# 一 图像变换技术为了解决什么

其目的是,图像经变换后处理问题会更简单.

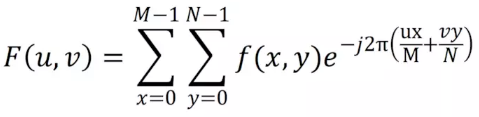
**图像变换有哪些?**

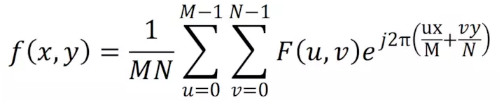
1. 频率变化, 为了减小数据相关性
   1. 离散傅立叶变换
   2. 离散余弦变换.
2. 几何变换, 应用在图像质量,尺寸,形状等.
   1. 图像位置变换
   2. 图像形状变换

# 二 离散傅立叶变换

## DFT的基本性质

基本公式:





特性:

1. 线性
2. 比例性质
3. 可分离性质
4. 平移性
5. 周期性
6. 共轭对称性

对于傅立叶平移性,matlab有如下函数

1. fftshift(X)
2. fftshift(X,dim)

### 实验, 傅立叶平移性:

|  |
| --- |
| clc**;**%清除工作区的程序  **% 傅立叶平移**   1. **需要把原图img转成gray图** 2. **先依据gray图生成一个傅立叶变换的频率表示.** 3. **在把这个频率表示做平移,一般都是平移到图像的w/2,h/2的地方.** 4. **这样可以方便分析图像傅立叶的分布情况.**   I**=**imread**(**'lena.bmp'**);**%读入图片  I**=**rgb2gray**(**I**);**%图片进行二值化处理  subplot**(**1**,**3**,**1**);**%建立1\*3的图像显示第一个图  imshow**(**I**);**%读出图像  title**(**'原始图象'**);**%写标题  J**=fft2(**I**);**%快速傅里叶变换  subplot**(**1**,**3**,**2**)**%建立1\*3的图像显示第二个图  imshow**(**J**);**  title**(**'FFT变换结果'**)**  subplot**(**1**,**3**,**3**)**  K**=fftshift(**J**);**%频率变换  imshow**(**K**);**  title**(**'零点平移'**);** |

|  |
| --- |
| function f = fft2(x, mrows, ncols)  **由图生成频率表示.**   1. **本质采用fftn和fft实现.根据传参选择的.** 2. **DFT公式.** 3. **输入的图x, 可以是N-Dim的array.** 4. **X是傅立叶变换后的. 公式如下.**     %FFT2 Two-dimensional discrete Fourier Transform.  % FFT2(X) returns the two-dimensional Fourier transform of matrix X.  % If X is a vector, the result will have the same orientation.  %  % FFT2(X,MROWS,NCOLS) pads matrix X with zeros to size MROWS-by-NCOLS  % before transforming.  %  % Class support for input X:  % float: double, single  %  % See also FFT, FFTN, FFTSHIFT, FFTW, IFFT, IFFT2, IFFTN.    % Copyright 1984-2010 The MathWorks, Inc.    if ismatrix(x)  if nargin==1  f = fftn(x);  else  f = fftn(x,[mrows ncols]);  end  else  if nargin==1  f = fft(fft(x,[],2),[],1);  else  f = fft(fft(x,ncols,2),mrows,1);  end  end |

何为fftshift

|  |
| --- |
| function x = fftshift(x,dim)  **这个就是固定的,转成图像的中心.**   1. **是固定到图像中心的.** 2. **采用循环移动的方式.** 3. **size(X) 返回X的shape.** 4. **Circshift的例子:**   ***% Examples:***  ***% A = [ 1 2 3; 4 5 6; 7 8 9];***  ***% B = circshift(A,1) % circularly shifts first dimension values down by 1.***  ***% B = 7 8 9***  ***% 1 2 3***  ***% 4 5 6***  ***% B = circshift(A,[1 -1]) % circularly shifts first dimension values***  ***% % down by 1 and second dimension left by 1.***  ***% B = 8 9 7***  ***% 2 3 1***  ***% 5 6 4***  %FFTSHIFT Shift zero-frequency component to center of spectrum.  % For vectors, FFTSHIFT(X) swaps the left and right halves of  % X. For matrices, FFTSHIFT(X) swaps the first and third  % quadrants and the second and fourth quadrants. For N-D  % arrays, FFTSHIFT(X) swaps "half-spaces" of X along each  % dimension.  %  % FFTSHIFT(X,DIM) applies the FFTSHIFT operation along the  % dimension DIM.  %  % FFTSHIFT is useful for visualizing the Fourier transform with  % the zero-frequency component in the middle of the spectrum.  %  % Class support for input X:  % float: double, single  %  % See also IFFTSHIFT, FFT, FFT2, FFTN, CIRCSHIFT.    % Copyright 1984-2013 The MathWorks, Inc.    if nargin > 1  if (~isscalar(dim)) || floor(dim) ~= dim || dim < 1  error(message('MATLAB:fftshift:DimNotPosInt'))  end  x = circshift(x,floor(size(x,dim)/2),dim);  else  x = circshift(x,floor(size(x)/2)); **%% size可以认为取出shape.lena.jpg的宽高是 512\*512的.这里size(x)后返回的是[512 512].**  end |

### 实验:傅立叶旋转性.

引入极坐标后,时域和频域的旋转有如下关系:



|  |
| --- |
| f=zeros(30,30);  f(5:24,13:17)=1;  F=**fft2**(f,256,256);  **%% 传入参数**   1. **表示按照256,256的[m n]来算傅立叶** 2. **Fft2返回也是[256 256]的格式大小.**   F2=**fftshift**(F); **%% 从矩阵原点平移到矩阵中心**  F3=log(1+**abs**(F2)); **%% 傅立叶变换后,会有复数,利用abs对其取模.**  subplot(2,2,1);  imshow(f);  title('原图');  subplot(2,2,2);  imshow(F3);  title('傅立叶’);  f1= imrotate(f,90);  F1=fft2(**f1**,256,256);  F21=fftshift(F1);  F31=log(1+abs(F21));  subplot(2,2,3);  imshow(f1);  title('原图旋转');  subplot(2,2,4);  imshow(F31);  title('傅立叶旋转'); |

## 快速傅立叶的应用

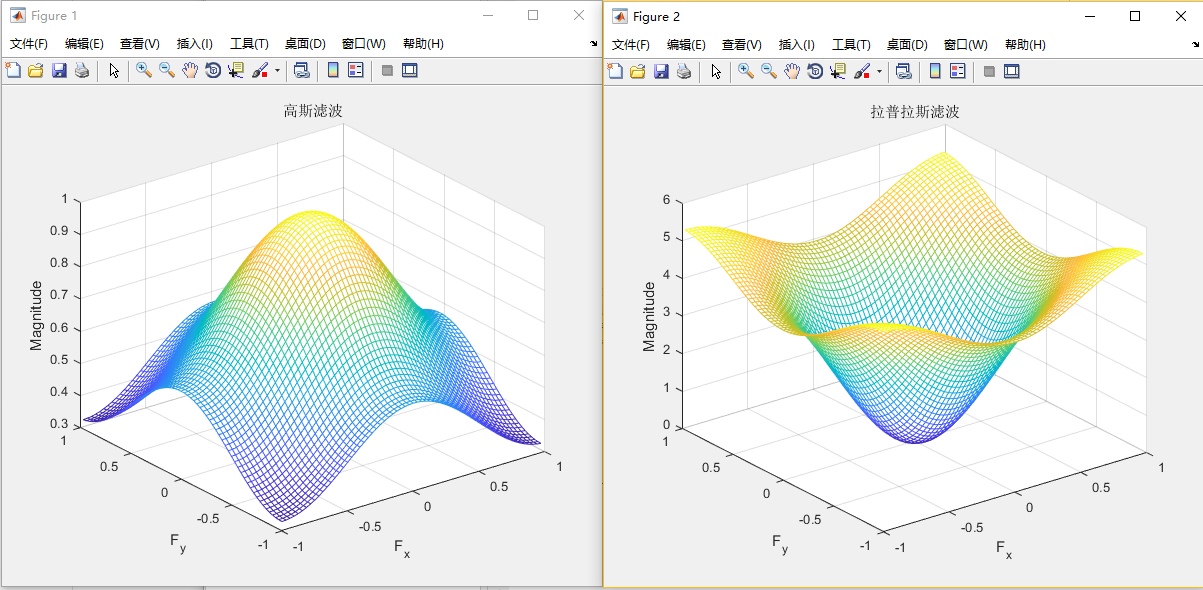
### 查看滤波器的频域响应

freqz2函数可以查看频域响应.

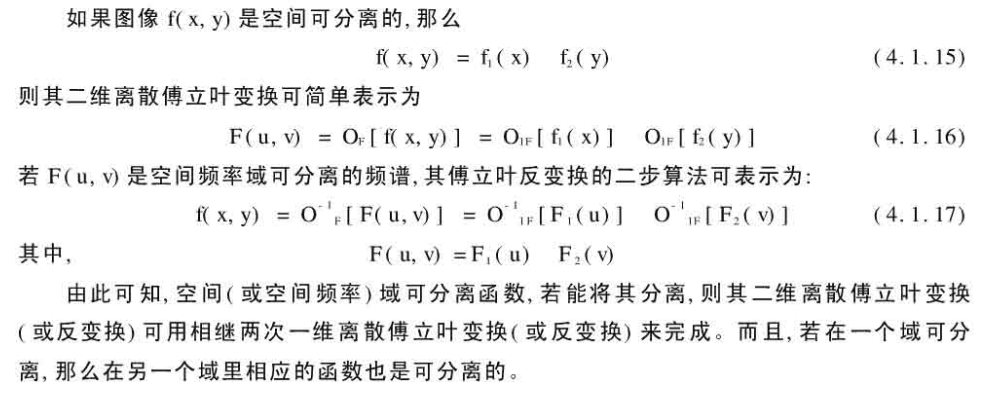
|  |
| --- |
| gaussian\_filter = fspecial('gaussian')  freqz2(gaussian\_filter)    laplacian\_filter = fspecial('laplacian')  freqz2(laplacian\_filter) |

可以看到

1.高斯低通滤波确实是低通的



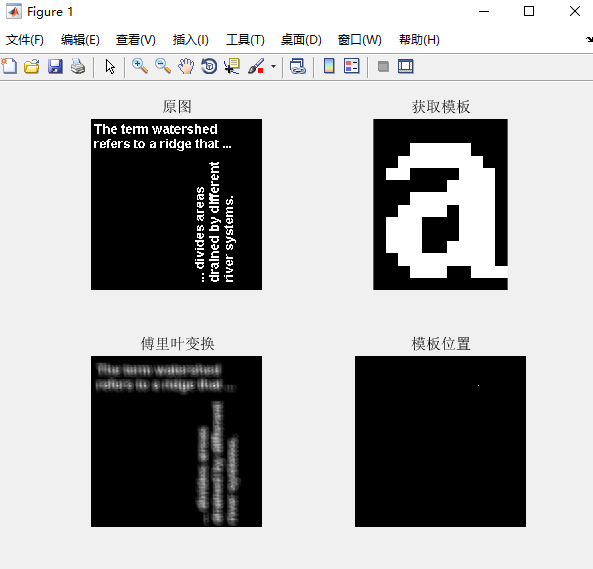
### 快速卷积



|  |
| --- |
| A **=** **[**1 2 3**;** 4 5 6**;** 7 8 9**];**  B **=** ones**(**3**);**  A**(**8**,**8**)** **=** 0**;**  B**(**8**,**8**)** **=** 0**;** % 补零  C **=** ifft2**(**fft2**(**A**).\***fft2**(**B**)); %% 利用的上面的4.1.17公式的性质.**   1. **单纯conv2d做卷积慢.** 2. **利用fft做乘法快.** 3. **利用ifft做傅立叶逆变换.**   C **=** C**(**1**:**5**,**1**:**5**);**  C **=** real**(**C**)** |

### 图像特征识别

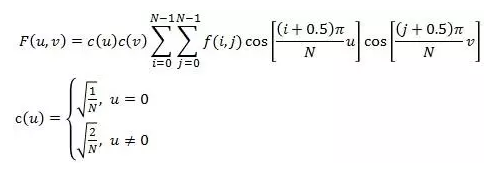
|  |
| --- |
| bw **=** imread**(**'text.png'**);**  subplot**(**221**);**  imshow**(**bw**);**  title**(**'原图'**);**  a **=** bw**(**32**:**45**,**88**:**98**);**  subplot**(**222**);**  imshow**(**a**);**  title**(**'获取模板'**);**  **%% 从原图中获取一个字符.把它作为待识别的模板**  **%% 然后把模板旋转90度.(做一个复杂化)**  **%% 将原图和模板做乘(卷积)**  **%% 然后取出阈值大于0.9的卷积值的元素位置.**  C **=** real**(**ifft2**(**fft2**(**bw**).\***fft2**(**rot90**(**a**,**2**),**256**,**256**)));**  subplot**(**223**);**  imshow**(**C**,[]);**  title**(**'傅里叶变换'**);**  c **=** max**(**C**(:));**  thresh **=** 0.9**\***c**;**  subplot**(**224**);**  imshow**(**C **>** thresh**);**  title**(**'模板位置'**);** |



离散余弦变换及应用

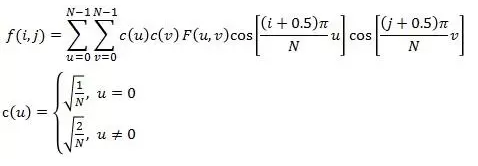
**何为离散余弦变换(DCT)?**

二维DCT变换是在一维的基础上再进行一次DCT变换，这个比较好理解，直接看公式：



这里我只讨论两个N相等的情况，也就是数据是方阵的形式，在实际应用中对不是方阵的数据都是先补齐再进行变换的。

**二维离散预先逆变换**



|  |
| --- |
| I1=imread('lena.jpg');  I1=rgb2gray(I1);  I1=imresize(I1,[300 300])  subplot(1,2,1)  imshow(I1);  title('Ô­Ê¼Í¼Ïñ');  I2=**dct2**(I1);% 二维离散余弦变换  subplot(1,2,2);  imshow(log(abs(I2)),[]); %%图像对数显示  title('ÀëÉ¢ÓàÏÒ±ä»»ºó'); |

|  |
| --- |
| function b=dct2(arg1,mrows,ncols)  %DCT2 2-D discrete cosine transform.  % B = DCT2(A) returns the discrete cosine transform of A.  % The matrix **B is the same size as A** and contains the  % discrete cosine transform coefficients.  % **另一种是用:**   1. **MN比图像尺寸大, 图像先pads(补零)** 2. **MN比图像尺寸小, 图像做crop.**   % B = DCT2(A,[M N]) or B = DCT2(A,M,N) **pads the matrix A with**  **% zeros to size M-by-N before transforming**. **If M or N is**  **% smaller than the corresponding dimension of A, DCT2 truncates**  **% A.**  %  % This transform can be inverted using **IDCT2**. **逆变换**  %  % Class Support  % -------------  % A can be numeric or logical. The returned matrix B is of  % class double.  %  % Example  % -------  % RGB = imread('autumn.tif');  % I = rgb2gray(RGB); **## 对灰度维度做操作.**  % J = dct2(I);  % imshow(log(abs(J)),[]), colormap(gca,jet), colorbar  %  % The commands below set values less than magnitude 10 in the  % DCT matrix to zero, then reconstruct the image using the  % inverse DCT function IDCT2.  %  % J(abs(J)<10) = 0;  % K = idct2(J);  % figure, imshow(I)  % figure, imshow(K,[0 255])  %  % See also **FFT2, IDCT2, IFFT2**.    % Copyright 1992-2016 The MathWorks, Inc.    % References:  % 1) A. K. Jain, "Fundamentals of Digital Image  % Processing", pp. 150-153.  % 2) Wallace, "The JPEG Still Picture Compression Standard",  % Communications of the ACM, April 1991.    [m, n] = size(arg1);  % Basic algorithm.  if (nargin == 1),  if (m > 1) && (n > 1),  b = **dct**(dct(arg1).').';  return;  else  mrows = m;  ncols = n;  end  end    % Padding for vector input. ## pad补零  a = arg1;  if nargin==2, ncols = mrows(2); mrows = mrows(1); end  mpad = mrows; npad = ncols;  if m == 1 && mpad > m, a(2, 1) = 0; m = 2; end  if n == 1 && npad > n, a(1, 2) = 0; n = 2; end  if m == 1, mpad = npad; npad = 1; end % For row vector.    % Transform.    b = **dct**(a, mpad);  if m > 1 && n > 1, b = dct(b.', npad).'; end |

#### 何为dct函数?

|  |
| --- |
| function b=dct(a,n)  %DCT Discrete cosine transform.  %  % Y = DCT(X) returns the **discrete cosine transform of X**. The  % vector Y is the same size as X and contains the discrete  % cosine transform coefficients. **%% X和Y的size相同**  %  % Y = DCT(X,N) pads or truncates the vector X to length N  % before transforming.  %  % If X is a matrix, the DCT operation is applied to each  % column. This transform can be inverted using IDCT.  %  % See also FFT,IFFT, and IDCT.    % Copyright 1993-2015 The MathWorks, Inc.    % References:  % 1) A. K. Jain, "Fundamentals of Digital Image  % Processing", pp. 150-153.  % 2) Wallace, "The JPEG Still Picture Compression Standard",  % Communications of the ACM, April 1991.    if ~isa(a, 'double')  a = double(a);  end    if min(size(a))==1  if size(a,2)>1  do\_trans = 1;  else  do\_trans = 0;  end  a = a(:);  else  do\_trans = 0;  end  if nargin==1,  n = size(a,1);  end  m = size(a,2);    % Pad or truncate a if necessary  if size(a,1)<n,  aa = zeros(n,m);  aa(1:size(a,1),:) = a;  else  aa = a(1:n,:);  end    if ((rem(n,2)==1) || (~isreal(a))) % odd case  % Form intermediate even-symmetric matrix.  y = zeros(2\*n,m);  y(1:n,:) = aa;  y(n+1:n+n,:) = flipud(aa);    % Perform FFT  yy = fft(y);    % Compute DCT coefficients %% 计算余弦变换系数  ww = (exp(-i\*(0:n-1)\*pi/(2\*n))/sqrt(2\*n)).';  ww(1) = ww(1) / sqrt(2);  b = ww(:,ones(1,m)).\*yy(1:n,:);    else % even case    % Re-order the elements of the columns of x  y = [ aa(1:2:n,:); aa(n:-2:2,:) ];    % Compute weights to multiply DFT coefficients  ww = 2\*exp(-i\*(0:n-1)'\*pi/(2\*n))/sqrt(2\*n);  ww(1) = ww(1) / sqrt(2);  W = ww(:,ones(1,m));    % Compute DCT using equation (5.92) in Jain  b = W .\* fft(y);  end    if isreal(a), b = real(b); end  if do\_trans, b = b.'; end |

#### Idct2压缩重构

|  |
| --- |
| I1=imread('pears.png');  I1=rgb2gray(I1);%灰度  subplot(1,2,1)  imshow(I1);  title('原图');  I2=dct2(I1);%先dct2做压缩  I2(abs(I2) < 10) = 0;  K = idct2(I2); % 逆变换,重构.  subplot(1,2,2);  imshow(K, [0 255]);  title('压缩重构图'); |

|  |
| --- |
| function a = idct2(arg1,mrows,ncols)  %IDCT2 2-D inverse discrete cosine transform.  % B = IDCT2(A) returns the two-dimensional inverse discrete  % cosine transform of A.  %  % B = IDCT2(A,[M N]) or B = IDCT2(A,M,N) pads A with zeros (or  % truncates A) to create a matrix of size M-by-N before  % transforming.  %  % For any A, IDCT2(DCT2(A)) equals A to within roundoff error.  %  % The discrete cosine transform is often used for image  % compression applications.  %  % Class Support  % -------------  % The input matrix A can be of class double or of any  % numeric class. The output matrix B is of class double.  %  % Example  % -------  % RGB = imread('autumn.tif');  % I = rgb2gray(RGB);  % J = dct2(I);  % imshow(log(abs(J)),[]), colormap(gca,jet), colorbar  %  % The commands below set values less than magnitude 10 in the  % DCT matrix to zero, then reconstruct the image using the  % inverse DCT function IDCT2.  %  % J(abs(J)<10) = 0;  % K = idct2(J);  % figure, imshow(I)  % figure, imshow(K,[0 255])  %  % See also DCT2, DCTMTX, FFT2, IFFT2.    % Copyright 1992-2016 The MathWorks, Inc.    % References:  % 1) A. K. Jain, "Fundamentals of Digital Image  % Processing", pp. 150-153.  % 2) Wallace, "The JPEG Still Picture Compression Standard",  % Communications of the ACM, April 1991.    [m, n] = size(arg1);  % Basic algorithm.  if (nargin == 1),  if (m > 1) && (n > 1),  a = idct(idct(arg1).').';  return;  else  mrows = m;  ncols = n;  end  end    % Padding for vector input.    b = arg1;  if nargin==2,  ncols = mrows(2);  mrows = mrows(1);  end    mpad = mrows; npad = ncols;  if m == 1 && mpad > m, b(2, 1) = 0; m = 2; end  if n == 1 && npad > n, b(1, 2) = 0; n = 2; end  if m == 1, mpad = npad; npad = 1; end % For row vector.    % Transform.    a = idct(b, mpad);  if m > 1 && n > 1, a = **idct**(a.', npad).'; end |

Idct

|  |
| --- |
| function a = idct(b,n)  %% 利用Fundamentals Of Digital Image Processing的公式计算.   1. 先算ifft. 2. 算ifft系数 3. 转换成idct   %IDCT Inverse discrete cosine transform.  %  % X = IDCT(Y) inverts the DCT transform, returning the original  % vector if Y was obtained using Y = DCT(X).  %  % X = IDCT(Y,N) pads or truncates the vector Y to length N  % before transforming.  %  % If Y is a matrix, the IDCT operation is applied to each  % column.  %  % See also FFT,IFFT,DCT.    % Copyright 1993-2015 The MathWorks, Inc.    % References:  % 1) A. K. Jain, "Fundamentals of Digital Image  % Processing", pp. 150-153.  % 2) Wallace, "The JPEG Still Picture Compression Standard",  % Communications of the ACM, April 1991.    if ~isa(b, 'double')  b = double(b);  end    if min(size(b))==1  if size(b,2)>1  do\_trans = 1;  else  do\_trans = 0;  end  b = b(:);  else  do\_trans = 0;  end  if nargin==1,  n = size(b,1);  end  m = size(b,2);    % Pad or truncate b if necessary  if size(b,1)<n,  bb = zeros(n,m);  bb(1:size(b,1),:) = b;  else  bb = b(1:n,:);  end    if rem(n,2)==1 || ~isreal(b), % odd case  % Form intermediate even-symmetric matrix.  ww = sqrt(2\*n) \* exp(j\*(0:n-1)\*pi/(2\*n)).';  ww(1) = ww(1) \* sqrt(2);  W = ww(:,ones(1,m));  yy = zeros(2\*n,m);  yy(1:n,:) = W.\*bb;  yy(n+2:n+n,:) = -j\*W(2:n,:).\*flipud(bb(2:n,:));    y = ifft(yy);    % Extract inverse DCT  a = y(1:n,:);    else % even case  % Compute **precorrection factor**  ww = sqrt(2\*n) \* exp(j\*pi\*(0:n-1)/(2\*n)).';  ww(1) = ww(1)/sqrt(2);  W = ww(:,ones(1,m));    % **Compute x tilde using equation (5.93) in Jain**  y = **ifft**(W.\*bb);    % **Re-order** elements of each column according to equations (5.93) and  % (5.94) in Jain  a = zeros(n,m);  a(1:2:n,:) = y(1:n/2,:);  a(2:2:n,:) = y(n:-1:n/2+1,:);  end    if isreal(b), a = real(a); end  if do\_trans, a = a.'; end |