An Introduction to Digital Image Processing with MATLAB

Alasdair McAndrew

School of Computer Science and Mathematics

Victoria University of Technology

مدخل إلى معالجة الصور مع الماتلاب ـ ترجمة فهد أل قاسم fhdalqasem@yahoo.com

المحتوى:

الفصل الثاني معالجة النقطة Point Processing

مقدمة Introduction

إن عمليات معالجة الصور تقوم بتحويل قيم المستوى الرمادي لكل بكسل (عنصر الصورة)، وهذه العمليات على معالجة الصورة تصنف إلى ثلاثة تصنيفات اعتمادا على المعلومات المطلوبة لانجاز عملية التحويل transformation تلك، وهذه التصنيفات من الأكثر تعقيدا إلى الأبسط :

١) التحويلات Transforms:

يقوم التحويل بتمثيل قيم البكسلات مع بعضها البعض، ولكن بطريقة مكافئة، وهو يسمح لبعض الخوارزميات الأكثر كفاءة وقوة بالعمل على الصورة، وكما سنرى لاحقا، يتعامل التحويل مع الصورة كاملة كأنها وحدة واحدة كبيرة، والشكل التالي يوضح الفكرة باختصار.

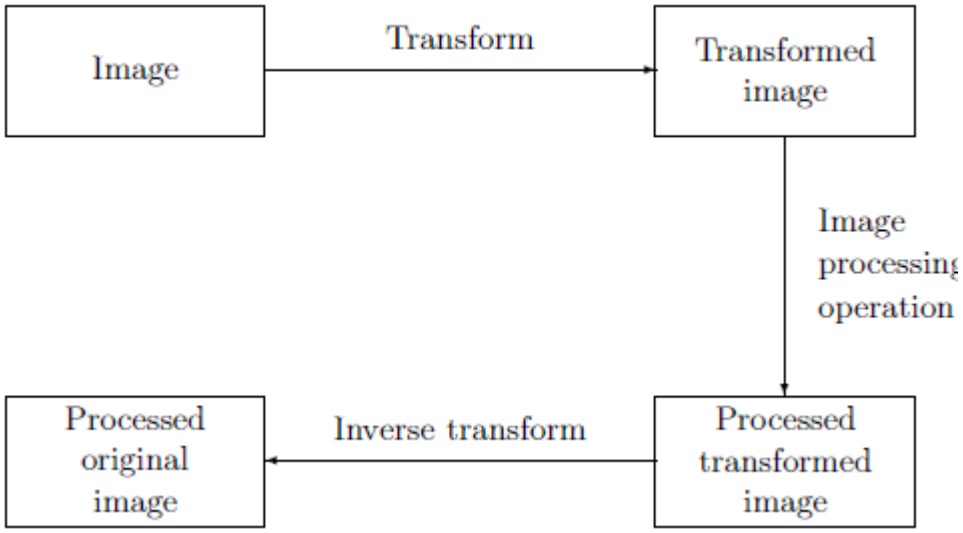


Figure 2.1: Schema for transform processing

٢) معالجة جوار النقطة Neighborhood processing :
من أجل تغيير مستوى اللون (الرمادي) للبكسل المحددة نحتاج فقط أن نعرف قيمة المستويات الرمادية لمجموعة صغير من النقاط المجاورة حول البكسل نفسها.

٣) العمليات على النقطة Point operations:
يتم تغيير قيمة اللون الرمادي دون أي معرفة عن البكسلات المحيطة بها، وهذه هي ابسط العمليات التي تقوم على الصورة، ولها استخدامات عديدة ومهمة، وخاصة عمليات التمهيد لما قبل المعالجة pre-processing، حيث يكون هناك احتياج لتعديل الصورة قبل تطبيق العملية الرئيسية عليها.

العمليات الرياضية على الصورة:
يمكننا تطبيق العمليات الرياضية على الصورة باعتبارها معادلة بسيطة:

$$y = f(x)$$

حيث ان الدالة f(x) تنطلق من المجال 255.....0 إلى هذا المجال نفسه، باعتباره المستويات الرمادية لقيم البكسلات، بنفس الترتيب في الصورة طبعا.
وكمثال على هذه الدالة يمكن تطبيق ابسط العمليات بإضافة أو طرح قيمة ثابتة C لكل بكسل:

$$y = x \pm C$$

وكمثال آخر على العمليات المطبقة على النقطة هناك عملية ضرب الثابت في كل قيم بكسلات الصورة:

$$y = Cx.$$

وبطبيعة الحال فالقيم الناتجة تحتاج إلى معالجة بسيطة حتى نضمن ان تكون مخرجات الدالة ضمن النطاق المحدد للدالة، الذي يمثل لنا مدى اللون الرمادي المفترض لهذا النوع من الصور. وهذا يعني ان قيم y التي تتجاوز النطاق يجب ان تكون ضمن الشرط التالي:

$$y \leftarrow \begin{cases} 255 & \text{if } y > 255, \\ 0 & \text{if } y < 0. \end{cases}$$

وكتوضيح على ذلك لنتفحص الرسم البياني التالي:

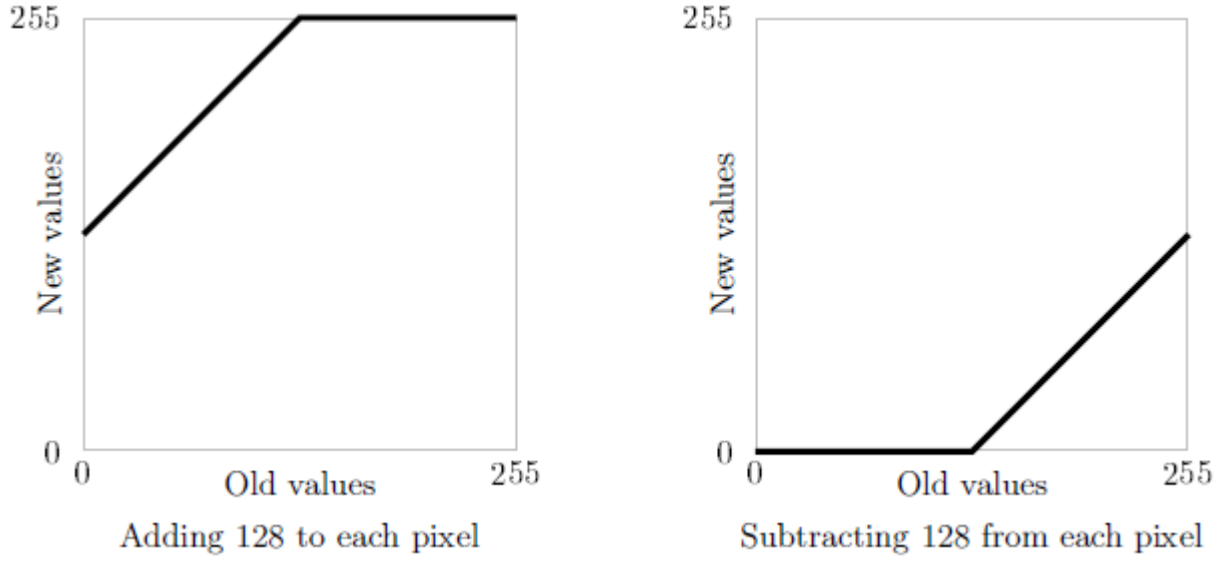


Figure 2.2: Adding and subtracting a constant

الرسم اعلاه يوضح العلاقة بين القيم الجديدة والقيم القديمة للبكسلات بعد عمليتي الجمع والطرح كلا على حده، وكيف ان تطبيق الشرط اعلاه يقوم بعملية تحويل القيم الأكبر من ٢٥٥ إلى ٢٥٥ والأقل من الصفر إلى صفر.
إن قيمة الثابت المضاف والمطروح في الرسمين البيانيين هي ١٢٨، فبإضافته يتم تحويل كل قيمة للبكسل من ١٢٧ واعلى إلى ٢٥٥، كما في الرسم الأول. وعند طرح قيمة الثابت فإننا نقوم بتحويل كل بكسل قيمته اقل من ١٢٨ أو تساويها إلى ٠.
وهذين الرسميين يوضحان لنا ايضا ان اضافة الثابت الموجب سوف يزيد من اضاء سطح الصورة lighten، أما الطرح فإنه يقوم فإنه يعكس الإضاءة وذلك بتغميق الصورة darken، ويمكن تطبيق ذلك على الماتلاب

على احدى الصورة القياسية من النوع الرمادي grayscale، والتي اسمها block.tif، الموضحة في الشكل ادناه، كما هو موضح في الكود بعدها.

The ’blocks’ image

```
>> b=imread('blocks.tif');
>> whos b
  Name      Size            Bytes  Class

  b       256x256           65536  uint8 array
```

لقد استخدمنا الأمر whos b لمعرفة نوع البيانات المخزنة في المصفوفة b، وهذا مهم من أجل معرفة كيفية التعامل معها، وواضح انها تستخدم بيانات من نوع uint8 وهذا النوع من البيانات يستخدم لتخزين البيانات فقط، فلا يمكننا اجراء العمليات عليه مباشرة، إذا انه سيصدر رسالة خطأ.

```
>>  b1=b+128
??? Error using ==> +
Function '+' not defined for variables of class 'uint8'.
```

ونستطيع تجاوز هذه المعضلة باحدى طريقتين، إما يتحويل المصفوفة b إلى مصفوفة من النوع double ، ومن ثم نطبق عملية الاضافة ثم نحول الناتج إلى uint8 من أجل عرض الناتج:

```
>>  b1=uint8(double(b)+128);
```

أو بطريقة أخرى أكثر أناقة باستخدام دالة الماتلاب التي تتعامل مع الصور بشكل أدق، بغض النظر عن نوع بيانات البسكلات فيها، وهي الدالة imadd .

```
>>  b1=imadd(b,128);
```

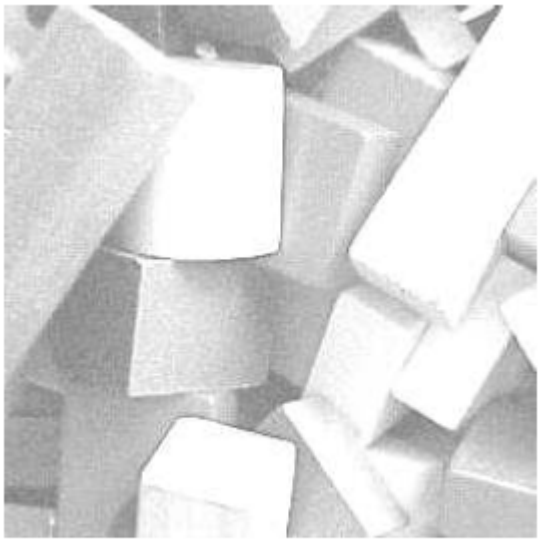
وبشكل مشابه نستخدم دالة الطرح imsubtract:

```
>> b2=imsubtract(b,128);
```

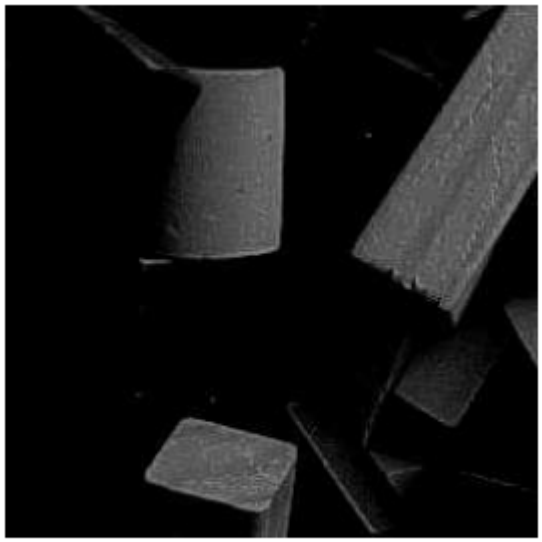
ثم نقوم بعرض الصورتين باستخدام الكود:

```
>> imshow(b1) , figure , imshow(b2)
```

وفيما يلي نرى الصورتين الناتجتين عن عمليتي الطرح والإضافة.



b1: Adding 128

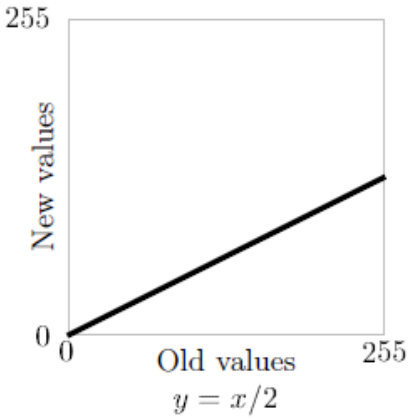


b2: Subtracting 128

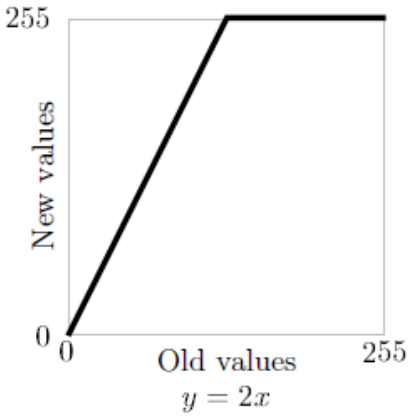
وبالإمكان ايضا استخدام الدالتين immultiply و imdivide لضرب الصورة أو قسمتها على الترتيب، كما هو موضح في المعادلات التالية:

$y = x/2$	`b3=immultiply(b,0.5);` or `b3=imdivide(b,2)`
$y = 2x$	`b4=immultiply(b,2);`
$y = x/2 + 128$	`b5=imadd(immultiply(b,0.5),128);` or `b5=imadd(imdivide(b,2),128);`

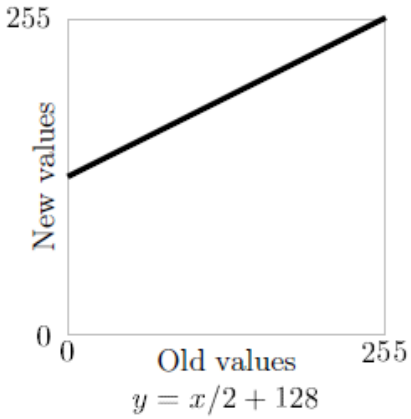
بعض المعادلات هذه زادت من تعتيم أو تغميق الصورة وبعضها زادت من السطوع، والرسوم البيانية التالية توضح عملية تغيير القيم بعد تطبيق العمليات لكل معادلة، وبعدها توضيح الصورة الثلاث الناتجة حسب الكود السابق:



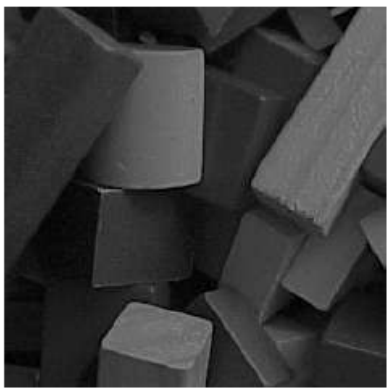
$y = x/2$



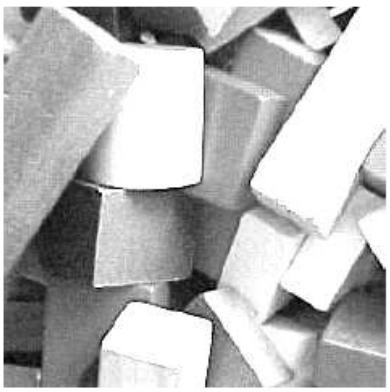
$y = 2x$



$y = x/2 + 128$



b3: $y = x/2$



b4: $y = 2x$

b5: $y = x/2 + 128$

نلاحظ أن هناك معادلات أظهرت نتائج مبالغ بها من حيث ضياع تفاصيل الصورة وذلك بسبب تكبير قيم معظم البكسلات او تصغيرها وهذا يظهر خصوصا في الصور b2 و b3 و b5.
التكبير المبالغ به يحول جميع القيم الكبيرة إلى ٢٥٥ (أبيض) وعكس ذلك يحول جميع القيم إلى ٠ (أسود).

الإتمام Complements:
إن متمم الصورة من النوع الرمادي grayscale هو الصورة السلبية لها negative كما هو مصطلح عليه في أفلام التصوير القديمة، فإذا كانت بيانات الصورة من النوع double فإن قيمها الرمادية سوف تنعكس بطرح مصفوفة الصورة من الواحد، هكذا:

```
>> 1-m
```

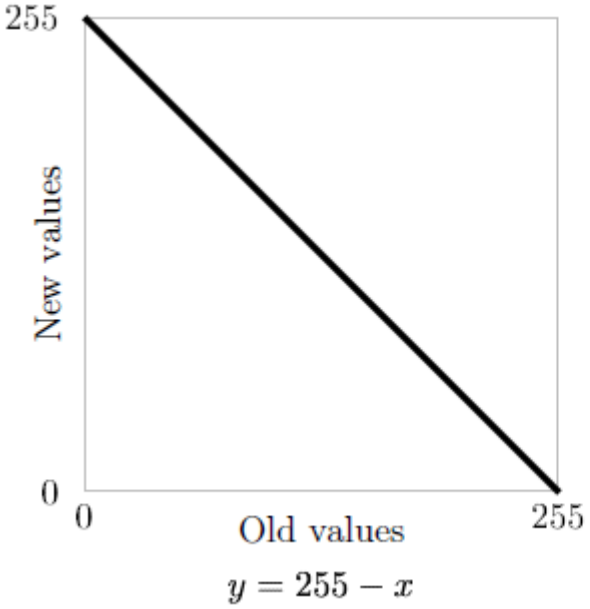
إما إذا كانت من النوع الثنائي فنستخدم مؤثر النفي ~ بالكود:

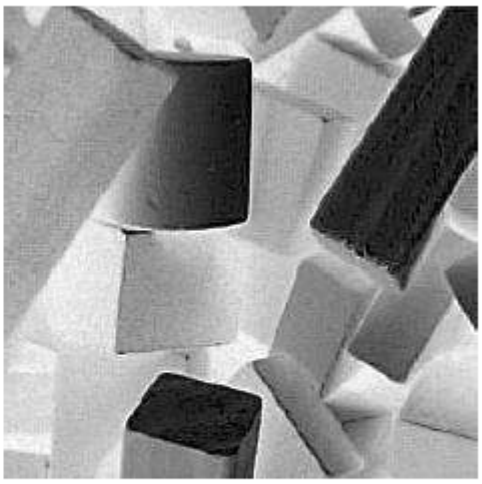
```
>> ~m
```

إما اذا كانت بيانات الصورة من النوع الصحيح uint8 فإننا نستخدم الدالة imcomplement، و هذا هو المستحسن دائما كما بالكود:

```
>> bc=imcomplement(b);
>> imshow(bc)
```

فيما يلي شكل يوضح الصورة blocks.tif والرسم البياني لتطبيق عملية الإتمام عليها، لاحظ عملية انعكاس البيانات في ارسم البياني حيث تحولت كل قيمة إلى القيمة المعاكسة لها.

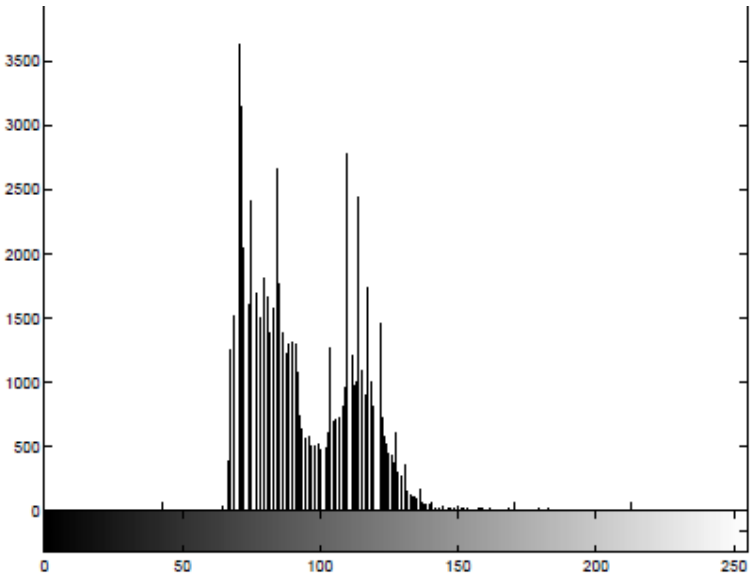




الهيستوجرام Histograms:
إن الهيستوجرام لأي صورة ملونة هو شكل بياني يوضح نسبة وجود الألوان فيها، بحيث يكون المحور الأفقي مبينا لنسبة هذه الأولون بشكل متدرج والمحور العمودي هو القيمة الإجمالية لظهور ذلك اللون في الصورة، وبالنسبة لصور اللون الرمادي فإننا نتحدث عن مستويات الرمادي بدلا من الألوان، فيكون الهيستوجرام هو تمثيل بياني لمستويات الرمادي المختلفة في الصورة. وعلى هذا فلكل لون (أو قيمة رمادية) قيمة عددية تمثل في الشكل البياني هي قيمة حدوثه في الصورة، وبمعالجة نسب الالوان والتدرجات يمكن عمل تأثيرات كثيرة على الصورة.
فمثلا في الصور الغامقة أو المعتمة dark image من صور اللون الرمادي فإن الهيستوجرام يكون متجمعا في النهاية الدنيا له حيث يقترب من اللون الاسود، والعكس بالعكس، اما الصور ذات التباين الجيد فإنها تكون موزعة بشكل افضل في بيان الهيستوجرام.
نستطيع إيجاد هيستوجرام الصورة في الماتلاب باستخدام الدالة imhist، كالتالي:

```
>> p=imread('pout.tif');
>> imshow(p),figure,imhist(p);
```

والنتائج مبينة في الشكل التالي :



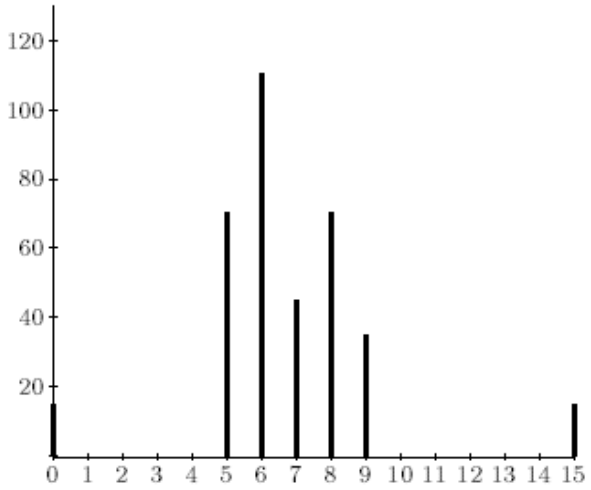
لاحظ أن القيم الرمادية قد تجمعت في مركز الهيستوجرام، وهذا يعني ان التباين contrasted الخاص بها سيء، وفيما يلي سنرى أن تحسين انتشار أعمدة الهيستوجرام ينعكس تلقائيا على تباين الصورة مما يساعد على تحسينها.

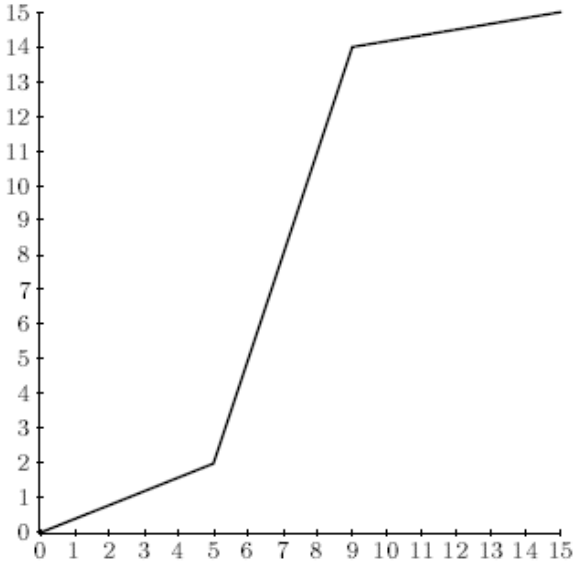
توسيع الهيستوجرام (**توسيع التباين**) (Histogram stretching (Contrast stretching:

لنفترض أن لدينا صورة لها الهيستوجرام الموضح في الشكل أدناه، حيث ان قيم المستوى الرمادي الموجودة فيه ترتبط مع القيم n_i المقابلة لها حسب الجدول التالي.

Grey level i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_i	15	0	0	0	0	70	110	45	70	35	0	0	0	0	0	15

اما الهيستوجرام ودالة التوسيع فباالشكل:





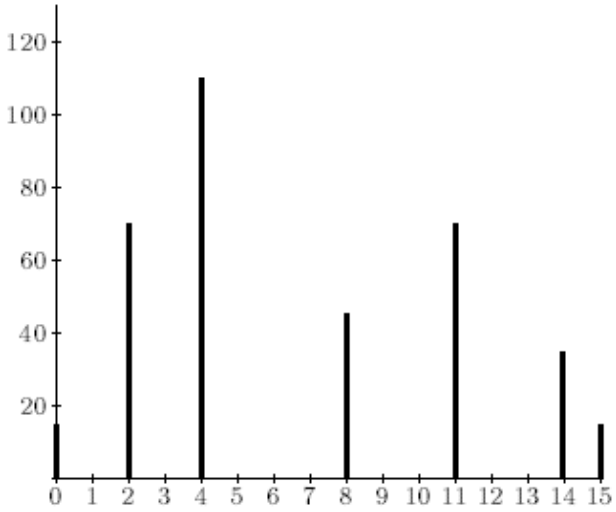
عملية التوسيع تتم بتطبيق ما يسمى بدالة القطعة الخطية piecewise linear function الموضحة على يمين الشكل السابق، حيث نستطيع عن طريقها توسيع المستوى الرمادي من المدى الموضح في الهيستوجرام وهو من ٥- ٩ ، إلى مدى أوسع وهو ٢-١٤ ، وهذه المعادلة هي:

$$j = \frac{14-2}{9-5}(i-5)+2$$

حيث أن i هي القيمة الحالية لمستوى الرمادي و j هي القيمة الجديدة بعد التحويل. وقد نتجت لنا قيما جديدة نستطيع الآن تطبيقها على الهيستوجرام الجديد، مع ملاحظة ان القيم الأخرى خارج المدى المحول إما ان تترك كما هي (كما في حالتنا هذه) او تدمج في عملية التحويل:

i	5	6	7	8	9
j	2	5	8	11	14

أما الهيستوجرام الجديد فسيكون بالقيم الجديدة التي يجب أن تكون تباينها أفضل من الأولى:

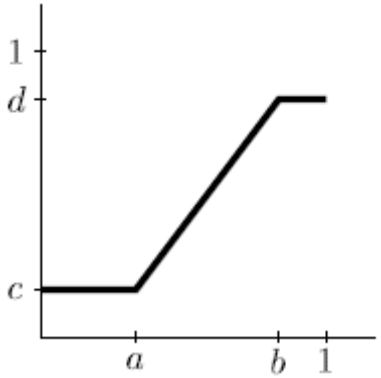


استخدام دالة الضبط (imadjust) Use of :

من أجل انجاز عملية توسيع الهيستوجرام في الماتلاب بسهولة، نستخدم الدالة imadjust التي لها ابسط استخداماتها هي حسب الشكل العام:

```
imadjust(im,[a,b],[c,d])
```

حيث ان النقطتين a و b هما النقطتان التي يتم تحولهما إلى c و d بالترتيب ووفق الشكل:



كما أن هذه الدالة تعمل بشكل جيد على الصور من التي نوع بياناتها double أو uint8 أو uint16 كما أن قيم المتغيرات a,b,c,d يجب ان تكون بين الصفر والواحد، بحيث تقوم الدالة بتحويلها حسب الحاجة.
لكن هذه الدالة تختلف قليلا عن ما ذكرنا اعلاه حيث انها تحول جميع القيم أقل من a إلى c، وجميع القيم اكبر من b إلى d. لهذا نقوم عادة عند كتابة الكود في الماتلاب إما بذكر المدى كاملا [0,1]، أو بترك القوسين بدون [] ، فالسطر التالي مثلا لا ينفذ شيء:

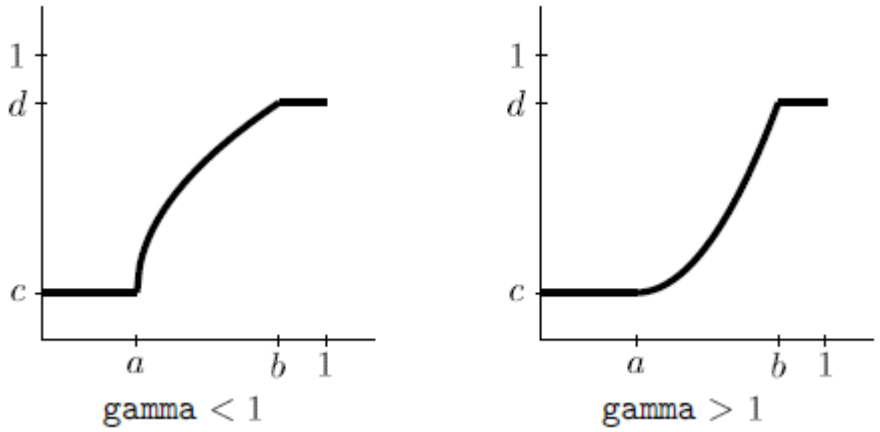
```
>> imadjust(im,[],[]);
```

أما السطر التالي:

```
>> imadjust(im,[],[1,0])
```

فيقوم بعكس مستويات الرمادي للصورة بحيث تتحول إلى الصورة السلبية photographic negative التي نفذنا عليها سابقا عملية التحويل إلى المتمم .
ولأداء أفضل لهذه الدالة نستخدم البارامتر أو الوسيط الرابع الذي لم يستخدم في المثالين أعلاه، حيث أن قيمته التلقائية default هي الواحد، حيث ان هذا المؤتر يتحكم بشكل مستقيم التحويل كما في الشكل البياني السابق الذي يكون كذلك عند تكون قيمة الوسيط (الذي يسمى جاما gamma) مساوية للواحد، وهو التحويل الخطي.

وكلما كانت قيمة الوسيط جاما اكبر من واحد كان التحويل أكثر تقعرا للأسفل، وكلما كانت قيمته أقل من الواحد كان التحويل أكثر تقعرا للأعلى، كما هو موضح بالشكل التالي:



وقيمة جاما في الدالة imadjust هي قيمة الأس γ في الشكل العام للدالة الموضح بالمعادلة:

$$y = \left(\frac{x-a}{b-a}\right)^{\gamma}(d-c)+c.$$

لاحظ طبعا أن هناك اختلاف طفيفا بين هذه الدالة وبين التطبيق المستخدم في التحويل أعلاه.
الكود التالي يوضح مثالا باستخدام جاما فقط:

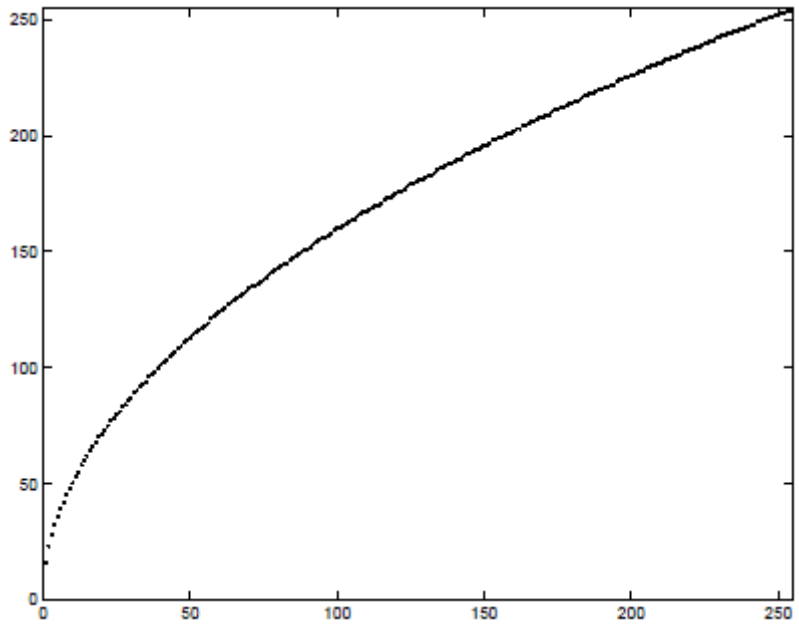
```
<<t=imread('tire.tif');
<<th=imadjust(t,[],[],0.5);
<<imshow(t),figure,imshow(th)
```

كما أنه من الممكن رؤية الشكل البياني للعلاقة بين القيم الجديدة والقيم القديمة لمستويات الرمادي باستخدام الدالة plot التي ترسم الرسم البياني للعلاقة بين القيمتين .

```
>> plot(t,th,'.'),axis tight
```

الشكل التالي يوضح الصورة tire.tif المستخدمة في المثال السابق والتحسين الطارئ عليها عندما gamma=0.5:

أما دالة الـ plot في الكود أعلاه فسوف تظهر بالشكل البياني:



دالة التوسيع الخطي حسب القطعة A piecewise linear stretching function

وعودة إلى دالة التوسيع الخطي حسب القطعة التي تبدو في بعض الحالات انسب من دالة الضبط imadjust، فإننا سنوضح فيما يلي كيفية كتابة دالة في الماتلاب لتنفيذ فكرتها، ثم طريقة استدعاء الدالة. سنستخدم الدالة find المحجوزة التي تتعامل مع المصفوفات، وتستخدم لإيجاد قيم البكسلات في الصورة، وتعيد دالة find موقع العنصر المطلوب وتتعامل المصفوفة كأنها متجه، هذه البكسلات التي تكون بين a_i و a_{i+1} ، حيث ان الخط بين الاحداثي (a_i,b_i) و (a_{i+1}, b_{i+1}) يعتمد على المعادلة:

$$y = \frac{b_{i+1} - b_i}{a_{i+1} - a_i}(x - a_i) + b_i$$

هذه المعادلة يمكن أن تكتب على في الكود بالشكل:

```
y= (x-a(i))*(b(i+1)-b(i))/(a(i+1)-a(i))+b(i);
```

وسيتم تنفيذ المعادلة هذه باستخدام كود الماتلاب الذي يستخدم الدالة find ضمن دالة من تعريفنا اسمها histpwl واهم اسطر هذه الدالة هي الجزء الذي يبحث عن القيم ثم يستدعي دالة القطعة الخطية:

```
pix=find(im >= a(i) & im < a(i+1));
out(pix)=(im(pix)-a(i))*(b(i+1)-b(i))/(a(i+1)-a(i))+b(i);
```

حيث ان im هي الصورة المدخلة، و out هي الصورة الناتجة عن عملية التحويل transformation operation.

وسيتم استدعاء الدالة histpwl ضمن كود الماتلاب التالي:

```
>> th=histpwl(t,[0 .25 .5 .75 1],[0 .75 .25 .5 1]);
>> imshow(th)
>> figure,plot(t,th,'.'),axis tight
```

بافتراض ان th هي الصورة tire.tif المستخدمة سابقا، وفيما يلي كود الدالة histpwl :

```
function out = histpwl(im,a,b)
%
% HISTPWL(IM,A,B) applies a piecewise linear transformation to the pixel values
% of image IM, where A and B are vectors containing the x and y coordinates
% of the ends of the line segments. IM can be of type UINT8 or DOUBLE,
% and the values in A and B must be between 0 and 1.
%
% For example:
%
```

```
%histpwl(x,[0,1],[1,0])
%
%simply inverts the pixel values.
%
classChanged = 0;
if ~isa(im, 'double'),
  classChanged = 1;
  im = im2double(im);
end
if length(a) ~= length (b)
  error('Vectors A and B must be of equal size');
end
N=length(a);
out=zeros(size(im));
for i=1:N-1
  pix=find(im>=a(i) & im<a(i+1));
  out(pix)=(im(pix)-a(i))*(b(i+1)-b(i))/(a(i+1)-a(i))+b(i);
end
pix=find(im==a(N));
out(pix)=b(N);
if classChanged==1
  out = uint8(255*out);
end
```

للتذكير فقط: الرمز % يستخدم لعمل التعليقات comments بين المبرمجين، كما أن اهميته في الماتلاب تكمن في أن وجوده في بداية الدالة يفيد في تخزين تعليمات الـ help الخاصة بالدالة.

مساواة الهيستوجرام Histogram equalization

مشكلة كل واحدة من الطرق السابقة في توسيع الهيستوجرام أنها تحتاج إدخالات المستخدم المخصصة لكل حالة على حده، في حين انه كطريقة أفضل يتم استخدام مساواة او تسوية الهيستوجرام، كأجراء آلي تماما، والفكره هنا هي تغيير الهيستوجرام بشكل موحد بحيث يصبح على جزء من أعمدة الهيستوجرام بنفس الطول، او بكلمة أخرى تكون مستويات الرمادي في الصورة كلها بنفس التكرار. من الناحية العملية هذا ممكن عموما، مع اننا قد نجد ان نتائج ذلك جيدة جدا.

لنفترض ان لدينا صورة فيها مستويات رمادي مختلفة عددها L من ٠،١،٢، حتى L-1، وان ذلك التدرج الرمادي ليكن i قد ظهر n_i مرة في هذه الصورة، وأن العدد الإجمالي للبكسلات في الصورة هو n، وهذا يعني أن:

$$n_0 + n_1 + n_2 + \cdots + n_{L-1} = n.$$

وأنه لكي نحول المستويات الرمادية للحصول على أفضل تباين للصورة فإننا نغير المستوى الرمادي i إلى:

$$\left(\frac{n_0 + n_1 + \cdots + n_i}{n}\right)(L-1).$$

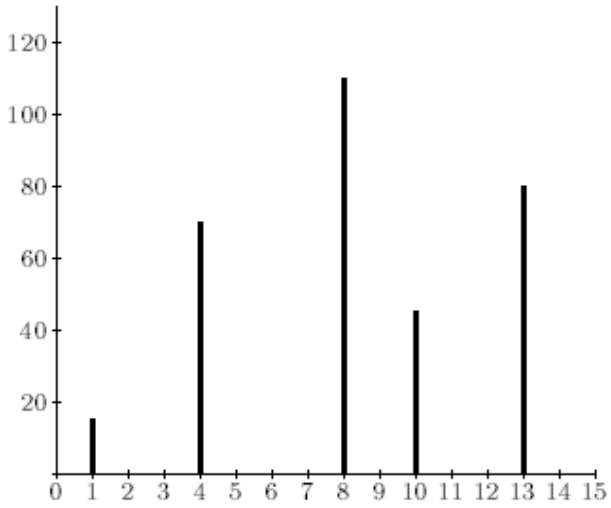
كما ان العدد الناتج سوف تقريبه (تدويره rounded) إلى أقرب عدد صحيح integer.

ما سبق يعتبر تعريفا رياضيا، ولتوضيحه أكثر لننظر للمثال التالي:

ليكن لدينا صورة من نوع تدرج الرمادي grayscale طول قيمة البكسل فيها 4-bit، الهيستوجرام الخاص بها موضح في المثال ادناه، وذلك وفق القيم الموضحة في الجدول التالي:

Grey level i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_i	15	0	0	0	0	0	0	0	0	70	110	45	80	40	0	0

أما الهيستوجرام فهو:



نلاحظ أن إجمالي ظهور البكسلات في هذا المثال هو n=360.

Grey level i	n_i	Σn_i	$(1/24)\Sigma n_i$	Rounded value
0	15	15	0.63	1
1	0	15	0.63	1
2	0	15	0.63	1
3	0	15	0.63	1
4	0	15	0.63	1
5	0	15	0.63	1
6	0	15	0.63	1
7	0	15	0.63	1
8	0	15	0.63	1
9	70	85	3.65	4
10	110	195	8.13	8
11	45	240	10	10
12	80	320	13.33	13
13	40	360	15	15
14	0	360	15	15
15	0	360	15	15

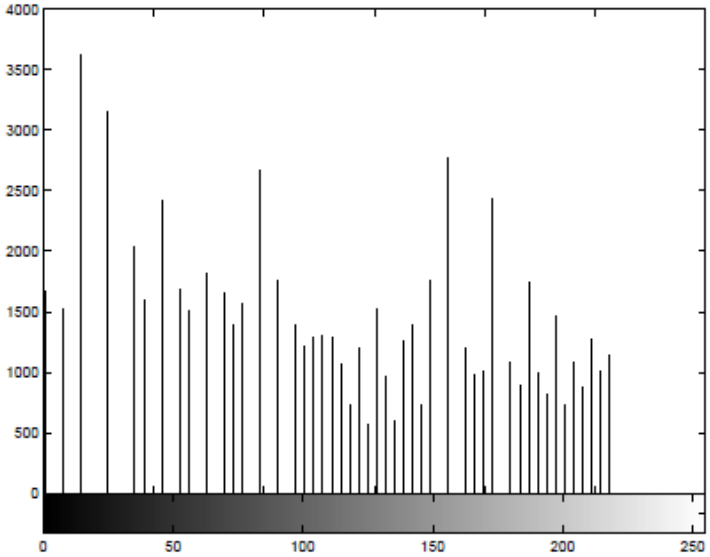
وفي الجدول السابق النتيجة الحاصلة من عملية مساواة الهيستوجرام بتطبيق الصيغة في التعريف.
كل هذا يمكن تلخيصه بالجدول النهائي التالي الذي يبين القيم الأولى والقيم النهائية:

Original grey level i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Final grey level j	1	1	1	1	1	1	1	1	1	4	8	10	13	15	15	15

كل هذا يمكن تطبيقه في الماتلاب باستخدام الدالة histeq كما في الأسطر التالية:

```
>> p=imread('pout.tif');
>> ph=histeq(p);
>> imshow(ph),figure,imhist(ph),axis tight
```

وبالتأكيد فإن نتيجة مساواة الهيستوجرام هنا هي افضل النتائج كما يوضح الشكل التالي، لكننا نؤكد ان الطرق المختلفة تظهر نتائج مختلفة على الحالات المختلفة مما يعني انه قد نحصل على نتائج افضل في بعض الحالات الخاصة رغم اعتبارنا ان هذه الطريقة هي الطريقة الاكثر تعميما ولذلك فهي الأسهل تطبيقا.



الجدول المرجعي Lookup tables

يمكن انجاز معالجة النقطة بشكل فعال باستخدام جدول النظر (الجدول المرجعي)، والذي يعرف اختصارا بـ LUT، فللتعامل مع صورة نوع بياناتها صحيحة uint8 ، يكون الجدول المستخدم مصفوفة احادية طولها ٢٥٥ قيمة، كل واحدة من هذه القيم قيمة صحيحة من المدى ٠.....٢٥٥، وبهذا تكون العملية المطبقة على الصورة متمثلة بتبديل كل قيمة بكسل p بالقيمة المناظرة في الجدول t_p.

وكمثال على ذلك ليكن لدينا جدول الـLUT التالي الذي يناظر تطبيق عملية القسمة على اثنين:

Index:	0	1	2	3	4	5	...	250	251	252	253	254	255
LUT:	0	0	1	1	2	2	...	125	125	126	126	127	127

وهذا يعني أن القيمة ٤ مثلا سوف تستبدل بالقيمة ٢، والقيمة ٢٥٣ سوف تستبدل بـ ١٢٦، ونطبق ذلك في الماتلاب باعتبار أن الجدول المرجعي هو المصفوفة T، وأن الصورة هي im فإننا الجدول المرجعي يطبق ببساطة بالأمر: T(im)، ولنقم على سبيل المثال بتطبيق الجدول اعلاه فإننا ننشئ الجدول بالأمر:

```
>> T=uint8(floor(0:255)/2);
```

والتطبيق سيكون بالأمر:

```
>> b2=T(b);
```

فتكون الصورة b2 من النوع unit8 ويمكننا عندئذ الاطلاع على هذه الصورة بالأمر imshow.

تمارین

Exercises

Image Arithmetic

1. Describe lookup tables for

   (a) multiplication by 2,

   (b) image complements

2. Enter the following command on the blocks image `b`:

```
>> b2=imdivide(b,64);
>> bb2=immultiply(b2,64);
>> imshow(bb2)
```

   Comment on the result. Why is the result not equivalent to the original image?

3. Replace the value 64 in the previous question with 32, and 16.

Histograms

4. Write informal code to calculate a histogram $h[f]$ of the grey values of an image $f[row][col]$.

5. The following table gives the number of pixels at each of the grey levels 0–7 in an image with those grey values only:

0	1	2	3	4	5	6	7
3244	3899	4559	2573	1428	530	101	50

   Draw the histogram corresponding to these grey levels, and then perform a histogram equalization and draw the resulting histogram.

6. The following tables give the number of pixels at each of the grey levels 0–15 in an image with those grey values only. In each case draw the histogram corresponding to these grey levels, and then perform a histogram equalization and draw the resulting histogram.

   (a)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	40	60	75	80	75	65	55	50	45	40	35	30	25	20	30

   (b)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	40	80	45	110	70	0	0	0	0	0	0	0	0	15

7. The following small image has grey values in the range 0 to 19. Compute the grey level histogram and the mapping that will equalize this histogram. Produce an 8×8 grid containing the grey values for the new histogram-equalized image.

```
12   6   5  13  14  14  16  15
11  10   8   5   8  11  14  14
```

```
 9   8   3   4   7  12  18  19
10   7   4   2  10  12  13  17
16   9  13  13  16  19  19  17
12  10  14  15  18  18  16  14
11   8  10  12  14  13  14  15
 8   6   3   7   9  11  12  12
```

8. Is the histogram equalization operation idempotent? That is, is performing histogram equalization *twice* the same as doing it just once?

9. Apply histogram equalization to the indices of the image `emu.tif`.

10. Create a dark image with

```
>> c=imread('cameraman.tif');
>> [x,map]=gray2ind(c);
```

The matrix `x`, when viewed, will appear as a very dark version of the cameraman image. Apply histogram equalization to it, and compare the result with the original image.

11. Using `p` and `ph` from section 2.3.2, enter the command

An Introduction to Digital Image Processing with MATLAB

Alasdair McAndrew

School of Computer Science and Mathematics

Victoria University of Technology

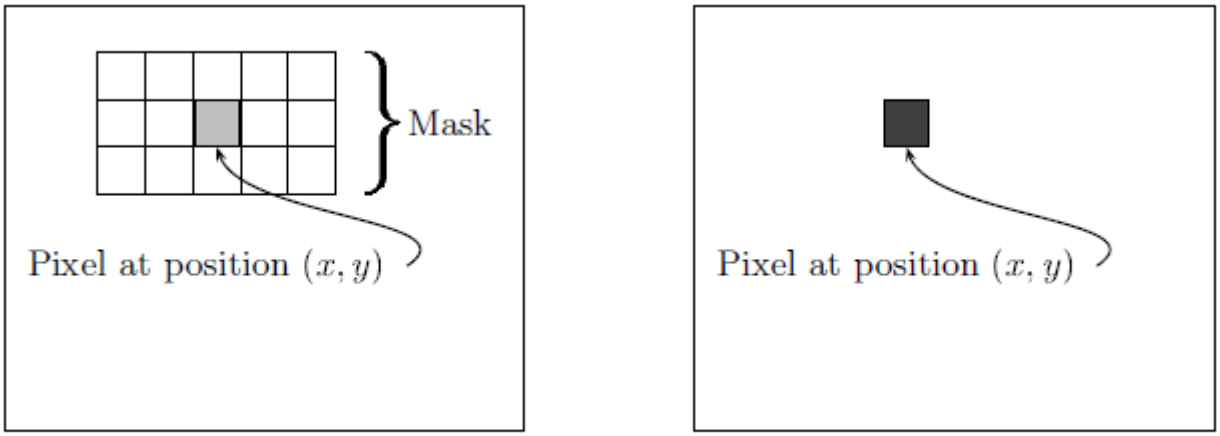
مدخل إلى معالجة الصور مع الماتلاب ـ ترجمة فهد أل قاسم fhdalqasem@yahoo.com

الفصل الثالث معالجة جوار النقطة Neighborhood Processing

كما شاهدنا في الفصل الثاني أن الصورة يمكن تعديلها عن طريق تطبيق معادلة محددة على كل قيم عناصر الصورة البكسلات، ويمكننا اعتبار معالجة جوار الصورة عبارة عن تمديد لذات الموضوع، حيث ان الدالة سوف تطبق لكل قيم البكسلات المجاورة لكل النقاط!.

الفكرة هنا هي في تحريك قناع mask على الصورة، هذا القناع يمكن ان يكون مستطيلا (عادة ما يكون بأطوال فردية) أو أي شكل آخر، وكما هو موضح في الشكل أدناه، سوف نقوم بإنشاء صورة جديد من

صورة سابقة بحساب قيم جديدة لبكسلاتها وفق البكسلات المحسوبة في القناع، نسمي كلا من القناع والدالة التي تغير القيم بالفلتر filter.



Original image Image after filtering

فإذا كانت الدالة المستخدمة لتغيير القيم تحت القناع دالة خطية linear function ، فإننا الفلتر يسمى فلترا خطيا، ويمكننا تطبيق فلترا خطيا عن طريق ضرب كل عناصر القناع بالعناصر المقابلة في الجوار، ثم إضافة جميع نتائج الضرب.

لنفترض أن لدينا قناعا أبعاده ٣×٥ كما هو موضح بالشكل السابق، وأن قيم القناع تعطى عن طريق:

$m(-1,-2)$	$m(-1,-1)$	$m(-1,0)$	$m(-1,1)$	$m(-1,2)$
$m(0,-2)$	$m(0,-1)$	$m(0,0)$	$m(0,1)$	$m(0,2)$
$m(1,-2)$	$m(1,-1)$	$m(1,0)$	$m(1,1)$	$m(1,2)$

وأن قيم البكسلات المناظرة هي:

$p(i-1,j-2)$	$p(i-1,j-1)$	$p(i-1,j)$	$p(i-1,j+1)$	$p(i-1,j+2)$
$p(i,j-2)$	$p(i,j-1)$	$p(i,j)$	$p(i,j+1)$	$p(i,j+2)$
$p(i+1,j-2)$	$p(i+1,j-1)$	$p(i+1,j)$	$p(i+1,j+1)$	$p(i+1,j+2)$

ولنقم الآن بالضرب ثم الجمع كما بالمعادلة:

$$\sum_{s=-1}^{1}\sum_{t=-2}^{2} m(s,t)p(i+s,j+t).$$

المخطط التالي يوضح عملية تنفيذ الفلترة الحيزية (الفراغية) spatial filtering، وهو يحتاج إلى ثلاث خطوات:

١. وضع القناع فوق البكسل الحالي.
٢. مطابقة كل نتائج الفلتر المضروبة مع العناصر المناظرة في الجوار.
٣. جمع كل النتائج.

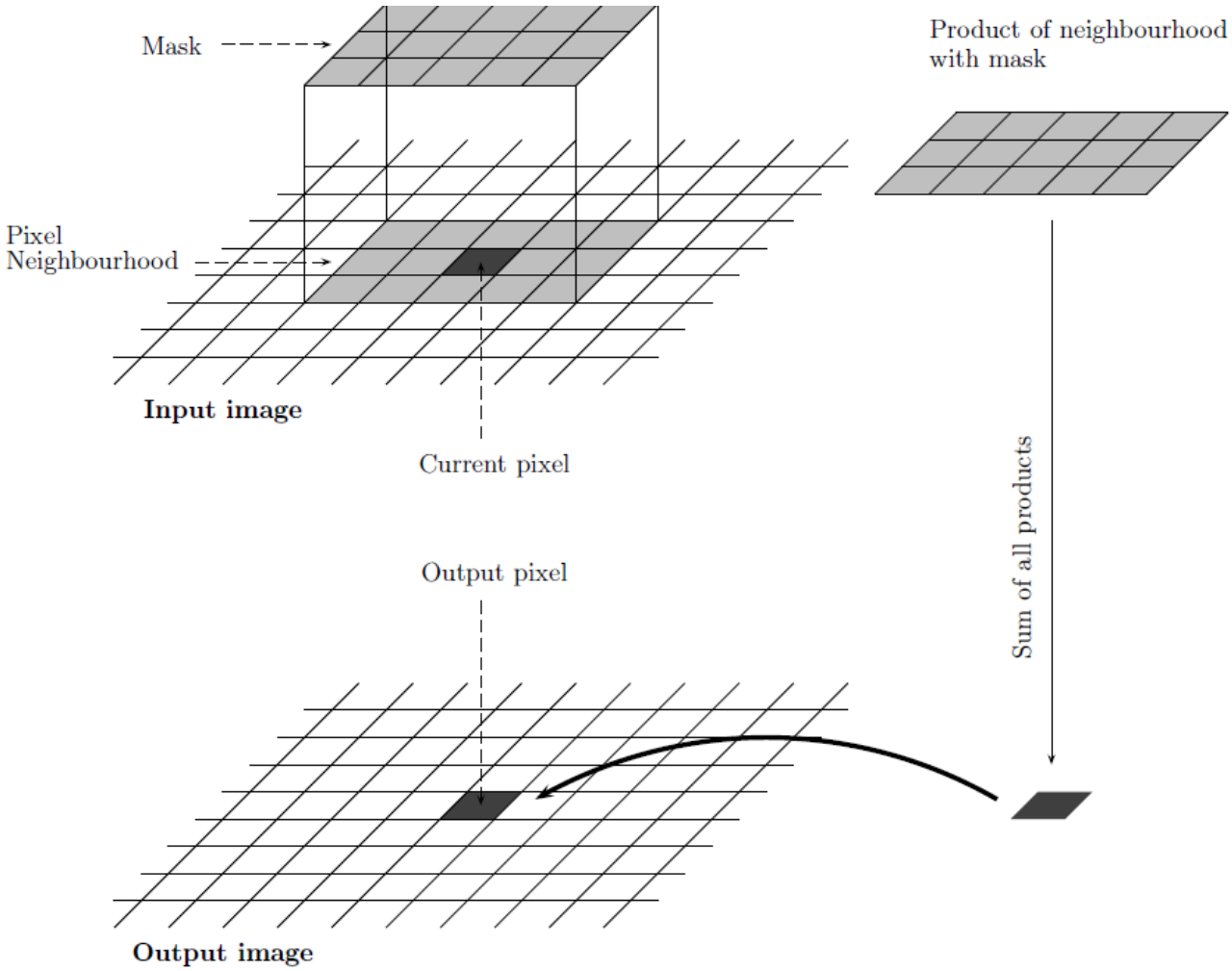
ويجب تكرار ذلك على كل البكسلات في الصورة.

وهناك عملية شبيهة بالفلترة الحيزية تسمى الالتفاف الحيزي spatial convolution ، ويتم انجاز الالتفاف الحيزي بشكل مشابه للفلترة الحيزية عدا انه يجب عمل تدوير للفلتر ١٨٠ درجة قبل عمليتي الضرب والإضافة.

وباستخدام الرمزين m(i,j) وp(i,j) كما سبق، ستكون مخرجات الالتفاف الحيزي على قناع ٣×٥ المطبق على بكسل واحد هي:

$$\sum_{s=-1}^{1}\sum_{t=-2}^{2} m(-s,-t)p(i+s,j+t).$$

لاحظ الإشارة السالبة في عناصر m والتي يمكن استبدالها بإجراء عمليات الطرح في عناصر p.



مثال تخطيطي يوضح فكرة الفلترة الحيزية

وكمثال على الفلاتر الشهيرة الفلتر ' average ' هذا الفلتر الذي اصغر أحجامه 3×3 ويمكن ان يكون اكبر من ذلك بأي حجم، ويكون غالبا مربعا، يقوم هذا الفلتر بأخذ متوسط كل القيم التسعة التي في الجوار المطبق عليه، ويقوم بعد ذلك القيم المتوسطة تلك هي القيمة الجديدة للنقطة المركزية في الجوار.

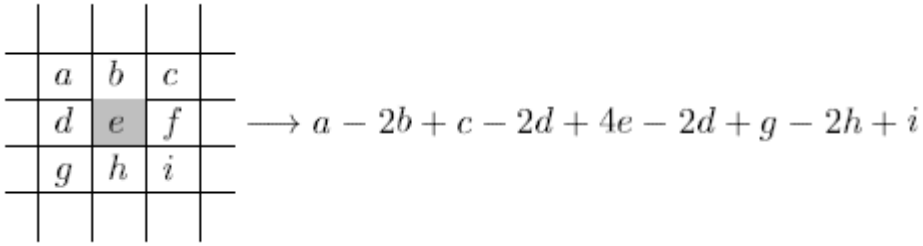
	a	b	c
	d	e	f
	g	h	i

$$\longrightarrow \frac{1}{9}(a+b+c+d+e+f+g+h+i)$$

وكمثال آخر الفلتر الشهير التالي:

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

يمكن تلخيص تطبيق هذا الفلتر بالشكل التالي:



مشكلة أطراف الصورة Edges of the image:
تظهر دائما مشكلة عند تطبيق الفلتر حيث نتساءل هل سيتم تطبيق الفلتر على النقاط التي في العمودين الأول والأخير من الصورة وكذلك الصفين الأول والأخير (هذا بالنسبة لفلتر بعده ٣×٣)، فجوار نقاط هذه الصفوف والأعمدة يقع جزء منه خارج مصفوفة الصورة، وهنالك في الحقيقة ثلاث تقنيات لحل هذه المشكلة:

١. تجاهل الأطراف Ignore the edges
ويرمز لهذا الأسلوب كما سيأتي بـ 'valid' حيث يتم تجاهل الصفوف والأعمدة على الأطراف وينتج عن هذا الأسلوب مصفوفة حجمها اصغر من المصفوفة الأصلية.
٢. الحشو بالأصفار Padding with zeros: وهذا التقنية تعتمد على إضافة أصفار خارج إطار المصفوفة مما يكبر من حجمها ويجعل أطرافها كأنها نقاط داخلية بصناعة جوار صفري لكل عمود وصفر لا جوار له، وبعد انجاز الفلتر لدينا خيارين:
   a. الخيار 'same' وفيه يتم إلغاء الحشو بعد انجاز عملية الفلترة بحيث يكون الناتج مصفوفة بنفس حجم المصفوفة الأصلية.
   b. الخيار 'full' وفيه يتم الإبقاء على الصفوف والأعمدة الزائدة بقيمها الجديدة بعد الفلتر مما يسبب الحصول على مصفوفة مفلترة اكبر من الاصلية.

الفلترة في الماتلاب Filtering in Matlab
يتم استخدام الدالة filter2 ويمرر إليها ثلاثة وسائط parameters

```
Resulted_image=filter2(filter,image,shape)
```

المدخل الأول filter هو مصفوفة الفلتر والثاني image هو الصورة الاصلية والثالث sahpe هو إحدى ثلاث كلمات محجوزة لتحديد نوع المصفوفة الناتجة حسب طريقة الحشو إما same وهي السلسلة النصية الافتراضية default أو valid أو full، حسب الموضح أعلاه.
مثال : تطبيق الفلتر average ثلاثي البعد على المصفوفة x كما يلي:

```
>> x=uint8(10*magic(5));
```

هذه المصفوفة عبارة عن صورة افتراضية تم توليدها عبر الدالة magic.

```
>> a=ones(3,3)/9
```

والمصفوفة a هي الفلتر المطلوب تطبيقه.

```
>> filter2(a,x,'same');
```

ولنعيد تطبيق نفس السطر مع تغيير السلسلة النصية shape مره valid ومره full ثم لاحظ الفارق.
الدالة fspecial **لتوليد الفلاتر القياسية**
هناك مجموعة كبيرة من الفلاتر القياسية نستطيع توليدها عبر الدالة fspecial حسب اسمائها المخزونة مسبقا.
فالسطر التالي:

```
>> f= fspecial('average',[5,7]);
```

يولد فلتر المتوسط الذي طبقناه سابقا ولكن بأبعاد مختلفة، وإذا لم يتم تحديد أبعاد فإن البعد القياسي هو ٣×٣:

```
>> aa= fspecial('average');
```

المصفوفة aa هي نفسها المصفوفة a التي تم حجزها سابقا.
وهناك فلاتر اخرى مثلا فلتر لابلاسيان:

```
>> f=fspecial('laplacian')
```
والفلتر جاوسيان على لابلاسيان ويختصر log:
```
>> f1=fspecial('log')
```
الفلاتر عالية ومنخفضة العبور Frequencies; low and high pass filters
توجد في كل صورة مجموعة من المكونات، ولو نظرنا إلى المصفوفة نفسها لوجدنا ان هذه المكونات تنعكس على قيم المصفوفة وإجمالا نستطيع ان نقول ان لدينا نوعين من المكونات هي المكونات عالية التردد (التكرار)، والمكونات منخفضة التكرار أو التردد.
المكونات عالية التكرار High frequency components: هي مكونات تتميز بأنها تتغير بصورة كبيرة في قيم المستوى الرمادي في مسافات صغيرة، ومن أمثلتها الحواف edges والتشوشات noise .
المكونات منخفضة التكرار Low frequency components: هي مكونات ثابتة او شبه ثابتة داخل الصورة وتغيراتها قليلة في قيم المستوى الرمادي ومن امثلتها الخلفيات backgrounds ونسيج الصورة textures عموما.
وهذا يقودنا إلى تعريف نوعين من الفلاتر المطبقة على الصورة:
high pass filter: it 'passes over' the high frequency components, and reduces or eliminates low frequency components,
الفلاتر عالية العبور HPF تتجاوز المكونات عالية التكرار بدون تغيير ولكنها تقلل أو تلغي المكونات منخفضة التكرار.
low pass filter: it 'passes over' the low frequency components, and reduces or eliminates high frequency components,
الفلاتر منخفضة العبور LPF تتجاوز المكونات المنخفضة التكرار بدون تغيير، ولكنها تقلل او تلغي المكونات عالية التكرار.
سنقول اختصار فلاتر عالية وفلاتر منخفضة.
ومن أمثلة الفلاتر المنخفضة الفلتر average الذي يزداد تأثيره المخفض كلما زاد حجمه، كما يلي:
```
>> c=imread('cameraman.tif');
>> f1=fspecial('average');
>> cf1=filter2(f1,c);
```
لاحظ شكل الصورة الناتجة مع الفلتر ذي البعد ٣×٣ ثم اعد تطبيقه مع نفس الفلتر ولكن بحجم اكبر ٩×٩ ثم اكبر ٢٥×٢٥، كما في الأشكال التالية:

(a) Original image

(b) Average filtering

(c) Using a 9×9 filter

(d) Using a 25×25 filter

والفلترين laplacian و log هما فلترات عاليا العبور، كما في المثالين التاليين:

```
>> f=fspecial('laplacian');
>> cf=filter2(f,c);
>> imshow(cf/100);
>> f1=fspecial('log');
>> cf1=filter2(f1,c);
>> figure,imshow(cf1/100);
```

والشكل التالي يوضح نتيجة تطبيقهما على الصورة c صورة رجل الكاميرا كما سيأتي:

(a) Laplacian filter

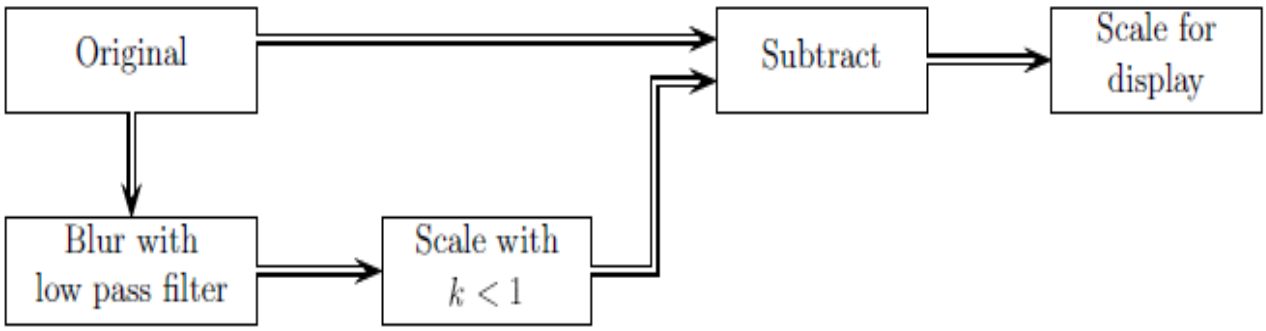
(b) Laplacian of Gaussian ("log") filtering

لاحظ كيف يقوم الفلتر عالي العبور بتغيير المناطق المنخفضة التكرار وتقليل قيمها حتى صارت اقرب إلى الصفر، وهذا هو سبب ظهور اللون الأسود في الصورتين الأخيريتن، أما الفلتر منخفض العبور فهو يقوم بتقليل قيم المكونات المرتفعة مما يعني اقتراب الصورة من المستوى الرمادي المتوسط أكثر وأكثر.

تحديد الحواف Edge sharpening

هي إحدى أهم تطبيقات الفلاتر المنخفضة حيث يتم في هذه التقنية عمل فلتر منخفض ثم طرحه من الصورة الأصلية فينتج لدينا صورة اصلية اكثر تحديدا لحواف مكوناتها، وتسمى هذه التقنية أيضا بتحسين الحواف أو القناع غير الحاد unsharp masking.

والمخطط في الشكل التالي يوضح هذه العملية:



تطبيق هذه الفكرة في الكود التالي، لاحظ اننا استخدمنا فلترا منخفضا هو الفلتر average:

```
>> f=fspecial('average');
>> xf=filter2(f,x);
>> xu=double(x)-xf/1.5;
>> imshow(xu/70);
```

ولقد حصلت تعديلات عديدة على هذه الفكرة، وتم تطوير فلتر يختصر خطوتي التحجيم scale والطرح subtracat بخطوة واحدة عبر الفلتر unsharp الذي يقبل وسيطا parameter واحد تحدد نسبة التحديد sharping المطلوب.

المثال التالي يطبق هذا الفلتر على صورة الصبي العابس:

```
>> p=imread('pout.tif');
>> u=fspecial('unsharp',0.5);
>> pu=filter2(u,p);
>> imshow(p),figure,imshow(pu/255);
```

لاحظ قسمة المصفوفة pu على 255 قبل عرضها بسبب زيادة قيم الصورة عن القيمة القصوى للون الرمادي.

وبطبيعة الحال فكل صورة تحتاج إلى قيمة مناسبة لتعديل القيم، في حين يتم استخدام الدالة im2gray بعض الأحيان لتقوم بنفس الدور.

الفلاتر غير الخطية Non-linear filters:

كما تحدثنا سابقا فإن الدالة المستخدمة في القناع هي التي تحدد نوع الفلتر فإذا كانت الدالة خطية فإن الفلتر يغدو خطيا وإذا كانت الدالة غير خطية فإننا نسمي الفلتر فلترا غير خطيا، كما هو حال هذا النوع من الفلاتر. وتتميز عنها الفلاتر الخطية بأنها أسهل في التوليد وأسرع في التنفيذ.

يتم تطبيق الفلاتر غير الخطية باستخدام الدالة nlfilter وابسط أمثلة الفلاتر غير الخطية فلتر القيمة العظمى maximum حيث يقوم باستبدال قيمة البكسل بأكبر قيمة من الجوار، وعكس ذلك يقوم فلتر القيمة الصغرى minimum. وكلا الفلترين يقومان بجزء من مهام الفلتر rank-order الذي يرتب قيم جوار النقطة المطبق عليها تصاعديا بحيث تكون القيمة الأولى هي العظمى والأخيرى هي الصغرى.

المثال التالي يستخدم الدالة nlfilter من أجل تطبيق فلتر القيمة العظمى على الصورة c:

```
>> cmax=nlfilter(c,[3,3],'max(x(:))');
```

والكود التالي يطبق فلتر القيمة الصغرى:

```
>> cmin=nlfilter(c,[3,3],'min(x(:))');
```

لاحظنا اننا مررنا اسم الصورة وحجم القناع ودالة تحدد نوع الفلتر على متغير افتراضي x.

الدالة nlfilter تعتبر دالة بطيئا نوعا ما خاصة إذا كلما كبر حجم قناع الفلتر، لهذا السبب سنستخدم الدالة colfilt لتطبيق الأمر السابق بسرعة اكبر:

```
>> cmax=colfilt(c,[3,3],'sliding',@max);
```

ولفلتر القيمة الصغرى، سنستخدم البارامتر @min

أما الدالة ordfilt2 فتقوم بتطبيق تقنية rank-order التي يمكننا من خلالها الحصول على فلتر القيمة العظمي او الصغرى والأهم من ذلك أي قيمة بينهما، كفلتر القيمة الوسطى ذا الأهمية الأكبر منهما، كلما في الأمر ان علينا اختيار الرقم الوسيط الثاني بعد الصورة، الرقم تسعة التالي يحدد القيمة العظمى بين ٩ قيم حيث أن حجم الفلتر هو ٣×٣:

```
>> cmax=ordfilt2(c,9,ones(3,3));
```

أما الرقم ١ فيحدد فلتر القيمة الصغرى:

>> cmin=ordfilt2(c,1,ones(3,3));

وللحصول على فلتر القيمة الوسطى نختار الرقم الأوسط وفي هذا المثال الرقم 5:

>> cmed=ordfilt2(c,5,ones(3,3));

وهناك دالة أخرى تستخدم فلتر القيمة الوسطى هي medfilt2 ونحتاجها من أجل ازالة التشوش كما سيأتي في الفصل الخامس، رغم ان هذه الدالة اصلا تستدعى الدالة الأصلية ordfilt2 ولهذا يستحسن استخدام الدالة الأصلية واختيار القيمة المتوسطة إن أمكن ذلك.

فيما صورة رجل الكاميرا بعد تطبيق فلتري القيمة الصغرى والعظمى عليه.

(a) Using a maximum filter

(b) Using a minimum filter

==========

انتهى بحمد الله وتوفيقه ، اعداد وترجمة فهد ال قاسم.

```
>> figure,plot(p,ph,'.'),grid on
```

What are you seeing here?

12. Experiment with some other greyscale images.

13. Using LUTs, and following the example given in section 2.4, write a simpler function for performing piecewise stretching than the function described in section 2.3.1.

An Introduction to Digital Image Processing with MATLAB

Alasdair McAndrew

School of Computer Science and Mathematics

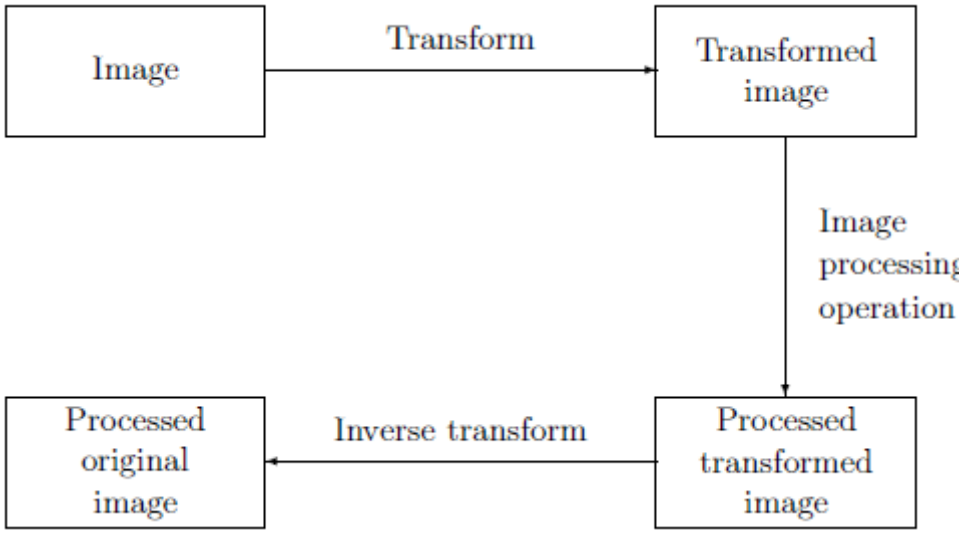
Victoria University of Technology

مدخل إلى معالجة الصور مع الماتلاب ـ ترجمة فهد آل قاسم fhdalqasem@yahoo.com

الفصل الرابع تحويل فورييه
The Fourier Transform

الفصل الرابع تحويل فورييه The Fourier Transform

إن لتحويل فورييه أهمية أساسية في معالجة الصور، فهو يسمح لنا بانجاز مهام قد تكون مستحيلة الانجاز بأي طريقة أخرى، كما أن من فاعليته أنه يسمح لنا بإنجاز مهام أخرى بسرعة أكبر.
ويقدم لنا تحويل فورييه من خلال أشياء كثيرة، بديلا قويا في الفلترة الحيزية الخطية، فهو أكثر فاعلية مقارنة بطرق الفلترة الأخرى عند العمل على فلاتر كبيرة، وهذا التحويل يتيح لنا عزل ومعالجة فترات محددة في الصورة، وإجراء عمليات الفلترة بنوعيها عالي العبور high-pass ومنخفض العبور low-pass بدرجة عالية من الدقة.
وقبل مناقشة علاقة تحويل فورييه بالصور، سوف نقدم تحويل فورييه أحادي البعد وبعض خصائصه one-dimensional Fourier transform.
تذكير من الفصل الثاني: يقوم التحويل[1] بتمثيل قيم البكسلات مع بعضها البعض، ولكن بطريقة مكافئة، وهو يسمح لبعض الخوارزميات الأكثر كفاءة وقوة بالعمل على الصورة، وكما سنرى لاحقا، يتعامل التحويل مع الصورة كاملة كأنها وحدة واحدة كبيرة، والشكل التالي يوضح الفكرة باختصار.



Schema for transform processing

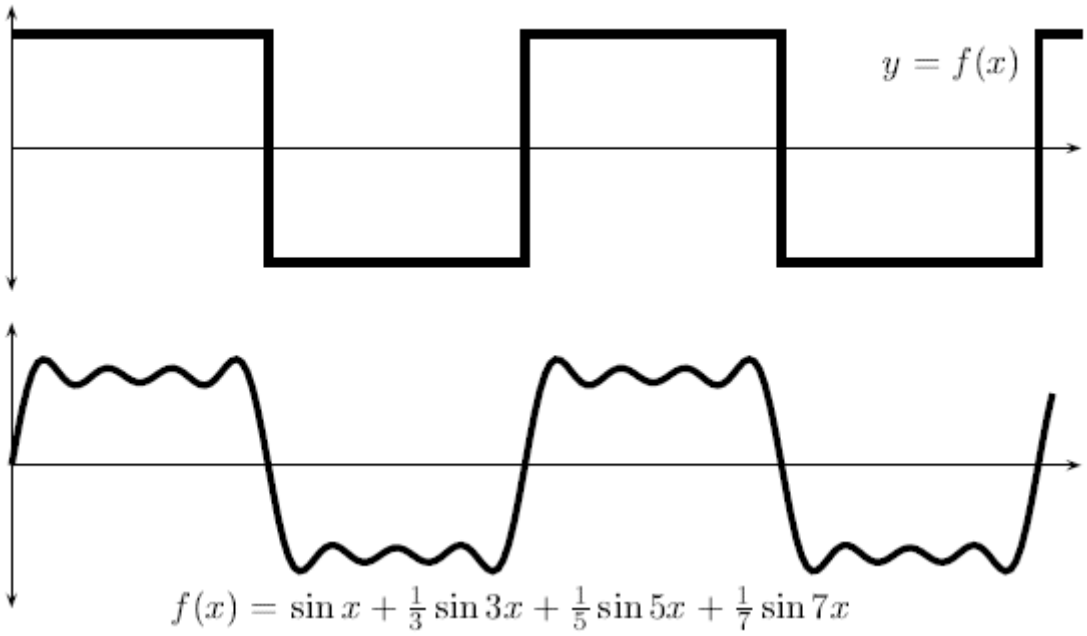
خلفية نظرية Background

سوف نبدأ بتوضيح مسألة أن الدالة الدورية يمكن أن تكتب كمجموعة مكون من دالتي الجيب وجيب التمام لمختلف الفترات (الدورات) frequencies والنطاقات.
بعض الدوال تحتاج إلى عدد محدود من الدوال في عملية التجزئة هذه، والبعض الآخر يحتاج إلى عدد غير منتهي من الدوال، مثلا الموجه المربعة الموضحة في الشكل البياني التالي، تتجزئ بتحويلها إلى الصيغة الرياضية:

$$f(x) = \sin x + \frac{1}{3}\sin 3x + \frac{1}{5}\sin 5x + \frac{1}{7}\sin 7x + \frac{1}{9}\sin 9x + \cdots$$

حسب الشكل:

[1] كلمة (التحويل) في هذا الفصل يقصد بها دائما (تحويل فورييه) وليس أي تحويل آخر ...حسب نوع التحويل المفهوم من السياق.



في الشكل السابق، قمنا بأخذ الأجزاء الأربعة الأولى فقط من أجل عملية التقريب، وكلما أخذنا أجزاء أكثر، كلما كان ناتج مجموع الدوال أقرب إلى الدالة الأصلية.
وبالمناسبة فإن الترددات أو التكرارات على مستوى الصورة او المصفوفة خصوصا يقصد بها : مقياس لمقدار التغير في قيم الصورة فإذا كانت التغيرات كثيرة على مستوى القيم كانت التكرارات frequencies كثيرة والعكس بالعكس.

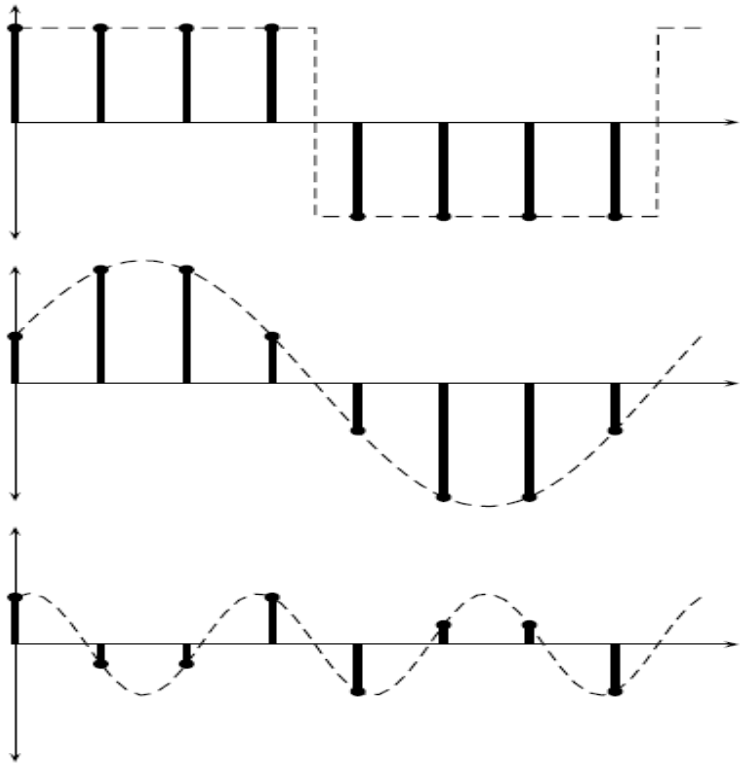
تحويل فورييه المتقطع من أحادي البعد

The one-dimensional discrete Fourier transform

عندما نقوم بالتعامل مع الدالة المتقطعة، كما سوف نفعل مع الصور، فإننا الحالة ستكون فيها اختلاف بسيط عن الحالة السابقة، حيث أن لدينا فقط عدد محدد من القيم، ونحتاج إلى عدد محدود من الدوال لتقريبها.
لنأخذ في الاعتبار المتسلسلة المتقطعة التالية:

1, 1, 1, 1, -1, -1, -1, -1

سنقوم هنا بتحويل المتسلسلة (تقريبها) وفق الشكل التالي، الذي يوضح هذه المتسلسلة كمجموعة من دالتي جيب sin فقط، فتحويل فورييه يسمح لنا بالحصول على دوال الجيب هذه وحدها حيث تتحول في دالة الجمع إلى الدالة أو المتسلسلة الأصلية.



ولأننا سوف نهتم بالسلاسل المتقطعة وهذا ينطبق على الصور، فإننا سوف نتحدث عن تحويل فورييه المتقطع discrete Fourier transform فقط والذي يختصر إلى DFT.

التعريف الرياضي لتحويل فورير احادي البعد one dimensional DFT

ليكن لدينا

$$\mathbf{f} = [f_0, f_1, f_2, \ldots, f_{N-1}]$$

هي متسلسلة من الطول N ونعرف تحويل فورييه المتقطع المقابل لها بأنه:

$$\mathbf{F} = [F_0, F_1, F_2, \ldots, F_{N-1}]$$

حيث أن إيجاد كل قيمة منه يتم عبر الصيغة:

$$F_u = \frac{1}{N} \sum_{x=0}^{N-1} \exp\left[-2\pi i \frac{xu}{N}\right] f_x. \qquad (4.2)$$

أما دالتها العكسية فهي:

$$x_u = \sum_{x=0}^{N-1} \exp\left[2\pi i \frac{xu}{N}\right] F_u. \qquad (4.3)$$

إذا قمنا بمقارنة دالة التحويل والدالة العكسية لها سوف نجد أن هناك فرقين بسيطين هما:

١. غياب معامل التدرج scaling factor $\frac{1}{N}$ في العكسية.
٢. تغير الإشارة داخل قوس الأس من سالب في التحويل إلى موجب في العكسية.

تحويل فورييه السريع The Fast Fourier Transform

إن أحد أسباب التي تجعل من الـ DFT مستخدما في معالجة الصور هو وجود خوارزمية سريعة جدا لحسابه، وهناك عدد من الخوارزميات السريعة للغاية والفعالة لحساب الـ DFT وإحدى هذه الخوارزميات تسمى تحويل فورييه السريعة The Fast Fourier Transform أو اختصارا FFT، وباستخدام الـ FFT نستطيع تقليل الوقت الذي نحتاجه لحوسبة DFT.

وإحدى طرق الـ FFT تعمل تعادويا recursively (تكراريا) بواسطة تقسيم المتجه الأصلي إلى نصفين، وحساب الـ FFT لكل نصف، ثم وضع النتائج سويا. وهذا يعني اكثر كفاءة عندما يكون طول المتجه من الأس ٢.
الجدول التالي يوضح ميزات استخدام خوارزمية الـ FFT مقارنة بالتعريف الرياضي بالمعادلات المباشرة، وذلك بمقارنة عدد عمليات الضرب المطلوبة لكل طريقة. بالنسبة لمتجه من الطول 2^n تأخذ الطريقة المباشرة 2^{2n} عملية ضرب، أما الـ FFT ففقط $n2^n$. يكون كلفة الوقت الموفرة من الرتبة $2^n/n$. ومن الواضح أن ميزة استخدام الخوارزمية الأسرع الـ FFT يصبح اكبر كلما زاد حجم المتجه/المصفوفة/ السلسلة التي نتعامل معها.وبسبب ميزات الحوسبة الموضح اعلاه فإننا التنفيذ سيكون باستخدام الـ FFT.
الجدول التالي يوضح المقارنة بين الطريقة السريعة الطرق المباشرة من حيث كفاءة الحوسبة:

2^n	Direct arithmetic	FFT	Increase in speed
4	16	8	2.0
8	84	24	2.67
16	256	64	4.0
32	1024	160	6.4
64	4096	384	10.67
128	16384	896	18.3
256	65536	2048	32.0
512	262144	4608	56.9
1024	1048576	10240	102.4

Table 4.1: Comparison of FFT and direct arithmetic

تحويل فورييه المتقطع ثنائي البعد The two-dimensional DFT
في المصفوفات ثنائية البعد، تأخذ DFT المصفوفة كمدخل، وتعيد مصفوفة أخرى من نفس الحجم، كمخرج، فإذا كانت قيم المصفوفة ممثلة بالدالة f(x,y) حيث x وy هي أدلة المصفوفة، فسنرمز لمصفوفة المخرجات بـ F(u,v)، ونسمي المصفوفة F تحويل فورييه للمصفوفة f ثم نكتب:

$$F = \mathcal{F}(f).$$

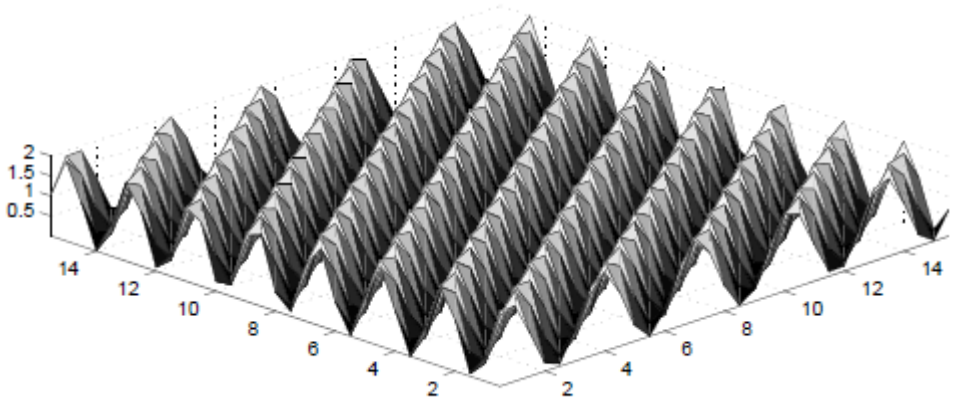
وبهذا تكون الدالة الأصلية f هي معكوس التحويل أي معكوس الدالة F وتكتب:

$$f = \mathcal{F}^{-1}(F).$$

وكما شاهدنا سابقا حيث أن الدالة أحادية البعد يمكن ان تكتب كمجموع من دوال الجيب وجيب التمام، ونعلم أن الصورة المعطاة يمكن النظر إليها كدالة مصفوفة ثنائية البعد f(x,y) ، فإنه من المبرر أن نفترض أنها يمكن أن تكتب كمجموع من الدوال المجعدة corrugation functions التي لها الشكل العام:

$$z = a\sin(bx + cy).$$

وكمثال للدوال المجعدة للنظر إلى الشكل البياني التالي:



وهذا بالضبط ما تفعله دوال تحويل فورييه ثنائية البعد، حيث أنها تقوم بإعادة كتابة الدالة الأصلية إلى مجموعة من هذه الأجزاء/ التجعدات.

إن تعريف تحويل فورييه المتقطع ثنائي البعد يكون مشابها للتحويل أحادي البعد، حيث سيكون التحويل وعكس التحويل لمصفوفة من البعد M×N ، حيث سنفترض لأجل الترميز أن الدليل x قيمه من 0 إلى M-1، وأن الدليل y قيمه من 0 إلى N-1، فيكون لدينا:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]. \quad (4.4)$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]. \quad (4.5)$$

يجدر الإشارة أن الصيغة السابقة ليست مستخدمة من قبل جميع المؤلفين، هناك اختلافات جزئية كثيرة.

بعض خصائص تحويل فورييه ثنائي البعد Some properties of the two dimensional DFT

كل خصائص التحويل أحادية البعد تظهر هنا، ولكن هناك بعض الخصائص التي لم نتطرق لها سابقا، خاصة تلك المتعلقة بمعالجة الصور، ومنها:

التشابه Similarity نلاحظ اولا أن التحويل والتحويل العكسي متشابه جدا، باستثناء معامل التدرج $\frac{1}{MN}$ في دالة التحويل العكسية، والإشارة السالبة في الأس بدالة التحويل (وتسمى بدالة التحويل الأمامية forward transform).

وهذا يعني أن نفس الخوارزمية سوف تستخدم، باستثناء بعض الضبط البسيط، لكلا التحويلين الأمامي والعكسي.

استخدام الـ DFT كفلتر حيزي: لاحظ الصيغة

$$\exp\left[\pm 2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]$$

ولاحظ أن حسابها سيكون مستقلا عن القيمتين f و F وهذا يعني انه بالإمكان حسابها مسبقا، ووضعها فقط بعد ذلك في الصيغة أعلاه، ويعني أيضا أن كل قيمة في F(u,v) يمكن الحصول عليها بضرب كل قيمة في f(x,y) بقيمة ثابتة، وإضافة كل النتائج. ولكن هذا بكل دقة هو ما تفعله الفلترة الحيزية الخطية linear spatial filter، إنها كما رأينا سابقا تقوم بضرب كل العناصر تحت القناع بقيم ثابتة ثم تضيف النتائج مع بعض، بمعنى انه يمكننا اعتبار أن الـ DFT كفلتر حيزي خطي يكون كبيرا كلما كبرت الصورة. وللتعامل مع مشكلة حواف الصورة image edges، سوف نفترض أن الصورة قد مطت في كل الاتجاهات بحيث يكون القناع mask قادرا على رؤية قيم للصورة لكي يستخدمها.

قابلية الفصل separability

لاحظ أن (عناصر الفلتر) في تحويل فورييه يمكن أن يصاغ كجداء عمليات ضرب منفصلة:

$$\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right] = \exp\left[2\pi i\frac{xu}{M}\right]\exp\left[2\pi i\frac{yv}{N}\right].$$

وتكون قيمة الجداء الأولى التالية

$$\exp\left[2\pi i\frac{xu}{M}\right]$$

معتمدة على قيمتي x وu ومستقلة عن القيمتين y و v . وبشكل معاكس نستطيع ملاحظة الجداء الثاني

$$\exp\left[2\pi i\frac{yv}{N}\right]$$

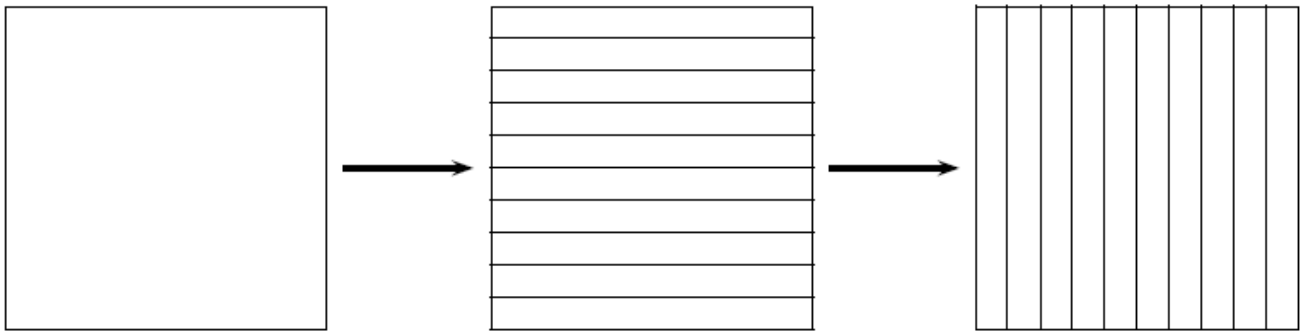
حيث تعتمد على قيمتي y و v، وهي مستقلة عن x وu، وهذا يعني أننا يمكن أن نجزئ الصيغة أعلاه إلى صيغ أبسط تعمل فقط على صف rows واحد أو عمود columns واحد كالتالي :

$$F(u) = \sum_{x=0}^{M-1} f(x)\exp\left[-2\pi i\frac{xu}{M}\right], \quad (4.6)$$

$$f(x) = \frac{1}{M}\sum_{u=0}^{M-1} F(u)\exp\left[2\pi i\frac{xu}{M}\right]. \quad (4.7)$$

فإذا قمنا بتبديل x و u بالمتغيرين y و v، فسنحصل على الصيغة المقابلة للتحويل DFT على أعمدة المصفوفة، هذه الصيغ تعرف تحويل DFT المتقطع أحادي البعد one-dimensional لمتجه vector ، وهو ما نختصره بـ DFT.

يمكننا إذن حساب تحويل فورييه ثنائي البعد (2-D DFT) باستخدام هذه الخاصية (قابلية الفصل separability) من أجل الحصول على مصفوفة الـتحويل، فنقوم أولا بحساب الـ DFT (أحادية البعد) لكل الصفوف، ثم نحسبها لكل الأعمدة الناتجة، كما هو موضح بالشكل التالي:



(a) الصورة الأصلية Original
(b) تحويل الصفوف DFT of each row of (a)
(c) تحويل الأعمدة DFT of each column of (b)

وحيث أن الجداء (ناتج الضرب product) لا يتأثر بالترتيب، فإنه يمكن القيام بشكل مكافئ بحساب الـ 2-D DFT، عن طريق حساب الـ DFT لكل الأعمدة أولا ومن ثم حساب الـ DFT لكل الصفوف الناتجة.

الخطية Linearity: من الميزات المهمة في التحويل الأحادي DFT أنه خطي، فالتحويل الحاصل للمكونات المجموعة هو نفسه مجموع التحويل الناتج عن كل مكون، ونفس الأمر بالنسبة للضرب في عنصر ثابت:

$$\begin{aligned} \mathcal{F}(f+g) &= \mathcal{F}(f)+\mathcal{F}(g) \\ \mathcal{F}(kf) &= k\mathcal{F}(f) \end{aligned}$$

حيث أن k عنصر وحيد (متجه بعنصر واحد scalar)، بينما الدوال f وg هي مصفوفات، وكل صيغة أعلاه تنتج مباشرة من التعريف المعطى في المعادلة ٤.٤ سابقا، هذه الخاصية لها استخدامات كبيرة للتعامل مع الصور كما في image degradation، كالضوضاء noise التي يمكن نمذجتها كالمجموع:

d = f + n

حيث أن f هي الصورة الأصلية، وn هي الضجيج، أما d فهي الصورة الناتجة من تجريد الصورة degraded image من الضجيج، حيث أن:

$$\mathcal{F}(d) = \mathcal{F}(f) + \mathcal{F}(n)$$

وربما استطعنا حذف أو تقليل n عن طريق تعديل التحويل المطبق.

نظرية الالتفاف The convolution theorem

تلك النتائج السابقة تقدم واحدا من أهم فوائد استخدام تحويل فورييه المتقطع DFT .

ليكن لدينا الصورة M ونرغب في تطبيق فلتر الالتفاف عليها، باستخدام الفلتر الحيزي S spatial filter، فإن طريقتنا ستكون بوضع الفلتر S على كل بكسل في M على الدور، وحساب الجداء لجميع القيم الرمادية المناظرة في M مع عناصر S، وجمع ناتج الجداء. نتيجة ذلك تسمى الالتفاف الرقمي لـ M وS، والتي تصاغ بالشكل:

M * S.

لكن هذه الطريقة لتطبيق الالتفاف تعتبر بطيئة جدا، خاصة لو كانت S كبيرة، وقد نصت نظرية الالتفاف على أن الجداء M * S يمكن الحصول عليه بالخطوات المتسلسلة التالية:

١. القيام بحشو Pad الفلتر S بالأصفر مما يجعله بنفس حجم M ويرمز للناتج بـ له بـ 'S.
٢. طبق التحويل الأحادي DFT على كلا من M و 'S فنحصل على F(M) و F('S).
٣. قم بإيجاد جداء التحويلين عنصر ضرب عنصر element-by-element product، لهذين التحويلين بالصيغة:

$$\mathcal{F}(M) \cdot \mathcal{F}(S').$$

٤. خذ التحويل العكسي للناتج:

$$\mathcal{F}^{-1}(\mathcal{F}(M)\cdot\mathcal{F}(S')).$$

واعتمادا على هذه الخوارزمية فإننا نلخص نظرية الالتفاف بالصيغة:

$$M * S = \mathcal{F}^{-1}(\mathcal{F}(M)\cdot\mathcal{F}(S'))$$

أو بشكل مكافئ بـ:

$$\mathcal{F}(M * S) = \mathcal{F}(M)\cdot\mathcal{F}(S').$$

ورغم أنها قد تبدو طريقة مكلفة وغير ملائما لحساب أمر يبدو بسيطا كما نعرف عن فلتر الالتفاف، فإنه تمكننا أن نحصل على ميزة سرعة عالية إذا كانت S كبيرة.
مثلا، ليكن لدينا رغبة في حساب الالتفاف على صورة بعدها 512 × 512، مع فلتر بعده ٣٢×٣٢، لفعل ذلك بالطريقة المباشرة فإنه يتطلب $٣٢^{٢}$ أي ١٠٢٤ عملية ضرب لكل بكسل أي ٢٦٢١٤٤= 512 × 512 ، وهذا يعطينا أجمالي من عمليات الضرب قدره ٢٦٨٤٣٥٤٥٦ عملية ضرب مطلوبة في الطريقة المباشرة.
والآن بالنظر إلى كلفة استخدام الـ DFT (باستخدام الخوارزمية FFT)، حيث سيكون لدينا ٤٦٠٨ عملية لكل صف، بالنظر إلى الجدول ٤.١ أعلاه الذي يوضح ميزات خوارزمية التحويل السريعة، ولأن معنا ٥١٢ صفا فهذا يعني ٥١٢×٤٦٠٨= ٢٣٥٩٢٩٦ عملية ضرب، ومثلها بالنسبة لضرب الأعمدة، وهذا يعني اننا نحتاج للحصول على التحويل DFT إلى ٤٧١٨٥٩٢ عملية ضرب، ومثلها للتحويل الخاص بالفلتر، إضافة إلى ايجاد معكوس الفلتر،وأخيرا نحتاج إلى 512 × 512 عملية ضرب من أجل إنجاز ضرب التحويلين لإيجاد الجداء النهائي.
وهكذا فإن إجمال عمليات الضرب التي نحتاجها لإنجاز الالتفاف convolution باستخدام DFT هي:

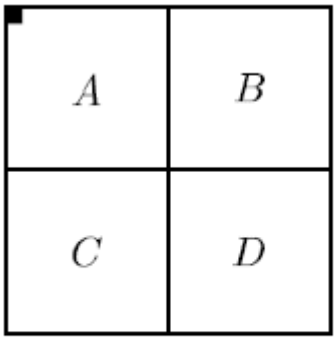
$$4718592 \times 3 + 262144 = 14,417,920$$

وهي قيمة تعني توفيرا كبيرا في الكلفة مقارنة بالطريقة المباشرة التي أوضحنا سابقا أنها $268,435,456$.

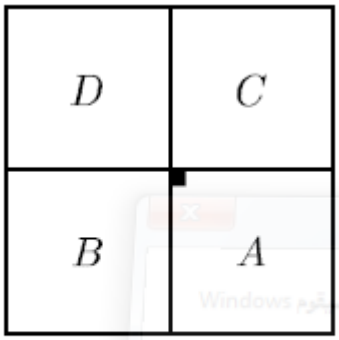
المعامل دي سي DC coefficient:
إن قيمة F(0,0) للتحويل DFT يسمى بالمعامل دي سي (DC coefficient)، فإذا وضعنا u=v=0 في التعريف المعطى بالمعادلة ٤.٤، فإننا نحصل على :

$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp(0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y).$$

وهذا يعني أن هذا المعامل يساوي مجموعة كل القيم في المصفوفة الأصلية.
التبديل Shifting:
من أجل أغراض العرض فإنه من المتعارف عليه أن نجعل المعامل DC في مركز المصفوفة، هذا يتم في المصفوفة f(x,y) إذا قمنا بضرب جميع أدلة عناصر المصفوفة بـقيمة قبل حساب التحويل، هذه القيمة نتتج من دالة معينة، والذي يحصل دائما، أنه إذا كانت المصفوفة احادية فإن النصف الثاني من عناصر يتجه للبداية والنصف الأول يعود بعده، أما في المصفوفة ثنائية البعد فإن كل ربعين متناضرين يتبادلان مواقعهما، كما هو موضح في المثال الذي بالشكل التالي، يتم تبديل shifting المصفوفة الثنائية، حيث أن A هي جزء المصفوفة التي تحتوي على المعامل DC المقصود.
لاحظ كيف تمت تقسيم المصفوفة إلى أربعة أرباع A,B,C,D وكيف تبادلت هذه المصفوفات الجزئية مواقعها، ولاحظ موقع المعامل DC الذي كان يعتبر أول العناصر في المصفوفة الجزئية A والموضح بالمربع الأسود، وكيف انه انتقل تماما إلى مركز المصفوفة بعد تبديلها.



An FFT



After shifting

التناظر المقترن Conjugate symmetry

إن تحليل تحويل فوريه يقودنا إلى الخاصية المتناظرة، فإذا قمنا بتغيير قيم المعادلة ٤.٤ بحيث يكون u=- u و v=-v، فإن المعادلة ستكون بالصيغة:

$$\mathcal{F}(u,v) = \mathcal{F}^*(-u + pM, -v + qN)$$

وذلك لأي عددين صحيحين p و q، أما M و N فهما كما نعلم بعدا المصفوفة الأصلية.

المصفوفة الناتجة تدل على أن نصف التحويل هو نسخة معكوسة، كصورة في المرآة، لاقتران النصف الآخر، نستطيع أن نفكر أن النصفين العلوي والسفلي أو أن النصفين اليمين والشمال هما نسخة معكوسة للاقتران الناتج من النصف الآخر. الشكل التالي يوضح هذا التناظر المتعلق بالتحويل المبدل shifted DFT، وكما في الشكل السابق فإن المربع الأسود الصغير يوضح موقع المعامل DC، والمقصود بالتناظر أن المعلومات يمكن أن تعطى لنصف واحد وتكرر فقط للنصف الثاني.

	a		a^*
b^*	B^*	d^*	A^*
	c		c^*
b	A	d	B

عرض التحويلات Displaying transforms

لدينا تحويل فوريه F(u,v) للصورة f(x,y)، ونرغب بمعرفة كيف يبدو كصورة، فإذا كانت عناصر F(u,v) أعداد مركبة complex numbers، فإننا لن نستطيع مشاهدة قيمها مباشرة، لكننا نستطيع رؤية المقدار |F(u,v)|، وبما أنها أعداد من النوع double، وهي عموما ذات مدى كبير، فإن لدينا طريقتين :

١. إيجاد القيمة العظمى m لـ |F(u,v)|، وهذا سيكون المعامل DC، واستخدام دالة عرض الصور imshow من أجل العرض بالشكل:

```
imshow (|F(u,v)| / m);
```

٢. أو، استخدام الدالة mat2gray من أجل مشاهدة المقدار |F(u,v)| مباشرة.

إحدى المعضلات هي أن المعامل DC هو بشكل عام كبير جدا أكثر من باقي القيم، وهذا يؤثر على عرض الصورة الناتجة من التحويل جاعلا إياه نقطة بيضاء وحيدة محاطة بالأسود، وإحدى طرق توسيع القيم تتم بإيجاد لوغاريتم |F(u,v)| وعرض الصيغة:

$$\log(1 + |F(u,v)|).$$

إن عرض المقدار |F(u,v)| يسمى طيف التحويل the spectrum of the transform، وسوف نرى بعض الأمثلة عن ذلك لاحقا.

تحويل فوريه في الماتلاب Fourier transforms in Matlab

الدوال المقابلة للصيغ الرياضية أعلاه في الماتلاب هي:
الدالة ()fft التي تحسب تحويل فورييه DFT على متجه vector (مصفوفة أحادية البعد).
الدالة () ifft التي تحسب معكوس التحويل DFT على متجه.
الدالة () fft2 التي تحسب التحويل DFT على مصفوفة matrix.
الدالة () ifft2 التي تحسب معكوس التحويل DFT على مصفوفة.
الدالة ()fftshift التي تحسب تحويل فورييه DFT ثم تقوم بتبديل التحويل shifting.
وقبل أن نقوم بتطبيق هذه الدوال على الصور سوف نبدأ بتطبيقها على مصفوفات بسيطة حتى نفهم كيفية عمل التحويل DFT:

أمثلة١. Example 1

لنفترض أن لدينا المصفوفة الثابتة f(x,y)=1، وبالعودة إلى فكرة جمع الدوال المجعدة corrugations، بعدها لن يكون من المطلوب استخدام الثابت، وكذلك نحن نريد أن تتكون دالة التحويل DFT من المعامل DC محاطا بقيم صفرية ما أمكن، فسوف نستخدم الدالة ones التي تنتج لنا مصفوفة واحدات (كل قيمها واحد)، حسب أبعاد مربعة معطاة لها.

```
>> a=ones(8);
>> fft2(a)
```

ولنتتبع تنفيذ السطرين أعلاه في الماتلاب في الشكلين التاليين:

```
>> aa=ones(8)

aa =

     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
>> fft2(aa)

ans =

    64     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
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

لاحظ معامل DC المعرف سابقا بأنه مجموع قيم عناصر المصفوفة الأصلية، وأنه أو عناصر التحويل.

أمثلة٢. Example 2

والآن لنطبق على مصفوفة تتكون من تجعدات فردية، ولذلك نفترض المصفوفة a الصغيرة ثم نقوم بعمل تكبير لها عبر الدالة repmat التي تولد مصفوفة اكبر حجما بتكرار قيم مصفوفة سابقة.

```
>> a = [100 200; 100 200];
>> a = repmat(a,4,4)
```

وسينتج عن ذلك المصفوفة:

a =

100	200	100	200	100	200	100	200
100	200	100	200	100	200	100	200
100	200	100	200	100	200	100	200
100	200	100	200	100	200	100	200
100	200	100	200	100	200	100	200
100	200	100	200	100	200	100	200
100	200	100	200	100	200	100	200
100	200	100	200	100	200	100	200

أما ناتج تطبيق التحويل عليها فسوف يولد لنا المصفوفة af كما في الكود التالي:

```
>> af = fft2(a)
```

af =

9600	0	0	0	-3200	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	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

يبدو ما فعلناه هنا كأننا جمعنا مصفوفتين ، المصفوفة الثابتة التي تساوي ١٥٠ والمصفوفة المجعدة التي تتبادل قيمها بين ٥٠ و -٥٠ من اليسار إلى اليمين. المصفوفة الثابتة سوف تنتج لنا كما في المثال السابق المعامل DC وحيدا وقيمته ٦٤×١٥٠=٩٦٠٠، قيمة التجعد الوحيدة corrugation a single value. ووفق الخاصية الخطية linearity فسوف يكون التحويل عبارة عن قيمتين فقط.

Example 2 أمثلة٢.

وسوف نأخذ هنا مصفوفة أحادية الحافة، أو حافة من خطوة واحدة single step edge

```
>> a = [zeros(8,4) ones(8,4)]
```

a =

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

الآن سوف ننجز تحويل فوريه مع التبديل، وذلك لجعل معامل DC في المركز، وحيث أنه يحتوي على بعض القيم المركبة، فسوف نقوم لتبسيط ذلك بعرض القيم المطلقة المحولة.

```
>> af=fftshift(fft2(a));
>> round(abs(af))
ans =
```

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	9	0	21	32	21	0	9
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

إن معامل DC هو بالطبع مجموع جميع قيم a، أما باقي القيم فيمكن اعتبارها معاملات دوال الجيب sin المطلوبة لتشكيل الحافة edge، كما هو معطى في المعادلة ٤.١. أما نسخ القيم حول المعامل DC فهو نتيجة للتناظر الموجود في التحويل DFT.

تحويل فورييه مع الصور Fourier transforms of images

سوف ننشئ القليل من الصور البسيطة، وسوف نرى ما ينتج عن معامل فورييه عندما يطبق عليها.

المثال ١: سوف ننشئ صورة بسيطة مكونة من حافة واحدة.

```
>> a=[zeros(256,128) ones(256,128)];
```

إذا نظرنا إلى هذه المصفوفة باعتبارها صورة فسوف تكون عبارة عن صورة نصفها الأيمن أسود والنصف الأيسر أبيض، والآن لنأخذ تحويل الصورة ثم نطبق عليها التبديل.

```
>> af=fftshift(fft2(a));
```

الآن سوف نعرض طيف التحويل spectrum وذلك بطريقتين:

الأولى:

```
afl=log(1+abs(af));
imshow(afl/afl(129,129))
```

استخدمنا السطر الأول لكي يسمح السطر الثاني بعرض المصفوفة وذلك بعد أن تخلصنا من القيم المركبة وبالطبع بعد التبديل فإن المعامل DC سوف يكون في الموقع (x,y)=(129,129)، ولقد قمنا بتوسيع التحويل باستخدام اللوغاريتم log وتقسيم النتيجة على القيمة المتوسطة للحصول على مصفوفة من النوع double مع قيم في المدى 0.1-0.0، وهذا يجعل المصفوفة قابلة للعرض باستخدام دالة عرض الصورimshow.

الثانية:

```
imshow(mat2gray(log(1+abs(af))))
```

وكما شاهدنا في الفصل الثالث فإن الدالة mat2gray تقوم بشكل تلقائي بتكبير المصفوفة للعرض كصورة.

وفي الحقيقة من المتعارف عليه أن يتم كتابة دالة صغيرة من قبل المبرمج لمشاهدة التحويل، وهذه الدالة fftshow الموضحة أدناه نقدمها على سبيل المثال.

```
function fftshow(f,type)
% Usage: FFTSHOW(F,TYPE)
%
% Displays the fft matrix F using imshow, where TYPE must be one of
% 'abs' or 'log'. If TYPE='abs', then then abs(f) is displayed; if
% TYPE='log' then log(1+abs(f)) is displayed. If TYPE is omitted, then
% 'log' is chosen as a default.
%% Example:
% c=imread('cameraman.tif');
% cf=fftshift(fft2(c));
% fftshow(cf,'abs')
%
if nargin<2,
type='log';
end
if (type=='log')
fl = log(1+abs(f));
```

```
fm = max(fl(:));
imshow(im2uint8(fl/fm))
elseif (type==’abs’)
fa=abs(f);
fm=max(fa(:));
imshow(fa/fm)
else
error(’TYPE must be abs or log.’);
end;
```

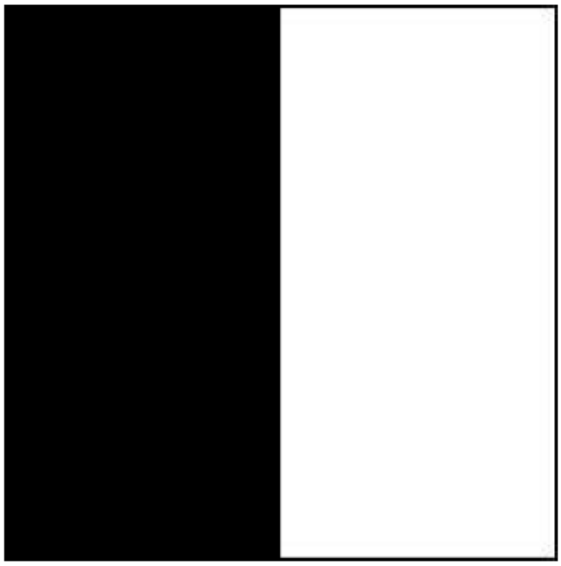
بالنسبة للتحويل af فإننا سوف نستدعي الدالة fftshow على خيارين، إما:

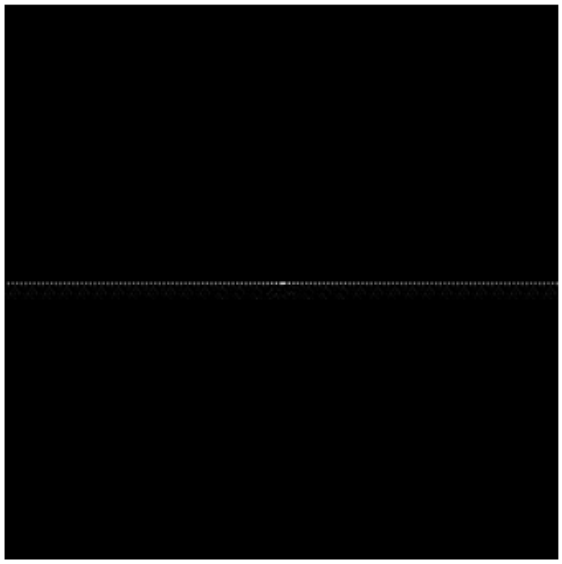
```
>>fftshow(af,’log’);
```

سوف تعرض الصورة بعد تعديلها بدالة اللوغاريتم، أو:

```
>>fftshow(af,’abs’);
```

التي ستعرض الصورة باستخدام دالة المقياس abs، والشكل التالى يوضح الصورة قبل وبعد التحويل:



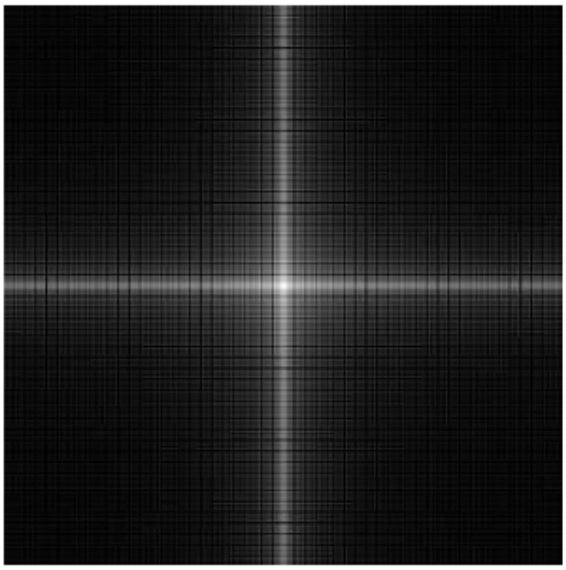


المثال ٢:

والآن سوف ننشئ صندوقا (مربعا) والتحويل الخاص به

```
>> a=zeros(256,256);
>> a(78:178,78:178)=1;
>> imshow(a)
>> af=fftshift(fft2(a));
>> figure,fftshow(af,’abs’)
```

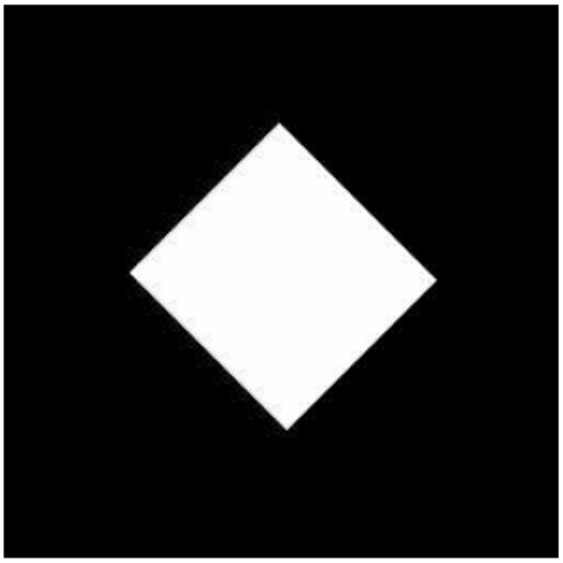
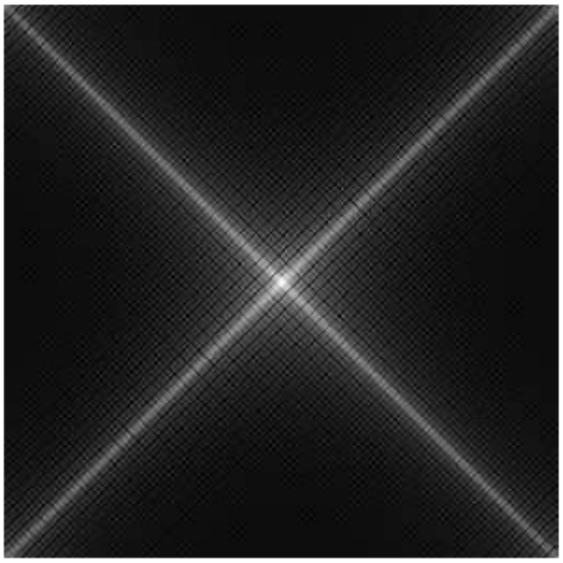
الشكل التالي يوضح الصورة المقصودة، والتحويل على اليمين.

المثال ٣: والآن سوف ننظر إلى الصندوق المربع بعد تدويره بحيث يصبح معينا (الماسي الشكل diamond)

```
>> [x,y]=meshgrid(1:256,1:256);
>> b=(x+y<329)&(x+y>182)&(x-y>-67)&(x-y<73);
>> imshow(b)
>> bf=fftshift(fft2(b));
>> figure,fftshow(bf)
```

تقوم الدالة meshgrid بتوليد مصفوفتين اعتمادا على تكرار معين لقيم المتجهين المدخلين إليها، وغالبا ما تكون هناك علاقة بين هاتين المصفوفتين تستخدم في رسم دوال معينة، في هذا المثال قمنا بتوليد المصفوفتين x وy من أجل عرض المصفوفة كالصورة المطلوبة.
ونتيجة ذلك موضحة بالشكل التالي الصورة الأصلية والتحويل، لاحظ أن التحويل هو تدوير التحويل الخاص بالصورة السابقة.

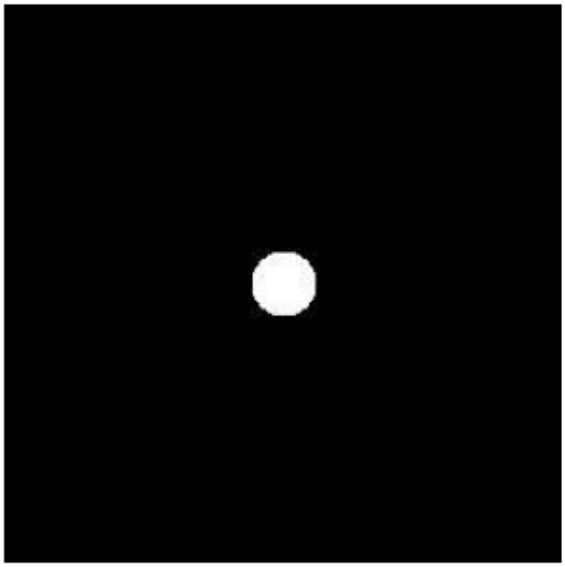
المثال ٤:
سوف ننشئ في هذا المثال دائرة، ثم نقدم التحويل الخاص بها.

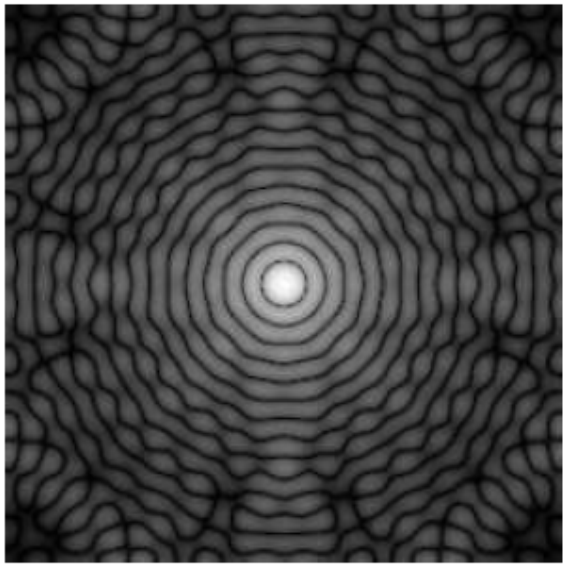
```
>> [x,y]=meshgrid(-128:217,-128:127);
>> z=sqrt(x.^2+y.^2);
>> c=(z<15);
```

الكود السابق قام بتوليد المصفوفة c التي تظهر صورتها في الشكل أدناه كدائرة بيضاء داخل محيط اسود.
أما تحويل فوريه الخاص بها فكالتالي[٢]:

[٢] لاحظ أن أول سطر في هذا المقطع البرمجي مكتوب في المادة الأصلية بشكل خطئ.

```
>> cf=fftshift(fft2(c));
>> fftshow(cf,'log');
```





لاحظ وجود تقطعات في حواف صورة التحويل، سواء في صورة الدائرة أو المربع في المثال السابق، يمكننا اعتبار التقطعات التي على الخط معاملات دوال التجعد في التحويل، اما بالنسبة للدائرة هناك خطوط للقيم تشع من الدائرة، مما يجعلها تظهر كأنها دوائر في التحويل.

إذا كانت التقطعات في الدائرة خفيفة نوعا بحيث تظهر الحواف كأنها ملطخة، فسيكون التحويل بدون حلقات، دائرة كهذه يمكن الحصول عليها بالمعادلة التالية (اعتمادا على قيمة z لا c).

```
b=1./(1+(z./15).^2);
```

ولنجرب تحويل هذه الصورة كيف سيبدو.

```
>> cf=fftshift(fft2(b));
>> fftshow(cf,'log');
```

فلترة النطاق المتكرر Filtering in the frequency domain

لقدت تعلمنا سابقا ميزة استخدام تحويل فورييه في تحسين نظرية الالتفاف، حيث يتم انجاز الفلترة الحيزية باستخدام الضرب حسب العنصر بمصفوفة فلتر تحويل خاصة.

في هذا الجزء سوف نرى نتفحص بعض طرق الفلترة باستخدام تحويل فورييه.

الفلت**رة المثالية** Ideal filtering

الفلترة منخفضة العبور Low pass filtering

ليكن لدينا مصفوفة تحويل فورييه، واننا قمنا بتبدلها فأصبح معامل DC في مركز المصفوفة، فإذا كانت المكونات منخفضة النطاق/ التردد حول المركز،فإننا نستطيع عمل فلتر منخفض العبور بضرب التحويل بالمصفوفة بطريقة نحافظ فيها على قيم المركز، ويتم تقليل أو حذف القيم البعيدة عن المركز، واحدى طرق فعل ذلك تتم بالضرب بمصفوفة العبور المنخفض المثالية ideal low-pass matrix وهي مصفوفة ثنائية m تعرف كالتالي:

$$m(x,y) = \begin{cases} 1 & \text{if } (x,y) \text{ is closer to the center than some value } D, \\ 0 & \text{if } (x,y) \text{ is further from the center than } D. \end{cases}$$

الدائرة c التي عرضناها سابقا، هي مصفوفة من هذا النوع فيها D=15، فيكون معكوس تحويل فورييه للمصفوفة الناتجة للضرب حسب العنصر بين F و m هي النتيجة المطلوبة:

$$\mathcal{F}^{-1}(F \cdot m).$$

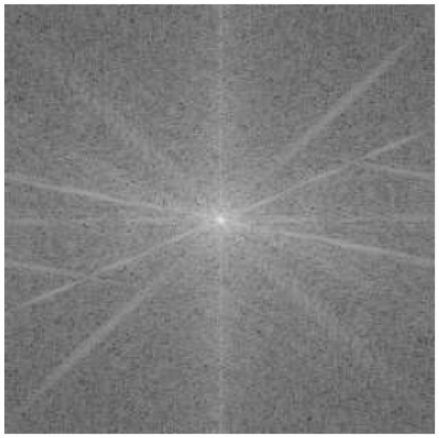
دعنا نرى ما يحدث إذا قمنا بتطبيق هذا الفلتر على الصورة، نحصل أولا الصورة ثم على التحويل الخاص بها:

▪

! ▪ _ _ $ __

```
>> cm=imread('cameraman.tif');
>> cf=fftshift(fft2(cm));
>> figure,fftshow(cf,'log')
```

الشكل التالي يوضح صورة رجل الكاميرا الرمادية التي نعرفها مع صورة التحويل في الكود السابق:



والآن لنقم بإجراء الفلتر منخفض العبور عليها بضرب مصفوفة التحويل بمصفوفة الدائرة، تذكر استخدام مؤثر الدوت dot النقطة (.)التي تستخدم في الماتلاب لضرب مصفوفتين عنصرا عنصرا (حسب العنصرelement-wise) أو استخدام الدالة times.

```
>> cfl=cf.*c;
>> figure,fftshow(cfl,'log');
```

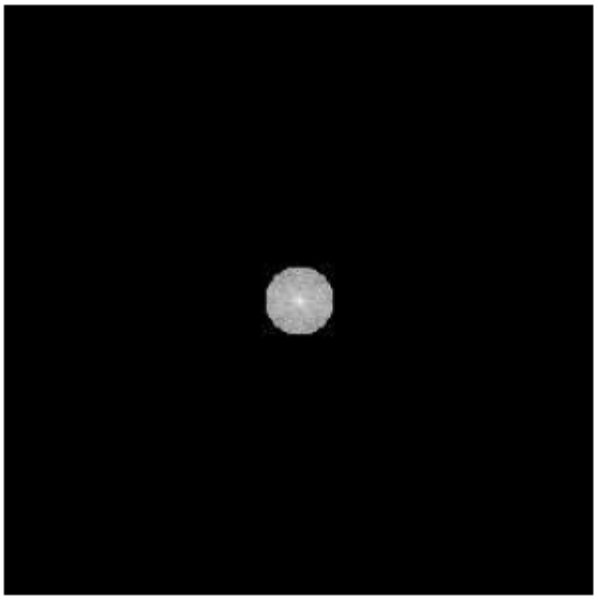
إذا ظهرت لك رسالة خطأ ترفض إجراء عملية الضرب ;cf.*c فقم بتعديل حجم المصفوفة c بالأمر:

```
>> c=c(:,1:256);
```

من الضروري الآن القيام بعكس التحويل باستخدام الدالة ifft2 وذلك لمعرفة تأثير الفلتر على الصورة، وذلك حسب الكود التالي:

```
>> cfli=ifft2(cfl);
>> figure,fftshow(cfli,'abs');
```

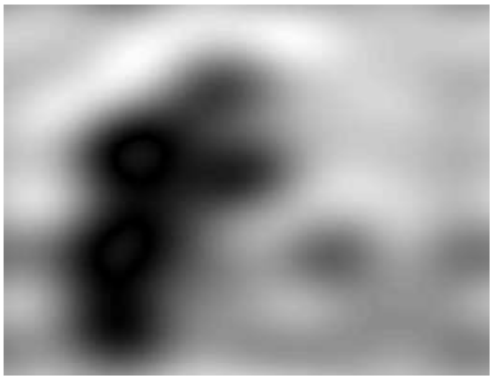
الشكل التالي يوضح الفلتر ثم عكسه أي الصورة المطبق عليها الفلتر:



لاحظ ان المصفوفة cfli التي يفترض انها مصفوفة اعداد حقيقية، إلا أننا لازلنا نستخدم الدالة التي صممناها مسبقا fftshow لعرضها، وهذا بسبب كون الدالة fft2 لا تقدم لنا نتائج عددية كاملة، لكنها تقدم نتائج عددية تقريبية جدا. لهذا نستخدم fftshow مع الوسيط 'abs' الذي يقوم بتقريب أي خطأ حصل للتحويل أو معكوسه.

لاحظ في الصورة الناتجة عن الفلتر وجود الحلقات ringing على حواف الصورة، هذا ناتج عن التقطعات الحاصلة من الدائرة، والذي ينتقل إلى الصورة.

وسوف نتوقع دائما انه كلما صغرت الصورة كلما زادت التلطخات في الصورة، وأنه كلما كبرت الدائرة كلما قلت اللطخات، الشكل التالي يوضح ذلك، باستخدام القطع ٥ والقطع ٣٠، لاحظ استمرار وجود الحلقات مع تغيير قيمة القطع.



(a) Cutoff of 5

(b) Cutoff of 30

الفلترة عالية العبور High pass filtering

بنفس اسلوب الفلترة منخفضة العبور إلا أننا سوف نعكس العملية، فسوف نحذف أو نتجاهل القيم حول المركز ونحافظ على القيم الأخرى، سنطبق ذلك بتعديل ثانوي على ما فعلناه سابقا، ولنبدأ بكود إنشاء الدائرة:

```
>> [x,y]=meshgrid(-128:127,-128:127);
>> z=sqrt(x.^2+y.^2);
>> c=(z>15);
```

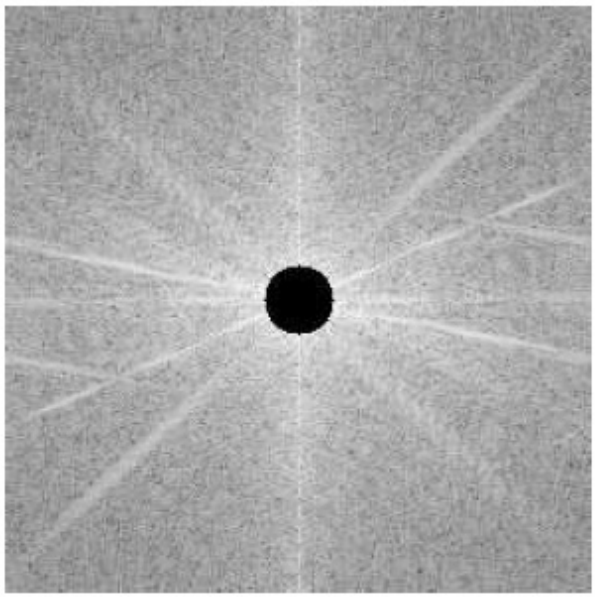
لاحظ الفارق بين هذه الدائرة وسابقتها، ثم لنقم بضرب الفلتر/الدائرة بالتحويل الذي حصلنا عليه سابقا.

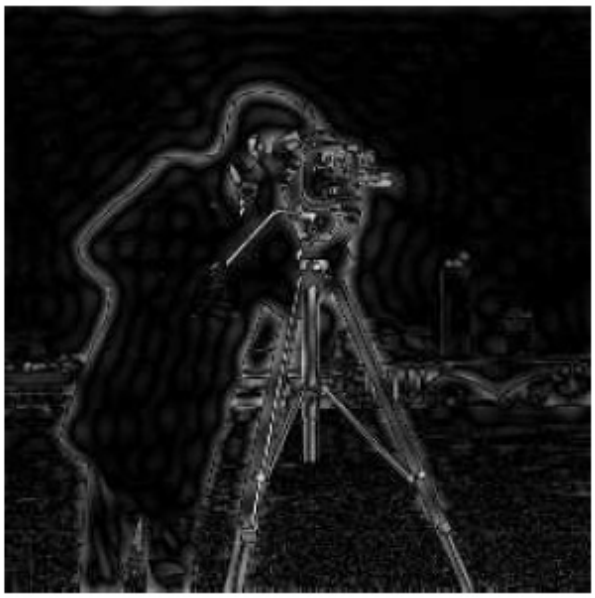
```
>> cfh=cf.*c;
>> figure,fftshow(cfh,'log')
```

ثم لنقم بعكس التحويل كما في الكود التالي:

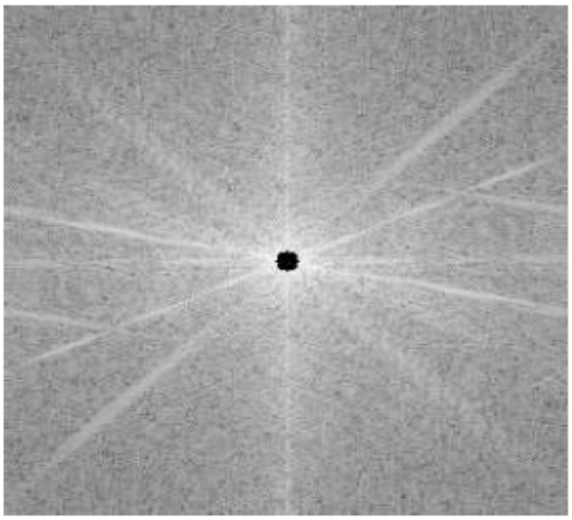
```
>> cfhi=ifft2(cfh);
>> figure,fftshow(cfhi,'abs')
```

ونتيجة ذلك موضحة بالشكل:



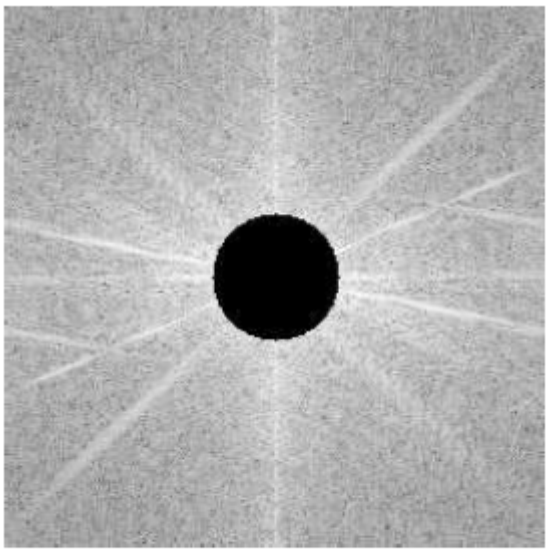


وكما حصل في الفلترة المنخفضة حجم الدائرة تأثر بالمعلومات المتاحة لعكس التحويل، وكذلك بالنسبة للنتائج النهائية مع اختلافها. الشكل التالي يوضح تطبيق الفلتر عالي العبور على مختلف قيم القطع (قيمة D في الدائرة) فإذا تم تكبير حجم القطع فإن كثير من المعلومات يتم حذفها من التحويل، وتترك فقط الترددات الأعلى، وهذا يمكن ملاحظته في الشكل التالين حيث تبقت الصورة فقط، وعكس ذلك بالنسبة للقطع cutoff الأقل. وهناك تفاصيل على المستوى الرمادي في الصورة النهائية لكن باقي الترددات المنخفضة قريبة من الصفر.



(a) Cutoff of 5

(b) The resulting image



(a) Cutoff of 30

(b) The resulting image

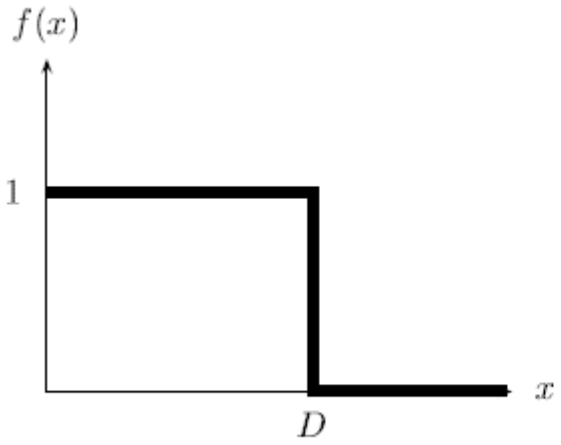
فلتر البتروورث Butterworth filtering
الفلترة المثالية الموضحة سابقا بنوعيها تقوم بقطع تحويل فوريه عند بعض مسافات من المركز، وهذا سهل جدا للتنفيذ، كما شاهدنا سابقا، لكن لديه بعض العيوب بعرض بعض العيوب أو التشويشات غير المطلوبةِ، كالحلقات التي تظهر في النتائج. وهناك خيار شائع هو استخدام فلاتر البتروورث.
وقبل الحديث عنها، سوف نعيد النظر إلى الفلاتر المثالية، وحيث انها متناظرة شعاعيا radially symmetric حول مركز التحويل، يمكن وصفها ببساطة باقسامها المتقاطعة، وهذا يعني، يمكننا وصف الفلتر كدالة توضح المسافة عن المركز، بالنسبة للفلتر المثالي منخفض العبور، هذه الدالة يمكن التعبير عنها بـ:

$$f(x) = \begin{cases} 1 & \text{if } x < D, \\ 0 & \text{if } x \geq D \end{cases}$$

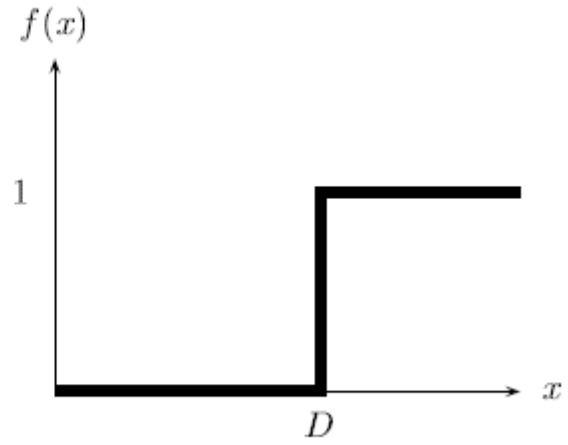
حيث أن D هي نصف قطر القطع. وبالنسبة للفلتر المثالي عالي العبور، فإن الدالة ستكون:

$$f(x) = \begin{cases} 1 & \text{if } x > D, \\ 0 & \text{if } x \leq D \end{cases}$$

هاتين الدالتين موضحتان في الاشكال البيانية التالية:



(a) Low pass



(b) High pass

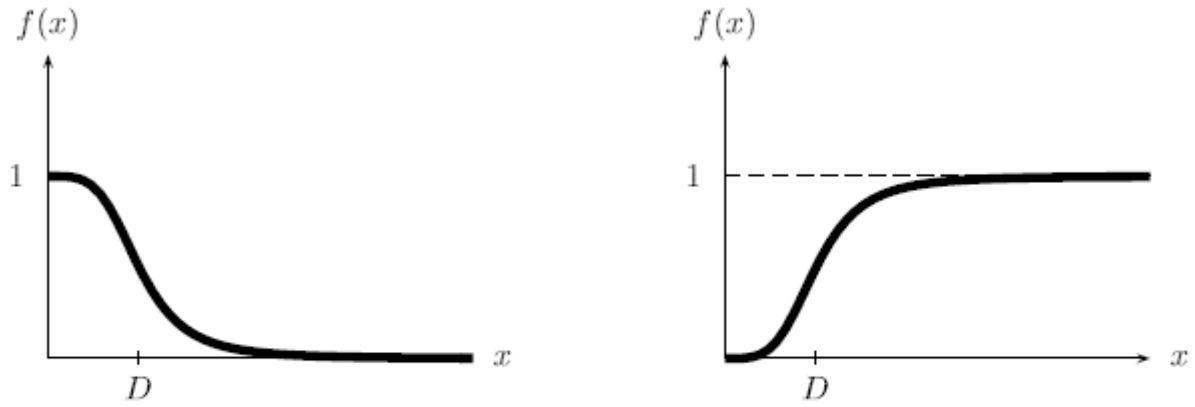
أما دوال البتروورث فستكون بالنسبة للفلتر منخفض العبور:

$$f(x) = \frac{1}{1 + (x/D)^{2n}}$$

وبالنسبة للفلتر عالي العبور:

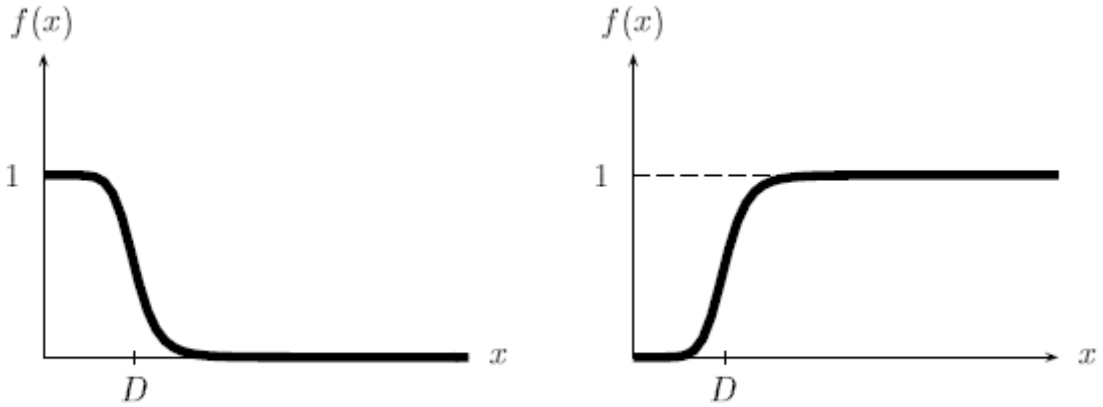
$$f(x) = \frac{1}{1 + (D/x)^{2n}}$$

حيث أنه وفي الحالتين فإن الوسيط n يسمى برتبة الفلتر filter order، وحجم n يتحكم بحدة sharpness القطع، هذه الدوال موضحة في الشكل التالي.



(a) Low pass (b) High pass

Figure 4.19: Butterworth filter functions with $n = 2$



(a) Low pass (b) High pass

وهذه الدوال سهلة التنفيذ في بيئة الماتلاب، وفيما يلي كود تطبيق البتروورث منخفض المرور حجم الفلتر ٢٥٦×٢٥٦، مع D=15 والرتبة n=2:

```
>> [x,y]=meshgrid(-128:217,-128:127));
>> bl=1./(1+((x.^2+y.^2)/15).^2);
```

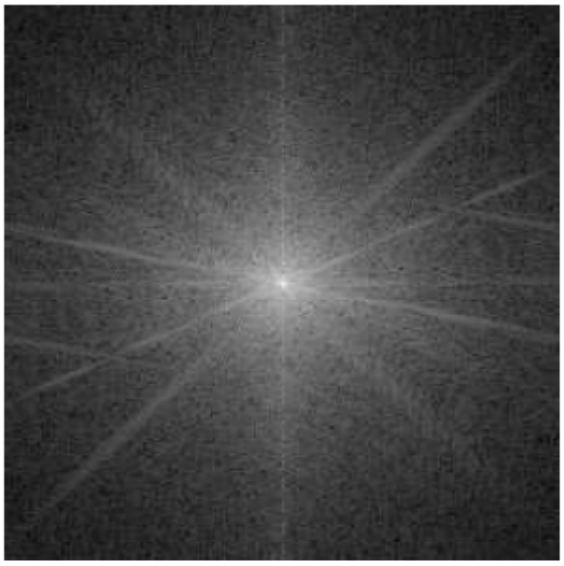
ولأن فلتر البتروورث عالي العبور يمكن أن ينتج بطرح منخفض العبور من ١، فإنه بالإمكان كتابة دوال عامة في الماتلاب لتطبيق فلاتر البتروورث بأحجام عامة، ودالة منخفض العبور موضحة كالتالي:

```
function out=lbutter(im,d,n)
% LBUTTER(IM,D,N) creates a low-pass Butterworth filter
% of the same size as image IM, with cutoff D, and order N
%
% Use:
% x=imread('cameraman.tif');
% l=lbutter(x,25,2);
%
height=size(im,1);
width=size(im,2);
[x,y]=meshgrid(-floor(width/2):floor((width-1)/2),-floor(height/2): ...
floor((height-1)/2));
out=1./(1+(sqrt(2)-1)*((x.^2+y.^2)/d^2).^n);
end;
```

وعلى هذا فإننا سوف نستدعيها في الكود التالي ولنطبقها هنا على التحويل الناتج من صورة رجل الكامير التي خزنها سابقا في المتغير c:

```
>> bl=lbutter(c,15,1);
>> cfbl=cf.*bl;
>> figure,fftshow(cfbl,'log')
```

ونتيجة ذلك تظهر في الشكل التالي:



لاحظ عدم وجود قطوع حادة في هذه الصورة no sharp cutoff ، وأن الجزء الخارجي للتحويل لا يساوي الصفر، رغم أن يعتبر معتما.

وتنفيذ التحويل العكسي وعرضه يعطي الصورة أعلاه يمين، وهي صورة ملطخة لكن الحلقات التي شاهدناها سابقا غائبة تماما، قارن بين التحويل بعد تطبيق الفلتر بتروورث مع التحويل الاصلي الذي رأيناه سابقا.

يقوم هذا الفلتر بطرد القيم بعيدا عن المركز، حتى لو لم تصبح صفرية بشكل غير متوقع، وكما قمنا بتطبيق الفلتر منخفض العبور فسوف نقوم بالفلتر عالي العبور، أولا بإنشاء الفلتر في الدالة التالية، ثم تطبيق الدالة على تحول الصورة.

الدالة hbutter

```
function out=hbutter(im,d,n)
% HBUTTER(IM,D,N) creates a high-pass Butterworth filter
% of the same size as image IM, with cutoff D, and order N
%%
Use:
% x=imread('cameraman.tif');
% l=hbutter(x,25,2);
%
out=1-lbutter(im,d,n);
```

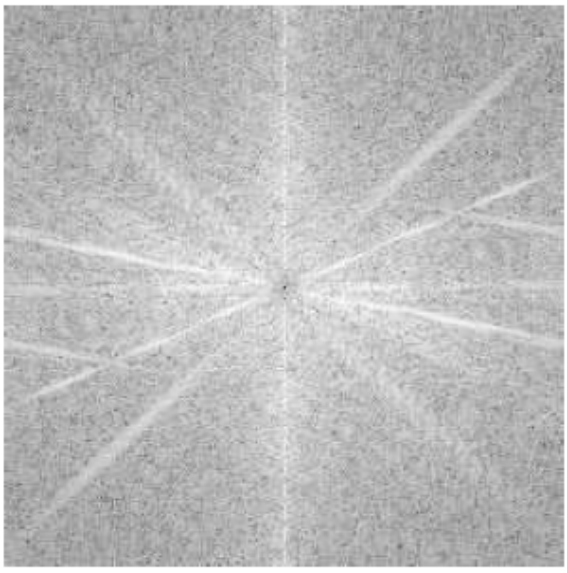
استدعاء الدالة:

```
>> bh=hbutter(cm,15,1);
>> cfbh=cf.*bh;
>> figure,fftshow(cfbh,'log')
```

ثم العكس وعرض النتائج:

```
>> cfbhi=ifft2(cfbh);
>> figure,fftshow(cfbhi,'abs')
```

الصور موضحة بالشكل التالي:



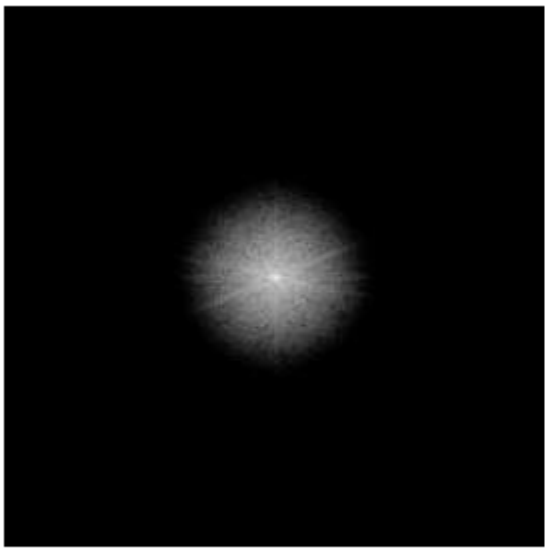
فلتر **جاوسيان** Gaussian filtering
لقد طبقنا فلتر جاوسيان في الفصل الثالث، ورأينا كيف انه يمكن ان يستخدم للفلترة منخفضة العبور، مع أنه من الممكن أن يستخدم في النطاقات المتكررة frequency domain، كما هو الحال مع الفلترات المثالية وفلترات بتروورث، والتنفيذ سيكون سهلا: قم بإنشاء فلتر جاوسيان، قم بضربه مع تحويل الصورة، ثم قم بعكس النتائج، طالما كان فلتر جاوسيان يملك تلك الميزة الرياضية المهمة وهي تحويل فوريه لفلتر جاوسيان ينتج عنها فلتر جاوسيان آخر، فسوف نحصل على نفس نتائج فلتر جاوسيان الفراغي الخطي linear Gaussian spatial filter.
ويعتبر فلتر جاوسيان اكثر الفلاتر التي درسناها نعومة smooth، والفلاتر المثالية هي الأقل نعومة، ويصنف فلتر بتروورث في الوسط.
نستطيع إنشاء فلتر جاوسيان باستخدام الدالة fspecial وتطبيقها على التحويل الخاص بنا:

```
>> g1=mat2gray(fspecial('gaussian',256,10));
>> cg1=cf.*g1;
>> fftshow(cg1,'log')
>> g2=mat2gray(fspecial('gaussian',256,30));
>> cg2=cf.*g2;
>> figure,fftshow(cg2,'log');
```

لاحظ استخدام الدالة mat2gray من اجل التحويل، وكذلك استخدام البارامتر 10 كوسيط ثالث للدالة fspecial، والذي يسمى بالمعامل سيجما، وله عدة فوائد.

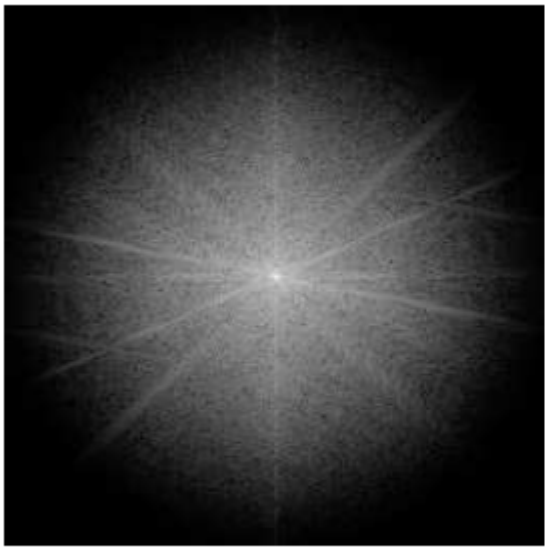
```
>> g=fspecial('gaussian',256,10);
>> format long, max(g(:)), format
ans =
0.00158757552679
```

لاحظ القيمة العظمى التي حصلنا عليها من الفلتر جوسيان حيث يقوم بتوليده للحصول على فلتر العبور المنخفض، وقد استخدمنا دالة التحويل mat2gray لكي لا نضطر لتكبير المعامل سيجما، حيث حصلنا الآن على اقرب قيمة للواحد يمكن الحصول عليها تلقائيا.
نتيجة التحويل السابق نجدها في الشكل التالي، لاحظ الفارق بين نتيجتي الفلتر حسب قيمة سيجما.:



(a) $\sigma = 10$

(b) Resulting image



(c) $\sigma = 30$

(d) Resulting image

والصور الناتجة تم الحصول عليها باستخدام الأوامر السابقة لعكس التحويل نفسها:

```
>> cgi1=ifft2(cg1);
>> cgi2=ifft2(cg2);
>> fftshow(cgi1,'abs');
>> fftshow(cgi2,'abs');
```

ونتائج هذه الكود هي الموضحة أعلاه.

ونستطيع الحصول على الفلتر عالي العبور عبر فلتر جاوسيان بطرح الفلتر المنخفض الناتج من العدد الصحيح ١، ثم نقوم بتطبيقه مباشرة ثم عكسه لإظهار النتائج كما يلي:

```
>> h1=1-g1;
>> h2=1-g2;
>> ch1=cf.*h1;
>> ch2=cf.*h2;
>> ch1i=ifft2(ch1);
>> chi1=ifft2(ch1);
>> chi2=ifft2(ch2);
>> fftshow(chi1,'abs');
>> figure,fftshow(chi2,'abs');
```

والشكل التالي يوضح نتائج فلتر جاوسيان عالي العبور على نفس الصورة:

(a) Using $\sigma = 10$

(b) Using $\sigma = 30$

Exercises

1. By hand, compute the DFT of each of the following sequences:

   (a) $[2, \quad 3, \quad 4, \quad 5]$ (b) $[2, \quad -3, \quad 4, \quad -5]$ (c) $[-9, \quad -8, \quad -7, \quad -6]$

   (d) $[-9, \quad 8, \quad -7, \quad 6]$

   Compare your answers with those given by MATLAB's `fft` function.

2. For each of the transforms you computed in the previous question, compute the inverse transform by hand.

3. By hand, verify the convolution theorem for each of the following pairs of sequences:

   (a) $[2, \quad 4, \quad 6, \quad 8]$ and $[-1, \quad 2 \quad -3, \quad 4]$ (b) $[4, \quad 5, \quad 6, \quad 7]$ and $[3, \quad 1 \quad 5, \quad -1]$

4. Using MATLAB, verify the convolution theorem for the following pairs of sequences:

   (a) $[2, \quad -3, \quad 5, \quad 6, \quad -2, \quad -1, \quad 3, \quad 7]$ and $[-1, \quad 5, \quad 6, \quad 4, \quad -3, \quad -5, \quad 1, \quad 2]$

   (b) $[7, \quad 6, \quad 5, \quad 4, \quad -4, \quad -5, \quad -6, \quad -7]$ and $[2, \quad 2, \quad -5, \quad -5, \quad 6, \quad 6, \quad -7, \quad -7]$

5. Consider the following matrix:

$$\begin{bmatrix} 4 & 5 & -9 & -5 \\ 3 & -7 & 1 & 2 \\ 6 & -1 & -6 & 1 \\ 3 & -1 & 7 & -5 \end{bmatrix}$$

Using MATLAB, calculate the DFT of each row. You can do this with the commands:

```
>> a=[4 5 -9 -5;3 -7 1 2;6 -1 -6 1;3 -1 7 -5];
>> a1=fft(a')'
```

(The `fft` function, applied to a matrix, produces the individual DFTs of all the columns. Here we transpose first, so that the rows become columns, then transpose back afterwards.)

Now use similar commands to calculate the DFT of each column of `a1`.

Compare the result with the output of the command `fft2(a)`.

6. Perform similar calculations as in the previous question with the matrices produced by the commands `magic(4)` and `hilb(6)`.

7. How do you think filtering with an averaging filter will effect the output of a Fourier transform?

Compare the DFTs of the cameraman image, and of the image after filtering with a 5×5 averaging filter.

Can you account for the result?

What happens if the averaging filter increases in size?

8. What is the result of two DFTs performed in succession? Apply a DFT to an image, and then another DFT to the result. Can you account for what you see?

9. Open up the image `engineer.tif`:

```
>> en=imread('engineer.tif');
```

Experiment with applying the Fourier transform to this image and the following filters:

(a) ideal filters (both low and high pass),

(b) Butterworth filters,

(c) Gaussian filters.

What is the smallest radius of a low pass ideal filter for which the face is still recognizable?

10. If you have access to a digital camera, or a scanner, produce a digital image of the face of somebody you know, and perform the same calculations as in the previous question.

An Introduction to Digital Image Processing with MATLAB

Alasdair McAndrew

School of Computer Science and Mathematics

Victoria University of Technology

مدخل إلى معالجة الصور مع الماتلاب ـ ترجمة فهد آل قاسم fhdalqasem@yahoo.com

المحتوى:

- استعادة الصورة Image Restoration
- تقسيم الصورة image segmentation
- معالجة الصور الملونة color image processing

استعادة الصورة Image Restoration

يهتم استعادة الصورة بازالة وتقليل التآكلات والتشوهات image degradation التي تلحق بالصورة أثناء الحصول عليها image acquisition. ومن هذه التآكلات:

١. التشوهات أو التشويشات image noise.
٢. التمويه اللطخ image bluring.

التشوشات هي اخطاء تحدث في قيم البكسلات pixel values محدثة تشوهات في اشارات الصورة image signals، بسبب عوامل خارجية.

ومن التشويشات الشهيرة:

١. تشويش الملح والفلفل
٢. تشويش البقع.
٣. تشويش جاوسيان.
٤. التشويش الدوري.

ويمكن محاكات الانواع الثلاثة الاولى في بيئة الماتلاب باستخدام الدالة imnoise التي تولد تشوشا للصورة كما في المثال التالي:

```
t=imread('cameraman.tif');
```

تشوش الملح والفلفل:

```
tsp=imnoise(t,'salt & pepper');
```

تشوش البقع:

```
tsk=imnoise(t,'speckle');
```

تشوش جاوسيان:

```
tga=imnoise(t,'gaussian');
```

قم بتطبيق الامثلة الثلاثة اعلاه على اكثر من صورة لمعرفة شكل التشوش.

أما التشوش الدوري فهو تشوش منتظم يمكن افتعاله باستخدام دالة دورية مثل دالة الجيب كما يلي:

```
>> s=size(t);
>> [x,y]=meshgrid(1:s(1),1:s(2));
>> p=sin(x/3+y/5)+1;
>> t_pn=(im2double(t)+p/2)/2;
```

كيفية التخلص من التشوش noise removal:

التشوشات الثلاثة الأولى يتم معالجتها باستخدام الفلاتر الحيزية spatial fitler اما التشوشات الدورية فيتم معالجتها باستخدام فلاتر النطاق المتكرر frequency domain filtering.

إزالة تشوش (الملح والفلفل salt & pepper)
كما لا بد انك لاحظت في تطبيق مثال محاكاة هذا التشوش فإنه ينتج لنا نقاطا او مكونات عالية التردد في الصورة مما يعني أننا نستخدم لإزالته فلترا منخفض العبور LPF.
وعلى هذا يمكننا تجربة الفلتر المنخفض المعروف average :

```
>> a3=fspecial('average');
>> t_sp_a3=filter2(a3,t_sp);
```

كما أنه بالإمكان تجربة الفلتر غير الخطي medfilt2 مباشرة:

```
>> t_sp_m3=medfilt2(t_sp);
```

يقوم هذا الفلتر بأخذ القيمة المتوسطة من جوار كل نقطة وهو افضل اداء من فلتر القيمة العظمي وفلتر القيمة الوسطى المستخدمة سابقا.
إزالة تشوش (**جاوسيان** Gaussian)
احد أشهر التشوشات الشائعة وهو يشبه التشوش الناتج عن التلفاز عندما لا يلتقط البث التلفزيوني، وفكرة ازالته تكمن في جمع مصفوفات عدة صورة متشابهة بها نفس التشوش ثم اخذ المتوسط الذي يفترض ان يكون افضل صورة.
الكود التالي يولد عشرة صور مشوشة بجاوسيان لكي نقوم بعد ذلك بازالة التشوش، لاحظ اننا لازلنا نستخدم الصورة أعلاه ويمكننا في المعمل تجربة صور مختلفة:

```
>> s=size(t);
>> t_ga10=zeros(s(1),s(2),10);
>> for i=1:10
t_ga10(:,:,i)=imnoise(t,'gaussian');
end;
```

والآن لنقم بمعالجة المصفوفة التي تحتوي على عشرة صورة متشابهة بها نفس التشوش كالتالي:

```
>> t_ga10_av=mean(t_ga10,3);
```

الرقم ثلاثة في الدالة mean يشير إلى ان المصفوفة ثلاثية البعد واننا نرغب في الحفاظ على القيمة الثالثة لها بدون اخذ المتوسط:
والآن ماذا لو اخذنا الرقم ١٠٠ بدلا من ١٠ الا يفترض ان تكون النتيجة افضل: جرب ذلك.
الفتر التكيفي adaptive filter
هو فلتر يقوم بتكييف نفسه حسب الصورة نفسها بمعنى ان دالة القناع يتم توليدها من الصورة التي يطبق عليها ، ومن امثلة ذلك فلتر واينر الذي يستخدم في ازالة تشوش جاوسيان.
استخدام فلتر واينر wiener كمثال على فلتر تكيفي لازالة تشوش جاوسيان:
نستخدم الدالة wiener2 لتوليد فلتر وينر وتطبيقه بنفس الوقت.

```
>> t2=imnoise(t,'gaussian',0,0.005);
>> imshow(t2)
>> t2w=wiener2(t2,[7,7]);
>> figure,imshow(t2w)
```

معالجة التشوش الدوري periodic noise removal
الصورة المشوشة اعلاه بالتشوش الدوري periodic noise يمكن معالجتها باستخدام النطاقات التكرارية أي تحويل فورييه، وكما اوضحنا اعلاه فإن التشوش الدوري يتم محاكاته باستخدام الدالة الجيبية اما في الواقع فهو يحدث بسبب أي خلل الكتروني في اجهزة التقاط الصورة او مسحها او تخزينها، ويمكن توليد التشوش الدوري كالتالي على افترض لدينا الصورة t اعلاه او أي صورة اخرى:

```
>> [x,y]=meshgrid(1:256,1:256);
>> p=1+sin(x+y/1.5);
>> tp=(double(t)/128+p)/4;
```

ويتم قبل معالجة الصورة tp المشوشة دوريا تحويلها تحويل فورييه:

```
>>tf = fftshift( fft2 (t_pn);
```

نقوم الآن بتوليد فلتر يسمى بفلتر رفض الحزمة band reject filtering حيث يتكون هذا الفلتر من مصفوفة واحدات توجد فيها حلقة اصفار بالمركز نتحكم نحن بقطر هذه الدائرة وليكن هنا ٤٩.

```
>> z=sqrt((x-129).^2+(y-129).^2);
>> br=(z<47 | z>=51);
```

ولنقوم بالمعالجة الآن:

```
>> tbr=tf.*br;
```

ثم لنقوم بعكس ناتج التحويل ولنراقب النتيجة.

هناك بالطبع عدة طرق مختلفة ومطولة لمعالجة التشوش الدوري الذي يعتبر من اهم المواضيع في معالجة الصور لكننا نكتفي بهذا المثال كنموذج نظرا لضيق الوقت.

تقسيم الصورة image segmentation

يقصد بالتقسيم تجزئة الصورة إلى على عدة مكونات أو إلى كائنات منفصلة واهم مواضيعه هي:

١) القيمة الهدف (العتبة) threshold
٢) اكتشاف حواف كائنات الصورة edge detection

البحث عن القيمة الهدف (العتبة) thresholding

القيمة الهدف او العتبة هي قيمة من قيم كثافات المستوى الرمادي في الصورة والتي تحدد الكائنات الأمامية foreground في الصورة من الكائنات الخلفية background.

مثلا عندما نقول أن تكون عندنا t بيانات صورة رجل الكاميرا واردنا اظهار الكائنات التي في الصورة فقط بدون الخلفية فإننا قد نكتب:

```
>> imshow( t>100);
```

الأمر السابق يظهر لنا الصورة بقيمتين ابيض واسود فقط .. اسود للخلفية وابيض لكائنات الصورة حسب الثريشولد المحدد وهو القيمة ١٠٠.

وبالطبع لكل صورة قيمة هدف (ثريشولد threshold) خاصة بها، هذه القيمة نحتاجها دائما عند تحويل الصورة الرمادية (تدرج الرمادي grayscale) إلى صورة ثنائية binary، وذلك باستخدام الدالة im2bw التي لها الشكل:

```
Bw= im2bw ( im , level);
```

حيث ان المُعامل level يقصد به القيمة الهدف التي تحدد فصل كائنات الصورة عن خلفيتها.

ويعبر رياضيا عن القيمة الهدف بالصيغة:

$$f(x,y) = \begin{cases} 0 & \text{if its grey level is} > T, \\ 1 & \text{if its grey level is} \le T. \end{cases}$$

وهناك دالة جاهزة تحاول اكتشاف القيمة الهدف (العتبة) باستخدام طرق رياضية احصائية هذه الدالة هي:

```
thr=graythresh(t);
```

حيث t هي الصورة المطلوبة.

هذه الدالة تنفذ افتراضيا في الدالة im2bw لاحتساب قيمة المُعامل level، لهذا نكتفي عادة بكتابة:

```
Bw= im2bw (t);
```

أو:

```
Bw= im2bw (t,graythresh(t));
```

ولأننا نتعامل حسابيا في الماتلاب دائما مع قيمة حقيقية double فإن قيمة الثريشولد تكون (0 < T < 1)، ويمكن تحويلها إلى النطاق [0,255] بضربها في ٢٥٥.

الثريشولد المزدوج :

احيانا نحتاج إلى وضع قيمتي هدف للصورة لتحويلها إلى النظام الثنائي نستخدم عندئذ T1 و T2 بحيث تتحقق الصيغة:

$$\text{A pixel becomes} \begin{cases} \text{white if its grey level is between } T_1 \text{ and } T_2, \\ \text{black if its grey level is otherwise.} \end{cases}$$

تطبيقات الثريشولج/ العتبة/ القيمة الهدف هي:

١. اكتشاف الاجسام الرئيسية في الصورة.
٢. اظهار التفاصيل المخفية في الصورة.
٣. حذف الخلفية ووضع خلفية بديلة.

وسنكتفي هنا بمثال يوضح التطبيق الأول.(لبقية التطبيقات يرجى العودة للكتاب).

```
>> r=imread('rice.tif');
>> imshow(r),figure,imshow(r>110);
```

حيث استخدمنا هنا T=110 لاكتشاف الاجسام في الصورة الموضحة.

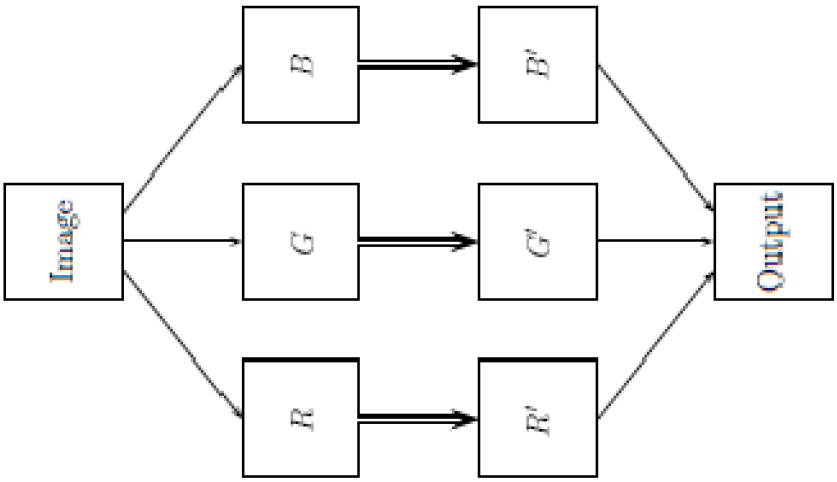
اكتشاف حواف كائنات الصورة edge detection

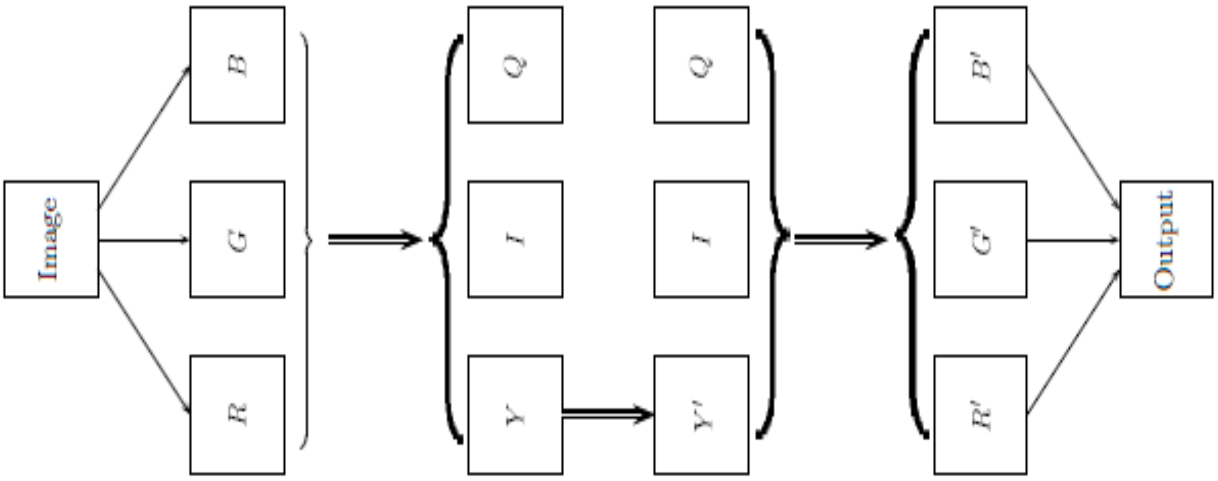
يتم استخدام الدالة edge لاكتشاف حواف كائنات الصورة، أي لرسم الصورة مع وضع حدود بيضاء على الاجسام التي تتكون منها الصورة.

```
>> edge (im, 'method', parameters);
```
كما في دالة الفلتر فإن الدالة edge تستخدم عدة طرق والطريقة الافتراضية هي 'sobel' وهي احدى الطرق الكلاسيكية لاكتشاف الحواف، كما ان لكل طريقة المُعاملات الخاصة بها.
المثال التالي يوضح تطبيق الدالة على الصورة r سابقا،
```
>> r=imread('rice.tif');
>> redg=edge(r);
>> imshow (redg);
```
ولاحظ معي ان الدالة edge تقوم بعرض الصورة المحددة افتراضيا أي لسنا بحاجة للأمر imshow، هكذا:
```
>> edge(r);
```
لاحظ ايضا أن الـ 'method' الافتراضية هنا هي طريقة سوبل لاكتشاف الحواف وهناك طرق أخرى تستطيع التعرف عليها بالبحث عن الدالة في بيئة الماتلاب نذكر منها:
فلتر بريويت :
```
>> redg=edge(r,'prewitt');
```
فلتر روبرت
```
>> redg=edge(r,'robert');
```
وهذه من الفلترات الكلاسيكية ايضا التي تسخدم تقنية المشتقة الأولى، اما الفلترين المهمين والمستخدمين عادة فهما:
فلتر لوج (لابلاسيان على جاوسيان)
```
>> redg=edge(r,'log');
```
وفلتر كاني
```
>> redg=edge(r,'canny');
```
تمرين:
يرجى تطبيق جميع هذه الفلاتر ومقارنة الفارق وايضا اضافة معاملاتها والتطبيق على صور مختلفة.

معالجة الصور الملونة color image processing

هناك عدة طرق لتمثيل اللون في معالجة الصورة الرقمية، وأشهر هذه الطرق هي RGB وهي الطريقة المعتمدة لعرض الصورة الحقيقية، إلا أنها للأسف لا تمثل الطريقة المثلى في المعالجة، فكثافة اللون الأكثر حضورا مثلا هي الخيار الذي يؤثر أكثر على الصورة، وعلى هذا فمن الصعب معرفة ذلك اللون رياضيا ثم استخدامه.
سوف نستعرض فيما يلي طريقتين أساسيتين لتمثيل الالوان من ضمن جملة كبيرة من الطرق المتبعة:
الطريقة الأولى HSV الطريقة **الثانية** YIQ
أسلوب HSV
هذا النظام يمثل اللون بثلاثة قيمة هي:hue ، saturation ، value.
حيث hue هي اللون الطيفي المعروف، و saturation نسبة الاشباع باللون الابيض، اما value فشدة اللمعان او درجة السطوع.
وهذه هي أفضل وأشهر الطرق المتبعة للتعامل مع الألوان، ويتم استخدام معادلات رياضية احصائية معروفة للتحويل من الأسلوب RBG إلى HSV، أما في الماتلاب فنستخدم الدالتين hsv2rbg أو rgb2hsv.
أسلوب YIQ
هو النظام المعتمد لدى هيئة ntsc الامريكية، ويتميز بفصل الكثافة intensity عن اللون color، حيث يتم وضع الكثافة في المتغير الأول Y ومعلومات الالوان تكون في المتغيرات I و Q.
وهذا هو النوع المستخدم في معالجة الصورة الملونة حيث يتم التعامل مع المتغير Y بدلا من التعامل مع الصورة كاملة.
ويتم التحويل من وإلى الاسلوب RGB باستخدام الدالتين: ntsc2rgb و rgb2ntsc كالتالي:
```
>> p=imread('peppers.png');
>> pyiq= rgb2ntsc (p);
```
حيث ان الصورة التي تحتوي على الكثافة Y هنا هي : pyiq(:,:,1)
```
imshow(puiq(:,:,1));
```
معالجة الصور الملونة
يتم معالجة الصورة الملونة RGB بعدة طرق اشهرها:
١. فصل كل لون على حده والتعامل معه ثم دمج الألوان مجددا.
٢. التحويل إلى صيغة YIQ ثم التعامل مع اللون Y فقط ثم اعادة التحويل.
الشكلين التاليين يوضحان الفكرتين السابقتين:





يرجى ملاحظة ان تعقيد الرسمة لا علاقة له بتعقيد العملية، فالطريقة الثانية اقل تعقيدا من الأولى (لماذا؟).

أمثلة العمليات على الصورة الملونة:

١. تحسين التباين contrast stretching:

ليكن لدينا الصورة :

```
p = imread (’peppers.png’ );
```

ولنقم الآن بتحويلها إلى صيغة YIQ:

```
py=rgb2ntsc(p);
```

وسوف نستخدم لتحسين التباين فقط الصيغة py(:,:,1)، كالتالي:

```
pyc=histeq(py(:,:,1));
```

ثم لنقم الآن بإعادة التحويل:

```
pq = ntsc2rgb (pyc);
```
وسوف نستخدم فيما يلي الطريقة الثانية لتحسين التباين:
```
pr=p(:,:,1);
pg=p(:,:,2);
pb=p(:,:,3);
```
وذلك بتطبيق التحسين على الصور الثلاث:
```
prq=histeq(pr);
pgq=histeq(pg);
pbq=histeq(pb);
```
ولنقم اخيرا بدمج الصور الثلاث:
```
pqc= cat (3, prq,pgq,pbq);
```
والآن لسنتعرض الصورتين ونقارن بينهما:
```
Imshow (p), figure, imshow(pq), figure, imshow(pqc);
```
٢. الفلترة الحيزية spatial filtering:

من أجل الفلاتر المنخفضة فلا فرق بين الطريقتين لكن من اجل الفلاتر العالية فيفضل دائما استخدام الطريقة الأولى (اسلوب YIQ).

ولنطبق الآن الفلتر المنخفض 'average' بالطريقة الثانية:
```
>> c=p;
>> a15=fspecial('average',15);
>> cr = filter2 (a15,c(:,:,1));
>> cg = filter2 (a15,c(:,:,2));
>> cb = filter2 (a15,c(:,:,3));
>> blur = cat (3,cr,cg,cb);
>> imshow (blur);
```
ثم لنطبق الفلتر unsharp كفلتر عالي العبور بالطريقة الأولى:
```
>> cn=rgb2ntsc(c);
>> a=fspecial('unsharp');
>> cn(:,:,1)=filter2(a,cn(:,:,1));
>> cu=ntsc2rgb(cn);
>> imshow(cu)
```
٣. إزالة التشويش noise removal

الكود التالي يوضح طريقة تطبيق محاكاة التشويش ثم طريقة ازالته:
```
>> tw=p;
>> tn=imnoise(tw,'salt & pepper');
>> imshow(tn);
>> figure,imshow(tn(:,:,1));
>> figure,imshow(tn(:,:,2));
>> figure,imshow(tn(:,:,3));
```
استخدام الفلتر غير الخطي medfilt2 لإزالة التشوش
```
>> trm=medfilt2(tn(:,:,1));
>> tgm=medfilt2(tn(:,:,2));
>> tbm=medfilt2(tn(:,:,3));
>> tm=cat(3,trm,tgm,tbm);
>> imshow(tm)
```
نفس ما سبق ولكن بالطريقة الأولى:
```
>>tnn=rgb2ntsc(tn);
>> tnn(:,:,1)=medfilt2(tnn(:,:,1));
>> tm2=ntsc2rgb(tnn);
>> imshow(tm2);
```

٤. تقسيم الصورة واكتشاف حواف كائنات الصورة

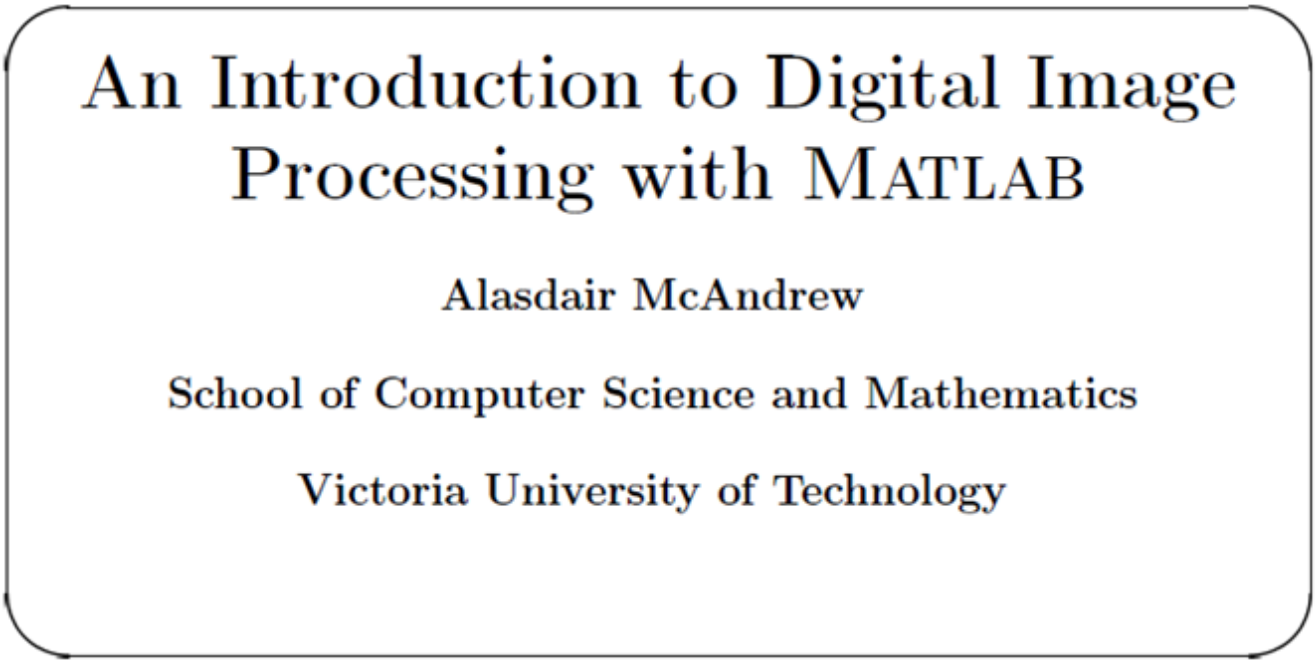
في حالة التقسيم يفضل عادة تحويل الصورة الملونة إلى تدرج الرمادي

```
>> f=p;
>> fg=rgb2gray(f);
>> fe1=edge(fg);
>> imshow(fe1);
```

كما انه من الممكن التعامل مع الالوان الثلاثة حسب الطريقة الثانية:

```
>> f1=edge(f(:,:,1));
>> f2=edge(f(:,:,2));
>> f3=edge(f(:,:,3));
>> fe2=f1 | f2 | f3;
>> figure,imshow(fe2)
```

==========



مدخل إلى معالجة الصور مع الماتلاب ـ ترجمة فهد أل قاسم fhdalqasem@yahoo.com

المحتوى: