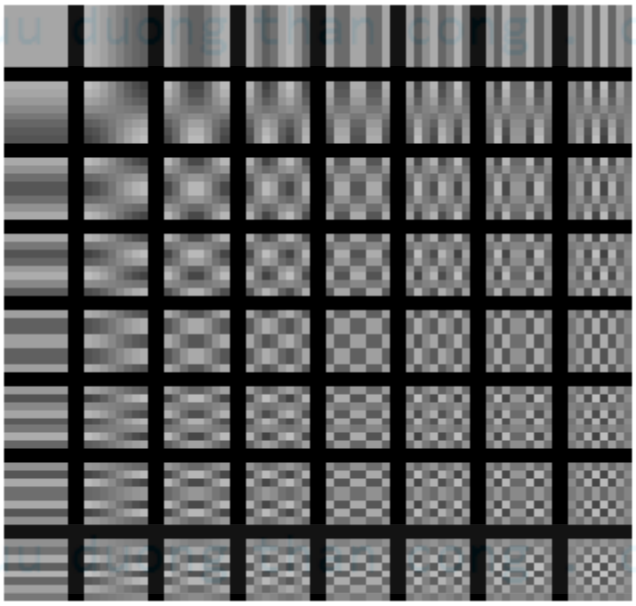
$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

- Biểu thức DCT ngược có thể được xem xét khi coi rằng mọi ma trận A kích thước MxN như là tổng của MN hàm có dạng :

$$\alpha_p \alpha_q \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

- Những hàm này được gọi là những hàm cơ sở của DCT . Hệ số DCT B_{pq} có thể được xem như trọng số cho mỗi hàm cơ sở . Với các ma trận 8x8 , 64 hàm cơ sở được minh hoạ bởi ảnh sau :



Ma trận biến đổi DCT

- Toolbox xử lý ảnh sử dụng 2 cách để tính DCT . Cách thứ nhất là dùng hàm dct2 . Hàm dct2 sử dụng giải thuật dựa trên FFT để tăng tốc tính toán với các ảnh có kích thước lớn . Cách thứ hai là sử dụng ma trận biến đổi DCT . Ma trận này được trả về từ hàm dctmtx và được sử dụng hiệu quả hơn với các ảnh có kích thước nhỏ như 8x8 , 16x16 . Ma trận biến đổi MxM – T được cho bởi :

$$T_{pq} = \begin{cases} \dfrac{1}{\sqrt{M}} & p = 0, \quad 0 \le q \le M-1 \\ \sqrt{\dfrac{2}{M}} \cos \dfrac{\pi(2q+1)p}{2M} & 1 \le p \le M-1, \quad 0 \le q \le M-1 \end{cases}$$

Với ma trận A có kích thước MxM , T*A là một ma trận MxM mà các cột của nó là DCT một chiều của các cột trong A . DCT hai chiều của A có thể thu được từ biểu thức : B=T*A*T' . Do T là ma trận trực giao nên ma trận nghịch đảo của nó cũng là ma trận chuyển vị nên DCT hai chiều của B là T'*B*T

DCT và nén ảnh

- Trong giải thuật nén ảnh JPEG , ảnh vào được chia làm các khối có kích thước 8x8 hoặc 16x16 và DCT hai chiều được tính cho mỗi khối . Hệ số DCT sau đó được lượng tử hoá , mã hoá và truyền tải . Bộ nhận JPEG (hoặc chương đọc file JPEG) sẽ giải mã hệ số đã được lượng tử của DCT , tính DCT ngược cho mỗi khối và sau đó đặt các khối lại với nhau trong một ảnh duy nhất . Với các ảnh điển hình , nhiều hệ số DCT có giá trị gần 0 , những hệ số này có thể bị bỏ qua mà không ảnh hưởng nhiều đến chất lượng của ảnh nén
- Ví dụ sau tính DCT hai chiều của các khối 8x8 trong một ảnh , bỏ qua (đặt bằng 0) tất cả ngoại trừ 10 trong số 64 hệ số DCT của mỗi khối và sau đó xây dựng lại ảnh bằng DCT hai chiều ngược của mỗi khối . Phương pháp ma trận chuyển đổi DCT được sử dụng :

```
I = imread('cameraman.tif');
I = im2double(I);
```

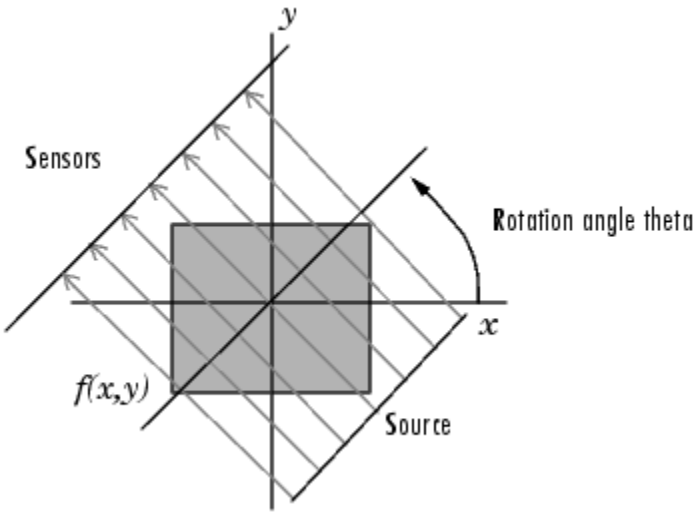
```
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
imshow(I), figure, imshow(I2)
```

- Mặc dù chất lượng của ảnh nén bị suy giảm , nó vẫn rõ nét thậm chí gần 85% hệ số DCT bị bỏ qua .

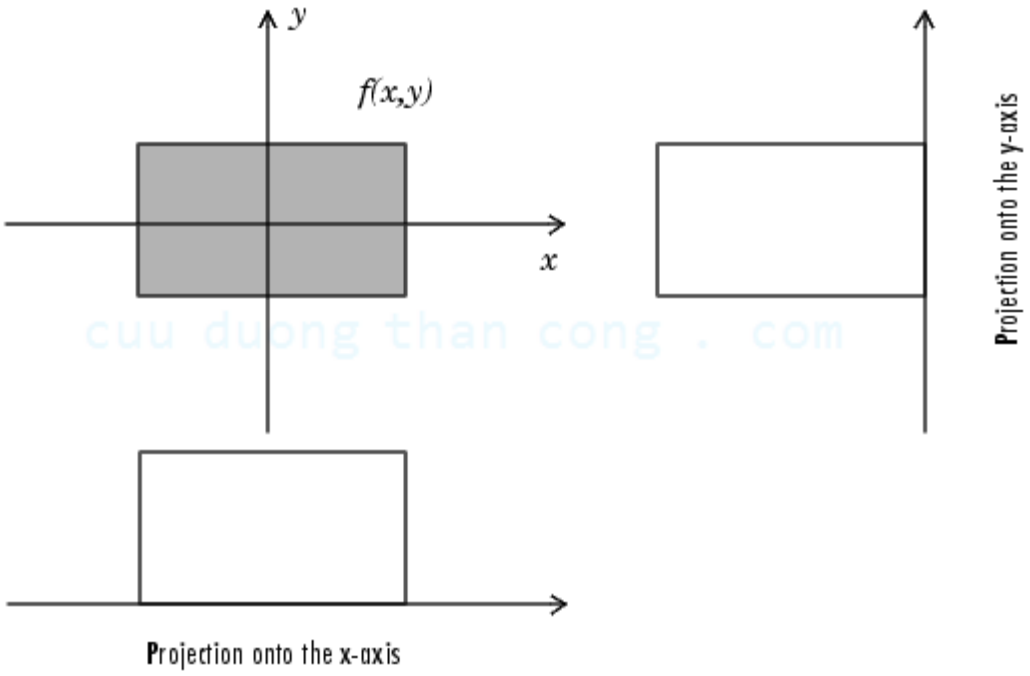
4 . Biến đổi Radon

- Hàm radon trong toolbox xử lý ảnh tính ***projections*** (phép chiếu) của một ma trận ảnh theo một chiều nhất định . Phép chiếu của một hàm f(x,y) là một tập các tích phân đường . Hàm radon tính tích phân đường từ nhiều nguồn dọc theo các đường song song hay tia theo một chiều nào đó . Các tia này cách nhau 1 đơn vị pixel . Để biểu diễn một ảnh ,

hàm radon lấy nhiều phép chiếu song song của một ảnh từ các góc độ khác nhau bằng cách quay nguồn xung quanh một tâm quay là tâm của ảnh . Hình sau chỉ ra một phép chiếu đơn ở một góc quay nhất định .



Chẳng hạn , tích phân đường (line integral) của f(x,y) theo chiều thẳng đứng là phép chiếu của f(x,y) trên trục x , tích phân đường theo chiều nằm ngang là phép chiếu của f(x,y) trên trục y . Hình sau mô tả các phép chiếu dọc , ngang của hàm hai chiều :



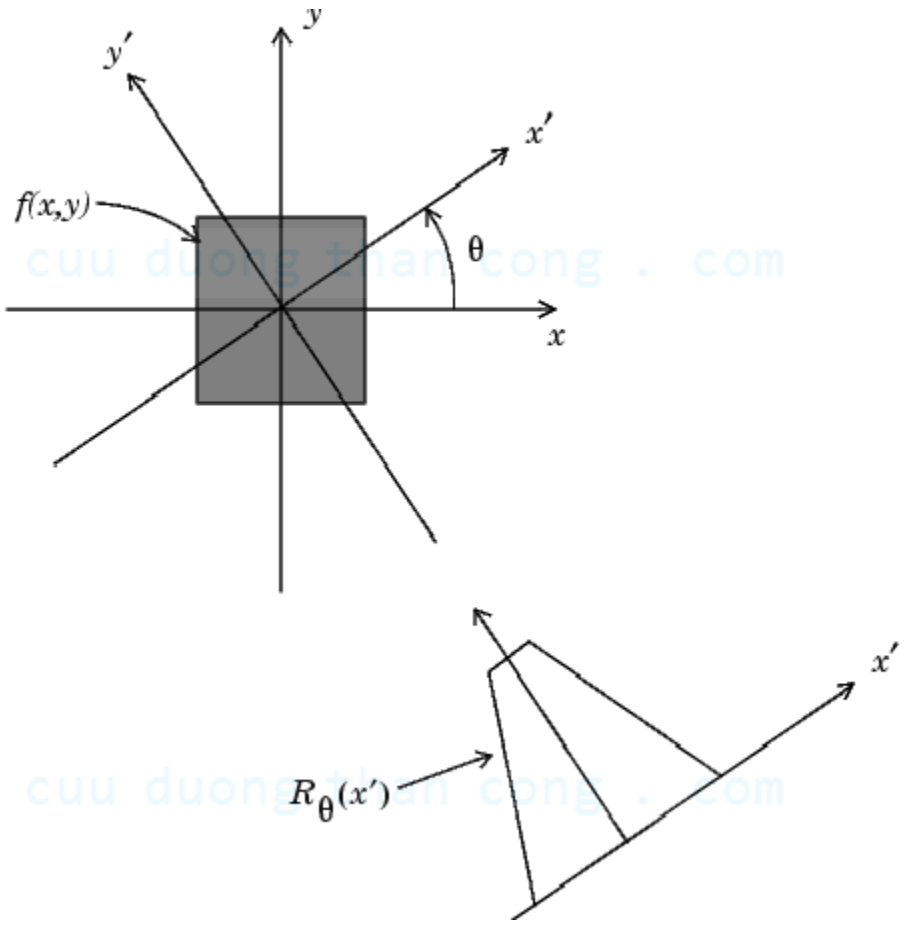
- Phép chiếu có thể được tính dọc theo bất kì góc thê-ta nào . Nhìn chung , biến đổi Radon của hàm f(x,y) là tích phân đường của f song song với trục y .

$$R_\theta(x') = \int_{-\infty}^{\infty} f(x'\cos\theta - y'\sin\theta,\ x'\sin\theta + y'\cos\theta)\, dy'$$

where

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

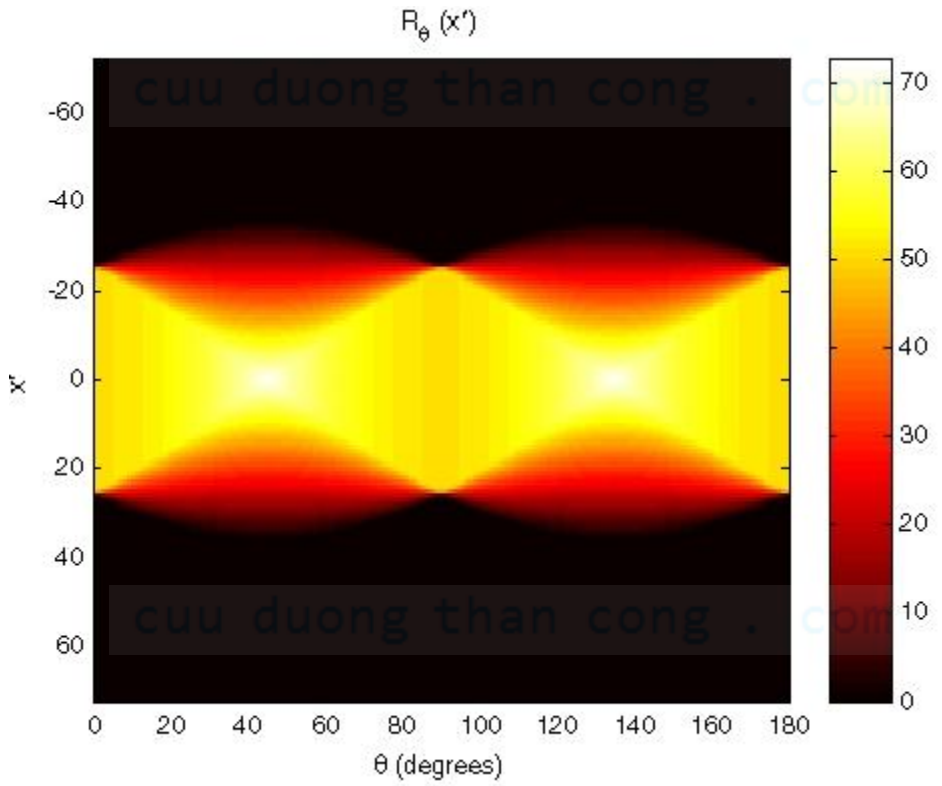
Hình sau đây mô tả hình học của biến đổi Radon



Xem biến đổi Radon như một ảnh

- Biến đổi Radon cho một số lượng lớn các góc thường được hiển thị như một ảnh . Trong ví dụ này , biến đổi Radon cho ảnh vuông được tính giữa các góc từ 0 đến 180 độ , bước góc là 1 độ .

```
theta = 0:180;
[R,xp] = radon(I,theta);
imagesc(theta,xp,R);
title('R_{\theta} (X\prime)');
xlabel('\theta (degrees)');
ylabel('X\prime');
set(gca,'XTick',0:20:180);
colormap(hot);
colorbar
```



Dùng biến đổi Radon để phát hiện đường thẳng

- Biến đổi Radon có quan hệ gần gũi với các thao tác trực quan trên máy tính được biết đến với biến đổi Hough . Ta có thể sử dụng hàm radon để thi hành một dạng của biến đổi Hough phục vụ cho việc phát hiện các đường thẳng . Các bước như sau :

1. Tính các cạnh của ảnh nhị phân dùng hàm edge

```
I = fitsread('solarspectra.fts');
I = mat2gray(I);
BW = edge(I);
imshow(I), figure, imshow(BW)
```

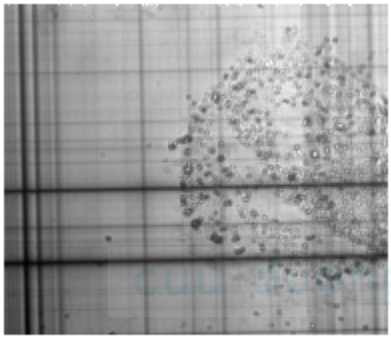
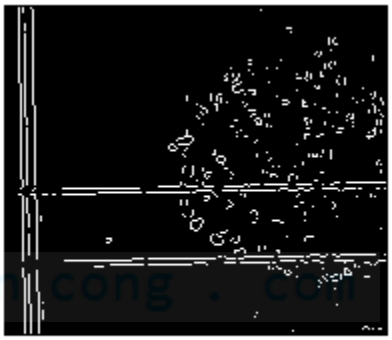


Image Courtesy of Ann Walker

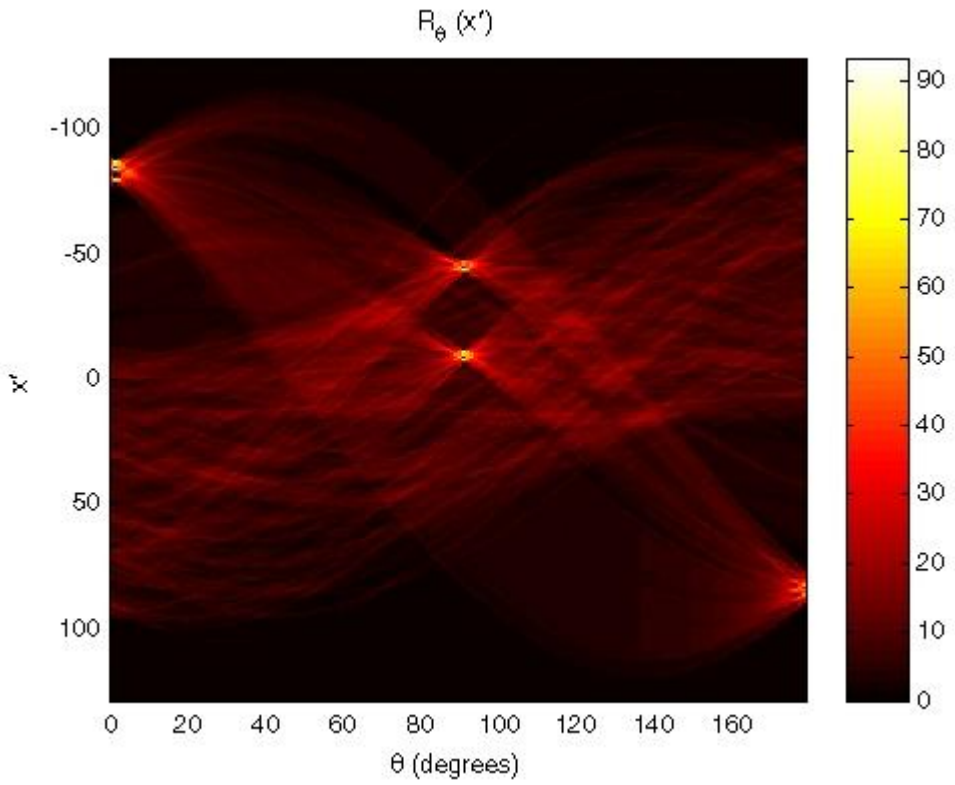
Original Image



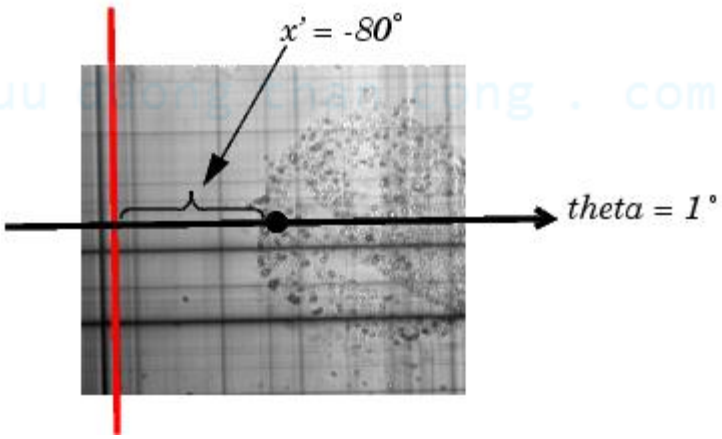
Edge Image

2. Tính biến đổi radon của ảnh thu được

```
theta = 0:179;
[R,xp] = radon(BW,theta);
figure, imagesc(theta, xp, R); colormap(hot);
xlabel('\theta (degrees)'); ylabel('x\prime');
title('R_{\theta} (x\prime)'); colorbar
```



3. Tìm vị trí của đỉnh trong ma trận biến đổi Radon . Những vị trí này tương ứng với các vị trí của đường thẳng trong ảnh gốc . Trong hình sau , đỉnh cao nhất R tương ứng với thê-ta=1 độ và x'=-80 . Đường thẳng vuông góc tại thê-ta=1 và được xác định bởi x'=-80 như hình vẽ dưới đây . Biểu diễn hình học của biến đổi Radon là đường thẳng có mũi tên nằm ngang . Chú ý rằng các đường thẳng song song khác với đường thẳng đứng trong hình cũng xuất hiện tại đỉnh mà thê-ta=1 độ trong phép biến đổi . Cũng vậy , những đường thẳng vuông góc với chúng xuất hiện ở đỉnh mà thê-ta=91 độ



Biến đổi Radon ngược

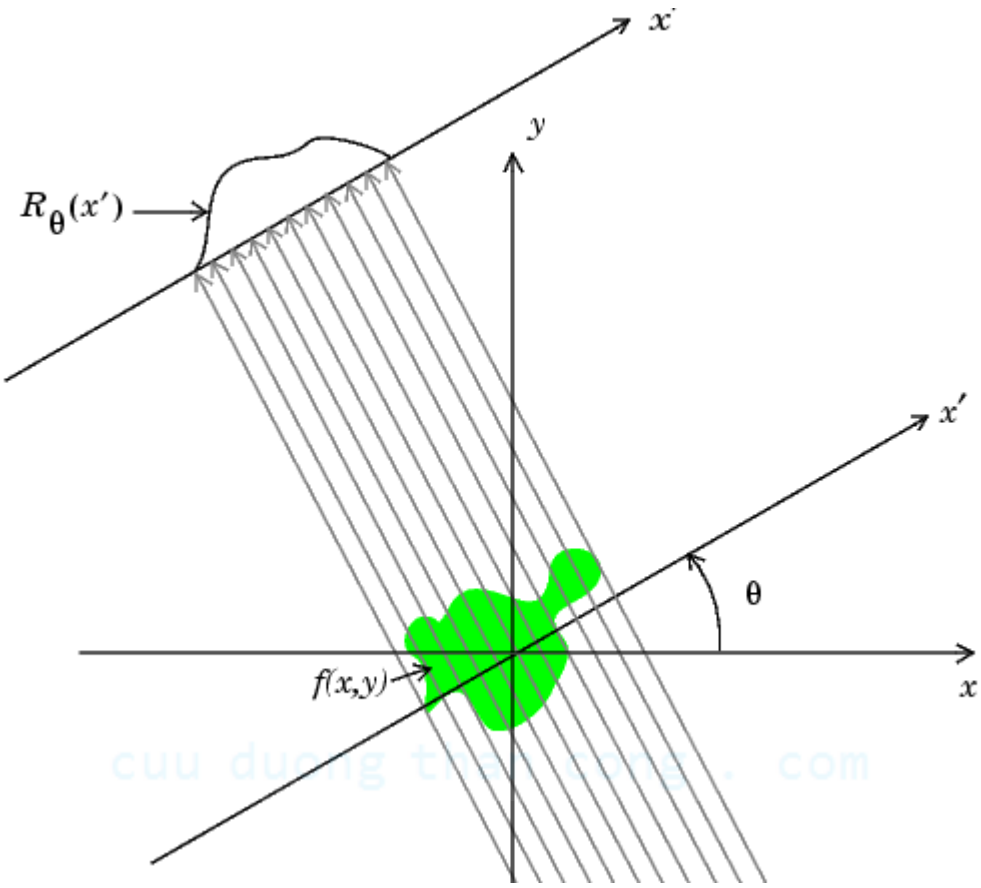
- Hàm iradon thi hành biến đổi Radon ngược , chúng thường được sử dụng trong các ứng dụng ***tomography*** . Biến đổi Radon ngược có thể được sử dụng để tạo các ảnh từ dữ liệu phép chiếu .
- Như đã đề cập trước đây , đưa ra ảnh I và thiết lập góc thê-ta , hàm radon có thể được sử dụng để tính biến đổi Radon

R=radon(I,theta);

Hàm iradon sau đó được gọi để tạo lại ảnh I

IR=iradon(R,theta);

- Trong ví dụ trên , phép chiếu được tính toán từ ảnh gốc I . Trong hầu hết các ứng dụng , không có ảnh gốc nào cả để từ đó phép chiếu được định dạng . Chẳng hạn , **tomography** hấp thụ tia X , phép chiếu được tạo ra bằng cách đo đạc sự suy giảm phát xạ xuyên qua một mẫu vật lý ở các góc độ khác nhau . Ảnh gốc có thể xem như vùng giao nhau với vật mẫu , trong đó , cường độ đại diện cho mật độ của vật mẫu . Phép chiếu được thu thập sử dụng phần cứng đặc biệt và sau đó ảnh bên trong vật mẫu được tạo lại với hàm iradon . Điều này ngăn ngừa việc tràn ảnh trong các vùng trong suốt khác bên trong vật mẫu .
- Hàm iradon tạo lại một ảnh từ phép chiếu song sogn . Trong hình học chiếu song song , mỗi phép chiếu được tạo thành bởi một tập hợp các tích phân đường thông qua một ảnh ở một góc nhất định .
- Hình sau đây minh hoạ hình học song song được sử dụng ra sao trong kĩ thuật hấp thụ tia X . Chú ý rằng số lượng bộ phát và số lượng sensor là bằng nhau – n . Mỗi sensor sẽ đo sự phát xạ được phát ra từ các bộ phát tương ứng và độ suy giảm phát xạ cho phép đo đạc mật độ khối lượng của vật thể . Điều này tương ứng với tích phân đường được tính toán trong biến đổi Radon .



- Một hình học khác cũng hay được sử dụng là hình học fan-beam , trong đó có một nguồn và n sensor .

Cải thiện kết quả

- Hàm iradon sử dụng giải thuật ***filtered backprojection*** để tính toán biến đổi Radon ngược . Giải thuật này tạo một xự xấp xỉ cho ảnh I trên cơ sở của phép chiếu trong các cột của R . Một kết quả chính xác hơn có thể thu được bằng cách sử dụng nhiều phép chiếu khi tái tạo ảnh . Khi số lượng phép chiếu tăng (chiều dài của thê-ta) , ảnh được tái tạo IR chính xác hơn . Véctơ theta phải chứa giá trị góc tăng đều với một hằng số delta

theta . Khi delta theta được biết , nó có thể được truyền tới hàm iradon thay cho chuỗi giá trị theta . Đây là một ví dụ :

IR = iradon(R,Dtheta);

Giải thuật kể trên sẽ lọc phép chiếu trong R và sau đó tạo lại ảnh sử dụng phép chiếu đã được lọc . Trong một số trường hợp , nhiễu có thể xuất hiện trong phép chiếu . Để loại bỏ nhiễu tần số cao , đặt một cửa sổ tới bộ lọc để làm suy giảm nhiễu . Nhiều bộ lọc cửa sổ như vậy có thể dùng được với hàm iradon . Ví dụ sau gọi hàm iradon áp đặt một cửa sổ Hamming tới bộ lọc :

IR = iradon(R,theta,'Hamming');

iradon cũng cho phép ta chỉ ra tần số đã được chuẩn hoá , D - ở trên , với bộ lọc có đáp ứng 0 . D phải có giá trị nằm trong khoảng [0,1] . Với lựa chọn này , trục tần số được định tỉ lệ lại để cho toàn bộ bộ lọc được nén đầy trong khoảng tần số [0,D] . Điều này có thể hữu ích trong các trường hợp mà phép chiếu bao gồm một chút thông tin về tần số cao nhưng có nhiễu tần số cao . Trong trường hợp này , nhiễu có thể hoàn toàn bị loại bỏ mà không làm ảnh hưởng đến sự khôi phục ảnh . Lời gọi sau đến hàm iradon sẽ thiết lập tần số được tiêu chuẩn hoá bằng 0.85

IR=iradon(R,theta,0.85)

Ví dụ : Tái tạo một ảnh từ các dữ liệu phép chiếu song song

- Những lệnh dưới đây chỉ ra cách tái tạo một ảnh từ dữ liệu của các phép chiếu song song . Ảnh kiểm tra có thể được tạo ra bởi hàm **phantom** . Ảnh phantom minh hoạ nhiều đặc tính được tìm thấy trong ảnh y học về đầu người . Hình elíp sáng bao bề ngoài của ảnh giống như một hộp sọ và các elíp bên trong giống như các chi tiết bên trong não bộ .

1. Tạo ảnh phantom

P = phantom(256);

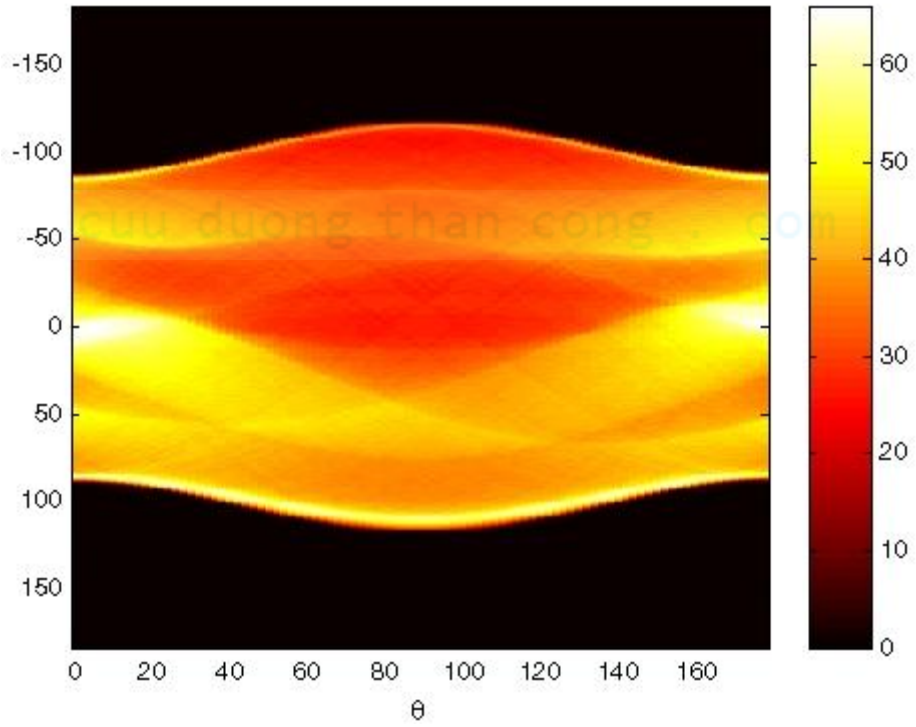
imshow(P)

2. Tính biến đổi Radon của ảnh phantom với 3 tập giá trị khác nhau của theta . R1 có 18 phép chiếu , R2 có 36 và R3 có 90 .

```
theta1 = 0:10:170; [R1,xp] = radon(P,theta1);
theta2 = 0:5:175;  [R2,xp] = radon(P,theta2);
theta3 = 0:2:178;  [R3,xp] = radon(P,theta3);
```

3. Hiển thị đồ thị của biến đổi Radon của ảnh phantom . Hình sau chỉ ra R3 biến đổi với 90 phép chiếu .

```
figure, imagesc(theta3,xp,R3); colormap(hot); colorbar
xlabel('\theta'); ylabel('x\prime');
```

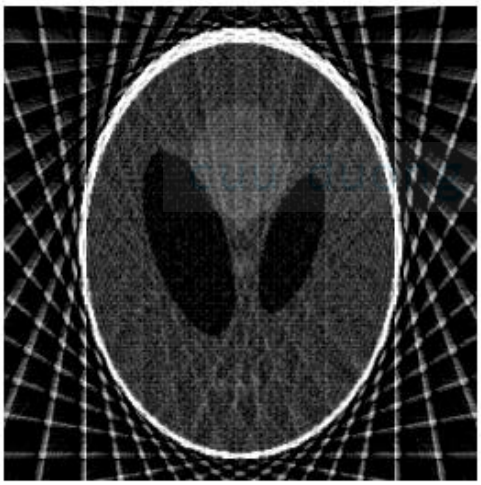


Chú ý rằng một số đặc điểm của ảnh ban đầu xuất hiện trong ảnh này đã bị biến đổi . Cột đầu tiên trong biến đổi Radon tương ứng với phép chiếu ở 0 độ mà được lấy tích phân theo chiều thẳng đứng . Cột gần giữa nhất tương ứng với phép chiếu ở 90 độ được lấy tích phân theo chiều nằm ngang . Phép chiếu ở 90 độ có một profile rộng hơn so với phép chiếu ở 0 độ .

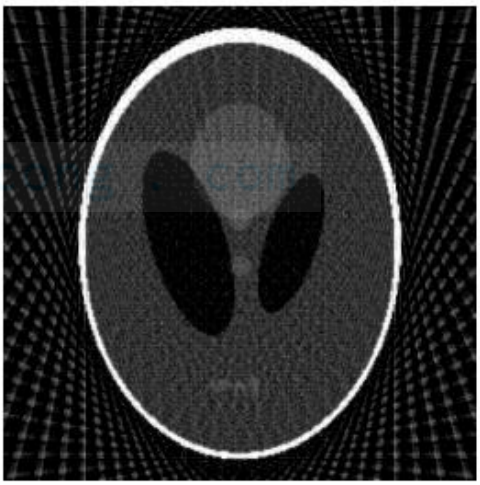
4. Tạo lại ảnh gốc từ dữ liệu phép chiếu đã tạo trong bước 2 và hiển thị kết quả .

```
I1 = iradon(R1,10);
I2 = iradon(R2,5);
I3 = iradon(R3,2);
imshow(I1)
figure, imshow(I2)
figure, imshow(I3)
```

- Hình sau biểu diễn kết quả của 3 ảnh tái tạo lại . Chú ý rằng ảnh I1 được tạo lại chỉ từ 18 phép chiếu thì kém chính xác nhất . Ảnh I2 với 36 phép chiếu – khá hơn nhưng vẫn không đủ trong để phân biệt các hình elíp nhỏ . Ảnh I3 , sử dụng 90 phép chiếu gần giống với ảnh gốc .



I1



I2

I3

Ngoài phép chiếu tia song song , người ta cũng dùng phép chiếu tia hình quạt để biến đổi ảnh (fan-beam projection) . Để sử dụng phép chiếu này , dùng hàm **fanbeam** của toolbox .

VI - Biến đổi không gian ảnh

- Biến đổi không gian ảnh là thực hiện ánh xạ giữa vị trí các pixel trong ảnh vào với các pixel trong ảnh ra .

1 . Bảng thuật ngữ :

Tên thuật ngữ	Diễn giải
Aliasing	Răng cưa - xuất hiện khi giảm kích thước ảnh . Khi kích thước của một ảnh bị giảm , các pixel gốc bị lấy mẫu giảm để tạo ra ít pixel hơn . Aliasing xảy ra như kết quả của việc giảm kích thước ảnh thường xuất hiện dưới dạng bậc thang (đặc biệt trong các ảnh có độ tương phản cao)
Antialiasing	Các biện pháp chống răng cưa cho ảnh
Bicubic interpolation	Giá trị của các pixel ra được tính toán từ giá trị trung bình của 4x4 pixel lân cận
Bilinear interpolation	Gía trị của pixel ra được tính toán từ giá trị trung bình của 2x2 pixel lân cận
Geometric operation	Một thao tác sửa đổi quan hệ hình học gữa các pixel trong một ảnh. Chẳng hạn thay đổi kích thước ảnh , quay ảnh và xén ảnh
Interpolation	Quá trình được sử dụng để ước lượng giá trị ảnh ở một vị trí giữa các pixel
Nearest-neighbor interpolation	Các giá trị pixel ra được gán giá trị của pixel nằm trong một vùng gần pixel đó .

2. Nội suy

- Nội suy là quá trình sử dụng để ước lượng một giá trị ảnh ở một vị trí giữa các pixel . Chẳng hạn , nếu ta thay đổi kích thước một ảnh , nó sẽ chứa nhiều pixel hơn ảnh gốc , toolbox sử dụng sự nội suy để tính giá trị cho các pixel thêm vào . Hàm **imresize** và **imrotate** sử dụng nội suy hai chiều để thực hiện thao tác của mình . Hàm **improfile** cũng sử dụng sự nội suy hoá .

Các phương pháp nội suy

- Toolbox sử lý ảnh cung cấp 3 cách nội suy hoá
+ Nội suy các pixel gần nhất (nearest –neighbor interpolation)
+ Nội suy song tuyến tính (Bilinear interpolation)
+ Nội suy song khối (Bicubic interpolation)
Các phương pháp nội suy làm việc theo một cách giống nhau . Trong mỗi trường hợp , để tính giá trị của một pixel đã được nội suy , chúng tìm điểm trong ảnh ra mà pixel nằm tại đó . Sau đó , chúng gán một giá trị tới các pixel ra bằng cách tính toán giá trị trung bình có trọng số của một số pixel lân cận . Trọng số dựa trên cơ sở khoảng cách tới điểm đang xét .
- Các phương pháp này khác nhau ở tập các pixel mà chúng xem xét :
+ Với nội suy các pixel gần nhất : pixel ra được gán giá trị của các pixel ở gần nó nhất . Các pixel khác không được xem xét .
+ Nội suy song tuyến tính , giá trị của pixel ra là giá trị trung bình theo trọng số của 2x2 pixel lân cận .
+ Nội suy song khối : giá trị của pixel ra là trung bình có trọng số của 4x4 pixel lân cận .
Số lượng các pixel được xem xét ảnh hưởng đến độ phức tạp tính toán . Vì vậy , phương pháp song tuyến tính mất nhiều thời gian hơn phương pháp thứ nhất và phương pháp song khối mất nhiều thời gian hơn song tuyến tính . Tuy nhiên , số lượng pixel lớn hơn , độ chính xác sẽ tốt hơn .

Kiểu ảnh

- Các hàm sử dụng tuyến tính yêu cầu một tham số chỉ ra phương pháp nội suy . Với hầu hết các hàm , phương pháp mặc định được sử dụng là nearest-neighbor interpolation .

Phương pháp này tạo ra một kết quả có thể chấp nhận được cho hầu hết các ảnh và là phương pháp duy nhất thích hợp với ảnh chỉ số . Với ảnh cường độ hay RGB , tuy nhiên ta thường chỉ ra kiểu song tuyến tính hoặc song khối bởi vì những phương pháp này cho kết quả tốt hơn

Với ảnh RGB , nội suy thường được thực hiện trên mặt phẳng R,B,G một cách riêng biệt

Với ảnh nhị phân , nội suy gây ra những ảnh hưởng mà ta có thể nhận thấy được . Nếu sử dụng nội suy song tuyến tính hoặc song khối , giá trị tính toán được cho pixel trong ảnh ra sẽ không hoàn toàn là 0 hoặc 1. Ảnh hưởng trên ảnh kết quả phụ thuộc vào lớp của ảnh vào :

+ Nếu lớp ảnh vào là double , ảnh ra là một ảnh đen trắng thuộc lớp double . Ảnh ra không là ảnh nhị phân bởi vì nó bao gồm các giá trị khác 0 và 1 .

+ Nếu ảnh vào là uint8 , ảnh ra là một ảnh nhị phân thuộc lớp uint8 . Giá trị của các pixel được nội suy được làm tròn thành 0 hoặc 1 . Vì vậy , ảnh ra thuộc lớp uint8 .

Nếu sử dụng phương pháp nearest-neighbor interpolation , ảnh ra luôn là ảnh nhị phân bởi vì những giá trị của pixel được nội suy được lấy trực tiếp từ ảnh vào .

3. Thay đổi kích thước ảnh

- Để thay đổi kích thước của một ảnh , sử dụng hàm imresize . Sử dụng hàm này ta có thể :

+ Chỉ ra kích thước của ảnh kết quả

+ Chỉ ra phương pháp nội suy được sử dụng

+ Chỉ ra bộ lọc được sử dụng để ngăn ngừa hiện tượng răng cưa

Chỉ ra kích thước cho ảnh kết quả

- Sử dụng hàm imresize , ta chó thể chỉ ra kích thước của ảnh kết quả theo hai cách :

+ Bằng cách chỉ ra hệ số phóng đại được sử dụng trên ảnh

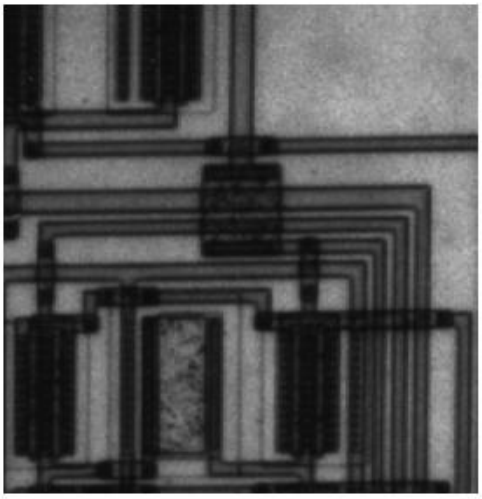
+ Bằng cách chỉ ra chiều của ảnh kết quả

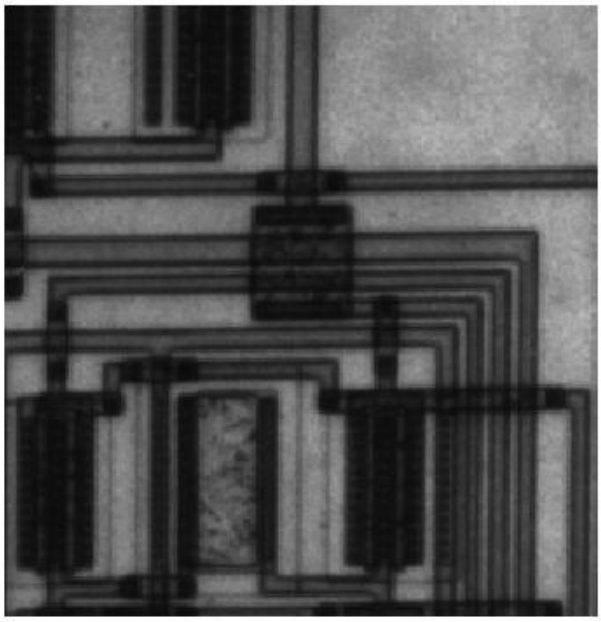
Sử dụng hệ số phóng đại ảnh

- Để mở rộng một ảnh , chỉ ra hệ số phóng đại lớn hơn 1 . Để thu nhỏ một ảnh , chỉ ra hệ số phóng đại nằm giữa 0 và 1 . Chẳng hạn , lệnh sau tăng kích thước của ảnh I lên 1.25 lần :

```
I = imread('circuit.tif');
J = imresize(I,1.25);
imshow(I)
```

figure, imshow(J)





Chỉ định kích thước của ảnh ra

- Ta có thể chỉ ra kích thước của ảnh ra bằng cách truyền một véc tơ chứa số lượng hàng và cột của ảnh sau cùng . Những lệnh sau đây tạo một ảnh ra Y với 100 hàng và 150 cột .

Y = imresize(X,[100 150])

Chú ý : Nếu kích thước được chỉ ra không có cùng tỉ lệ với ảnh vào , ảnh ra sẽ bị biến dạng

Chỉ đinh phương pháp nội suy được sử dụng

- Theo mặc định , hàm imresize sử dụng phương pháp nội suy các pixel gần nhất (nearest – neighbor interpolation) để tính giá trị các pixel của ảnh ra . Tuy nhiên , ta có thể chỉ định các phương pháp nội suy khác .Bảng sau đây liệt kê các phương pháp nội suy được trợ giúp theo thứ tự của độ phức tạp .

Giá trị tham số	Phương pháp nội suy
‘nearest’	Nội suy các phixel gần nhất (mặc định)

‘bilinear’	Nội suy song tuyến tính
‘biculic’	Nội suy song khối

Trong ví dụ sau , hàm imresize sử dụng phương pháp nội suy song tuyến tính :

Y=imresize(X, [100 150],’bilinear’);

Sử dụng bộ lọc để ngăn chặn hiện tượng răng cưa

- Việc giảm kích thước (hình học) của một ảnh có thể gây ra những ảnh hưởng nhất định lên ảnh chẳng hạn như hiện tượng xuất hiện răng cưa tại biên của ảnh . Điều này là do thông tin luôn bị mất khi ta giảm kích thước một ảnh . Răng cưa xuất hiện như những gợn sóng trong ảnh sau cùng .

- Khi giảm kích thước của ảnh sử dụng nội suy song tuyến tính hoặc song khối , hàm imresize tự động áp đặt một bộ lọc thông thấp lên ảnh trước khi nội suy . Điều này để giảm ảnh hưởng của răng cưa trong ảnh ra . Ta có thể chỉ ra kích thước của bộ lọc này hoặc chỉ ra một bộ lọc khác thay thế .

Chú ý : Thậm chí đã sử dụng một bộ lọc thông thấp , chất lượng của ảnh vẫn bị ảnh hưởng do thông tin luôn bị mất trong quá trình nội suy

- Hàm imresize không áp đặt một bộ lọc thông thấp lên ảnh nếu phương pháp nội suy các pixel gần nhất được sử dụng . Phương pháp nội suy này ban đầu được sử dụng với các ảnh chỉ số và bộ lọc thông thấp không thích hợp cho kiểu ảnh này .

- Ta cũng có thể chỉ ra một bộ lọc tự tạo thay cho các bộ lọc có sẵn .

Hàm imresize

Cú pháp của hàm này như sau :

B = imresize(A,m)

B = imresize(A,m,method)

B = imresize(A,[mrows ncols],method)

B = imresize(...,method,n)

B = imresize(...,method,h)

Diễn giải

+ B=imresize(A,m) : Trả lại một ảnh B lớn gấp m lần ảnh A (kích thước hình học) sử dụng phương pháp nội suy mặc định (nearest – neighbor interpolcation) . A có thể là một ảnh chỉ số , ảnh đen trắng , RGB hoặc ảnh nhị phân . Nếu m nằm giữa 0 và 1 , B sẽ nhỏ hơn A . Nếu m lớn hơn 1 , B sẽ lớn hơn A .

+ B=imresize(A,m,method) : Trả lại một ảnh lớn gấp m lần ảnh A sử dụng phương pháp nội suy method . method là một chuỗi chỉ ra phương pháp nội suy nào được sử dụng chẳng hạn : ‘nearest’,’bilinear’,’bicubic’ .

+B=imresize(A, [mrows ncols],method) : Trả lại một ảnh với kích thước được chỉ ra bởi véc tơ [mrows ncols] . Nếu kích thước được chỉ ra không cùng tỉ lệ với ảnh vào , ảnh sẽ bị biến dạng

Khi kích thước của ảnh ra nhỏ hơn kích thước của ảnh vào và phương pháp nội suy được sử dụng là ‘bilinear’ hoặc ‘bicubic’ , hàm imresize áp đặt một bộ lọc thông thấp trước khi tuyến tính hoá để giảm hiện tượng răng cưa . Kích thước mặc định là 11x11 .

Ta có thể chỉ ra một thứ tự khác cho bộ lọc mặc định sử dụng cấu trúc :

B=imresize(…,method,n) : n là một số nguyên chỉ ra kích thước của bộ lọc – nxn . Nếu n=0 , hàm imresize bỏ qua bước lọc . Ta cũng có thể chỉ ra bộ lọc riêng sử dụng cú pháp :

B=imresize(…,method,h) : Trong đó h là một bộ lọc FIR hai chiều (có thể được trả về bởi các hàm ftrans2, fwind1 , fwind2 hoặc fsamp2) .

4. Quay ảnh

- Để quay một ảnh , sử dụng hàm imrotate . Hàm này chấp nhận hai tham số chính :

+ Ảnh cần quay

+ Góc quay

-Góc quay tính theo độ . Nếu ta chỉ ra một giá trị dương , hàm imrotate quay ảnh theo chiều ngược chiều kim đồng hồ . Nếu chỉ ra giá trị âm , hàm quay ảnh theo chiều kim đồng hồ . Ví dụ sau quay một ảnh 35 độ theo chiều ngược chiều kim đồng hồ :

J=imrotate(I,35) ;

- Một số tham số tuỳ chọn ta có thể truyền vào cho hàm bao gồm :

+ Phương pháp nội suy được sử dụng

+ Kích thước của ảnh ra

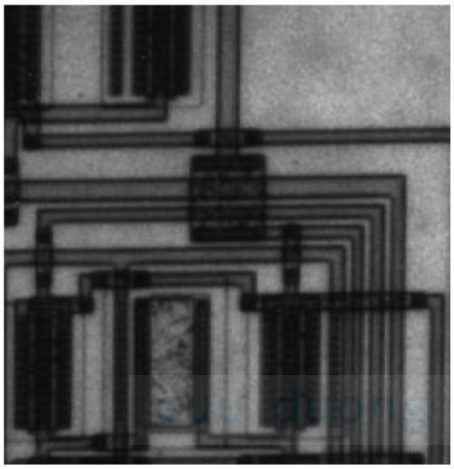
Chỉ định phương pháp nội suy được sử dụng

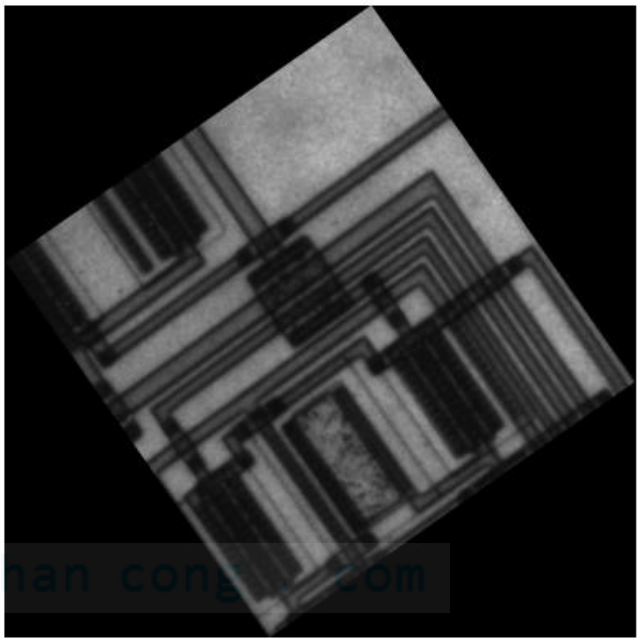
- Theo mặc định , hàm imrotate sử dụng phương pháp nội suy thứ nhất (nearest-neighbor interpolation) để tính giá trị các pixel trong ảnh ra . Tuy nhiên ,ta có thể chỉ ra các phương pháp nội suy khác như : ‘bilinear ‘,’bicubic’

Ví dụ sau quay một ảnh 35 độ ngược chiều kim đồng hồ sử dụng nội suy song tuyến tính :

I = imread('circuit.tif');

```
J = imrotate(I,35,'bilinear');
imshow(I)
figure, imshow(J)
```





Chỉ định kích thước của ảnh ra

Theo mặc định , hàm imrotate tạo một ảnh ra đủ lớn để có thể bao gồm toàn bộ các pixel của ảnh gốc . Các pixel nằm ngoài biên của ảnh gốc được gán giá trị 0 như thể nền màu đen trong ảnh ra . Nếu ta chỉ ra chuỗi ‘crop’ như một tham số , hàm imrotate sẽ xén ảnh ra tới kích thước như ảnh vào .

Hàm imrotate

Cú pháp của nó như sau :

```
B = imrotate(A,angle)
B = imrotate(A,angle,method)
B = imrotate(A,angle,method,bbox)
```

Diễn giải

+B=imrotate(A,angle) : Quay ảnh A một góc angle độ theo chiều ngược chiều kim đồng hồ , sử dụng phương pháp nội suy các pixel gần nhất . Để quay theo chiều kim đồng hồ hãy truyền giá trị âm cho tham số angle

+ B=imrotate(A,angle,method) : Quay ảnh A một góc angle độ theo chiều kim đồng hồ sử dụng phương pháp nội suy được chỉ ra trong method .

+ B=imrotate(A,angle,method,bbox) : Quay ảnh A một góc angle độ . Tham số bbox chỉ ra hộp biên của ảnh trả về . bbox là một chuỗi có thể nhận các giá trị sau :

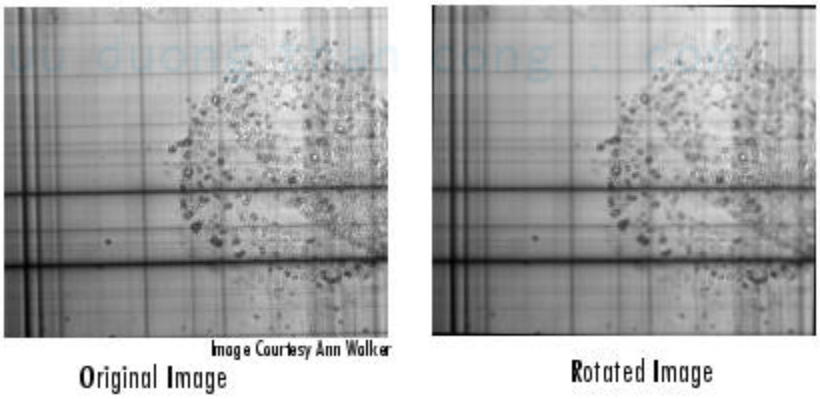
'crop' : Ảnh ra B chỉ bao gồm phần trung tâm của ảnh được quay và có cùng kích thước với ảnh A

'loose' : (Mặc định) : Ảnh ra B bao gồm toàn bộ ảnh được quay và lớn hơn ảnh A . imrotate thiết lập giá trị 0 cho các pixel ngoài biên của ảnh gốc .

Ví dụ

- Ví dụ này đọc một ảnh quang phổ ánh sáng mặt trời được lưu trong định dạng FITS và quay nó và căn nó theo chiều ngang .

```
I = fitsread('solarspectra.fts');
I = mat2gray(I);
J = imrotate(I,-1,'bilinear','crop');
imshow(I)
figure, imshow(J)
```



5. Xén ảnh (image cropping)

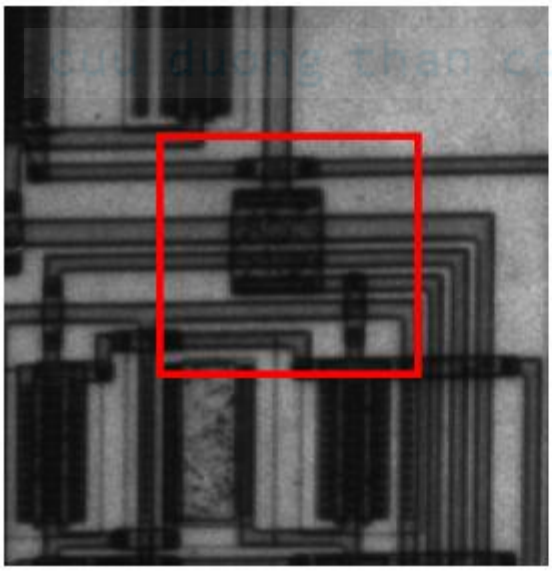
- Để trích một vùng chữ nhật của một ảnh , sử dụng hàm imcrop . Hàm imcrop chấp nhận hai tham số chính :

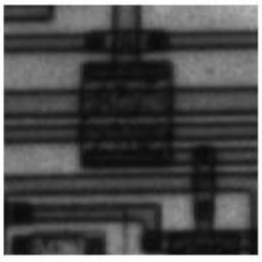
+ Ảnh cần xén

+ Các góc của hình chữ nhật xác định vùng xén

- Nếu ta gọi hàm imcrop mà không chỉ ra hình chữ nhật , ta có thể xén ảnh theo các tương tác . Trong trường hợp này ,ta sử dụng trỏ chuột để chọn vùng chữ nhật cần xén bằng cách nhấn và giữ phím chuột trái và di chuyển để chọn vùng xén . Khi chọn xong thì nhả chuột . Trong ví dụ sau , ta hiển thị một ảnh và gọi hàm imcrop . Hàm imcrop hiển thị ảnh trong một hình và đợi ta vẽ vùng chữ nhật cần xén trên ảnh .

```
imshow circuit.tif
I=imcrop;
Imshow(I);
```



Hàm imcrop

- Cú pháp của nó như sau :

I2 = imcrop(I)

X2 = imcrop(X,map)

RGB2 = imcrop(RGB)

I2 = imcrop(I,rect)

X2 = imcrop(X,map,rect)

RGB2 = imcrop(RGB,rect)

[...] = imcrop(x,y,...)

[A,rect] = imcrop(...)

[x,y,A,rect] = imcrop(...)

Diễn giải

- Hàm imcrop xén một ảnh theo một hình chữ nhật được chỉ định .

I2=imcrop(I) ;

X2=imcrop(X,map);

RGB2=imcrop(RGB);

Hàm imcrop sẽ hiển thị ảnh I và đợi ta chỉ ra hình chữ nhật cần xén bằng chuột

- Nếu ta bỏ qua các tham số , hàm imcrop thao tác trên ảnh của trục hiện tại .

- Để chỉ định một hình chữ nhật ta dùng trỏ chuột như đã nói ở trên

- Ta cũng có thể chỉ ra kích thước của hình chữ nhật mà không thao tác trực tiếp như các cú pháp sau :

I2 = imcrop(I,rect)

X2 = imcrop(X,map,rect)

RGB2 = imcrop(RGB,rect)

Trong đó : rect là một véctơ bốn phần tử dạng [xmin ymin width height], những giá trị này được chỉ ra trong toạ độ không gian . Để chỉ định các toạ độ không theo toạ độ không

gian cho ảnh vào , đặt trước các tham số khác với 2 véc tơ hai phần tử chỉ ra Xdata và Ydata . Chẳng hạn :

[…]=imcrop(x,y,…)

- Nếu ta cung cấp các tham số ra phụ , hàm imcrop sẽ trả lại thông tin về vùng chữ nhật được chọn và hệ toạ độ của ảnh vào . Chẳng hạn :

[A,rect] = imcrop(...)

[x,y,A,rect] = imcrop(...)

A là ảnh ra , x và y là Xdata và Ydata của ảnh vào

Chú ý :

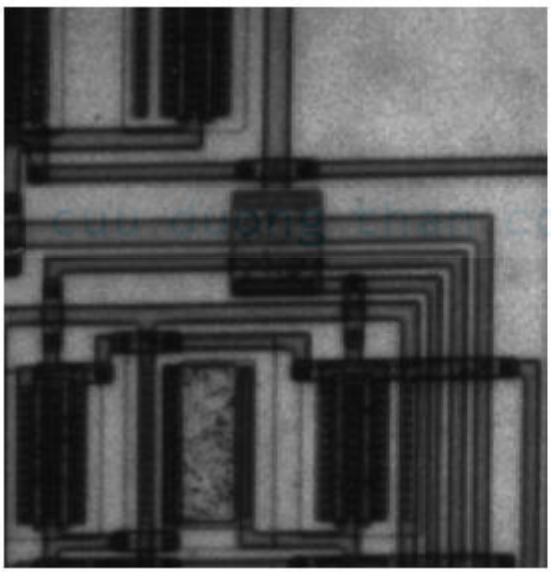
- Do rect là một tập hợp các toạ độ không gian , các phần tử width và height trong rect không luôn luôn tương ứng chính xác với kích thước của ảnh ra . Chẳng hạn , giả sử rect là [20 20 40 30] , sử dụng hệ toạ độ không gian theo mặc định . Góc trên trái của vùng chữ nhật được chọn là tâm của pixel (20,20) và góc dưới phải của vùng chữ nhật là tâm của pixel (50,60) . Ảnh ra là một ảnh có kích thước 31x41 chứ không phải 30x40 . Điều này là do ảnh ra bao gồm tất cả các pixel trong ảnh vào hoàn toàn hoặc một phần được bao bọc bởi vùng chữ nhật trên .

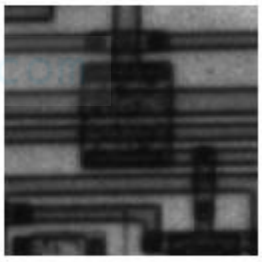
Ví dụ

I = imread('circuit.tif');

I2 = imcrop(I,[75 68 130 112]);

imview(I), imview(I2)





6. Các biến đổi ảnh thông dụng

- Để thực hiện các biến đổi không gian ảnh 2 chiều , sử dụng hàm imtransform . Hàm này chấp nhận hai tham số chính :

+ Ảnh cần biến đổi

+ Một cấu trúc biến đổi được gọi là TFORM chỉ ra kiểu biến đổi ta muốn thực hiện

Chỉ ra kiểu biến đổi

- Ta chỉ ra kiểu biến đổi trong cấu trúc TFORM . Có hai cách để tạo một cấu trúc TFORM :

+ Sử dụng hàm **maketform**

+ Sử dụng hàm **cp2tform**

Sử dụng hàm maketform

- Khi sử dụng hàm này , ta chỉ ra kiểu biến đổi ta muốn thực hiện . Các kiểu biến đổi mà maketform trợ giúp bao gồm :

+’ affine’ : Biến đổi có thể bao gồm : translation (dịch) , rotation (quay) , scaling , stretching và shearing . Các đường thẳng vẫn là đường thẳng , đường song song vẫn song song nhưng hình chữ nhật có thể bị biến đổi

+’box’ : Một trường hợp đặc biệt của affine khi mỗi chiều được dời và định tỉ lệ độc lập

+ ‘composite ‘ : Bao gồm tổ hợp của hai hay nhiều phép biến đổi

+ ‘custom ‘ : Biến đổi do người dùng tự định nghĩa , nó cung cấp các hàm thuận hoặc nghịch được gọi bởi hàm imtransform

+ ‘projective ‘ : Biến đổi trong đó các đường thẳng vẫn giữ nguyên nhưng các đường song song đồng quy lại thành một điểm .

Sử dụng cp2tform

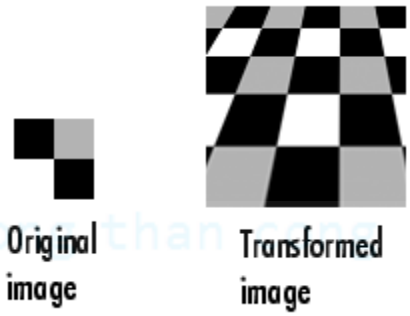
- Ta sử dụng hàm này để tạo ra cấu trúc TFORM khi ta muốn thi hành một biến đổi cần khít với các điểm dữ liệu như một biến đổi đa thức .

Chú ý : Khi sử dụng với hàm imtransform , cấu trúc TFORM phải định nghĩa một biến đổi 2 chiều . Nếu một ảnh chứa nhiều hơn một chiều chẳng hạn như ảnh RGB , cùng một biến đổi 2 chiều sẽ được áp đặt tới tất cả các mặt phẳng 2 chiều theo chiều cao hơn . Để định nghĩa một biến đổi n chiều sử dụng hàm imformarrray

Thực hiện biến đổi

- Khi ta đã định nghĩa một cấu trúc TFORM , ta có thể thi hành một sự biến đổi bằng cách gọi hàm imtransform . Chẳng hạn , đoạn mã sau sử dụng hàm này để thi hành một biến đổi projective cho một ảnh bàn cờ :

```
I = checkerboard(20,1,1);
figure; imshow(I)
T = maketform('projective',[1 1; 41 1; 41 41;   1 41],...
            [5 5; 40 5; 35 30; -10 30]);
R = makeresampler('cubic','circular');
K = imtransform(I,T,R,'Size',[100 100],'XYScale',1);
figure, imshow(K)
```



- Các tuỳ chọn của hàm imtransform cho phép ta điều khiển nhiều khía cạnh của việc biến đổi . Chẳng hạn , chú ý rằng ảnh bị biến đổi xuất hiện nhiều bản copy của ảnh gốc . Điều này nhận được bởi tuỳ chon ‘size’ .Xem thêm Help Online

Hàm imtransform

- Áp đặt một biến đổi không gian 2 chiều lên một ảnh

Cú pháp

```
B = imtransform(A,TFORM)
B = imtransform(A,TFORM,INTERP)
[B,XDATA,YDATA] = imtransform(...)
[B,XDATA,YDATA] = imtransform(...,param1,val1,param2,val2,...)
```

Diễn giải

+ B=imtransform(A,TFORM) : biến đổi ảnh A theo cấu trúc được định nghĩa trong TFORM . Cấu trúc này được trả về từ hàm maketform hoặc cp2tform . Nếu ndims(A)>2

như các ảnh RGB thì cùng một biến đổi không gian 2 chiều được áp đặt tới tất cả các mặt phẳng theo chiều cao hơn .

Khi sử dụng cú pháp này , hàm imtransform tự động dịch gốc của ảnh ra để ảnh ra có thể được hiển thị nhiều nhất có thể .

+ B=imtransform(A,TFORM , INTERP) : chỉ ra dạng của phép nội suy được sử dụng . INTERP có thể là một trong các giá trị 'nearest', 'bicubic' hoặc 'bilinear' .

Tương tự , INTERP có thể là một cấu trúc được trả về từ hàm makeresampler . Tuỳ chọn này cho phép điều khiển nhiều hơn lên việc lấy mẫu lại (resampling) .

+[B,XDATA,YDATA]= imtransform(…) : trả về vị trí của ảnh ra B trong không gian X-Y . XDATA và YDATA các véctơ hai thành phần . Những thành phần của XDATA chỉ ra toạ độ x của cột đầu và cuối của B . Những thành phần của YDATA chỉ ra toạ độ y của cột đầu và cuối của B . Bình thường , hàm imtransform tính toán XDATA và YDATA tự động vì vậy B chứa toàn bộ ảnh đã biến đổi A . Tuy nhiên , ta có thể đè chồng tính toán tự động này xem dưới đây :

[B,XDATA,YDATA] = imtransform(...,param1,val1,param2,val2,...) : Chỉ ra các tham số điều khiển nhiều khía cạnh khác nhau của biến đổi không gian . Bảng sau liệt kê các tham số mà ta có thể chỉ ra .

Tham số	Diễn giải
'UData' 'VData'	Cả hai tham số này là các véctơ hai phần tử thực . 'Udata' và 'Vdata' chỉ ra vị trí không gian của ảnh A trong không gian vào 2 chiều U-V . Hai phần tử của 'Udata' cho toạ độ u (hoành độ) của cột đầu tiên và cuối cùng của A . Hai phần tử của 'Vdata' cho toạ độ v (tung độ) của hàng đầu tiên và cuối cùng của A . Giá trị mặc định cho 'Udata' và 'Vdata' tương ứng là [1 size(A,2)] và [1 size(A,1)]
'Xdata' 'Ydata'	Cả hai tham số này là các véctơ hai phần tử thực chỉ ra vị trí không gian của ảnh ra B trong không gian ra 2 chiều X-Y . Hai phần tử của 'Xdata' chỉ ra hoành độ x của cột

	đầu tiên và cuối cùng của B . Hai phần tử của 'Ydata' chỉ ra tung độ của hàng đầu tiên và cuối cùng của B . Nếu 'Xdata' và 'Ydata' không được chỉ ra , hàm imtransform ước lượng giá trị cho chúng để có thể chứa toàn bộ ảnh ra đã bị biến đổi
'XYScale'	Là véctơ với một hoặc hai phần tử thực . Phần tử đầu tiên của 'XYScale' chỉ ra chiều rộng của mỗi pixel vào trong không gian X-Y . Phần tử thứ hai (nếu tồn tại) chỉ ra chiều cao của mỗi pixel ra . Nếu 'XYScale' chỉ có một phần tử , giá trị này sẽ được dùng cho cả chiều rộng và chiều cao . Nếu 'XYScale' không được chỉ định nhưng Size được chỉ ra thì 'XYScale' được tính toán từ 'Size' ,'Xdata' và 'Ydata' .
'Size'	Một véctơ hai phần tử nguyên không âm . 'Size' chỉ ra số hàng và cột trong ảnh ra B . Với chiều cao hơn , kích cỡ của B được lấy trực tiếp từ A . Nói cách khác , size(B,k) tương đương với size(A,k) với k>2. Nếu 'Size' không được chỉ định , nó sẽ được tính từ 'Xdata' ,'Ydata' và 'XYScale'
'FillValues'	Một mảng chứa một hoặc nhiều giá trị tô (fill values) . Fill values được sử dụng cho các pixel trên ảnh ra khi vị trí được biến đổi tương ứng trên ảnh vào hoàn toàn là viền ngoài của ảnh ra . nếu A là 2 chiều , 'Fillvalues' phải vô hướng . Tuy nhiên , nếu chiều của A lớn hơn 2 , 'FillValues' có

	thể là một mảng mà kích thước của nó thoả mãn ràng buộc sau : size(fill_values,k) phải bằng size(A,k+2) hoặc 1 . Chẳng hạn , nếu A là một ảnh RGB unit8 có kích thước 200x200x3 thì các khả năng của 'FillValues' bao gồm : + 0 : Tô với màu đen +[0;0;0] : Tô với màu đen +255 : Tô với màu trắng + [255;255;255] : Tô với màu trắng +[0;0;255] : Tô với màu xanh +[255;255;0] : Tô với màu vàng Nếu A là 4 chiều 200x200x3x10 thì 'FillValues' có thể là 1 vô hướng 1x10,3x1,3x10

Ví dụ

+ Ví dụ 1

Áp một phép dịch chuyển nganh tới một ảnh cường độ ;

```
I = imread('cameraman.tif');
tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
J = imtransform(I,tform);
imshow(I), figure, imshow(J)
```

+ Ví dụ 2

Một phép biến đổi chiếu có thể ánh xạ một hình vuông thành một hình bốn cạnh . Trong ví dụ này , thiết lập một toạ độ vào để cho ảnh vào tô đầy hình vuông sau đó biến đổi ảnh sang một hình 4 cạnh với các đỉnh (0,0) ,(1,0) ,(1,1) và (0,1) thành một hình 4 cạnh với các đỉnh là (-4,2) ,(-8,3) ,(-3,-5) và (6,3) . Tô với mày xám và sử dụng nội suy song khối . Tạo ảnh ra với kích thước bằng với ảnh vào .

```
I = imread('cameraman.tif');
udata = [0 1];  vdata = [0 1];  % input coordinate system
```

```
tform = maketform('projective',[ 0 0;  1  0;  1  1; 0 1],...
                        [-4 2; -8 -3; -3 -5; 6 3]);
[B,xdata,ydata] = imtransform(I, tform, 'bicubic', ...
                        'udata', udata,...
                        'vdata', vdata,...
                        'size', size(I),...
                        'fill', 128);
subplot(1,2,1), imshow(udata,vdata,I), axis on
subplot(1,2,2), imshow(xdata,ydata,B), axis on
```

VI – Phân tích và làm giàu ảnh (nâng cao chất lượng ảnh)

1. Bảng thuật ngữ

Tên thuật ngữ	Diễn giải
Adaptive filter	Bộ lọc mà tính chất của nó biến đổi qua một ảnh phụ thuộc vào đặc tính địa phương của các pixel trong ảnh
Contour	Đường trong một ảnh dọc theo đó giá trị cường độ của các pixel ảnh bằng hằng số
Edge	(Cạnh) Đường cong mà theo đó độ biến đổi cường độ các pixel ảnh rất nhanh . Edge thường được kết hợp với vùng biên của một đối tượng trong một cảnh . Sự phát hiện các edge được sử dụng để phân biệt các edge trong một ảnh
Property	Sự đo đạc định lượng của một ảnh hoặc một vùng ảnh .
Histogram	Đồ thị được sử dụng trong phân tích ảnh chỉ ra sự phân bố của cường độ của một ảnh . Ta có thể sử dụng thông tin này trong một histogram để lựa chọn một thao tác

	làm giàu ảnh thích hợp . Chẳng hạn , nếu đồ thị histogram của một ảnh chỉ ra rằng vùng các giá trị cường độ nhỏ , ta có thể sử dụng một hàm điều chỉnh cường độ để mở rộng các giá trị qua một vùng rộng .
Noise	Các sai số trong quá trình thu thập ảnh dẫn đến việc các giá trị pixel không phản ảnh đúng cường độ của cảnh thực
Profile	Một tập hợp các giá trị cường độ lấy từ các điểm cách đều nhau dọc theo một đoạn đường thẳng hoặc đường gấp khúc trong một ảnh . Với các điểm không nằm ở tâm của pixel , giá trị này được nội suy .
Quadtree decomposition	Kĩ thuật phân tích ảnh bằng cách chia ảnh ra nhiều khối giống nhau

2. Các giá trị pixel và thống kê

- Toolbox xử lý ảnh cung cấp một vài hàm trả lại thông tin về dữ liệu tạo nên một ảnh . Những hàm này trả lại thông tin về dữ liệu ảnh theo nhiều dạng khác nhau bao gồm :

+ Dữ liệu của những pixel được lựa chọn (hàm pixval , impixel)

+ Dữ liệu dọc theo một đường trong một ảnh (hàm improfile)

+ Đồ thị đường của dữ liệu ảnh (hàm imcontour)

+ Biểu đồ (histogram) của dữ liệu ảnh (hàm imhist)

+ Thống kê vắn tắt dữ liệu ảnh (các hàm mean2,std2,corr2)

+ Đo đạc tính chất của một vùng ảnh (hàm regionprops)

a – Lựa chọn các pixel

- Ta có hai hàm có thể cung cấp thông tin về dữ liệu màu của các pixel ảnh được chọn lựa :

+ Hàm pixval : Hiển thị tương tác giá trị dữ liệu của các pixel khi ta di chuyển con trỏ chuột trên ảnh . Hàm này có thể hiển thị khoảng cách Ơ-clít giữa hai pixel (Euclidean Distance)

+ Hàm impixel trả lại giá trị dữ liệu của một hoặc một tập các pixel được lựa chọn . Ta có thể cung cấp toạ độ của pixel như là tham số vào hoặc có thể lựa chọn pixel sử dụng chuột .

Chú ý : Với ảnh chỉ số , hàm pixval và impixel chỉ ra giá trị RGB được lưu trong bản đồ màu , không phải giá trị chỉ số :

- Để sử dụng hàm pixval , đầu tiên ta hiển thị một ảnh và sau đó nhập lệnh pixval. Pixval sẽ đặt một thanh màu đen ở cuối của khung hình , thanh này sẽ hiển thị toạ độ (x,y) của pixel khi con trỏ chuột đi qua nó và dữ liệu màu cho pixel đó .
- Nếu ta click chuột vào ảnh và dữ chuột trong khi ta di chuyển chuột , hàm pixal cũng hiển thị khoảng cách giữa điểm ta click và vị trí hiện tại của con trỏ chuột . Hàm pixval vẽ một đường thẳng giữa các điểm để chỉ ra khoảng cách đã được đo . Khi ta nhả chuột , đường thẳng và khoảng cách biến mất .
- Hàm pixval mang lại cho ta thông tin nhiều hơn hàm impixel tuy nhiên , hàm impixel có ưu điểm là trả lại thông tin trong một biến và nó có thể được gọi theo cách tương tác hoặc không tương tác . Nếu ta gọi hàm impixel mà không có tham số , con trỏ thay đổi hình dạng khi nó di chuyển trên ảnh . Ta có thể click các pixel quan tâm , hàm impixel sẽ hiển thị một ngôi sao nhỏ qua mỗi pixel ta đã chọn . Khi ta đã chọn xong pixel , nhấn nút Enter , hàm impixel trả lại giá trị màu của pixel đã chọn và ngôi sao biến mất .
- Ví dụ sau minh hoạ cách sử dụng hàm impixel :

1. Hiển thị một ảnh :

```
imshow canoe.tif
```

2. Gọi hàm impixel để lựa chọn điểm

```
vals=impixel
```

Sau đó click các điểm của ảnh để lựa chọn các pixel . Khi ta kết thúc việc lựa chọn , nhấn phím Enter

Giá trị trả lại là các giá trị của các pixel đã chọn trong biến vals

vals =

0.1294 0.1294 0.1294

0.5176 0 0

0.7765 0.6118 0.4196

b- Intensity Profile

- Hàm improfile tính toán và vẽ các giá trị cường độ dọc theo một đường thẳng hoặc một hình gấp khúc trong một ảnh . Ta có thể cung cấp toạ độ của đường thẳng như một tham số vào của hàm hoặc định nghĩa đường thẳng gấp khúc sử dụng chuột . Trong những trường hợp này , hàm improfile sử dụng nội suy để tính toán giá trị của các pixel cách đều nhau dọc theo một đường (mặc định , hàm này sử dụng phương pháp nội suy các pixel gần nhất tuy nhiên ta có thể thay đổi phương pháp khác – Xem cú pháp cụ thể của hàm) . Hàm improfile làm việc tốt nhất trên các ảnh RGB và ảnh cường độ .

- Với một đường thẳng đơn , hàm improfile vẽ đồ thị các giá trị cường độ trong một khung nhìn 2 chiều . Với một đường gấp khúc , hàm sẽ vẽ một đồ thị các giá trị cường độ trong một khung nhìn 3 chiều

- Nếu ta gọi hàm improfile không có tham số , con trỏ chuột sẽ thay đổi hình dạng khi nó di chuyển trên ảnh . Ta có thể chỉ ra đường thẳng bằng cách click chuột . Hàm improfile sẽ vẽ một đường thẳng giữa hai pixel gần nhau . Khi ta kết thúc lựa chọn , nhấn phím Enter . Hàm improfile hiển thị đồ thị trong một khung hình mới
- Trong ví dụ này , ta gọi hàm improfile và chỉ ra một đường thẳng đơn với chuột . Trong hình này , đường thẳng hiển thị màu đỏ và được vẽ từ trên xuống dưới :

```
I = fitsread('solarspectra.fts');
imshow(I,[]);
improfile
```

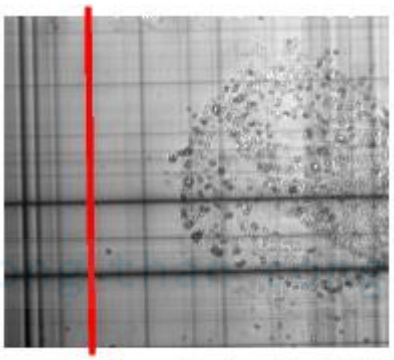
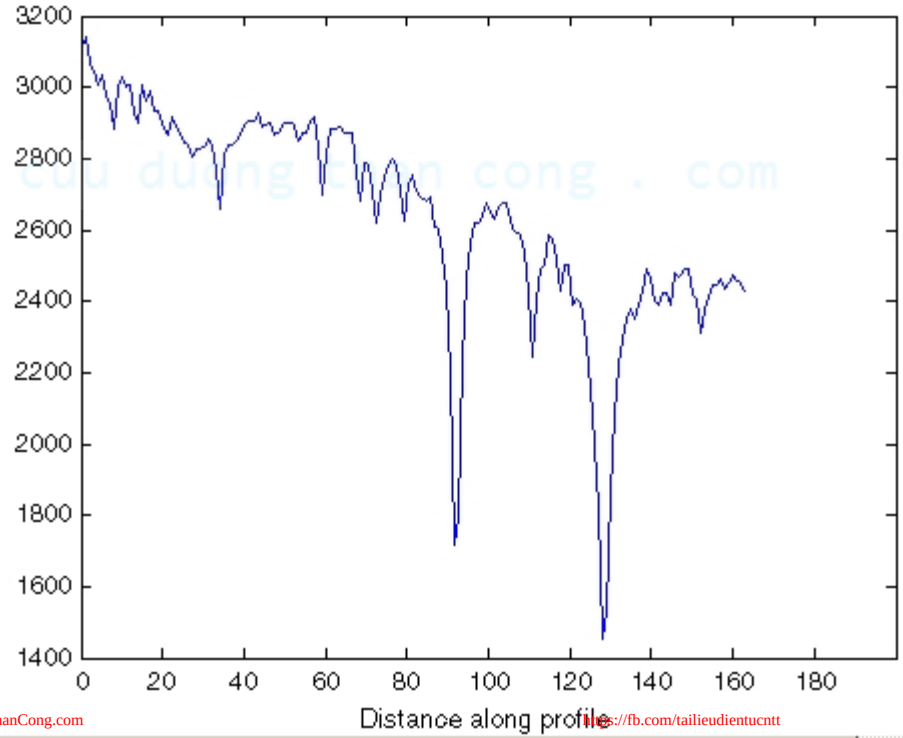


Image Courtesy of Ann Walker

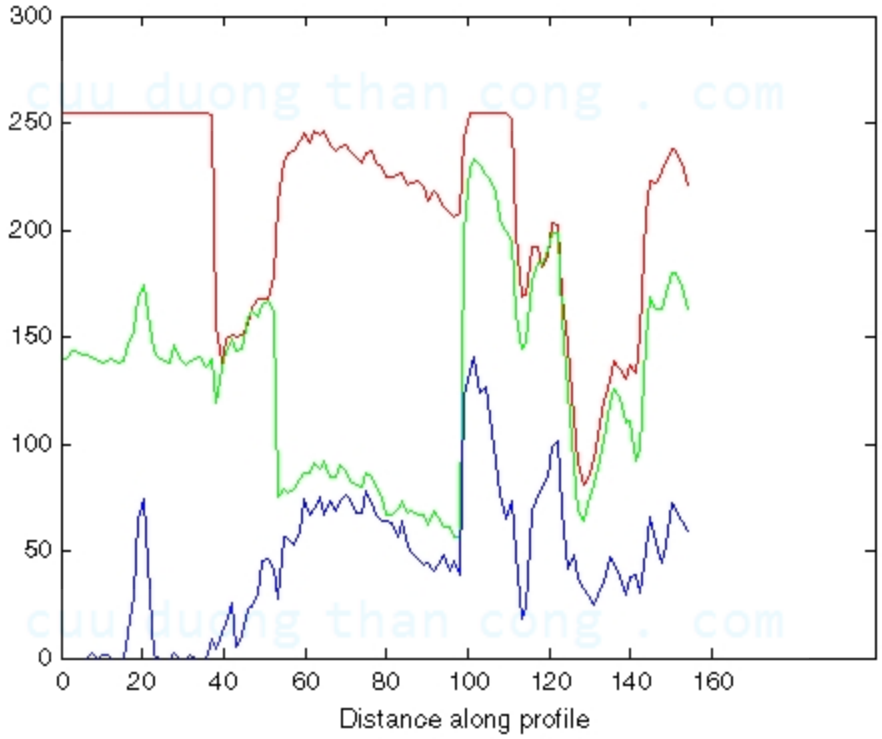
Hàm improfile hiển thị một đồ thị của các dữ liệu ảnh dọc theo đường thẳng . Chú ý rằng đỉnh , đáy cách chúng tương ứng với vùng sáng và tối trong ảnh



- Ví dụ dưới đây chỉ ra hàm improfile hàm việc ra sao với một ảnh RGB . Sử dụng hàm imshow để hiển thị ảnh trong một cửa số khung nhìn . Gọi hàm improfile không có tham số và vết của một đường thẳng theo cách tương tác . Trong hình , đường màu đen chỉ ra đoạn đường thẳng được vẽ từ trên xuống dưới .

imshow peppers.png
improfile

- Hàm improfile hiển thị một đồ thị của các giá trị cường độ dọc theo đoạn đường thẳng . Đồ thị bao gồm các đường thẳng riêng biệt cho các giá trị cường độ R,G,B .



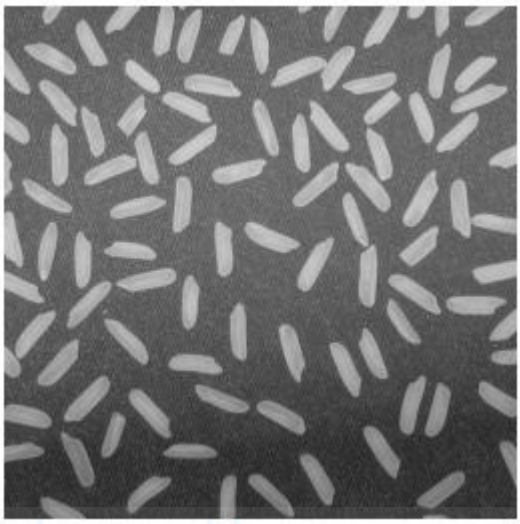
c- Đường viền ảnh (Image Contour)

- Ta có thể sử dụng hàm imcontour để hiển thị một đồ thị đường viền của các dữ liệu trong một ảnh cường độ . Hàm này tương tự như hàm contour trong Matlab nhưng nó tự động thiết lập trục vì vậy , chiều và tỉ lệ khớp với ảnh .

- Ví dụ này hiển thị một ảnh cường độ (ảnh đen trắng) của các hạt gạo và một đồ thị đường viền của dữ liệu ảnh :

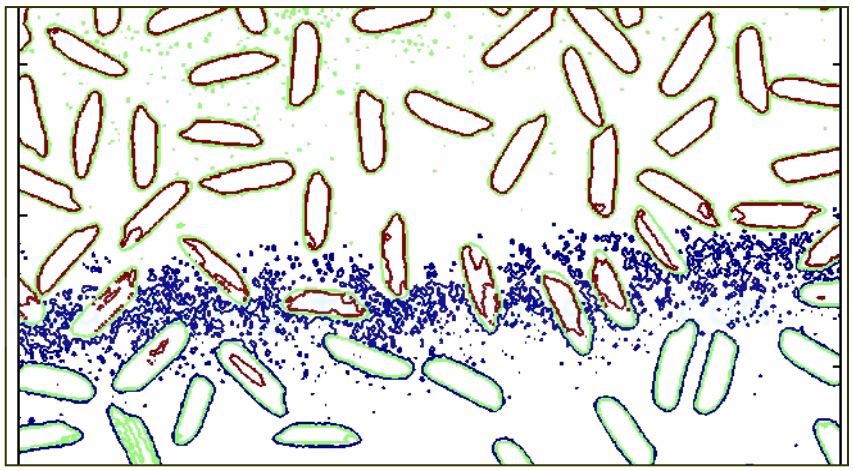
1. Đọc một ảnh cường độ và hiển thị nó :

```
I = imread('rice.png');
imshow(I)
```



2 . Hiển thị một đồ thị đường viền của ảnh cường độ :

```
figure , imcontour (I,3)
```



d- Biểu đồ ảnh (Image Histogram)

- Một biểu ảnh là một biểu đồ chỉ ra sự phân bố của cường độ của một ảnh chỉ số hoặc ảnh cường độ . Hàm biểu đồ ảnh imhist tạo ra biểu đồ này bằng cách tạo ra n thùng (bins

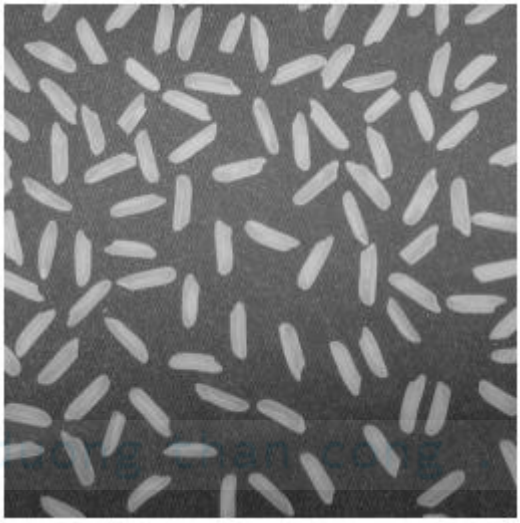
) cách đều nhau , mỗi cái đại diện cho một vùng các giá trị dữ liệu . Sau đó nó tính toán số lượng các pixel cho mỗi vùng .

- Ví dụ sau đây hiển thị một ảnh của các hạt gạo và biểu đồ với 64 thùng . Biểu đồ hiển thị một đỉnh xung quanh 100 tương ứng với nền xám tối trong ảnh .

1. Đọc vào một ảnh và hiển thị nó :

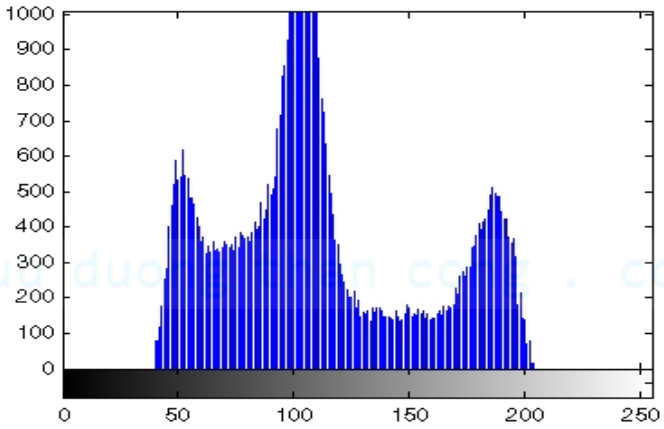
I = imread('rice.png');

imshow(I)



2. Hiển thị biểu đồ của ảnh

figure , imhist (I)



e- Thống kê vắn tắt

- Ta có thể tính thống kê chuẩn của một ảnh sử dụng các hàm mean2,std2 và corr2 . mean2 và std2 tính độ lệch trung bình và độ lệch tiêu chuẩn của các phần tử của một ma trận . Hàm corr2 tính hệ số tương quan giữa hai ma trận có cùng kích thước .
- Để biết chi tiết các hàm này , xem cú pháp của chúng trong Online Help

f- Đo đạc tính chất của một vùng ảnh

- Ta có thể sử dụng hàm regionprops để tính toán các tính chất của một vùng ảnh . Chẳng hạn , hàm regionprops có thể đo đạc những tính chất như : vùng , trọng tâm và khung bao viền của vùng mà ta chỉ ra .

3. Phân tích ảnh

- Kĩ thuật phân tích ảnh trả lại thông tin về cấu trúc của một ảnh . Các kĩ thuật này bao gồm :

+ Phát hiện cạnh (Edge Detection)

+ Tìm vết của đường biên (Boundary Tracing)

+ Quadtree Decomposition

a – Phát hiện cạnh (Edge Detection)

- Ta sử dụng hàm edge để phát hiện các cạnh trong một ảnh mà tương ứng với vùng biên của các đối tượng . Để tìm cạnh , hàm này tìm các vị trí trong ảnh có giá trị cường độ thay đổi rất nhanh sử dụng một trong những tiêu chuẩn sau :

+ Các vị trí mà tích phân đầu của cường độ lớn hơn biên độ của một số ngưỡng nào đó Các vị trí mà tích phân thứ hai của cường độ giao với 0

Hàm edge cung cấp một số bộ ước lượng tích phân , mỗi cái ứng dụng một trong các định nghĩa trên . Với một số bộ ước lượng này , ta có thể chỉ ra các thao tác có nhạy với cạnh nằm nganh , nằm dọc hoặc cả hai hay không . Hàm trả về một ảnh nhị phân chứa giá trị 1 tại nơi cạnh được tìm thấy và 0 ở các chỗ khác .

- Phương pháp phát hiện cạnh mạnh nhất mà hàm edge cung cấp laf phương pháp Canny . Phương pháp này khác với các phương pháp khắc ở chỗ nó sử dụng hai giá trị ngưỡng khác nhau (để phát hiện cạnh mạnh hay yếu) và bao gồm các cạnh yếu ở ảnh ra chỉ nếu chúng được kết nối với các cạnh khoẻ (strong edges) .
- Ví dụ sau minh hoạ sức mạnh của phương pháp Canny bằng cách chỉ ra kết quả của việc áp vào phép phát hiện cạnh Sobel và Canny lên cùng một ảnh :

1. Đọc ảnh và hiển thị nó

```
I = imread('coins.png');
```

imshow(I)

2. Áp phép phát hiện cạnh Sobel và Canny lên ảnh và hiển thị chúng :

BW1 = edge(I,'sobel');

BW2 = edge(I,'canny');

imshow(BW1)

figure, imshow(BW2)

Sobel Filter

Canny Filter

b- Tìm vết của biên (Boundary Tracing)

- Toolbox cung cấp hai hàm ta có thể sử dụng để tìm biên của các đối tượng trong một ảnh :

+ Hàm bwtraceboundary

+ Hàm bwboundaries

- Hàm bwtraceboundary trả lại toạ độ hàng và cột của tất cả các pixel trên biên của một đối tượng trong ảnh . Ta phải chỉ ra vị trí của một pixel biên trên đối tượng như là điểm bắt đầu của việc tìm vết .

- Hàm bwboundaries trả lại toạ độ hàng và cột của các pixel biên của tất cả các đối tượng trong một ảnh

- Với cả hai hàm , các pixel khác 0 thuộc về một đối tượng và pixel với giá trị 0 (zero) tạo thành nền (background)

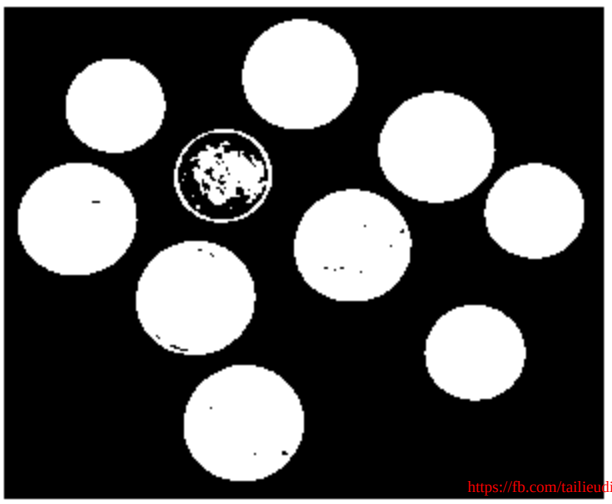
- Ví dụ sau sử dụng hàm bwtraceboundary để tìm vết của đường bao của một đối tượng trong một ảnh nhị phân sau đó sử dụng hàm bwboundaries để tìm vết của đường bao của tất cả các đối tượng trong ảnh :

1. Đọc vào một ảnh và hiển thị nó

```
I = imread('coins.png');
imshow(I)
```

2. Chuyển ảnh thành một ảnh nhị phân . Hàm bwtraceboundary và hàm bwboundaries chỉ làm việc trên các ảnh nhị phân

```
BW = im2bw(I);
imshow(BW)
```



3. Tính các toạ độ hàng và cột của một pixel trên biên của đối tượng ta muốn tìm vết . bwboundary sử dụng điểm này như là điểm bắt đầu của việc tìm vết .

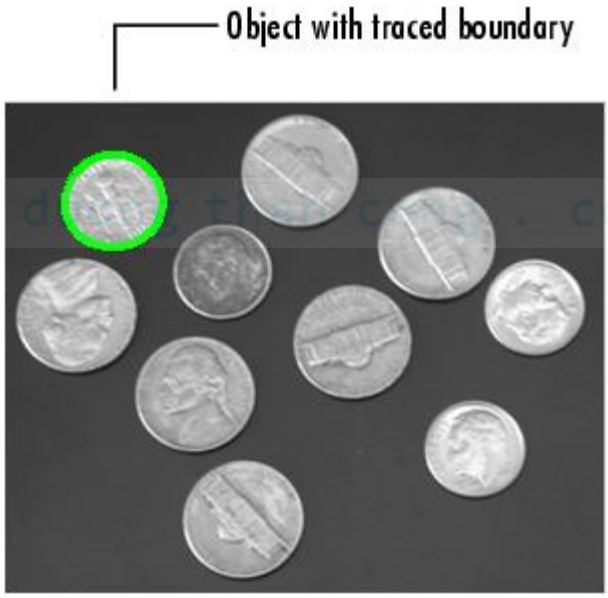
```
dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)))
```

4. Gọi hàm bwtraceboundary để tìm vết của biên từ điểm đã chọn . Như một tham số cần thiết , ta phải chỉ ra một ảnh nhị phân , các toạ độ hàng và cột của điểm bắt đầu và hướng của bước đầu tiên . Ví dụ chỉ ra hướng bắc (North)

```
boundary=bwtraceboundary (BW, [row ,col],'N');
```

5. Hiển thị ảnh gốc và sử dụng các toạ độ trả về để vẽ đồ thị trên ảnh

```
imshow(I)
hold on;
plot(boundary(:,2),boundary(:,1),'g','LineWidth',3);
```

6. Đề tìm vết của tất cả các đường biên của tất cả các đối tượng trong ảnh , sử dụng hàm bwboundaries . Theo mặc định , hàm này tìm biên của tất cả các đối tượng trong một ảnh bao gồm các đối tượng trong các vật khác . Trong ảnh nhị phân được sử dụng trong ví dụ này , một số đồng tiền chứa vùng màu đen mà hàm bwboundaries hiểu như các đối tượng riêng biệt . Để chắc chắn rằng hàm bwboundaries chỉ tìm vết với những đồng tiền , sử dụng hàm imfill để tô vùng trong mỗi đồng tiền

```
BW_filled = imfill(BW,'holes');
boundaries = bwboundaries(BW_filled);
```

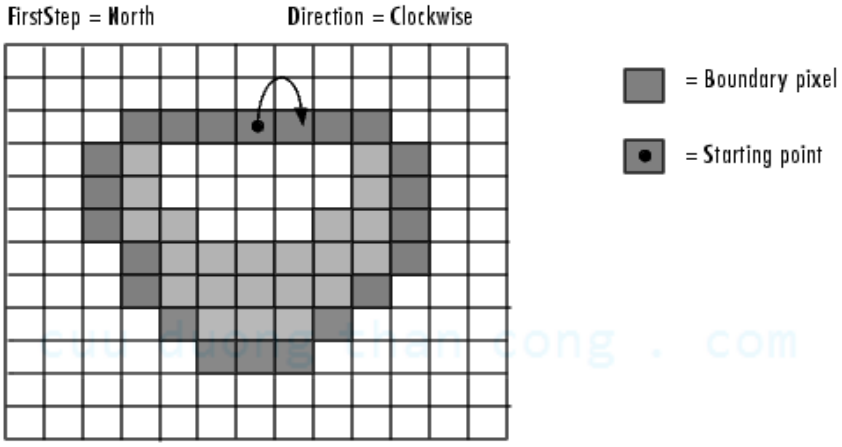
Hàm bwboundaries trả về một mảng mà mỗi phần tử chứa các toạ độ hàng , cột của mỗi đối tượng trong ảnh .

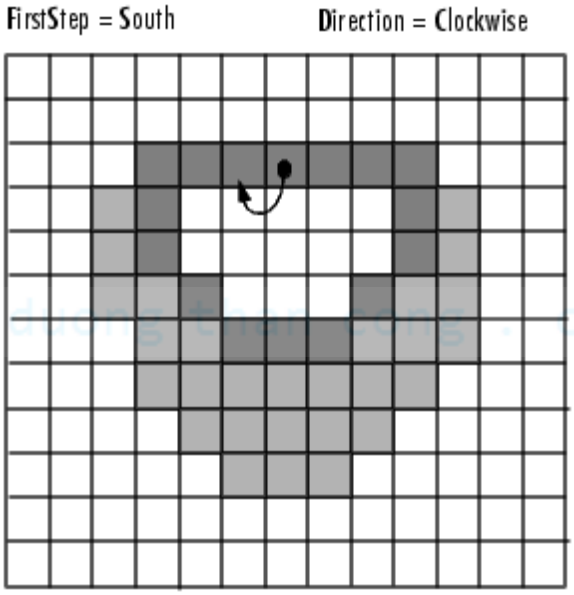
7. Vẽ các đường biên của tất cả các đồng tiền trên ảnh gốc sử dụng các toạ độ trả về từ hàm bwboundaries

```
for k=1:10
  b = boundaries{k};
  plot(b(:,2),b(:,1),'g','LineWidth',3);
end
```

Chọn bước đầu tiên và hướng cho việc tìm vết

- Với một số đối tượng , ta phải quan tâm khi lựa chọn pixel làm điểm xuất phát và hướng làm bước đầu tiên (North , South…)
- Chẳng hạn , nếu một đối tượng chứa một lỗ và ta chọn một pixel trên một phần mỏng của đối tượng làm điểm xuất phát , ta có thể tìm vết biên ngoài của đối tượng hoặc biên trong của lỗ phụ thuộc vào hướng ta chọn cho bước đầu tiên . Với các vật được tô , hướng mà ta chọn cho bước đầu tiên không quan trọng
- Để minh hoạ , hình sau đây chỉ ra các pixel đã được tìm vết khi pixel xuất phát trên một phần mỏng của vật và bước đầu tiên (first step) được thiết lập là north hoặc south .





c- Kĩ thuật chia 4 (Quadtree Decomposition Technique)

- Đây là một kĩ thuật phân tích bao gồm việc chia nhỏ một ảnh ra thành các khối đồng đều hơn ảnh . Kĩ thuật này thể hiện những thông tin về cấu trúc của ảnh . Nó cũng hữu dụng như là bước đầu tiên trong giải thuật nén thích nghi (adaptive compression)
- Ta có thể thực hiện kĩ thuật này bằng cách sử dụng hàm qtdecomp . Hàm này làm việc bằng cách chia một ảnh vuông thành 4 khối vuông có cùng kích cỡ và sau đó kiểm tra mỗi khối để xem nếu nó hợp với một số tiêu chuẩn đồng đều (chẳng hạn nếu tất cả các pixel trong khối là ở trong moọt khoảng riêng biệt) . Nếu một khối hợp tiêu chuẩn , nó sẽ không được chia thêm nữa . Nếu không , nó sẽ được chia tiếp thành 4 khối con và quá trình kiểm tra tiêu chuẩn lại được áp dụng lên các khối này . Quá trình này được lặp lại cho đến khi mỗi khối hợp tiêu chuẩnn . Kết quả có thể chứa các khối với kích thước khác nhau

Ví dụ : Thực hiện kĩ thuật Quadtree Decomposition

- Để minh hoạ , ví dụ này thực hiện kĩ thuật chia 4 trên một ảnh cường độ có kích cỡ : 512x512 .

1. Đọc vào một ảnh cường độ

```
I = imread('liftingbody.png');
```

2.Chỉ ra tiêu chuẩn kiểm tra được sử dụng để quyết định tính đồng nhất của mỗi khối trong phép phân tích . Chẳng hạn , tiêu chuẩn có thể là sự tính toán ngưỡng sau đây :

```
max(block(: ))-min(block(:))<=2
```

Ta cũng có thể cung cấp cho hàm qtdecomp một hàm (hơn là sử dụng giá trị ngưỡng) để quyết định có chia nhỏ các khối hay không , chẳng hạn ta dựa trên quyết định về sự thay đổi của khối . Để biết thêm , xem cú pháp của hàm qtdecomp

3. Thực hiện kĩ thuật chia 4 bằng cách gọi hàm qtdecomp , chỉ ra ảnh và gía trị ngưỡng như những tham số

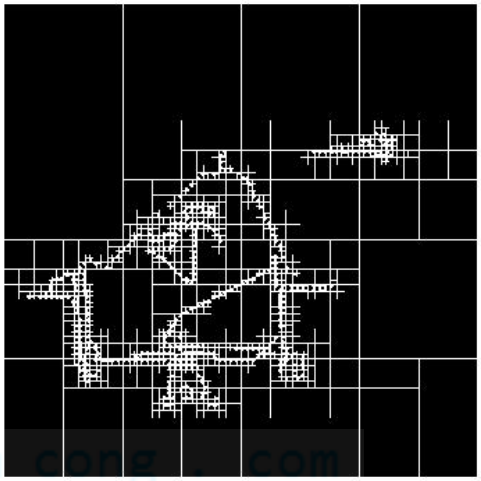
```
S = qtdecomp(I,0.27)
```

Ta chỉ ra ngưỡng là giá trị giữa 0 và 1 bất kể I thuộc lớp nào . Nếu I thuộc lớp uint8 , hàm qtdecomp nhân ngưỡng với giá trị 256 để tính giá trị ngưỡng được sử dụng . Nếu I thuộc lớp uint16 , hàm nhân ngưỡng với giá trị 65535

Hàm qtdecomp đầu tiên chia ảnh thành 4 khối 256x256 và áp đặt một sự kiểm tra tiêu chẩn lên mỗi khối . Nếu một khối không đủ tiêu chuẩn , hàm chia tiếp nó và áp đặt sự kiểm tra tiêu chuẩn lên các khối con này . qtdecomp tiếp tục thực hiện việc chia các khối cho đến khi tất cả các khối hội đủ tiêu chuẩn . Các khối có thể nhỏ 1x1 thậm chí nhỏ hơn

- Hàm qtdecomp trả về S như là một ma trận thưa có cùng kích thước với I . Các phần tử khác 0 của S đại diện cho góc trên trái của các khối , giá trị của mỗi phần tử khác không chỉ ra kích thước mỗi khối .

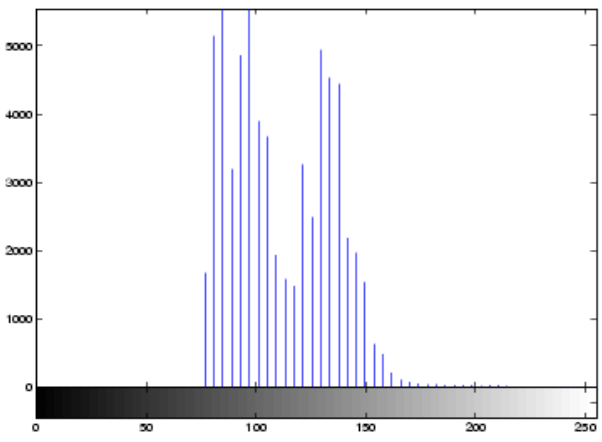
- Hình sau đây chỉ ra ảnh gốc và ảnh sau khi áp dụng kĩ thuật chia 4 . Mỗi hình vuông đen đại diện cho một khối đồng nhất và các đường trắng đại diện cho vùng bao giữa các khối .



4. Điều chỉnh cường độ ảnh (Intensity Adjustment)

- Kĩ thuật làm giàu ảnh được sử dụng để tăng chất lượng một ảnh .Sự điều chỉnh cường độ của một ảnh là một kĩ thuật làm giàu ảnh mà ánh xạ các giá trị cường độ của một ảnh tới một khoảng mới . Để minh hoạ , hình sau chỉ ra một ảnh có độ tương phản thấp với biểu đồ của nó . Chú ý trong biểu đồ của ảnh , tại sao tất cả các giá trị tụ tập tại tâm của vùng .

```
I = imread('pout.tif');
imshow(I)
figure, imhist(I,64)
```



- Nếu ta ánh xạ lại các giá trị dữ liệu để tô toàn bộ vùng cường độ [0,255] , ta có thể tăng độ tương phản của ảnh . Sau đây ta sẽ xem xét một số kĩ thuật điều chỉnh cường độ

a- Điều chỉnh các giá trị cường độ đến một khoảng xác định

- Ta có thể điều chỉnh các giá trị cường độ trong một ảnh bằng cách sử dụng hàm imadjust khi ta chỉ ra khoảng của các giá trị cường độ trong ảnh ra .
- Chẳng hạn , mã sau đây sẽ tăng độ tương phản trong một ảnh cường độ có độ tương phản thấp bằng cách ánh xạ lại các giá trị dữ liệu để điền đầy toàn bộ khoảng giá trị cường độ [0,255]

```
I = imread('pout.tif');
J = imadjust(I);
imshow(J)
figure, imhist(J,64)
```


- Hình sau hiển thị ảnh đã được điều chỉnh và biểu đồ của nó .

Chỉ định giới hạn điều chỉnh

- Ta có thể chỉ ra khoảng của các giá trị vào và giá trị ra sử dụng hàm imadjust . Ta chỉ ra những khoảng này trong hai véctơ và truyền đến hàm imadjust như là tham số . Véctơ đầu tiên chỉ ra các giá trị cường độ thấp và cao mà ta muốn ánh xạ . Véctơ thứ hai chỉ ra tỉ lệ qua đó ta muốn ánh xạ chúng

Chú ý : Ta phải chỉ ra các cường độ như là các giá trị giữa 0 và 1 bất kể ảnh vào I thuộc lớp nào .

- Chẳng hạn , ta có thể giảm độ tương phản của một ảnh bằng cách thu hẹp khoảng dữ liệu . Trong ví dụ dưới đây , áo choàng của người đàn ông có màu đen để lộ một số chi tiết nào . Hàm imadjust ánh xạ khoảng [0,51] trong ảnh vào thuộc lớp uint8 thành ảnh ra [128,255] . Điều này làm sáng đáng kể ảnh ra và cũng mở rộng khoảng động (dynamic range) của phần tối trong ảnh gốc , làm cho nó dễ dàng nhìn thấy các chi tiết trên chiếc áo choàng . Chú ý rằng , tuy nhiên bởi vì tất cả các giá trị lớn hơn 51 trong ảnh gốc được ánh xạ thành 255 (trắng) trong ảnh ra , ảnh ra xuất hiện như vừa bị rửa .

```
I = imread('cameraman.tif');
J = imadjust(I,[0 0.2],[0.5 1]);
imshow(I)
figure, imshow(J)
```

Thiết lập ngưỡng điều chỉnh tự động

- Để sử dụng hàm imadjust , ta phải thực hiện hai bước điển hình :

1. Quan sát biểu đồ của ảnh để quyết định giá trị cường độ giới hạn

2. Chỉ ra những ngưỡng này dưới dạng một phân số trong khoảng 0 đến 1 để ta có thể truyền chúng vào hàm imadjust trong véc tơ [low_in high_in]

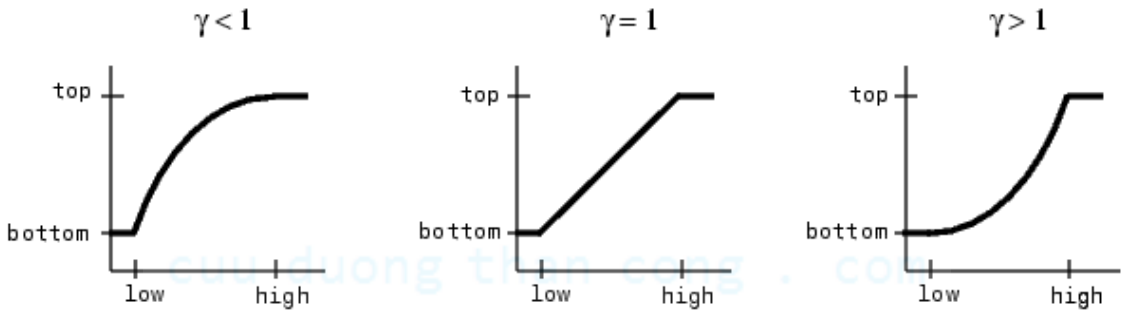
- Một cách thuận tiện hơn để chỉ ra các giá trị ngưỡng là sử dụng hàm stretchlim . Hàm này tính toán biểu đồ của ảnh và tính các giá trị ngưỡng điều chỉnh một cách tự động . Hàm trả về các giá trị dưới dạng phân số trong một véctơ mà ta có thể truyền theo dạng [low_in high_in] như là tham số tới hàm imadjust . Chẳng hạn :

```
I = imread('rice.png');
J = imadjust(I,stretchlim(I),[0 1]);
```

- Theo mặc định , hàm stretchlim sử dụng các giá trị cường độ đại diện cho đáy 1%(0.01) và đỉnh 1%(0.99) của khoảng như là các ngưỡng giới hạn điều chỉnh . Ta có thể chỉ ra các khoảng giới hạn khác như là tham số cho hàm stretchlim .

Tương quan Gamma

- Hàm imadjust ánh xạ low thành bottom , high thành top . Theo mặc định , những giá trị giữa low và high được ánh xạ một cách tuyến tính tới các giá trị giữa bottom và top . Chẳng hạn , giá trị giữa low và high tương ứng với giá trị giữa bottom và top
- Hàm imadjust có thể chấp nhận một tham số phụ chỉ ra tác nhân tương quan Gamma . Phụ thuộc vào giá trị của Gamma , sự ánh xạ giữa các giá trị trong ảnh vào và ra có thể là không tuyến tính . Chẳng hạn , giá trị giữa low và high có thể ánh xạ tới một giá trị lớn hơn hoặc nhỏ hơn giá trị giữa bottom và top . Hình sau minh hoạ mối quan hệ này . Ba đường cong chuyển đổi chỉ ra các giá trị được ánh xạ ra sao khi Gamma nhỏ hơn , bằng và lớn hơn 1 .



- Ví dụ dưới đây minh hoạ tương quan Gamma. Chú ý rằng trong lời gọi hàm imadjust , khoảng dữ liệu của ảnh vào và ra được chỉ định là các ma trận rỗng . Khi ta chỉ định một ma trận rỗng , hàm imadjust sử dụng khoảng giá trị mặc định [0 ,1] . Trong ví dụ này , cả hai khoảng đều rỗng về phía trái , điều này có nghĩa rằng tương quan Gamma được áp đặt mà không có bất kì sự điều chỉnh dữ liệu nào khác .

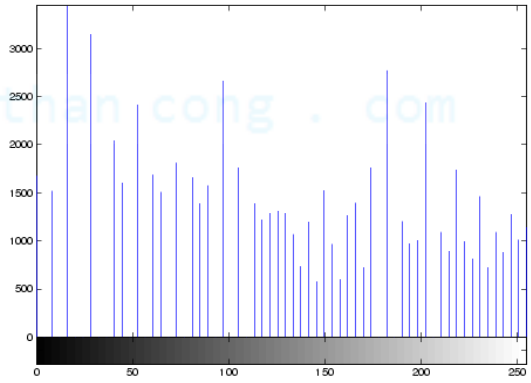
```
[X,map] = imread('forest.tif')
I = ind2gray(X,map);
J = imadjust(I,[],[],0.5);
imshow(I)
figure, imshow(J)
```

b- Sự làm cân bằng biểu đồ (Histogram Equalization)

- Quá trình điều chỉnh các giá trị cường độ có thể được thực hiện tự động bằng hàm histeq . Hàm histeq thực hiện một sự cân bằng biểu đồ bao gồm việc biến đổi các giá trị cường độ để biểu đồ của ảnh ra có thể xấp xỉ tương hợp với một biểu đồ xác định . (theo mặc định , hàm này cố gắng làm hợp tới một biểu đồ phẳng với 64 thùng – bins , nhưng ta có thể chỉ ra một biểu đồ khác)

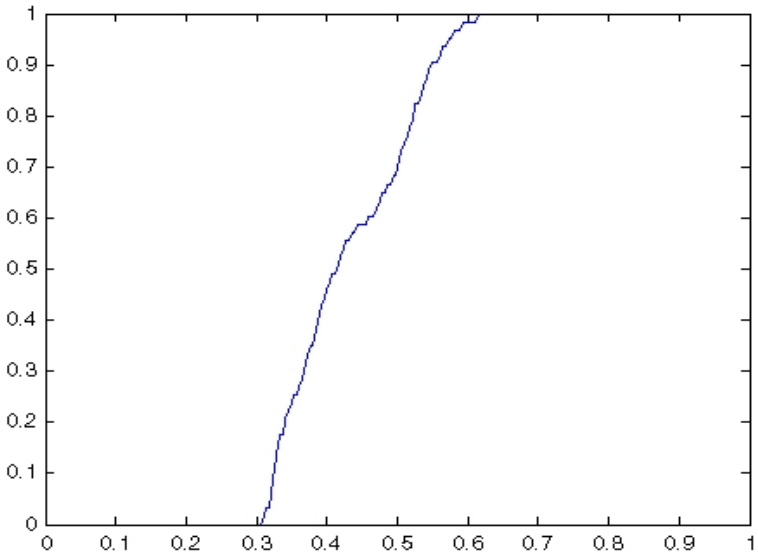
- Ví dụ sau đây minh hoạ việc sử dụng hàm histeq để điều chỉnh cường độ của một ảnh . Ảnh gốc có độ tương phản thấp với hầu hết các giá trị ở giữa của khoảng cường độ . Hàm histeq tạo ra một ảnh ra có các giá trị được phân bố đều trong ảnh

```
I = imread('pout.tif');
J = histeq(I);
imshow(J)
figure, imhist(J,64)
```



- Hàm histeq có thể trả về một véc tơ 1x256 chỉ ra mỗi giá trị vào có thể được cho ta kết quả của giá trị ra . (giá trị trong véctơ này nằm trong khoảng [0,1] bất kể ảnh vào thuộc lớp nào) . Ta có thể vẽ dữ liệu này để nhận đường cong biến đổi . Chẳng hạn :

```
I = imread('pout.tif');
[J,T] = histeq(I);
figure,plot((0:255)/255,T);
```



- Để ý đường cong này phản ánh biểu đồ trong hình trước với các giá trị vào nằm giữa 0.3 và 0.6 trong khi giá trị ra được phân bố giữa 0 và 1 .

c- Cân bằng biểu đồ thích ứng , độ tương phản giới hạn (Contrast – Limited Adaptive Histogram Equalization)

- Tương tự như sử dụng hàm histeq , ta có thể thực hiện sự cân bằng biểu đồ thích ứng độ tương phản giới hạn sử dụng hàm adapthisteq . Trong khi hàm histeq là việc trên toàn bức ảnh , hàm adapthisteq hoạt động trên một vùng nhỏ của ảnh được gọi là ***ngóii*** *(tiles)* . Độ tương phản của mỗi tile được làm giàu để cho biểu đồ của vùng ra xấp xỉ tương hợp với một biểu đồ xác định . Sau khi thi hành phép cân bằng , adapthisteq kết hợp các tile gần nhau sử dụng nội suy song tuyến thính để loại bỏ các ***artifact*** bao gồm các đường biên .

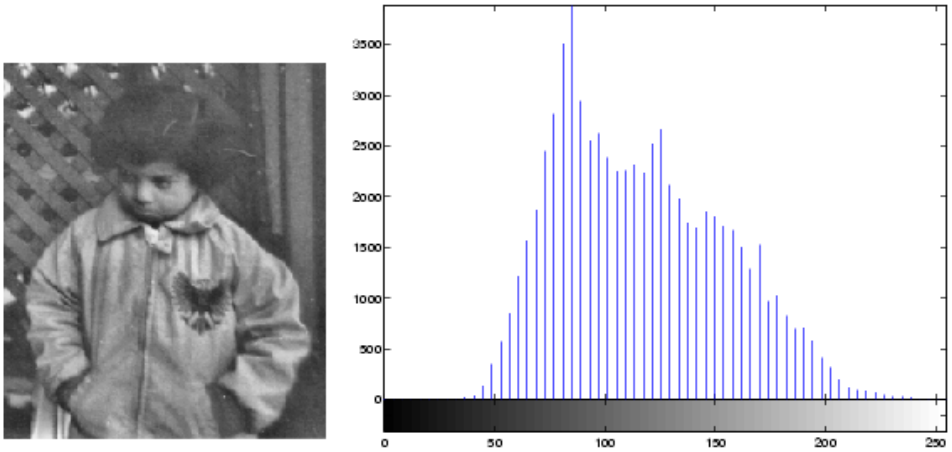
- Để tránh mở rộng nhiễu có trong ảnh , ta có thể sử dụng hàm adapthisteq với một số tham số tuỳ chọn để giới hạn độ tương phản đặc biệt trên các vùng thuần nhất .

- Để minh hoạ , ví dụ này sử dụng hàm adapthisted để điều chỉnh độ tương phản của một ảnh cường độ (ảnh đen trắng) . Ảnh gốc có độ tương phản thấp với hầu hết các giá trị nằm ở giữa của khoảng cường độ . Hàm adapthisteq tạo ra một ảnh có các giá trị được phân bố đều trong ảnh :

```
I = imread('pout.tif');
J = adapthisteq(I);
```

imshow(I)

figure, imshow(J)



d- Giãn không tương quan (Decorrelation Stretching)

- Ta áp dụng giãn không tương quan bằng cách sử dụng hàm decorrstretch . Số lượng giải màu , NBANDS trong một ảnh thường là 3 . Tuy nhiên , ta có thể áp đặt giãn không tương quan bất kì số giải màu nào ta muốn .

- Các giá trị màu gốc của ảnh được ánh xạ tới một tập các giá trị mới với một khoảng rộng hơn . Cường độ màu của mỗi pixel được biến đổi thành màu ***eigenspace*** của ma trận tương quan , được giãn để ngang bằng với sự biến đổi giải màu (band variances) sau đó được biến đổi lại thành giải màu gốc .

- Để định nghĩa một ***bandwise statistics*** , ta sử dụng toàn bộ ảnh gốc hoặc với tuỳ chọn subset , bất kì một vùng nào đó của ảnh .

Giãn không tương quan đơn giản

- Trong ví dụ này , ta thực hiện một phép giãn không tương quan trên một ảnh

1. Ảnh có 7 giải , nhưng ta chỉ đọc vào 3 màu nhìn thấy

A = multibandread('littlecoriver.lan', [512, 512, 7], ...

'uint8=>uint8', 128, 'bil', 'ieee-le', ...

{'Band','Direct',[3 2 1]});

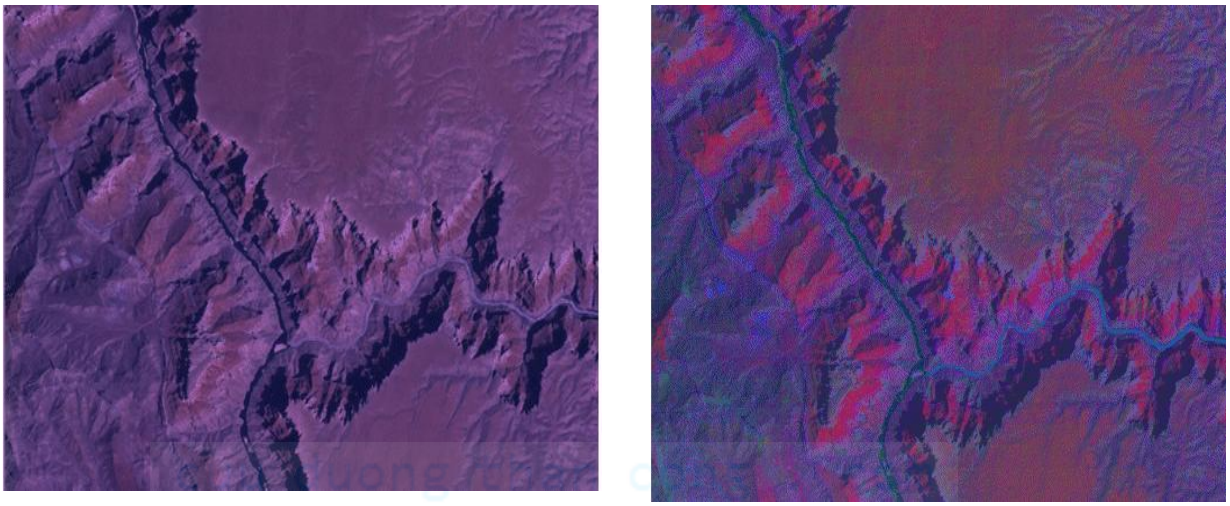
2. Thực hiện giãn không tương quan

B = decorrstretch(A);

3. Hiển thị kết quả

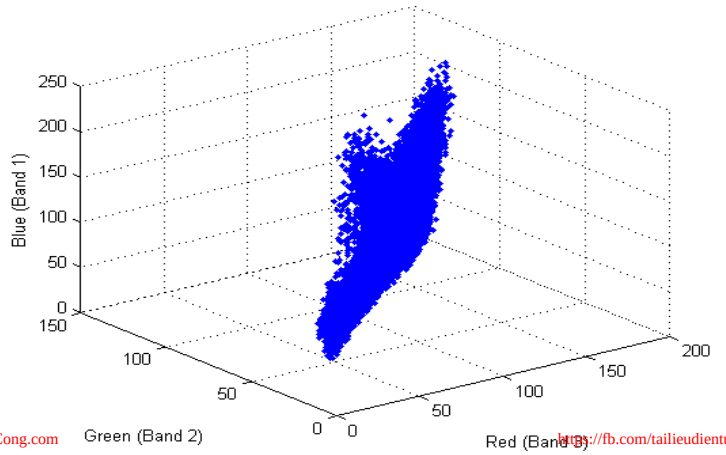
imshow(A); figure; imshow(B)

- So sánh hai ảnh , ảnh gốc có màu tím nhẹ trong khi ảnh bị biến đổi có một chút gì đó như được mở rộng khoảng màu



- Một đồ thị biểu diễn sự phân bố giải của ảnh sẽ chỉ ra các giải được làm mất tương quan và cân bằng như thế nào :

```
rA = A(:,:,1); gA = A(:,:,2); bA = A(:,:,3);
figure, plot3(rA(:),gA(:),bA(:),'.'); grid('on')
xlabel('Red (Band 3)'); ylabel('Green (Band 2)'); ...
zlabel('Blue (Band 1)')
rB = B(:,:,1); gB = B(:,:,2); bB = B(:,:,3);
figure, plot3(rB(:),gB(:),bB(:),'.'); grid('on')
xlabel('Red (Band 3)'); ylabel('Green (Band 2)'); ...
zlabel('Blue (Band 1)')
```

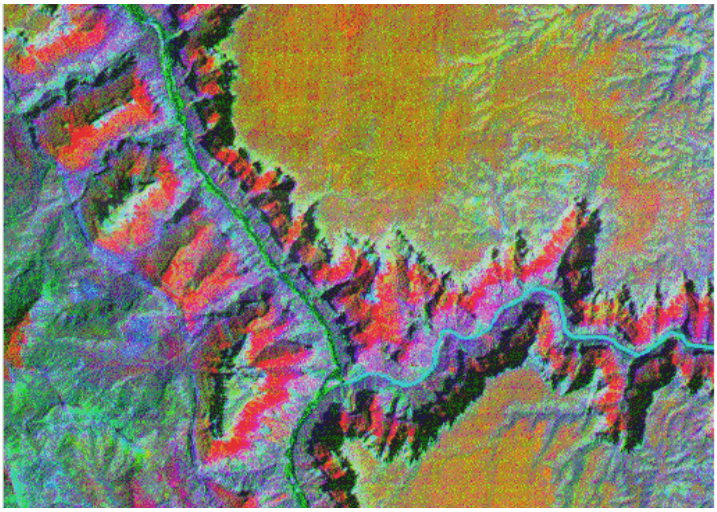


Giãn tương phản tuyến tính

- Bây giờ ta sẽ thực hiện một biến đổi tương tự nhưng với một phép giãn tương phản tuyến tính áp đặt lên ảnh sau khi giãn không tương quan .

imshow(A); C = decorrstretch(A,'Tol',0.01); figure; imshow(C)

- Hãy so sánh đã được biến đổi với ảnh gốc



- Việc thêm giãn tương phản tuyến tính làm giàu ảnh kết quả bằng cách mởi rộng hơn khoảng màu . Trong trường hợp này , khoảng màu được mở rộng được ánh xạ trong mỗi giải tới một khoảng chuẩn giữa 0.01 và 0.99 độ bão hoà 2%

5- Loại bỏ nhiễu

- Các ảnh số có thể có nhiều loại nhiễu khác nhau . Có một số cách mà nhiễu có thể thâm nhập vào trong ảnh phụ thuộc vào việc ảnh được tạo ra như thế nào . Chẳng hạn :

+ Nếu ảnh được scan từ một ảnh được chụp trên phim , các hạt trên phim là một nguồn nhiễu . Nhiễu có thể là kết quả của việc phim bị hư hỏng hoặc do chất lượng của scanner

+ Nếu ảnh được thu thập trực tiếp từ định dạng số , cơ chế của việc thu thập dữ liệu ảnh (chẳng hạn như bộ phát hiện CCD) có thể tạo ra nhiễu

+ Sự truyền dữ liệu ảnh cũng có thể bị nhiễu tác động

- Toolbox cung cấp một số cách để loại bỏ hoặc giảm nhiễu trong một ảnh . Các cách khác nhau được sử dụng cho các loại nhiễu khác nhau . Các cách đó bao gồm :

+Sử dụng các bộ lọc tuyến tính (Linear Filter)

+ Sử dụng các bộ median (Median Filter)

+ Sử dụng các bộ lọc thích nghi (Adaptive Filter)

- Để mô phỏng tác động của các vấn đề về nhiễu đã nói ở trên , toolbox cung cấp hàm imnoise mà ta có thể dùng để thêm nhiễu vào một ảnh .

a- Sử dụng các bộ lọc tuyến tính

- Ta có thể sử dụng các bộ lọc tuyến tính để loại bỏ nhiễu trong một ảnh . Các bộ lọc này chẳng hạn như bộ lọc trung bình hoặc bộ lọc Gauss là thích hợp . Chẳng hạn , một bộ lọc trung bình được sử dụng để loại bỏ các hạt nhiễu từ một ảnh chụp trên phim . Do mỗi pixel được thiết lập tới giá trị trung bình của các pixel xung quanh nó , do vậy sự biến động địa phương gây ra bởi các hạt nhiễu bị giảm bớt .

b- Sử dụng các bộ lọc median

-Sử dung các bộ lọc trung bình tương tự như việc sử dụng các bộ lọc trung bình (averaging filters) , mỗi pixel ra được thiết lập giá trị trung bình của các giá trị pixel lân cận của pixel vào tương ứng . Tuy nhiên , với bộ lọc này , giá trị của một pixel ra được quyết định bởi median của các pixel lân cận hơn là giá trị trung bình . Median thường nhỏ hơn nhiều so với trung bình các giá trị xa nhất (được gọi là outliers) . Bộ lọc median do đó tốt hơn để loại bỏ những outlier này mà không giảm độ sắc nét của ảnh . Hàm medfilt2 sử dụng phép lọc median

- Ví dụ sau đây so sánh việc sử dụng một bộ lọc trung bình và một bộ lọc median để loại bỏ nhiễu là các hạt "muối" và "hạt tiêu " . Loại nhiễu này bao gồm một tập hợp các pixel ngẫu nhiên được thiết lập thành màu đen hoặc trắng . Trong cả hai trường hợp , kích cỡ của vùng xung quanh được sử dụng để lọc là 3x3 .

1. Đọc ảnh và hiển thị nó

```
I = imread('eight.tif');
imshow(I)
```

2. Thêm nhiễu vào ảnh

```
J = imnoise(I,'salt & pepper',0.02);
figure, imshow(J)
```

3. Lọc nhiễu với bộ lọc trung bình sau đó hiển thị kết quả

K = filter2(fspecial('average',3),J)/255;

figure, imshow(K)

4. Sử dụng một bộ lọc median để lọc ảnh và hiển thị kết quả . Chú ý rằng hàm medfilt2 thực hiện việc loại bỏ nhiễu tốt hơn và ít làm mờ các cạnh

L = medfilt2(J,[3 3]);

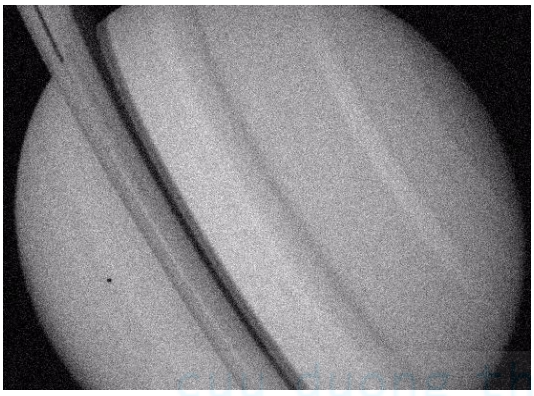
figure, imshow(K)

figure, imshow(L)

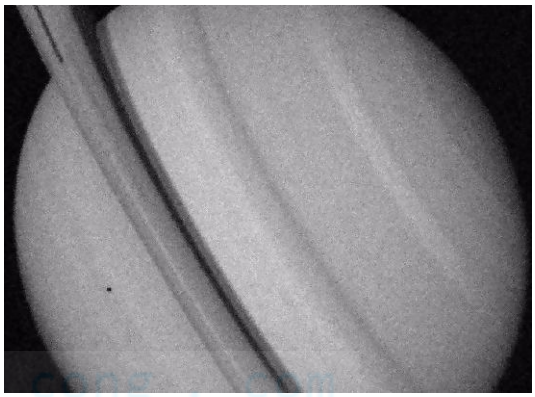
c- Sử dụng các bộ lọc thích nghi

- Hàm wiener2 áp đặt một bộ lọc Wiener (một kiểu của bộ lọc tuyến tính) lên một ảnh một cách thích nhi (adaptively) với sự biến động địa phương của ảnh . Những nơi có biến động lớn , hàm này thực hiện một số ít thao tác làm mượt ảnh . Những nơi có biến động nhỏ , hàm thực hiện nhiều các thao tác làm mượt ảnh .
- Cơ chế này thường tạo ra kết quả tốt hơn so với lọc tuyến tính . Các bộ lọc thích nghi thường được sử dụng nhiều hơn so với các bộ lọc tuyến tính trong việc bảo vệ các cạnh và vùng có tần số cao của một ảnh . Thêm vào , không có tác vụ thiết kế nào , hàm wiener2 điều khiển tất cả các tính toán ban đầu và thi hành phép lọc cho ảnh vào . Tuy nhiên , hàm này cần nhiều tính toán hơn các bộ lọc tuyến tính
- Hàm wiener2 làm việc tốt hơn khi nhiễu là nhiễu "trắng" chẳng hạnh như nhiễu Gauss . Ví dụ dưới đây áp đặt hàm wiener1 lên một ảnh có nhiễu Gauss tác động .

```
RGB = imread('saturn.png');
I = rgb2gray(RGB);
J = imnoise(I,'gaussian',0,0.005);
K = wiener2(J,[5 5]);
imshow(J)
figure, imshow(K)
```

Ảnh gốc



Ảnh kết quả

VII – Các thao tác hình thái (Morphological Operations)

- Morphology là một kĩ thuật xử lý ảnh dựa trên các hình dạng . Giá trị của mỗi pixel trong ảnh ra dựa trên cơ sở một sự so sánh của các pixel tương ứng trong ảnh vào với các pixel xung quanh nó . Bằng cách lựa chọn kích cỡ và hình dạng của vùng xung quanh , ta có thể xây dựng một thao tác hình thái rất nhạy cảm với một số hình dạng đặc thù trong ảnh vào .
- Ta sẽ xem xét các hàm xử lý hình thái của toolbox. Ta có thể sử dụng những hàm này để thực hiện các thao tác xử lý thông dụng như làm giàu độ tương phản (contrast enhancement) , loại bỏ nhiễu (nose removal) , làm mỏng (thinning) , tô (filling) và phân vùng (segmentation) .

1. Bảng thuật ngữ

Tên thuật ngữ	Diễn giảI
Background	Nền . Trong ảnh nhị phân , các pixel được thiết lập giá trị 0 được xem xét như là nền của ảnh . Khi ta nhìn một ảnh nhị phân , màu của nền là màu đen
Connectivity	Các tiêu chuẩn mô tả việc các pixel trong một ảnh hình thành nên một nhóm gắn kết như thế nào . Toolbox cung trợ giúp connectivity 2 chiều cũng như nhiều chiều
Foreground	Trong một ảnh nhị phân , các pixel được thiết lập giá trị 1 được xem như foreground . Khi quan sát một ảnh nhị phân , màu của foreground là màu trắng
Global maxima	Vùng cao nhất trong ảnh .
Global minima	Vùng thấp nhất trong ảnh`
Morphology	Một tập hợp rộng của các thao tác xử lý ảnh mà xử lý trên các hình dạng . Các thao tác này áp đặt một phần tử có cấu trúc lên một ảnh , tạo một ảnh ra có cùng kích thước với ảnh vào . Các thao tác hình thái cơ bản nhất là giãn nở (dilation) và ăn mòn (erosion)

Neighborhood	Vùng xung quanh – đó là tập các pixel được định nghĩa bởi vị trí tương đối của chúng với pixel quan tâm . Một vùng xung quanh có thể được định nghĩa bởi một phần tử có cấu trúc hoặc bằng cách chỉ ra một connectivity (tính kết nối)
Object	Tập các pixel trong một ảnh nhị phân hình thành nên một nhóm kết nối . Trong ngữ cảnh của xem xét này , “object” và “connected component” là đồng nhất .
Packed binary	Phương pháp nén ảnh nhị phân mà có thể làm tăng tốc quá trình xử lý ảnh
Regional maxima	Tập các pixel kết nối có cùng cường độ lớn nhất trong một vùng . Tất cả các pixel xung quanh đều có giá trị nhỏ hơn giá trị này
Regional minima	Tập các pixel kết nối có cùng cường độ nhỏ nhất
Structuring element	Ma trận được sử dụng để định nghĩa một hình dạng xung quanh và kích cỡ cho tác thao tác hình thái bao gồm giãn nở và xói mòn . Nó chỉ chứa các giá trị 0 và 1 và có thể có một hình dạng và kích cỡ bất kì . Các pixel với giá trị 1 định nghĩa vùng xung quanh

2. Giãn nở và xói mòn

- Giãn nở và xói mòn là hai thao tác xử lý hình thái cơ bản . Giãn nở cộng thêm các pixel tới vùng biên của các đối tượng trong một ảnh trong khi xói mòn loại bỏ các pixel trên vùng biên của các đối tượng . Số lượng pixel được thêm vào hoặc loại bỏ từ các đối tượng trong một ảnh phụ thuộc vào kích cỡ và hình dạng của phần tử cấu trúc được sử dụng để xử lý ảnh . Phần này sẽ :

+ Cung cấp những thông tin cơ bản về việc các hàm giãn nở và xói mòn hoạt động như thế nào

+ Mô tả phần tử có cấu trúc và cách tạo ra chúng

+ Mô tả cách thực hiện xói mòn

+ Mô tả một số thao tác cơ bản dựa trên giãn nở và xói mòn

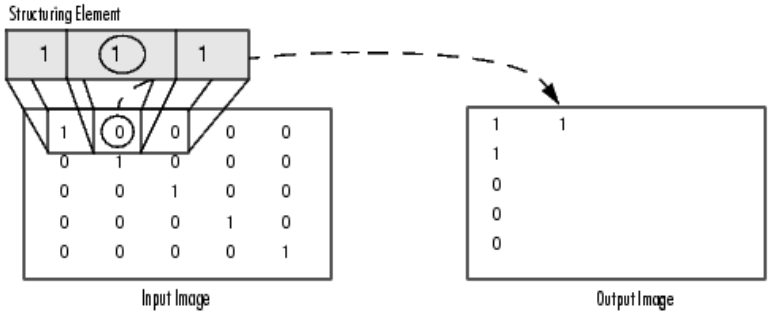
+ Mô tả các hàm của toolbox dựa trên cơ sở của giãn nở và xói mòn

a-Tổng quan về Dilation và Erosion

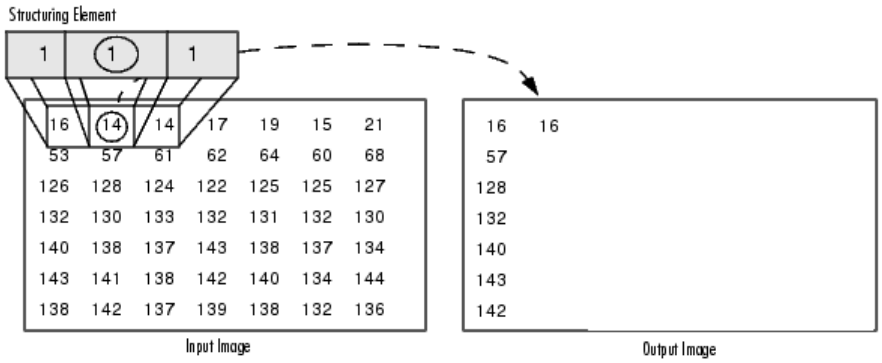
- Trong các thao tác giãn nở và xói mòn , trạng thái của một pixel nào đó trong ảnh ra được quyết định bằng việc áp đặt một quy tắc lên các pixel tương ứng và các pixel lân cận trong ảnh vào . Quy tắc này được sử dụng để xử lý các pixel định nghĩa thao tác giãn nở hay xói mòn . Bảng sau đây liệt kê các quy tắc cho cả hai phép xử lý này ;

Tao tác	Quy tắc
Dilation	Giá trị của các pixel ra là giá trị lớn nhất của tất cả các pixel trong vùng lân cận của pixel vào tương ứng . Trong một ảnh nhị phân , nếu bất kì pixel nào có giá trị 1 , pixel ra sẽ là 1
Erosion	Giá trị của pixel ra là giá trị nhỏ nhất của tất cả các pixel trong vùng lân cận của pixel vào tương ứng . Nếu trong một ảnh nhị phân có một pixel có giá trị 0 thì pixel ra có giá trị 0

- Hình sau đây minh hoạ sự giãn nở của một ảnh nhị phân . Chú ý cách phần tử cấu trúc định nghĩa vùng xung quanh của pixel quan tâm , đó là vùng hình tròn . Hàm giãn nở áp đặt quy luật tương ứng lên các pixel trong vùng lân cận và gán một giá trị tới pixel tương ứng trong ảnh ra . Trong hình , hàm giãn nở hình thái thiết lập các giá trị của pixel ra bằng 1 bởi vì một trong những phần tử trong vùng lân cận được định nghĩa bởi phần tử cấu trúc có giá trị 1



- Hình sau đây minh hoạ quá trình xử lý ảnh cho một ảnh đen trắng . Hình này chỉ ra quá trình xửlý của một pixel cụ thể trong ảnh vào . Chú ý việc hàm đã áp đặt quy tắc tới vùng lân cận của pixel vào và sử dụng giá trị cao nhất của tất cả các pixel trong vùng này như là giá trị tương ứng cho pixel ra như thế nào .



Xử lý các pixel ở biên ảnh

- Các hàm xử lý hình thái định vị gốc của phần tử cấu trúc , tâm của nó qua các pixel quan tâm trong ảnh vào . Với các pixel ở cạnh của một ảnh , các phần lân cận được định nghĩa bởi phần tử cấu trúc có thể mở rộng qua biên của ảnh

-Để xử lý các pixel trên biên ảnh ,các hàm xử lý hình thái gán một giá trị tới những pixel không xác định này giống như các hàm đã thêm vào ảnh với một số hàng và cột nào đó . Giá trị của những pixel thêm vào này thay đổi với cả thao tác giãn nở lẫn xói mòn . Bảng sau đây mô tả quy tắc thêm (padding) cho giãn nở và xói mòn trên cả ảnh nhị phân và ảnh đen trắng .

Thao tác	Quy tắc
Dilation	Các pixel ngoài biên ảnh được gán giá trị nhỏ nhất tuỳ theo kiểu dữ liệu . Với ảnh nhị phân , những pixel này được gán giá trị 0 . Với ảnh đen trắng , giá trị nhỏ nhất cho ảnh uint8 là 0
Erosion	Các pixel ngoài biên ảnh được gán giá trị lớn nhất tuỳ theo kiểu dữ liệu . Với ảnh nhị phân , những pixel này được gán giá trị 1 . Với ảnh đen trắng , gía trị lớn nhất của ảnh uint8 là 255

Chú ý : Bằng cách sử dụng giá trị nhỏ nhất cho thao tác giãn nở và giá trị lớn nhất cho thao tác xói mòn , các hàm toolbox tránh việc tác động lên biên . Các vùng gần biên của ảnh ra không đồng nhất với phần còn lại của ảnh . Chẳng hạn , nếu xói mòn được thêm với giá trị nhỏ nhất , việc xói mòn một ảnh sẽ dẫn đến đường biên đen xung quanh cạnh của ảnh ra .

b-Phần tử cấu trúc

- Một phần không thể thiếu được của các thao tác giãn nở và xói mòn là phần tử cấu trúc được sử dụng để thăm dò ảnh vào . Phần tử cấu trúc 2 chiều hoặc phẳng bao gồm một ma trận của 0 và 1 thường nhỏ hơn ma trận ảnh đang được xử lý . Pixel trung tâm của phần tử cấu trúc được gọi là gốc phân biệt với các pixel quan tâm – pixel đang được xử lý . Những pixel trong phần tử cấu trúc chứa giá trị 1 định nghĩa vùng lân cận của phần tử cấu trúc . Những pixel này cũng được xem xét trong quá trình giãn nở hoặc xói mòn . Phần tử 3 chiều hay không phẳng sử dụng 0 và 1 để định nghĩa quy mô của phần tử cấu trúc trong mặt phẳng x-y và thêm vào giá trị chiều cao để định nghĩa chiều thứ 3 .

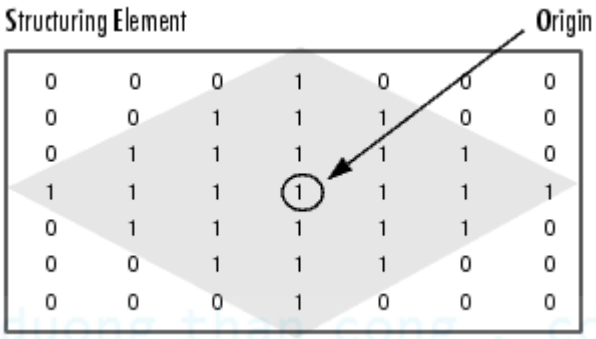
Gốc của phần tử cấu trúc

- Các hàm xửlý hình thái sử dụng mã sau đây để lấy toạ độ của gốc phần tử cấu trúc có kích thước và chiều bất kì

origin = floor((size(nhood)+1)/2)

(Trong mã trên , nhood là vùng lân cận định nghĩa phần tử cấu trúc . Do phần tử cấu trúc là đối tượng trong Matlab , ta không thể sử dụng kích thước của đối tượng STREL trong tính toán này . Ta phải sử dụng phương thức getnhood của STREL để nhận vùng lân cận của phần tử cấu trúc từ đối tượng STREL)

- Chẳng hạn , ví dụ sau minh hoạ một phần tử cấu trúc có hình thoi



Tạo phần tử cấu trúc

- Các hàm giãn nở và xói mòn của toolbox chấp nhận các đối tượng phần tử cấu trúc được gọi là STRELs . Ta sử dụng hàm strel để tạo ra các STRELs có kích thước và hình dạng bất kì . Hàm strel cũng bao gồm trợ giúp sẵn cho các hình dạng thông dụng khác như : đường thẳng , hình thoi , hình đĩa , đường thẳng điều hoà và hình quả bóng

- Chẳng hạn , đoạn mã sau tạo một phần tử cấu trúc phẳng có dạng hình thoi

```
se = strel('diamond',3)
se =

Flat STREL object containing 25 neighbors.
Decomposition: 3 STREL objects containing a total of 13 neighbors

Neighborhood:
     0     0     0     1     0     0     0
     0     0     1     1     1     0     0
```

0 1 1 1 1 1 0
1 1 1 1 1 1 1
0 1 1 1 1 1 0
0 0 1 1 1 0 0
0 0 0 1 0 0 0

Phân tích phần tử cấu trúc (Structuring Element Decomposition)

- Để nâng cao chất lượng , hàm strel có thể ngắt phần tử cấu trúc thành nhiều phần nhỏ hơn – kĩ thuật phân tích phần tử cấu trúc
- Chẳng hạn , giãn nở bởi một phần tử cấu trúc vuông 11x11 có thể thu được bằng cách đầu tiên , giãn nở với phần tử cấu trúc 1x1 và sau đó với một phần tử cấu trúc 11x11 . Điều này , theo lý thuyết , dẫn đến sự tăng tốc đáng kể của một nhân tố cỡ 5.5 mặc dù trong thực tế tốc độ thực tế là nhỏ hơn .
- Việc phân tích phần tử cấu trúc được sử dụng cho các hình đĩa và bóng là xấp xỉ , tất cả những phân tích khác là chính xác . Phân tích không được sử dụng với một phần tử cấu trúc bất kì trừ khi nó là phần tử cấu trúc phẳng - tất cả các pixel lân cận có giá trị 1
- Để quan sát thứ tự của phần tử cấu trúc được sử dụng trong phép phân tích , sử dụng phương thức STREL getsequence . Hàm getsequence trả lại một mảng của các phần tử cấu trúc tạo nên sự phân tích . Chẳng hạn , đây là phần tử cấu trúc được tạo ra trong phép phân tích một phần tử cấu trúc hình thoi .

sel = strel('diamond',4)

sel =

Flat STREL object containing 41 neighbors.

Decomposition: 3 STREL objects containing a total of 13 neighbors

Neighborhood:

0 0 0 0 1 0 0 0 0
0 0 0 1 1 1 0 0 0
0 0 1 1 1 1 1 0 0
0 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 0

```
     0     0     1     1     1     1     1     0     0
     0     0     0     1     1     1     0     0     0
     0     0     0     0     1     0     0     0     0
```

```
seq = getsequence(sel)
seq =
3x1 array of STREL objects

seq(1)
ans =
Flat STREL object containing 5 neighbors.

Neighborhood:
     0     1     0
     1     1     1
     0     1     0

seq(2)
ans =
Flat STREL object containing 4 neighbors.

Neighborhood:
     0     1     0
     1     0     1
     0     1     0

seq(3)
ans =
Flat STREL object containing 4 neighbors.

Neighborhood:
     0     0     1     0     0
```

0 0 0 0 0
1 0 0 0 1
0 0 0 0 0
0 0 1 0 0

c- Giãn nở ảnh (Dilating an Image)

- Để giãn nở một ảnh , sử dụng hàm imdilate . Hàm này nhận hai tham số chính :

+ Ảnh vào – là ảnh cần xử lý (đen trắng , nhị phân hoặc ảnh nhị phân được đóng gói)

+ Một phần tử cấu trúc được trả lại từ hàm strel hoặc một ma trận nhị phân định nghĩa vùng lân cận của một phần tử cấu trúc

- Hàm imdilate cũng chấp nhận hai tham số tuỳ chọn là : PADOPT và PACKDOPT . Tham số PADOPT tác động lên kích cỡ của ảnh ra . Tham số PACKDOPT phân biệt ảnh vào như ảnh nhị phân đóng gói .

- Ví dụ sau sẽ giãn nở một ảnh nhị phân chứa một đối tượng là một hình chữ nhật :

BW = zeros(9,10);

BW(4:6,4:7) = 1

BW =

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

- Để mở rộng tất cả các mặt của thành phần foreground , ví dụ này sử dụng một phần tử cấu trúc có kích thước 3x3 .

SE = strel('square',3)

SE =

Flat STREL object containing 3 neighbors.

Neighborhood:

1 1 1

1 1 1

1 1 1

- Để giãn nở ảnh , truyền ảnh BW và phần tử cấu trúc SE tới hàm imdilate .

BW2 = imdilate(BW,SE)

```
BW2 =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

d- Làm xói mòn ảnh (Eroding an Image)

- Để làm xói mòn một ảnh , sử dụng hàm imerode . Hàm này nhận hai tham số chính :

+ Ảnh vào – là ảnh cần được xử lý (đen trắng , nhị phân hay nhị phân đóng gói)

+ Một phần tử cấu trúc được trả về từ hàm strel hoặc một ma trận nhị phân định nghĩa vùng lân cận của phần tử cấu trúc

- Hàm imerode cũng chấp nhận 3 tham số tuỳ chọn là PADOPT , PACKDOP và M . PADOPT ảnh hưởng đến kích cỡ của ảnh ra . PACKDOPT phân biệt ảnh vào như là ảnh nhị phân đóng gói (packed binary image) . Nếu là ảnh nhị phân đóng gói , M phân biệt số lượng các hàng trong ảnh gốc .

- Ví dụ sau làm xói mòn một ảnh nhị phân :

1. Đọc ảnh vào trong không gian làm việc của Matlab

BW1 = imread('circbw.tif');

2. Tạo một phần tử cấu trúc . Đoạn mã sau tạo ra một phần tử cấu trúc chéo .

SE = strel('arbitrary',eye(5));

SE=

Flat STREL object containing 5 neighbors.

Neighborhood:

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

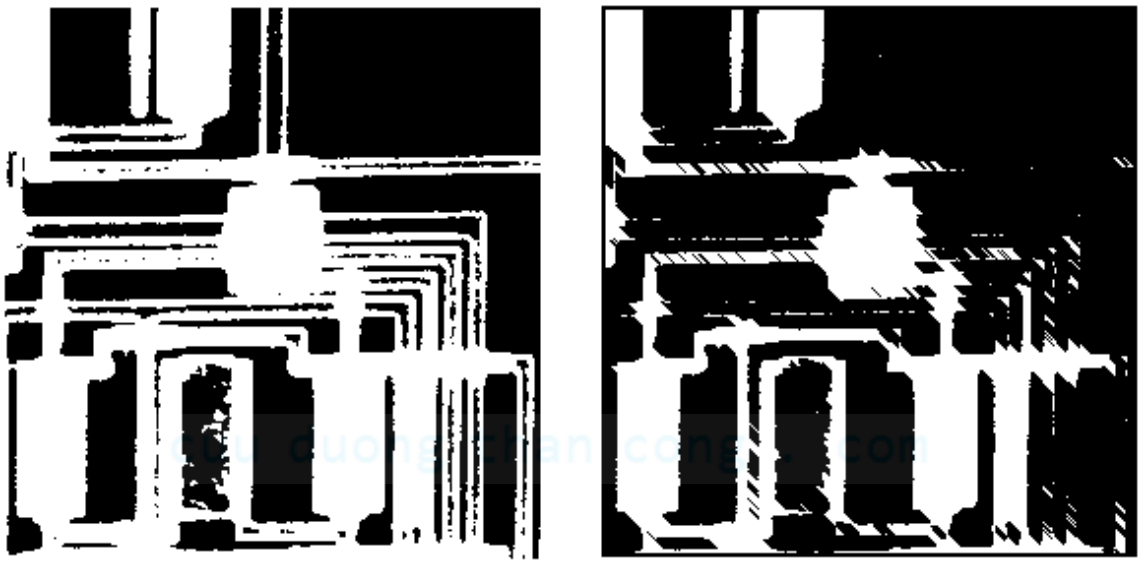
3. Gọi làm imerode , truyền ảnh BW và phần tử cấu trúc SE làm tham số cho hàm

BW2 = imerode(BW1,SE);

Để ý đường chéo trên mặt phải của ảnh ra . Chúng là do hình dạng của phần tử cấu trúc quyết định

imshow(BW1)

figure, imshow(BW2)



Original Image

Eroded Image

d- Kết hợp giữa giãn nở và xói mòn

- Giãn nở và xói mòn thường được sử dụng kết hợp để thực hiện các thao tác xử lý ảnh . Chẳng hạn , định nghĩa của mở hình thái (Morphological Opening) là một thao tác xói mòn tiếp theo đó là một thao tác giãn nở sử dụng cùng một phần tử cấu trúc . Tương tự , định nghĩa của đóng hình thái (Morphological Closing) bao gồm một thao tác giãn nở theo sau là xói mòn cũng với cùng phần tử cấu trúc

- Sau đây ta sẽ sử dụng hàm imdilate và imerode để minh hoạ việc bắt đầu một biến đổi hình thái như thế nào . Tuy nhiên , lưu ý , toolbox đã cung cấp hàm imopen để thực hiện xử lý này .

Mở biến đổi hình thái

- Ta có thể sử dụng mở biến đổi hình thái để loạ bỏ các đối tượng nhỏ từ một ảnh trong khi vẫn giữ nguyên hình dạng và kích thước của các đối tượng lớn . Chẳng hạn , ta có thể sử dụng hàm imopen để loại bỏ một đường tròn từ một ảnh , tạo ra một ảnh mà chỉ chứa các hình chữ nhật

- Để mở hình thái một ảnh , thực hiện những bước sau đây :

1. Đọc ảnh vào trong không gian làm việc của Matlab

```
W1 = imread('circbw.tif');
```

2. Tạo phần tử cấu trúc

```
SE = strel('rectangle',[40 30]);
```

- Phần tử cấu trúc phải đủ lớn để có thể loại bỏ các đường tròn khi ta xói mòn ảnh tuy nhiên không quá lớn để khỏi loại bỏ các hình chữ nhật .

3. Xói mòn ảnh với phần tử cấu trúc

```
BW2 = imerode(BW1,SE);
imshow(BW2)
```

- Hai dòng mã trên sẽ loại bỏ tất cả các đường cong tuy nhiên nó làm co các hình chữ nhật

4. Để phục hồi các hình chữ nhật với kích thước ban đầu , giãn nở ảnh vừa thu được sử dụng cùng phần tử cấu trúc ở trên

BW3 = imdilate(BW2,SE);

imshow(BW3

e- Các hàm dựa trên cơ sở của giãn nở và xói mòn

- Ta sẽ xem xét các thao tác xử lý ảnh thông dụng dựa trên cơ sở của giãn nở và xói mòn bao gồm :

+ Định khung cho ảnh (Skeletonization)

+ Xác định chu vi

- Bảng sau đây liệt kê các hàm khác trong toolbox thực hiện các thao tác biến đổi hình thái thông dụng dựa trên cơ sở của giãn nở và xói mòn

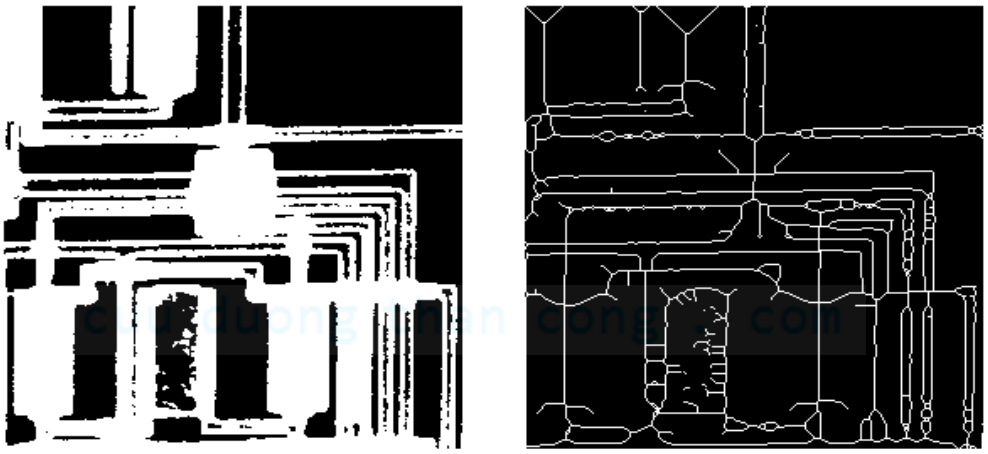
Hàm	Chức năng
Bwhitmisss	Và (AND) lô-gíc của một ảnh được xói mòn với cùng một phần tử cấu trúc và ảnh bù của nó được xói mòn với một phần tử cấu trúc thứ hai

Imbothat	Trừ ảnh gốc từ một phiên bản đóng hình thái của ảnh . Có thể được sử dụng để tìm các máng cường độ trong một ảnh
Imclose	Giãn nở một ảnh và sau đó xói mòn nó sử dụng cùng một phần tử cấu trúc
Imopen	Xói mòn một ảnh sau đó giãn nở nó sử dụng cùng một phần tử cấu trúc
Imtophat	Trừ một ảnh mở cấu trúc từ ảnh gốc . Có thể được sử dụng để làm tăng độ tương phản của một ảnh

Định khung cho ảnh

- Đó là thuật ngữ chỉ việc biến đổi tất cả các đối tượng trong ảnh thành các đường (bộ khung) mà không thay đổi cấu trúc ban đầu của ảnh - sử dụng hàm bwmorph .

```
BW1 = imread('circbw.tif');
BW2 = bwmorph(BW1,'skel',Inf);
imshow(BW1)
figure, imshow(BW2)
```



Original Image

Skeletonization of Image

Xác định các đường chu vi

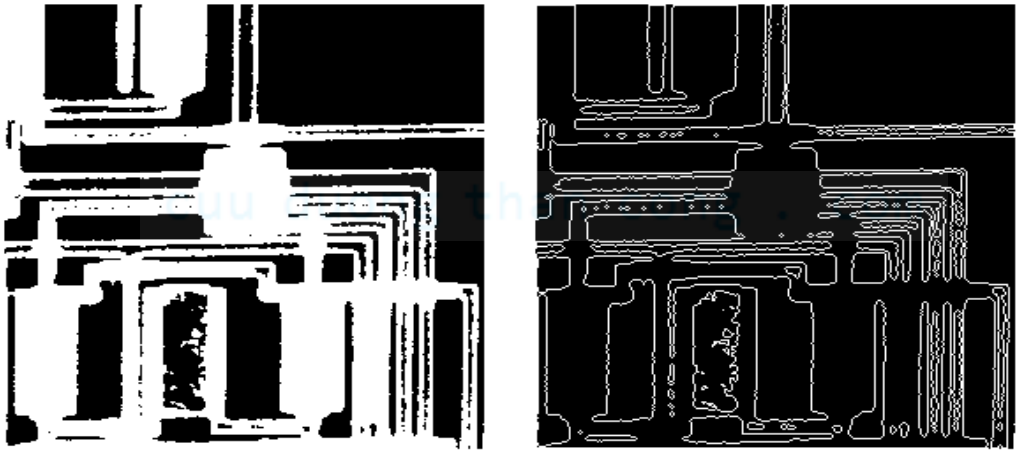
- Đây là thuật ngữ để chỉ việc thay thế tất cả các đối tượng trong ảnh bằng các đường chu vi của chúng . Hàm bwperim tính các pixel trên chu vi của các đối tượng trong một ảnh . Một pixel được xem như trên chu vi nếu nó thoả hai điều kiện :

+ Giá trị pixel bằng 1

+ Một hoặc nhiều pixel xung quanh nó có giá trị 0

Chẳng hạn , đoạn mã sau tìm các pixel trên chu vi của một ảnh nhi phân

```
BW1 = imread('circbw.tif');
BW2 = bwperim(BW1);
imshow(BW1)
figure, imshow(BW2)
```



Original Image Perimeters Determined

3- Tái tạo hình thái ảnh

- Tái tạo hình thái là một phần khác của kĩ thuật xử lý hình thái ảnh . Dựa trên giãn nở , tái tạo hình thái có 3 tính chất duy nhất :

+ Xử lý trên cơ sở hai ảnh , một ảnh ghi và một mặt nạ đúng hơn là một ảnh và một phần tử cấu trúc

+ Quá trình xử lý được lặp lại cho đến khi ổn định

+ Quá trình xử lý dựa trên cơ sở khái niệm về tính kết nối (connectivity) hơn là dựa trên cơ sở của phần tử cấu trúc

- Phần này sẽ :

+ Cung cấp các thông tin cơ sở về tái tạo hình thái và mô tả cách sử dụng hàm imreconstruct như thế nào

+ Mô tả tính kết nối các pixel (pixel connectivity) ảnh hưởng ra sao lên tái tạo hình thái

+ Mô tả cách sử dụng hàm imfill - dựa trên cơ sở của tái tạo hình thái

+ Mô tả một nhóm các hàm khác dựa trên cơ sở của tái tạo hình thái .

a- Ảnh ghi và mặt nạ (Marker and Mask)

- Xử lý tái tạo hình thái trên một ảnh , ảnh đó được gọi là ảnh ghi (marker) . Quá trình xử lý dựa trên cơ sở các đặc tính của một ảnh khác , ảnh đó được gọi là mặt nạ (mask) . Điểm cao nhất hay đỉnh trong ảnh ghi chỉ ra vị trí bắt đầu quá trình xử lý . Quá trình xử lý sẽ tiếp tục cho đến khi giá trị ảnh ổn định .

- Để minh hoạ tái tạo hình thái , xem xét ví dụ sau đây . Nó chứa hai vùng chính , các khối pixel bao gồm các giá trị 14 và 18 . Nền chủ yếu được thiết lập giá trị bằng 10 với một số pixel được thiết lập giá trị 11 .

```
A = [10   10   10   10   10   10   10   10   10   10;
     10   14   14   14   10   10   11   10   11   10;
     10   14   14   14   10   10   10   11   10   10;
     10   14   14   14   10   10   11   10   11   10;
     10   10   10   10   10   10   10   10   10   10;
     10   11   10   10   10   18   18   18   10   10;
     10   10   10   11   10   18   18   18   10   10;
     10   10   11   10   10   18   18   18   10   10;
     10   11   10   11   10   10   10   10   10   10;
     10   10   10   10   10   10   11   10   10   10];
```

- Để tái tạo hình thái ảnh này , thực hiện những bước sau :

1. Tạo một ảnh ghi . Với cùng phần tử cấu trúc trong giãn nở và xói mòn , các đặc tính của ảnh ghi quyết định quá trình xử lý đã được thực hiện trong tái tạo ảnh . Các đỉnh trong ảnh ghi sẽ phân biệt vị trí của các đối tượng trong ảnh mặt nạ mà ta muốn làm nổi bật .

Một cách để tạo một ảnh ghi là trừ một hằng số từ ảnh mặt nạ sử dụng hàm số học ảnh imsubtract

```
marker = imsubtract(A,2)
marker =
   8   8   8   8   8   8   8   8   8   8
   8  12  12  12   8   8   9   8   9   8
```

8	12	12	12	8	8	8	9	8	8
8	12	12	12	8	8	9	8	9	8
8	8	8	8	8	8	8	8	8	8
8	9	8	8	8	16	16	16	8	8
8	8	8	9	8	16	16	16	8	8
8	8	9	8	8	16	16	16	8	8
8	9	8	9	8	8	8	8	8	8
8	8	8	8	8	8	9	8	8	8

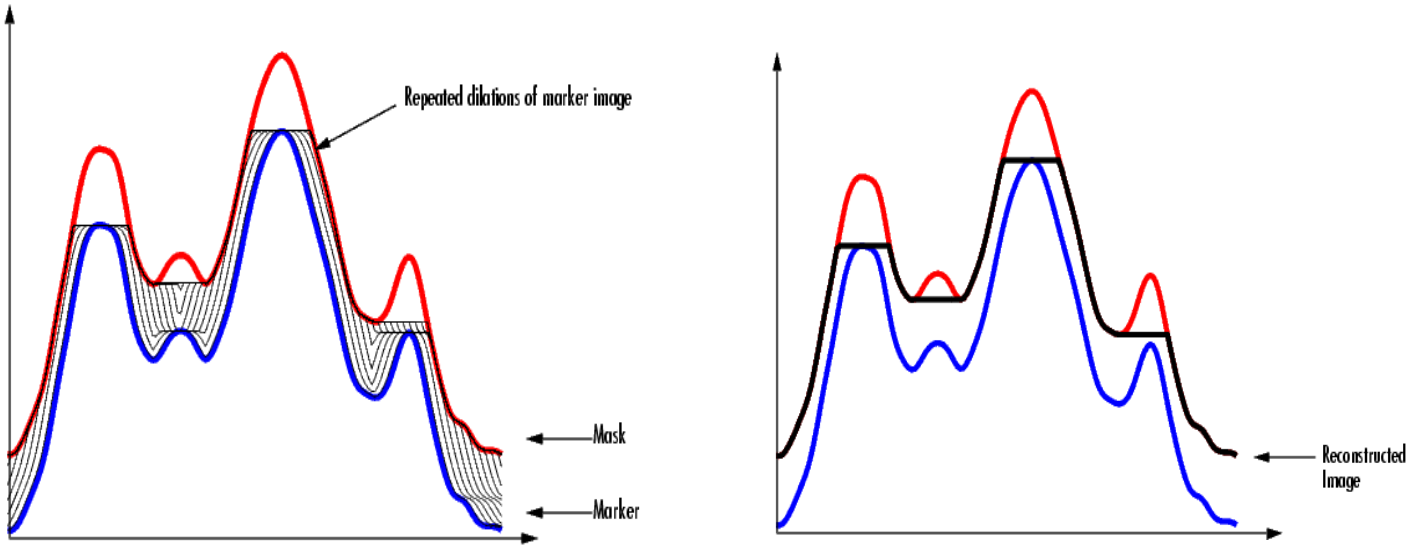
2. Gọi hàm imreconstruct để tái tạo hình thái ảnh . Trong ảnh ra , chú ý rằng tất cả những sự dao động về cường độ ngoại trừ giá trị đỉnh đã bị loại bỏ

recon = imreconstruct(marker, mask)

```
recon =
    10    10    10    10    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    16    16    16    10    10
    10    10    10    10    10    16    16    16    10    10
    10    10    10    10    10    16    16    16    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10
```

Tái tạo hình thái ảnh là gì ?

- Tái tạo hình thái có thể được xem như một quá trình giãn nở lặp lại trên ảnh ghi cho đến khi các đường viền của ảnh ghi vừa khít ở dưới ảnh mặt nạ . Theo cách này , các đỉnh trong ảnh ghi trải rộng ra hay giãn nở
- Hình sau đây minh hoạ quá trình xử lý này trong 1 chiều . Mỗi xự giãn nở liên tiếp được ràng buộc để nằm dưới mặt nạ . Khi các giãn nở tiếp sau đó không thay đổi ảnh , quá trình xử lý chấm dứt . Giãn nở cuối cùng là ảnh đã được tái tạo .



b- Tính liên kết các pixel

- Quá trình xử lý hình thái bắt đầu tại đỉnh của ảnh ghi và trải rộng ra các phần còn lại của ảnh trên cơ sở tính liên kết của các pixel . Tính liên kết định nghĩa các pixel được liên kết như thế nào với các pixel khác . Chẳng hạn , ảnh nhị phân này chứa một đối tượng foreground - tất cả các pixel có giá trị 1. Nếu foreground là 4 kết nối , ảnh có một đối tượng background và tất cả các pixel sẽ có giá trị 0 . Tuy nhiên , nếu foreground là 8 kết nối , foreground tạo một vòng đóng và ảnh có hai đối tượng background riêng rẽ : các pixel ở trong vòng và các pixel ở ngoài vòng

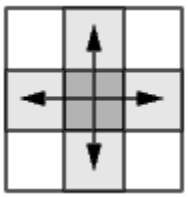
0 0 0 0 0 0 0 0
0 1 1 1 1 1 0 0
0 1 0 0 0 1 0 0
0 1 0 0 0 1 0 0
0 1 0 0 0 1 0 0
0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

Định nghĩa tính kết nối trong một ảnh

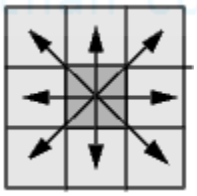
-Sau đây ta sẽ liệt kê tất các tính kết nối 2 và 3 chiều chuẩn được trợ giúp bởi toolbox .

Các kết nối hai chiều

+ 4 kết nối (4 connected) : Các pixel được kết nối với các pixel ở các cạnh của nó . Điều này có nghĩa rằng một gặp pixel liền nhau là một phần của cùng một đối tượng chỉ nếu chúng đều có giá trị 1 và được kết nối dọc theo chiều nằm ngang và chiều thẳng đứng .

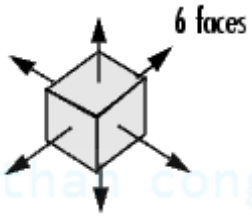


+ 8 kết nối (8 connected) : Các pixel được kết nối nếu các cạnh hoặc các góc của chúng chạm nhau . Điều này có nghĩa rằng nếu hai pixel liền nhau có giá trị 1 (on) , chúng sẽ là một phần của cùng một đối tượng bất kể chúng có được kết nối theo chiều ngang ,dọc hay chéo hay không .

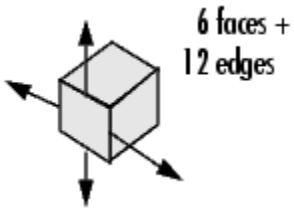


Các kết nối 3 chiều

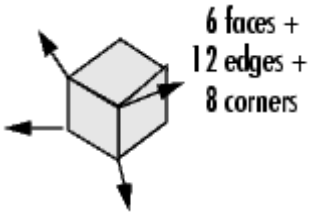
+ 6 kết nối : Các pixel được kết nối nếu các mặt của chúng tiếp xúc nhau



+ 18 kết nối : Các pixel được kết nối nếu các mặt hoặc các cạnh của chúng tiếp xúc nhau



+ 26 kết nối : Các pixel được kết nối nếu các mặt , cạnh hoặc góc của chúng chạm nhau



Chọn lựa một kết nối

- Kiểu của vùng lân cận mà ta chọn sẽ ảnh hưởng đến số lượng đối tượng được tìm thấy trong một ảnh và các vùng biên của chúng . Vì lý do này , kết quả của nhiều thao tác biến đổi hình thái thường khác nhau phụ thuộc vào kiểu kết nối mà ta đã chọn
- Chẳng hạn , nếu ta chọn kiểu 4 kết nối thì ảnh nhị phân này sẽ chứa hai đối tượng , nếu ta chọn kiểu 8 kết nối , ảnh chỉ có một đối tượng .

0 0 0 0 0 0
0 1 1 0 0 0
0 1 1 0 0 0
0 0 0 1 1 0
0 0 0 1 1 0

Chỉ định các kết nối tuỳ chọn

- Ta có thể định nghĩa các vùng lân cận bằng cách chỉ ra một mảng 3x3x3…x3 của các giá trị 0 ,1 . Các phần tử có giá trị 1 định nghĩa kết nối của lân cận trong quan hệ với tâm của phần tử . Chẳng hạn , mảng sau đây định nghĩa một kết nối "Bắc/Nam" có tác dụng ngắt một ảnh ra nhiều cột độc lập .

CONN = [0 1 0; 0 1 0; 0 1 0]

CONN =

0 1 0
0 1 0
0 1 0

c- Các thao tác tô lấp (Flood Fill)

- Hàm imfill thực hiện một thao tác tô lấp trên ảnh nhị phân và ảnh đen trắng . Với ảnh nhị phân , hàm imfill thay đổi các pixel nền được kết nối (có giá trị 0) thành các pixel foreground (có giá trị 1) . Quá trình thay đổi này dừng lại khi chúng đạt đến biên của các đối tượng . Với các ảnh đen trắng , hàm imfill thay các giá trị cường độ của vùng đen (được bao quanh bởi các vùng trắng hơn) thành cùng mức cường độ với các pixel xung quanh . Thao tác này sẽ có ích khi loại bỏ các ***artifact*** không thích hợp khỏi ảnh

Chỉ định kiểu kết nối trong các thao tác tô lấp

- Với ảnh nhị phân và ảnh đen trắng , biên của thao tác tô lấp được quyết định bởi kiểu kết nối mà ta đã chỉ định
- Mối ảnh hưởng của kiểu kết nối có thể được minh hoạ với ma trận sau :

```
BW = [ 0   0   0   0   0   0   0   0;
       0   1   1   1   1   1   0   0;
       0   1   0   0   0   1   0   0;
       0   1   0   0   0   1   0   0;
       0   1   0   0   0   1   0   0;
       0   1   1   1   1   0   0   0;
       0   0   0   0   0   0   0   0;
       0   0   0   0   0   0   0   0];
```

- Nếu nền là kiểu 4 kết nối , ảnh nhị phân này chứa hai phần tử background riêng biệt (phần trong vòng và ngoài vòng) . Nếu nền là kiểu 8 kết nối , các pixel kết nối chéo và chỉ có một phần tử background .

Chỉ định điểm bắt đầu

- Với các ảnh nhị phân , ta có thể chỉ ra điểm bắt đầu của thao tác tô lấp bằng cách sử dụng hàm imfill theo kiểu tương tác , lựa chọn các pixel khởi đầu với con trỏ chuột . Chẳng hạn , nếu ta gọi hàm imfill , chỉ ra pixel BW(4,3) là pixel khởi đầu , hàm imfill chỉ tô bên trong vòng bởi vì , theo mặc định nền là kiểu 4 kết nối (4 connected)

```
imfill(BW,[4 3])

ans =
   0   0   0   0   0   0   0   0
   0   1   1   1   1   1   0   0
```

0 1 1 1 1 1 0 0
0 1 1 1 1 1 0 0
0 1 1 1 1 1 0 0
0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

- Nếu ta chỉ ra cùng một điểm bắt đầu nhưng sử dụng nền kiểu 8 kết nối , hàm imfill sẽ tô đầy toàn bộ ảnh

imfill(BW,[4 3],8)

ans =

1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1

Tô lấp các lỗ hổng trong ảnh

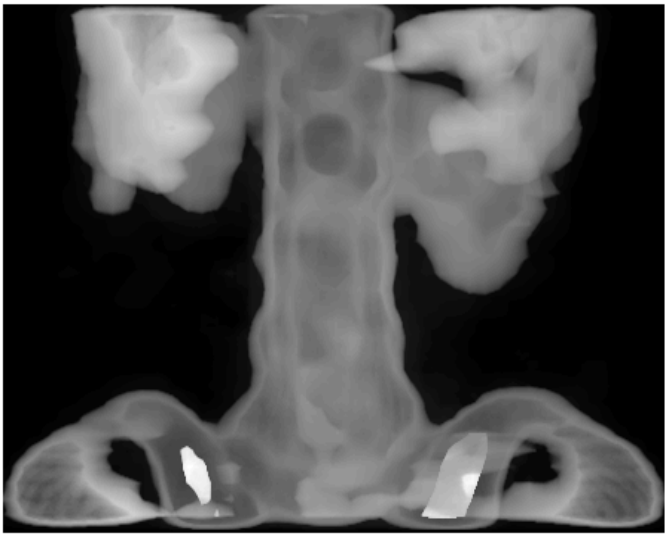
- Một thao tác tô lấp hay được sử dụng là tô lấp các lỗ hổng (holes)trong ảnh. Chẳng hạn , giả sử ta có một ảnh nhị phân hoặc đen trắng trong đó các đối tượng foreground đại diện cho các hình cầu . Trong ảnh , những đối tượng này xuất hiện như những đĩa . Trước khi thực hiện bất kì một thao tác xử lý nào , ta muốn đầu tiên tô đầy các lỗ hổng (xem ảnh) sử dụng hàm imfill

-Do thao tác tô lấp được sử dụng rất thường xuyên để tô lấp các lỗ , hàm imfill bao gồm các cú pháp đặc biệt để trợ giúp tác thao tác này trên cả ảnh nhị phân và ảnh đen trắng . Trong cú pháp này , ta chỉ cần chỉ ra tham số " holes " , ta không phải chỉ ra vị trí bắt đầu trong mỗi lỗ hổng . Để minh hoạ , ví dụ sau đây sẽ tô lấp các lỗ trong một ảnh đen trắng :

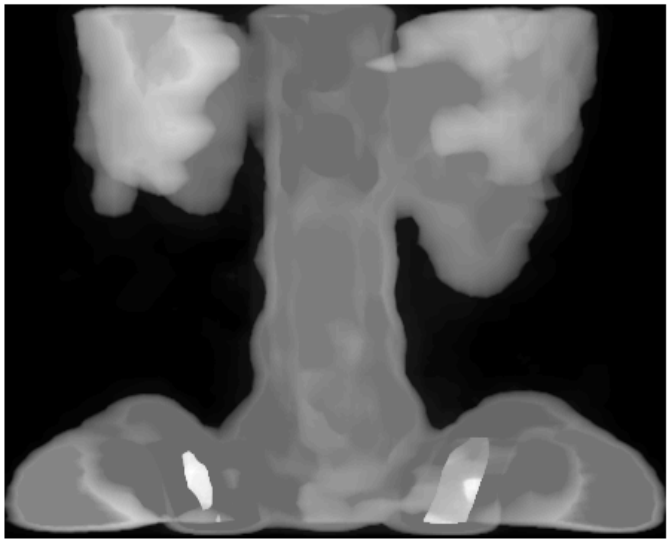
[X,map] = imread('spine.tif');

I = ind2gray(X,map);

```
Ifill = imfill(I,'holes');
imshow(I);figure, imshow(Ifill)
```



Original



After Filling Holes

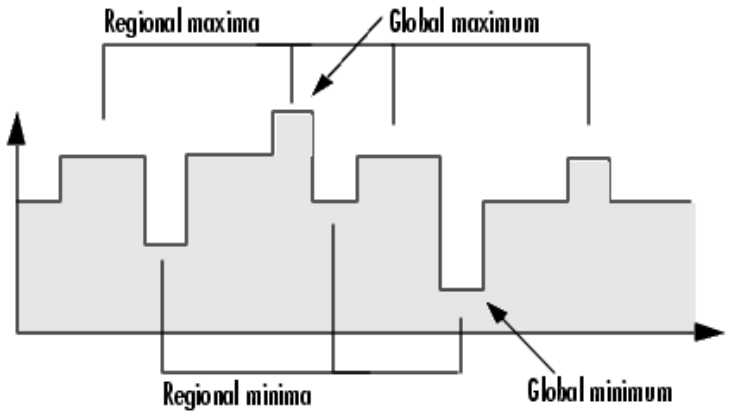
d- Tìm các đỉnh và đáy (Peak and Valley)

- Các ảnh đen trắng có thể được xem như trong không gian 3 chiều : trục x, y đại diện cho vị trí các pixel và trục z đại điện cho cường độ của mỗi pixel . Theo cách hiểu này , các giá trị cường độ đại diện cho độ cao giống như trong một bản đồ đo đạc địa hình . Những vùng có cường độ cao và thấp trong - ảnh tương ứng với đỉnh và đáy (thung lũng) , có thể là những đặc điểm hình thái quan trọng bởi vì chúng thường đánh dấu các đối tượng ảnh thích hợp

- Chẳng hạn , trong một ảnh với vài đối tượng có dạng hình cầu , các điểm có cường độ cao có thể đại diện cho đỉnh của các đối tượng . Sử dụng phép xử lý hình thái , các giá trị đỉnh này có thể được sử dụng để phân biệt các đối tượng trong một ảnh .

Các hàm cực đại và cực tiểu

- Một ảnh có thể có nhiều vùng lớn hoặc nhỏ nhưng chỉ một vùng cực đại và cực tiểu . Việc tính toán các đỉnh và đáy của ảnh có thể được sử dụng để tạo ra các ảnh ghi (marker images) - được sử dụng trong tái tạo hình thái .
- Hình sau đây minh hoạ khái niệm trong 1 chiều :



Tìm các vùng có cường độ cao hoặc thấp của ảnh

- Toolbox cung cấp các hàm mà ta có thể sử dụng để tìm các vùng có cường độ cao hoặc thấp trong một ảnh :

+ Hàm imregionalmax và imregionalmin phân biệt tất cả các cực tiểu hoặc cực đại vùng

+ Hàm imextendedmax và imextendedmin phân biệt tất cả các cực tiểu hoặc cực đại vùng mà lớn hơn hoặc nhỏ hơn một ngưỡng xác định

- Những hàm này chấp nhận một ảnh đen trắng như là tham số và trả về một ảnh nhị phân làm ảnh ra . Trong ảnh ra , cực tiểu hoặc cực đại vùng được thiết lập tới 1 , tất cả các pixel khác được thiết lập giá trị 0 . Chẳng hạn , ảnh đơn giản sau đây chứa hai cực đại vùng chính , các khối pixel chứa giá trị 13 và 18 và một vài cực đại nhỏ hơn được thiết lập giá trị 11

```
A = [10   10   10   10   10   10   10   10   10   10;
     10   13   13   13   10   10   11   10   11   10;
     10   13   13   13   10   10   10   11   10   10;
     10   13   13   13   10   10   11   10   11   10;
     10   10   10   10   10   10   10   10   10   10;
     10   11   10   10   10   18   18   18   10   10;
     10   10   11   10   10   18   18   18   10   10;
     10   10   11   10   10   18   18   18   10   10;
     10   11   10   11   10   10   10   10   10   10;
     10   10   10   10   10   10   11   10   10   10];
```

- Ảnh nhị phân được trả về từ hàm imregionalmax xác định tất cả các cực đại vùng

```
B =
     0     0     0     0     0     0     0     0     0     0
     0     1     1     1     0     0     1     0     1     0
     0     1     1     1     0     0     0     1     0     0
     0     1     1     1     0     0     1     0     1     0
     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     1     1     1     0     0
     0     0     0     1     0     1     1     1     0     0
     0     0     1     0     0     1     1     1     0     0
     0     1     0     1     0     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0
```

- Có thể ta chỉ muốn phân biệt các vùng của ảnh mà sự thay đổi cường độ là lớn nhất - ở đó sự chênh lệch giữa các pixel và các pixel lân cận của chúng ớn hơn một giá trị ngưỡng nào đó . Chẳng hạn , để tìm chỉ những cực đại vùng trong ảnh mẫu A mà lớn hơn ít nhất hai đơn vị so với các lân cận của chúng , sử dụng hàm imextendedmax

B = imextendedmax(A,2)

```
B =
     0     0     0     0     0     0     0     0     0     0
     0     1     1     1     0     0     0     0     0     0
     0     1     1     1     0     0     0     0     0     0
     0     1     1     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```