Practical Programming Tutorial of Two Dimensional Discrete Fourier Transform (DFT) Based on MATLAB® for Both 2D Signals and Images

Kornkamol Thakulsukanant* and Vorapoj Patanavijit
Assumption University, Suvarnabhumi campus

กรกมล ตระกูลสุขอนันต์* และวรพจน์ พัฒนวิจิตร
มหาวิทยาลัยอัสสัมชัญ วิทยาเขตสุวรรณภูมิฯ

Abstract

The two-dimensional (2-D) Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) represent mathematical models for 2-D signals (such as digital images and digital videos) in the frequency and spatial domains, respectively. Digital Image Processing (DIP) has been implemented globally over the past two decades. Thus, 2-D Discrete Fourier Transform (2-D DFT) is essential in terms of representing mathematical models and analyzing 2-D signals and systems. In light of its importance, this article presents a tutorial for 2-D DFT utilizing MATLAB® for both 2-D signals and images. The analysis of the discrete signals are based on both spatial and frequency domains. The theoretical basic of 2-D DFT is presented, followed by a tutorial based on synthetic and real examples using MATLAB®.

Keywords: 2-D Discrete Fourier Transform (DFT), 2-D Inverse Discrete Fourier Transform (IDFT), Digital Image Processing (DIP).

* ผู้ประสานงานหลัก (Corresponding Author)
e-mail: kthakulsukanant@yahoo.com

บทคัดย่อ

ผลการแปลงฟูเรียร์แบบวิยุตสองมิติและผลการแปลงผกผันฟูเรียร์แบบวิยุตสองมิติเป็นวิธีการคำนวณทางคณิตศาสตร์พื้นฐานสำหรับการสร้างแบบจำลองทางคณิตศาสตร์ของสัญญาณและระบบที่มีความไม่ต่อเนื่องเชิงเวลาแบบสองมิติเพื่อวิเคราะห์ทั้งในเชิงเวลาและความถี่ เนื่องจากการประยุกต์ใช้งานด้านการประมวลผลสัญญาณดิจิทัลแบบสองมิติ (อย่างเช่นสัญญาณภาพหรือสัญญาณวิดิทัศ) ในช่วงยี่สิบปีที่ผ่านมามีการเจริญเติบโตอย่างมาก ดังนั้นการวิเคราะห์โดยผลการแปลงฟูเรียร์แบบวิยุตสองมิติจึงกลายเป็นเทคนิคทางคณิตศาสตร์ที่มีประโยชน์อย่างมาก สำหรับการวิเคราะห์สัญญาณแบบสองมิติและระบบแบบสองมิติ บทความนี้จึงนำเสนอหลักการและแนวคิดเชิงคณิตศาสตร์ผลการแปลงฟูเรียร์แบบวิยุตสองมิติสำหรับสัญญาณสองมิติหรือระบบแบบสองมิติและยังนำเสนอตัวอย่างการคำนวณการแปลงฟูเรียร์โดยใช้โปรแกรม MATLAB® เพื่อให้ผู้อ่านสามารถวิเคราะห์สัญญาณที่มีความไม่ต่อเนื่องทั้งในเชิงเวลาและความถี่ บทความนี้จะนำเสนอทฤษฎีเกี่ยวกับการแปลงฟูเรียร์แบบวิยุตสองมิติแล้วจึงค่อยนำเสนอการวิเคราะห์ฟูเรียร์โดยใช้โปรแกรม MATLAB® กับตัวอย่างของสัญญาณหลายประเภทเพื่อให้ผู้อ่านมีความเข้าใจอย่างละเอียด

คำสำคัญ: ผลการแปลงฟูเรียร์แบบวิยุตสองมิติ, ผลการแปลงผกผันฟูเรียร์แบบวิยุตสองมิติ, การประมวลผลสัญญาณภาพ

Introduction

One-dimensional DFT (1-D DFT) and 2-D DFT have a great deal of similarity at the conceptual level. However, there are also considerable differences between these two types of signals, especially in terms of the amount of data involved in typical applications. Consequently, this impacts the computational efficiency of the signal processing algorithm for 2-D signals and images.

However, 2-D DFT and 2-D IDFT are essential approaches for converting between the spatial and frequency domains (and vice versa) of 2-D signal processing in many applications such as:

- ➢ Digital Image Processing (DIP).
- ➢ Image Enhancement.
  - Image Smoothing such as Idea Lowpass Filter, Butterworth Lowpass Filter, and Gaussian Lowpass Filter.

- Image Sharpening such as Ideal Highpass Filter, Butterworth Highpass Filter, Gaussian Highpass Filter, Unsharp Masking, Highboost Filtering, and Homomorphic Filtering.

➢ Image Restoration and Reconstruction such as Inverse Filter, Wiener Filter, and Constrained Least Squares Filter.

➢ Image Compression Coding such as JPEG standard coding.

➢ Video Compression Coding such as JPEG, JPEG2000, MPEG1, and MPEG2.

➢ Image Recognition.

DFT also helps in speeding up the calculation of video motion estimation or registration for implementation in real-time applications. These processed images are converted to the Fourier domain. Thus, the time/spatial shifting property of Fourier theory is employed for determining the displacement between each processed images. Therefore this estimation algorithm is usually called the phase-based motion estimation or registration (Patanavijit, 2011).

During the last decade, one of the attractive research topics in the digital image process (DIP) has been compressive sensing (CS) (Ha, Lee and Patanavijit, 2014). The sampling theory is based on the fact that the whole information/signal can usually be reconstructed using only some significant parts of information, instead of using several parts of information (for reconstruction using classical Nyquist Sampling Theory) if these significant parts of information distribute in the proper domain such as the Fourier domain by using 2-D DFT/IDFT (Ha & Patanavijit, 2010-1; Ha & Patanavijit, 2010-2; Ha, Lee and Patanavijit, 2012; Patanavijit & Ha, 2013-1; Patanavijit & Ha, 2013-2) or the Wavelet domain by using two dimensional Discrete Wavelet Transform (2-D DWT) and two dimensional Inverse Discrete Wavelet Transform (2-D IDWT) (Ha & Patanavijit, 2011; Sermwuthisarn, Gansawat, Patanavijit & Auethavekiat, 2012; Sermwuthisarn, Auethavekiat, Gansawat & Patanavijit, 2012).

In this article, DFT is emphasized to represent the frequency domain of discrete-time signals in 2-D (Castleman, 1996; Gonzalez & Woods, 2010; Lim, 1990). Moreover, the article provides 2-D DFT implementation rather than examining its theoretical basis. Therefore, several examples on 2-D signals based on synthetic and real cases are provided. The detailed concept of 1-D DFT and 1-D IDFT have already been presented in (Thakulsukanant & Patanavijit, accepted to SDURJ 8 (2), 2015).

The article is arranged as follows: section 2 presents DFT for 2-D signals, including its theoretical basis, and provides various examples of synthetic and real cases, and section 3 provides a conclusion.

2-D Discrete Fourier Transform (DFT)

1. Theory

A short theoretical discussion of 2-D signals is given in this section. 2-D signals also employ Fast Fourier Transform (FFT) (Duhamel & Vetterli, 1999) to determine the DFT. The details for DFT of 2-D signals can be found in (Castleman, 1996; Gonzalez & Woods, 2010; Lim, 1990). The mathematical representation of DFT and IDFT for 2-D signals are provided below

$$G(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)e^{-j2\pi(ux/M+vy/N)} \quad (1)$$

$$g(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \quad (2)$$

where *g (x, y)* is a digital image of size *M×N,* and *G (u,v)* is its corresponding spectrum. The discrete frequency variables of *u* and *v* have to be evaluated in the ranges of *u* = 0, 1, 2,..., *M*-1 and *v* = 0, 1, 2,..., *N*-1. The discrete time variables are *x* and *y*, which must be computed in the same ranges as *u* and *v*.

2-D DFT in this paper uses a function of *fft2*, which is a MATLAB® function and a fast algorithm. This paper does not intend to present the fast algorithm of *fft2*. Instead, the result of employing this function is demonstrated.

2. Synthetic Cases for Small Sample Number (n)

Several examples of synthetic cases for small number (n) of 2-D signals are demonstrated in this section.

Example 1: $g1(x,y)=\begin{bmatrix}1 & 0\\ 1 & 0\end{bmatrix}$

Figure 1 illustrates a program to generate this g1*(x, y)* signal of size 2×2. This spatial domain signal is created by employing the functions of *ones*( ) and *zeros*( ). A 2-D DFT of this signal is evaluated utilizing a function of *fft*2( ). The result of G1*(u, v)* is shown below:

$$G1(u,v)=\begin{bmatrix}2 & 2\\ 0 & 0\end{bmatrix}$$

```
>> g1 = [ones(1,1) zeros(1,1); ones(1,1) zeros(1,1)]; % Create g1(x,y) of size 2×2 matrix
>> G1 = fft2(g1);
```

Figure 1 MATLAB® program for g1(x,y).

Example 2: $g2(x,y)=\begin{bmatrix}1 & 0 & 0\\ 1 & 1 & 0\end{bmatrix}$

The program demonstrated in Figure 2 is employed to create *g2(x, y)* signal of size 3×2. Using the same function as mentioned above a 2-D DFT of *g2(x, y), G2(u, v),* is given by

$$G2(u,v)=\begin{bmatrix}3 & 1.5-0.866j & 1.5+0.866j\\ -1 & 0.5+0.866j & 0.5-0.866j\end{bmatrix}$$

```
>> g2 = [ones(1,1) zeros(1,2); ones(1,2) zeros(1,1)]; % Create g2(x,y) of size 3×2 matrix
>> G2 = fft2(g2);
```

Figure 2 MATLAB® program for g2(x, y).

Example 3: $g3(x,y)=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}$

A matrix of *g3(x, y)* with a dimension of 3×3 is used in this example. The program shown in Figure 3 reveals how to create *g3(x, y)* signal. The frequency domain of *g3(x, y), G3(u ,v),* is shown below:

$$G3(u,v)=\begin{bmatrix}1 & -0.5-0.866j & -0.5+0.866j\\ -0.5-0.866j & -0.5+0.866j & 1\\ -0.5+0.866j & 1 & -0.5-0.866j\end{bmatrix}$$

```
>> g3 = [zeros(1,3); zeros(1,1) ones(1,1) zeros(1,1); zeros(1,3)];
% Create g3(x,y) of size 3×3 matrix
>> G3 = fft2(g3);
```

Figure 3 MATLAB® program for g3(x, y).

Example 4: $g4(x,y)=\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 0 & 1 & 1\\ 1 & 1 & 0 & 0\\ 1 & 1 & 0 & 0\end{bmatrix}$

A signal *g4(x, y)* of size 4×4 is a representative of this example. Figure 4 provides a program to generate both spatial domain, *g4(x, y)*, and frequency domain, *G4(u, v)*, signals. The result of *G4(u, v)* is given below:

$$G4(x,y)=\begin{bmatrix}8 & 0 & 0 & 0\\ 0 & 4j & 0 & -4\\ 0 & 0 & 0 & 0\\ 0 & -4 & 0 & -4j\end{bmatrix}$$

```
>> g4 = [zeros(1,2) ones(1,2); zeros(1,2) ones(1,2); ones(1,2) zeros(1,2); ones(1,2)
zeros(1,2)];
% Create g4(x,y) of size 4×4 matrix
>> G4 = fft2(g4);
```

Figure 4 MATLAB® program for g4(x, y).

3. Synthetic Cases for Large Sample Number (n)

This section provides various examples of synthetic cases for a large sample number *(n)* of 2-D signals including all figures and MATLAB® programs.

Example 1: 2-D rectangular signal, *g5(x, y).*

Figure 5 (a) reveals a 2-D rectangular signal, *g5(x, y)*, of size 150×150 pixels. All values of the image's pixels are in the range of 0 -255, which corresponds to the darkest and the brightest areas, respectively. Thus, this signal is divided into 9 rectangles of size 50 ×50 pixels. In Figure 5 (a), a 2-D white rectangle of size 50×50 pixels is at the center of all black rectangles. Therefore, fifty rows and fifty columns of all pixels in this area are set to 255 (brightest). Consequently, this white rectangle appears at the center of these black rectangles. Outside the center area, all pixels are set to zero and thus, the darkest area is observed in this part.

Figure 5 (b) demonstrates a program for constructing this 2-D signal. *Zeros*( ) and *ones*( ) functions are used to create this 2-D rectangular signal. The function of *ones*(50,50) in the program of Figure 5 (b) corresponds to the white rectangle area in Figure 5 (a). All of the created *zeros*(50,50) functions correspond to all of the black areas in Figure 5 (a). A function of *imshow*( ) is used to view the created 2-D rectangle of *g5(x, y)*.

A three-dimensional (3-D) plot in the spatial domain for this rectangular signal is also provided in Figure 5 (c). The program shown in Figure 5 (d) is used to create this plot. The function of *meshgrid*( ) on the first line of the program is utilized to generate *X* and *Y* arrays (or reference co-ordinates), which is equal to the dimension of the image, for this 3-D plot. According to the second line of the program, the amplitude of this 2-D signal, *g5(x,y)*, is assigned to *Z*-axis (dimension). A 3-D plot is completed using a function of *mesh* ( ). All x-, y-, and z-labels are inserted by employing *xlabel*( ), *ylabel*( ), and *zlabel*( ) functions, respectively.

A 2-D magnitude spectrum (|*G5(u, v)*|) is provided in Figure 5 (e), which is plotted by using the program shown in Figure 5 (f). A 2-D DFT of *g5(x, y)* is determined using a function of *fft*2( ). The absolute and maximum values are achieved by employing the functions of *abs*( ) and *max*(*max*( )). All values of *G5(u, v)* are normalized by using two steps: first, dividing each absolute value by its maximum value, and second, multiplying the resulting value by 255. Figure 5 (e) on the left-hand side demonstrates the magnitude spectrum (|*G5(u, v)*|) after completing the normalizing step.

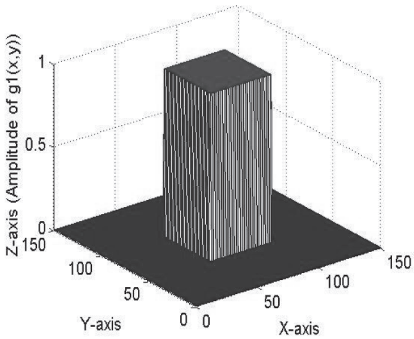
In general, the signals have a high magnitude spectrum (bright area) at low frequency ranges (around the origin). However, this obtained spectrum has high magnitude spectra around the four corners and the edges of the spectrum. Therefore, Figure 5 (e) (left-hand side) has to be reorganized (or requadrant). Figure 5 (e) right-hand side shows the same magnitude spectrum, however, with a reorganization of the 4-quadrants. In this figure, all bright areas are observed, which also indicates the high magnitude spectrum of this signal, at the origin (or at the center) of the spectrum. All dark areas corresponding to the low magnitude spectrum now appear at high frequency ranges.

Figure 5 (g) provides a 3-D plot of the same magnitude spectrum (|*G5(u,v)*| utilizing the program given in Figure 5 (h). All functions are the same as in the program mentioned in Figure 5 (d) to achieve this plot.

(a)

```
>> g5 = [zeros(50,50) zeros(50,50) zeros(50,
50);  zeros(50,50)  ones(50,  50)  zeros(50,
50);zeros(50,50)   zeros(50,   50)   zeros(50,
50) ];
% Create 2-D rectangular signal g5(x,y)
>>  imshow(g5)%  View  the  created  2-D  signal
of g5(x,y)
```

(b)

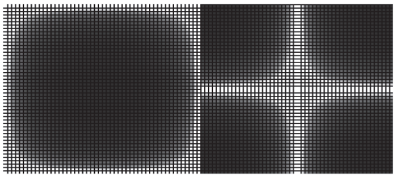


(c)

```
>> [X,Y] = meshgrid(1:1:150, 1:1:150);
% Create reference or co-ordinates of
X and Y arrays
>>  Z  =  g5;  %  All  values  of  g5  is
assigned to Z-axis
>>  mesh(X,Y,Z)%  Create  3-D  plot  of
g5(x,y)
>> xlabel('X-axis');% Insert x-label
>> ylabel('Y-axis');% Insert y-label
>>    zlabel('Z-axis    (Amplitude    of
g5(x,y))');% Insert z-label
```

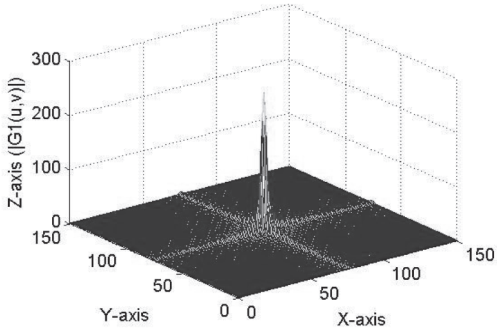
(d)

```
>>  G5  =  fft2(g5);%  Compute  2-D  DFT  of
g5(x,y) or G5(u,v)
>> absG5 = abs(G5); % Determine absolute
values of G5(u,v)
>>  G5max  =  max(max(absG5));  %Calculate
max. value of |G5(u,v)|
>> magG5 = (absG5/G5max)*255;
% Normalize all values of |G5(u,v)|
>> imshow(magG5)% View 2-D of |G5(u,v)|
>> ReQG5 = Requadrant(magG5);
%Reorganize all 4-quadrants of |G5(u,v)|
>> imshow(ReQG5)
```



(e)

(f)



(g)

```
>>  [X,Y]  =  meshgrid(1:1:150,
1:1:150);
>> Z = ReQG5;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis (|G5(u,v)|)');
```

(h)

Figure 5 (a) A 2-D rectangular signal, *g5(x, y)*, shown in 2-D view, (b) MATLAB® program for 2-D rectangular signal of Figure 5(a), (c) A 3-D plot for the 2-D rectangular signal *g5(x, y)*, (d) MATLAB® program for the 3-D plot of Figure 5(c), (e) 2-D plot of the magnitude spectrum, |*G5(u,v)*|, before requadrant (left) and after requadrant (right), (f) MATLAB® program for 2-D magnitude spectrum in Figure 5(e), (g) A 3-D plot of magnitude spectrum, |*G5(u, v)*| and (h) MATLAB® program for 3-D magnitude spectrum in Figure 5 (g).

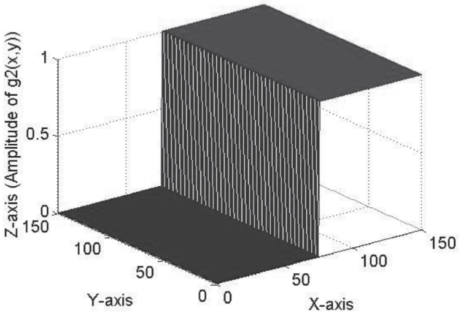
Example 2: 2-D unit step signal, *g6(x, y)*.

Figure 6 (a) reveals a 2-D unit step signal, *g6(x, y)*, of size 150×150 pixels. In this figure, half of this signal size is black colored (the area in this part is set to 0) and another half is white colored (the area in this part is set to 255). Thus, these two areas of the same size 75×150 pixels correspond to '0' and '1', respectively. Figure 6 (b) demonstrates a program for creating this *g6(x, y)* signal in a 2-D view. All functions, which have already been explained in previous examples can be applied here again. A 3-D view for this unit step signal is given in Figure 6 (c) using the program shown in Figure 6 (d). The 2-D magnitude spectrums (|*G6(u,v)*|), both before and after requadrants, are illustrated in Figure 6 (e). The program shown in Figure 6 (f) is utilized for generating the 2-D magnitude spectrum. By employing the program provided in Figure 6 (h), the magnitude spectrum in Figure 6 (g) is displayed.

(a)

```
>> g6 = [zeros(150,75) ones(150,75)];
% Create 2-D unit step signal g6(x,y)
>> imshow(g6)
```

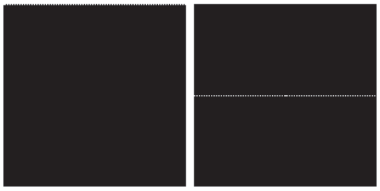
(b)



(c)

```
>>   [X,Y]   =   meshgrid(1:1:150,
1:1:150);
>> Z = g6;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>>zlabel('Z-axis  (Amplitude  of
g6(x,y))');
```

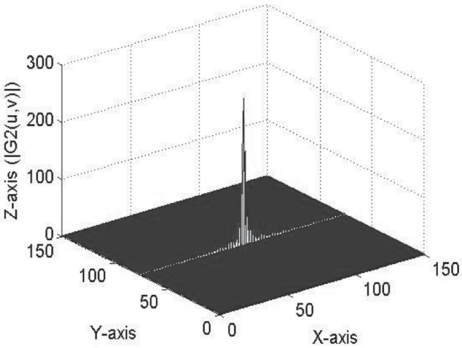
(d)



(e)

```
>> G6 = fft2(g6);
>> absG6 = abs(G6);
>> G6max = max(max(absG6));
>> magG6 = (absG6/G6max)*255;
>> imshow(magG6)
>> ReQG6 = Requadrant(magG6);
>> imshow(ReQG6)
```

(f)



(g)

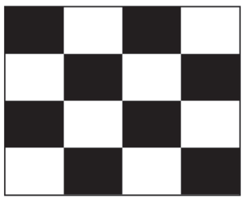
```
>>   [X,Y]   =   meshgrid(1:1:150,
1:1:150);
>> Z = ReQG6;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis (|G6(u,v)|)');
```

(h)

Figure 6 (a) A 2-D unit step signal, *g6(x, y)*, shown in 2-D view, (b) MATLAB® program for 2-D unit step signal of Figure 6(a), (c) A 3-D plot for 2-D unit step signal, *g6(x, y)*, (d) MATLAB® program for the 3-D plot of Figure 6 (c), (e) 2-D plot of magnitude spectrum, |*G6(u,v)*|, before requadrant (left) and after requadrant (right), (f) MATLAB® program for the 2-D magnitude spectrum in Figure 6 (e), (g) A 3-D plot of magnitude spectrum, |*G6(u,v)*| and (h) MATLAB® program for the 3-D magnitude spectrum in Figure 6 (g).

Example 3: 2-D checker board signal, *g7(x, y)*.

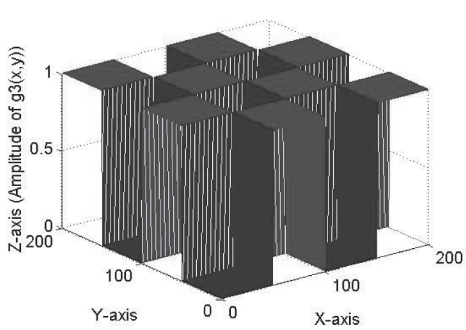
A 2-D checker board signal, *g7(x, y)* of size 200×200 pixels is demonstrated in Figure 7(a). This checker board consists of 8 black and 8 white rectangles of size 50×50 pixels. These rectangles have black and white colors arranged in an alternating pattern. This is done by setting all the pixels of the first rectangle (left most corner of Figure 7 (a)) to all '0s' and the second rectangle to all '1s'. This pattern is repeated until all black and white rectangles are completely arranged to form this 2-D checker board signal. Figure 7 (b) provides a program for generating this *g7(x, y)* signal. Figure 7 (c) illustrates a 3-D view for this signal employing a program given in Figure 7 (d). A 2-D magnitude spectrum (|*G7(u, v)*|) of this checker board signal and its corresponding program are provided in Figures 7 (e) and (f), respectively. This magnitude spectrum can be viewed in 3-D as shown in Figure 7 (g). The program shown in Figure 7 (h) indicates how to create this 3-D spectrum.



(a)

```
>>  g7  =  [zeros(50,50)  ones(50,50)
zeros(50,50)             ones(50,50);
ones(50,50)  zeros(50,50)  ones(50,50)
zeros(50,50);zeros(50,50)
ones(50,50)              zeros(50,50)
ones(50,50);              ones(50,50)
zeros(50,50)              ones(50,50)
zeros(50,50)];   % Created 2-D check
board signal g7(x,y)
>> imshow(g7)
```

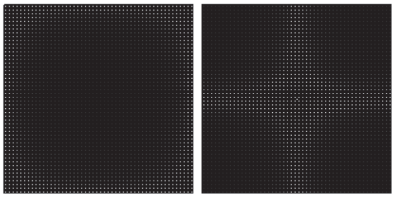
(b)



(c)

```
>> [X,Y] = meshgrid(1:1:200,
1:1:200);
>> Z = g7;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis (Amplitude of
g7(x,y))');
```

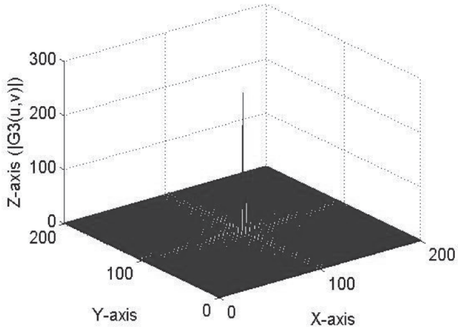
(d)



(e)

```
>> G7 = fft2(g7);
>> absG7 = abs(G7);
>> G7max = max(max(absG7));
>> magG7 = (absG7/G7max)*255;
>> imshow(magG7)
>> ReQG7 = Requadrant(magG7);
>> imshow(ReQG7)
```

(f)



(g)

```
>> [X,Y] = meshgrid(1:1:200,
1:1:200);
>> Z = ReQG7;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis (|G7(u,v)|)');
```

(h)

Figure 7 (a) A 2-D checker board signal, *g7(x, y)*, shown in 2-D view, (b) MATLAB® program for 2-D check board signal of Figure 7 (a), (c) A 3-D plot for the 2-D checker board signal, *g7(x, y)*, (d) MATLAB® program for 3-D plot of Figure 7 (c), (e) 2-D plot of magnitude spectrum, |*G7(u,v)*|, before requadrant (left) and after requadrant (right), (f) MATLAB® program for 2-D magnitude spectrum in Figure 7 (e), (g) A 3-D plot of magnitude spectrum, |*G7(u,v)*| and (h) MATLAB® program for a 3-D magnitude spectrum in Figure 7 (g).

Example 4: 2-D impulse signal, *g8(x, y)*.

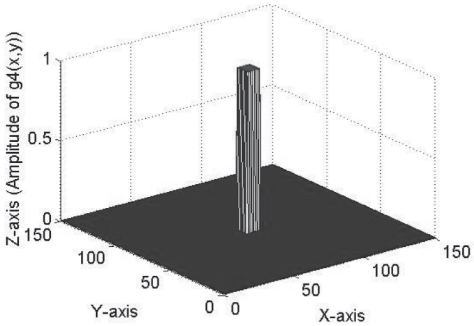
A 2-D impulse signal, *g8(x, y)*, of size 150×150 pixels is shown in Figure 8 (a). The size of the impulse is 10×10 pixels corresponding to a small white rectangular area at the center of this black rectangle. All values of the pixels in this center area are set to ‘1’ and thus it appears as a small white rectangular (impulse signal) as shown in Figure 8 (a). Other pixels apart from this area are set to ‘0’. Therefore, the area of this impulse signal is surrounded by black color. A program shown in Figure 8 (b) reveals how to generate this signal.

A 3-D view of *g8(x, y)* signal is illustrated in Figure 8 (c) using the program given in Figure 8 (d). Figure 8 (e) provides a 2-D view of the magnitude spectrum (|*G8(u, v)*|) and its corresponding program is given in Figure 8 (f). A 3-D view for this magnitude spectrum is also provided in Figure 8 (g) using the program shown in Figure 8 (h).

(a)

```
>> g8= [zeros(70,150); zeros(10, 70)
ones(10,10) zeros(10, 70);
zeros(70,150)];% Create 2-D impulse
signal g8(x,y)
>> imshow(g8)
```

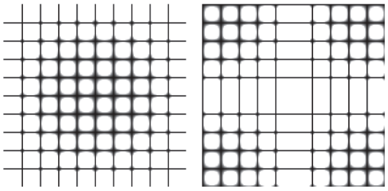
(b)



(c)

```
>> [X,Y] = meshgrid(1:1:150,
1:1:150);
>> Z = g8;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis (Amplitude of
g8(x,y))');
```

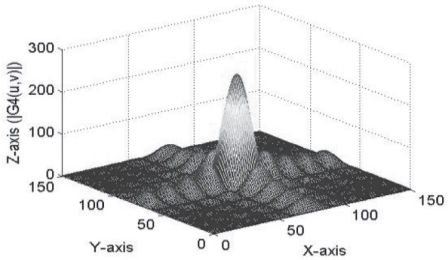
(d)



(e)

```
>> G8 = fft2(g8);
>> absG8 = abs(G8);
>> G8max = max(max(absG8));
>> magG8 = (absG8/G8max)*255;
>> imshow(magG8)
>> ReQG8 = Requadrant(magG8);
>> imshow(ReQG8)
```

(f)



(g)

```
>> [X,Y] = meshgrid(1:1:150,
1:1:150);
>> Z = ReQG8;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis
(|G8(u,v)|)');
```

(h)

Figure 8 (a) A 2-D impulse signal, *g8(x, y)*, shown in 2-D view, (b) MATLAB® program for the 2-D impulse signal of Figure 8 (a), (c) A 3-D plot for 2-D impulse signal, *g8(x, y)*, (d) MATLAB® program for the 3-D plot of Figure 8 (c), (e) A 2-D plot of magnitude spectrum, |*G8(u, v)*|, before requadrant (left) and after requadrant (right), (f) MATLAB® program for 2-D magnitude spectrum in Figure 8 (e), (g) A 3-D plot of magnitude spectrum, |*G8(u, v)*| and (h) MATLAB® program for a 3-D magnitude spectrum in Figure 8 (g).

4. Real Cases for 2-D Signals

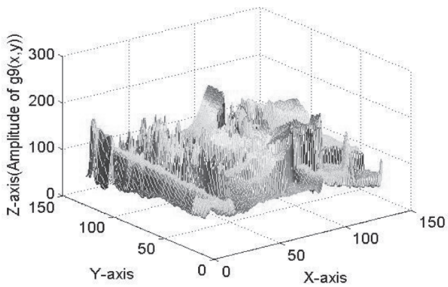
Three examples of real cases for 2-D signals are given in this section. All 2-D and 3-D views of all the signals and theirs corresponding MATLAB® programs are also provided.

Example 1: Lena standard gray image (128×128 pixels), *g9(x, y)*.

Figure 9 (a) reveals a Lena standard gray image, *g9(x, y)*, of size 128×128 pixels in a 2-D view. A spatial domain in a 3-D view of this image is shown in Figure 9 (b). The program shown in Figure 9 (c) is given for creating a 2-D and 3-D Lena image in the spatial domain of Figures 9 (a) and (b). All functions in the previous section are reapplied here.

From Figure 9 (c), an *imread* function on the first line of the program is used to read all data of this image. Note that the 3-D plot using *mesh*( ) function supports only double and single data types. Thus, all data apart from these two types are required to change the format by utilizing the function of *double(*or *single)*. In this case, a *double* function is selected to store a data of 8 bytes. Both 2-D and 3-D magnitude spectrums, |*G9(x, y)*|, of Lena image are illustrated in Figures 9 (d) and (f), respectively. A 2-D magnitude spectrum on the left-hand side of Figure 9 (d) is also required to requadrant as in previous examples. The programs for creating both figures are also given in Figures 9 (e) and (g), respectively.

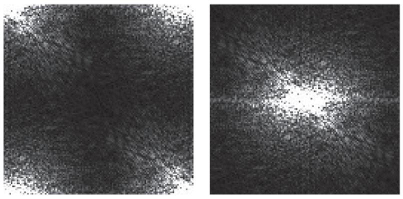
(a)



(b)

```
>> g9 = imread('Lena128.tif');
>> imshow(g9) % View 2-D Lena image
>> g9 = double(g9); % Change data
type to double
>> [X,Y] =
meshgrid(1:1:128,1:1:128);
>> Z = g9;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis(Amplitude of
g9(x,y))');
```

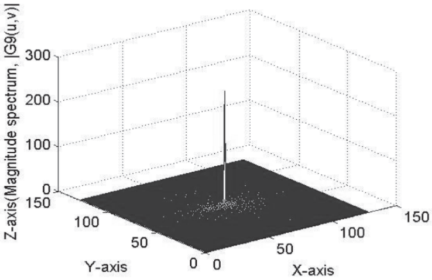
(c)



(d)

```
>> G9 = fft2(g9);
>> absG9 = abs(G9);
>> G9max = max(max(absG9));
>> magG9 = (absG9/G9max)*255;
>> imshow(magG9)
>> ReQG9 = Requadrant(magG9);
>> imshow(ReQG9)
```

(e)



(f)

```
>> [X,Y] =
meshgrid(1:1:128,1:1:128);
>> Z = ReQG9;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis(Magnitude
spectrum, |G9(u,v)|');
```

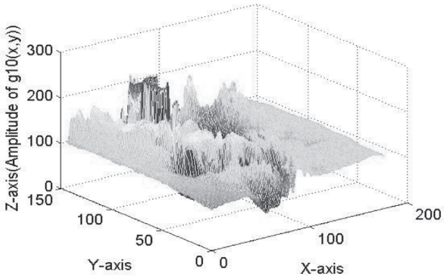
(g)

Figure 9 (a) Lena standard gray image, *g9(x, y)*, shown in 2-D view, (b) A 3-D plot for Lena image in the spatial domain, (c) MATLAB® program for the 3-D plot of Figure 9 (b), (d) 2-D plot for magnitude spectrum of Lena image, |*G5(u, v)*|, before requadrant (left) and after requadrant (right), (e) MATLAB® program for the 2-D magnitude spectrum in Figure 9 (d), (f) A 3-D plot for magnitude spectrum of Lena image, |*G9(u, v)*|, (g) MATLAB® program for the 3-D magnitude spectrum in Figure 9 (f).

Example 2: Susie image (40^{th} frame) (144×176 pixels), *g*10*(x, y)*.

A Susie image (40^{th} frame), *g*10*(x, y)*, of size 144×176 pixels is provided in Figure 10 (a). Figure 10 (b) provides a 3-D view of the spatial domain for this image. The program given in Figure 10 (c) is employed to create Figures 10 (a) and (b). The 2-D magnitude spectrum (|G10*(u, v)*|) for Susie image is provided in Figure 10 (d). A program for generating this spectrum is shown in Figure 10 (e). By utilizing the program provided in Figure 10 (g), a 3-D magnitude spectrum of Susie image can be plotted as shown in Figure 10 (f).

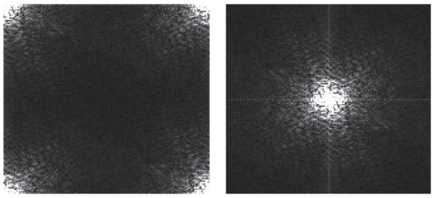
(a)



(b)

```
>> g10 =
imread('Susie_Frame_40.tif');
>> imshow(g10)
>> g10 = double(g10);
>> [X,Y] = meshgrid(1:1:176,1:1:144);
>> Z = g10;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis(Amplitude of
g10(x,y))');
```

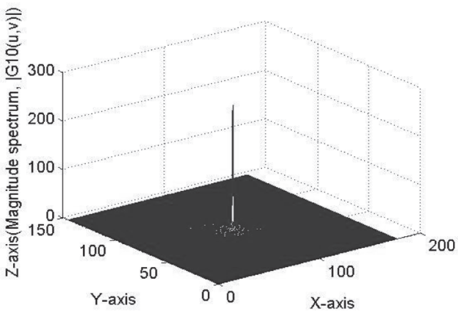
(c)



(d)

```
>> G10 = fft2(g10);
>> absG10 = abs(G10);
>> G10max = max(max(absG10));
>> magG10 = (absG10/G10max)*255;
>> imshow(magG10)
>> ReQG10 = Requadrant(magG10);
>> imshow(ReQG10)
```

(e)



(f)

```
>> [X,Y] =
meshgrid(1:1:176,1:1:144);
>> Z = ReQG10;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis(Magnitude
spectrum, |G10(u,v)|)');
```

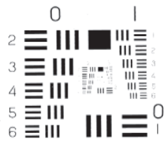
(g)

Figure 10 (a) Susie image, *g10(x, y)*, shown in 2-D view, (b) A 3-D plot for Susie image in the spatial domain, (c) MATLAB® program for the 3-D plot of Figure 10 (b), (d) 2-D plot for the magnitude spectrum of Susie image, |*G*10*(u, v)*|, before requadrant (left) and after requadrant (right), (e) MATLAB® program for 2-D magnitude spectrum in Figure 10 (d), (f) A 3-D plot for the magnitude spectrum of Susie image, |*G*10*(u, v)*| and (g) MATLAB® program for the 3-D magnitude spectrum in Figure 10 (f).

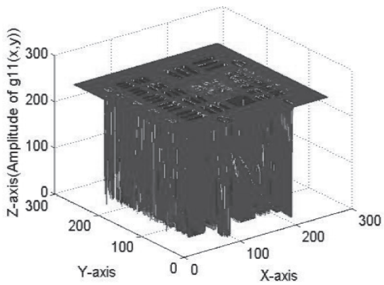
Example 3: Resolution chart (256×256 pixels), *g*11*(x, y)*.

Figure 11 (a) provides another example of a 2-D image. It is a Resolution chart, *g*11*(x, y)*, of size 256×256 pixels. Data in this figure is not a human face, however, it is a resolution chart. It is observed that the amount of gray color in this figure is less than the amount of gray color in the previous two examples.

A program shown in Figure 11 (c) is illustrated for constructing the 2-D and 3-D spatial domains of the resolution chart image in Figures 11 (a) and (b). 2-D and 3-D magnitude spectrums (|*G*11*(u, v)*|) of the resolution chart image are demonstrated in Figures 11 (d) and (f), respectively. The programs for both 2-D and 3-D magnitude spectrums are revealed in Figures 11 (e) and (g).



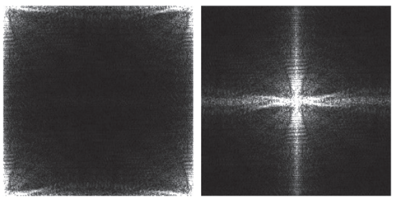
(a)



(b)

```
>> g11 = imread('Chart.tif');
>> imshow(g11)
>> g11 = double(g11);
>> [X,Y] = meshgrid(1:1:256,1:1:256);
>> Z = g11;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis(Amplitude of
g11(x,y))');
```

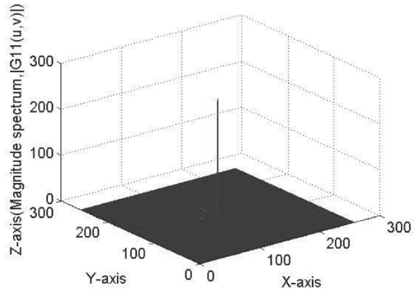
(c)



(d)

```
>> G11 = fft2(g11);
>> absG11 = abs(G11);
>> G11max = max(max(absG11));
>> magG11 = (absG11/G11max)*255;
>> imshow(magG11)
>> ReQG11 = Requadrant(magG11);
>> imshow(ReQG11)
```

(e)



(f)

```
>> [X,Y] =
meshgrid(1:1:256,1:1:256);
>> Z = ReQG11;
>> mesh(X,Y,Z)
>> xlabel('X-axis');
>> ylabel('Y-axis');
>> zlabel('Z-axis(Magnitude
```

(g)

Figure 11 (a) Resolution chart, *g*11*(x, y)*, shown in 2-D view, (b) A 3-D plot for the resolution chart image in the spatial domain, (c) MATLAB® program for the 3-D plot of Figure 11 (b), (d) A 2-D plot for magnitude spectrum of the resolution chart image, |*G*11*(u, v)*|, before requadrant (left) and after requadrant (right), (e) MATLAB® program for the 2-D magnitude spectrum in Figure 11 (d), (f) A 3-D plot for magnitude spectrum of the resolution chart image, |*G*11*(u, v)*| and (g) MATLAB® program for the3-D magnitude spectrum in Figure 11 (d).

Conclusion

This article introduces the concept of 2-D DFT and comprehensible implementations employing the MATLAB® program. Several examples of 2-D synthetic and real cases are given and explained step-by-step. 2-D and 3-D figures based on both spatial and frequency domains, with accompanying MATLAB® programs, are provided to reinforce the concept.

Examples of 2-D DFT in the research are also provided such as video motion estimation or registration and compressive sensing (Patanavijit, 2011; Ha, Lee and Patanavijt, 2014).

References

Castleman, K. R. (1996). *Digital image processing.* Prentice Hall International Editions.

Duhamel, P. & Vetterli, M. (1990). Fast fourier transforms: A tutorial review and a state of the art. *Journal of Signal Processing,* 19 (4), 259-299.

Gonzalez, R. C. & Woods, R. E. (2010). *Digital image processing.* Pearson Prentice Hall, 3rd Edition.

Hu, L., Zhou, J., Shi, Z. & Fu, Q. (2013). A fast and accurate reconstruction algorithm for compressed sensing of complex sinusoids. *IEEE Transactions on Signal Processing,* 61 (22), 5744-5754.

Ha, P.H. & Patanavijit, V. (2010-1). Performance evaluation of L1, L2 and SL0 on compressive sensing based on stochastic estimation technique. *In Proceedings of the 7th Annual International Conference on ECTI-CON, ECTI Association Thailand, Chiang Mai, Thailand,* pp. 688-692.

Ha, P.H. & Patanavijit, V. (2010-2). A novel robust compressive sensing based maximum likelihood with myriad stochastic norm. *In Proceeding of the 33rd EECON, Centara Duangtawan Hotel, Chiang Mai, Thailand,* pp. 1209-1212.

Ha, P.H. & Patanavijit, V. (2011). A novel iterative robust image reconstruction based on SL0 compressive sensing using Huber stochastic estimation technique in wavelet domain. *In Proceedings of the 34th EECON, Ambassador City Jomtien Hotel, Pataya, Chonburi, Thailand,* pp. 1029-1032.

Ha, P.H., Lee, W. & Patanavijit, V. (2012). The novel frequency domain Tikhonov regularization for an image reconstruction based on compressive Sensing with SL0 algorithm. *In Proceedings of the 9th Annual International conference on ECTI-CON, ECTI Assoc. Thailand,* pp.1-4.

Ha, P.H., Lee, W. & Patanavijit, V. (2014). An introduction to compressive sensing for digital signal reconstruction and its implementation on digital image reconstruction. *In Proceedings of the International Conference on iEECON, Pattaya, Thailand,* pp. 1-4.

Lim, J. S. (1990). *Two-dimensional signal and image processing.* Prentice Hall.

Patanavijit, V. (2011). The empirical performance study of a super resolution reconstruction based on frequency domain from aliased multi-low resolution images. *In Proceedings of the 34th EECON, Jomtien Hotel, Chonburi, Thailand,* pp. 645-648.

Patanavijit, V. & Ha, P.H. (2013-1). A novel frequency domain image reconstruction based on the Tikhonov regularization and robust estimation technique for compressive sensing. *In Proceedings of the 10th Annual International Conference on ECTI-CON, ECTI Assoc. Thailand, Krabi, Thailand,* pp. 1-6.

Patanavijit, V. & Ha, P.H. (2013-2). A novel application to image restoration based on regularized SL0 algorithm in frequency domain. *ECTI Transactions on Computer and Information Technology,* 7 (2), 181-195.

Sermwuthisarn, P., Gansawat. D., Patanavijit, V. & Auethavekiat, S. (2012). Impulsive noise rejection method for compressed measurement signal in compressed sensing. *EURASIP Journal on Applied Signal Processing (EURASIP JASP),* DOI: 10.1186/1687-6180-2012-68.

Sermwuthisarn, P., Auethavekiat, S., Gansawat. D. & Patanavijit, V. (2012). Robust reconstruction algorithm for compressed sensing in Gaussian noise environment using orthogonal matching pursuit with partially known support and random subsampling. *EURASIP Journal on Applied Signal Processing (EURASIP JASP),* DOI: 10.1186/1687-6180-2012-34.

Thakulsukanant, K. & Patanavijit, V. 2012. Tutorial of one dimensional discrete Fourier transform (DFT); theory, implementation, and MATLAB® programming (accepted to *SDURJ 8* (2), 2015).

Authors

Asst. Prof. Dr.Kornkamol Thakulsukanant

Business Information Systems (BIS) Department,

Martin de Tours School of Management and Economics (MSME Building 2^{nd} floor)

Assumption University (Suvarnabhumi Campus), 88 Moo 8 Bang Na-Trad Km.26,

Bangsaothong, Samuthprakarn, Thailand, 10540

e-mail: kthakulsukanant@yahoo.com

Asst. Prof. Dr.Vorapoj Patanavijit

Vincent Mary School of Engineering (VME Building 2^{nd} floor)

Assumption University (Suvarnabhumi Campus), 88 Moo 8 Bang Na-Trad Km.26,

Bangsaothong, Samuthprakarn, Thailand, 10540

e-mail: patanavijit@yahoo.com