Two-Dimensional Phase Unwrapping Problem

By Dr. Munther Gdeisat and Dr. Francis Lilley

Pre-requisite: In order to understand this tutorial it is necessary for you to have already studied and completed the "one-dimensional phase unwrapping problem" tutorial before reading this document.

There are many applications that produce wrapped phase images. Examples of these are synthetic aperture radar (SAR), magnetic resonance imaging (MRI) and fringe pattern analysis. The wrapped phase images that are produced by these applications are not usable unless they are first unwrapped so as to form a continuous phase map. This means that the development of a robust phase unwrapping algorithm is an important topic for all these applications. In this article, we will not discuss phase unwrapping only in the specific context of these applications, but we will instead explain the concept of the 2D phase unwrapping problem in general terms.

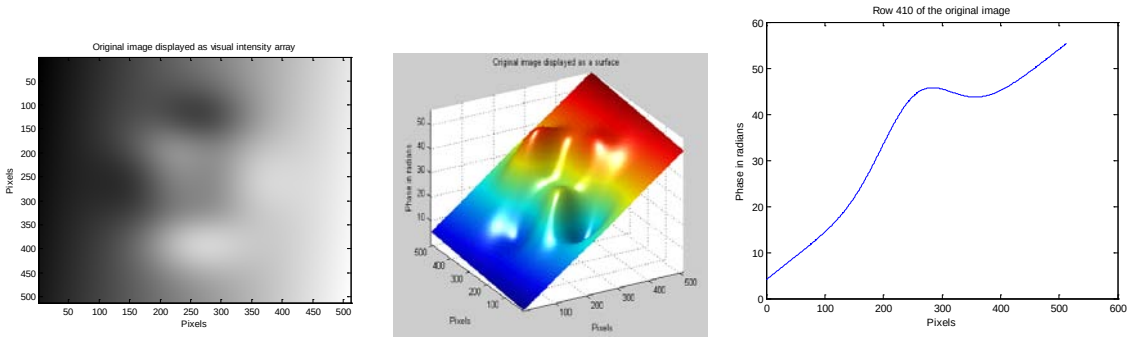
1. Introduction to 2D phase unwrapping

We shall explain the 2D phase unwrapping process as follows. Suppose that we have a computer-generated continuous phase image that does not contain any phase wraps (2π jumps). This image may be displayed as a visual intensity array, as shown in Figure 1(a). The same image may also be plotted as a 3D surface, as shown in Figure 1(b). The intensities from a single row of this image (row 410) are graphically plotted in Figure 1(c). The Matlab code that is used to generate this phase image is as follows. The `peaks` Matlab function is used to generate the continuous phase image. Please note that we are using the term "continuous" here to refer not to an analogue signal, but to a discrete 1D phase signal, or a discrete 2D phase image, that does not contain any phase wraps.

```
%This program is to simulate a continuous phase distribution to act as a dataset
%for use in the 2D phase unwrapping problem
clc; close all; clear
N = 512;
[x,y]=meshgrid(1:N);
image1 = 2*peaks(N) + 0.1*x + 0.01*y;
figure, colormap(gray(256)), imagesc(image1)
title('Continuous phase image displayed as a visual intensity array')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1,'FaceColor','interp', 'EdgeColor','none', 'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title(' Continuous phase map image displayed as a surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

figure, plot(image1(410,:))
title('Row 410 of the continuous phase image')
xlabel('Pixels'), ylabel('Phase in radians')
```



(a) *(b)* *(c)*

Figure 1: (a) A computer-generated continuous phase image displayed as a visual intensity array, (b) the same image plotted as a surface, (c) intensities from row 410 of the phase image.

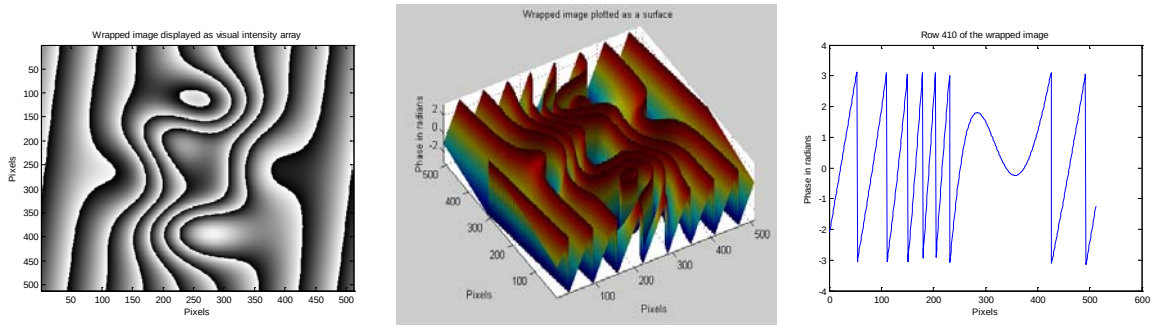
Now let us wrap the computer-generated continuous phase image. The Matlab code to perform this task is as follows;

```
%wrap the 2D image
image1_wrapped = atan2(sin(image1), cos(image1));
figure, colormap(gray(256)), imagesc(image1_wrapped)
title('Wrapped phase image displayed as a visual intensity array')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1_wrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,70), camlight left, axis tight
title('Wrapped phase image plotted as a surface')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

figure, plot(image1_wrapped(410,:))
title('Row 410 of the wrapped phase image')
xlabel('Pixels'), ylabel('Phase in radians')
```

The wrapped image is shown below.



(a) *(b)* *(c)*

Figure 2: (a) A wrapped phase image displayed as a visual intensity array, (b) the wrapped image plotted as a surface, (c) row 410 of the wrapped phase image.

Recall from the 1D phase unwrapping tutorial, that when we were dealing with lines of phase values, the phase wraps appeared as multiple 2π jumps forming a saw-tooth waveform like that shown in Figure 2(c). Note that in the 2D case, where we now have phase images in the form of a 2D array, the phase wraps appear as contour curves, as shown in Figure 2(a), which we shall refer to as wrap curves. These curves will appear in the form of either closed, or open, curves and you can see both types of curve in Figure 2(a). Note that in the latter case, if an open curve enters a wrapped phase image, it must therefore also leave it.

In order to unwrap the image we can use the Itoh 2D phase unwrapper. There are two main methods by which the Itoh 2D phase unwrapper may be implemented. The first method involves unwrapping the rows in the wrapped image sequentially (one at a time). This produces an intermediate image that is only partially phase unwrapped. Next we perform a similar process, but this time unwrap all the columns within the partially unwrapped image. The resultant unwrapped phase image, as produced by this first implementation of the Itoh unwrapper, is shown in Figures 3(a) & (b). The Matlab code to perform this task is as follows.

```
%Unwrap the image using the Itoh algorithm: the first method is performed
%by first sequentially unwrapping the all rows, one at a time.
image1_unwrapped =  image1_wrapped;
for i=1:N
    image1_unwrapped(i,:) = unwrap(image1_unwrapped(i,:));
end
%Then sequentially unwrap all the columns one at a time
for i=1:N
    image1_unwrapped(:,i) = unwrap(image1_unwrapped(:,i));
end
figure, colormap(gray(256)), imagesc(image1_unwrapped)
title('Unwrapped phase image using the Itoh algorithm: the first method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase image using the Itoh algorithm: the first method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')
```

The second method of implementing the Itoh unwrapper simply works the other way around. In other words, it involves first unwrapping all the columns within the wrapped phase image, one at a time. And this again produces a partially phase unwrapped image. Then we sequentially unwrap all rows of the partially unwrapped image. The resultant unwrapped phase image, produced using this second implementation of the Itoh unwrapper, is shown in Figure 3(b). The Matlab code to perform this task is as follows.

```
%Unwrap the image using the Itoh algorithm: the second method
%performed by first sequentially unwrapping all the columns one at a time.
image2_unwrapped =  image1_wrapped;
for i=1:N
    image2_unwrapped(:,i) = unwrap(image2_unwrapped(:,i));
end
%Then sequentially unwrap all the a rows one at a time
for i=1:N
    image2_unwrapped(i,:) = unwrap(image2_unwrapped(i,:));
end
figure, colormap(gray(256)), imagesc(image2_unwrapped)
title('Unwrapped phase image using Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image2_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase image using the Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')
```

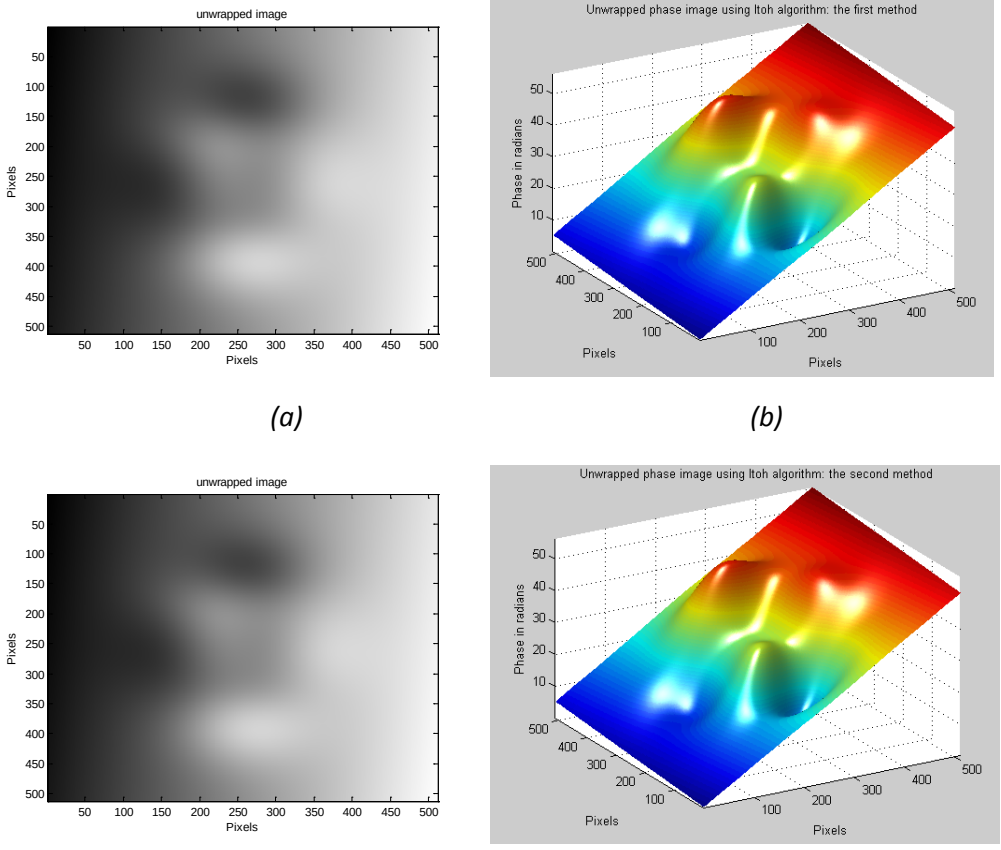


Figure 3: Unwrapped image using the 2D Itoh algorithm; implemented using the first method in (a) & (b), and implemented using the second method in (c) & (d).

It is obvious from the exercise that has been performed above that both of these implementations of the Itoh phase unwrapping algorithm actually produce the same output. This is because this wrapped phase image is not a real one, but is instead an artificial dataset that does not contain any errors.

The wrapped phase image that is shown in Figure 2(a) is a good example of an ideal phase image that does not contain any sources of error. We can easily process this image using any 2D phase unwrapper. As has been explained above, in this case we have processed the image using the 2D Itoh algorithm. This is a very simple phase unwrapping algorithm, which only works in cases where the phase images are virtually error free. Most real-world applications produce wrapped phase images that do contain errors. In this case, we need to use more complex 2D phase unwrappers in order to deal with these images.

In 2D phase unwrapping, there are four sources of errors that complicate the phase unwrapping process. These sources of errors are as follows.

1. Noise
2. Under sampling
3. When the continuous phase image contains sudden, abrupt phase changes
4. Errors produced by the phase extraction algorithm itself

In this tutorial we will discuss only the first three sources of errors and their effects upon the 2D phase unwrapping process. We will also explain how to successfully unwrap images in these three different situations. The fourth source of error depends on the specific algorithm that is used to extract the wrapped phase. The reader should be aware of this as another potential source of error when performing phase unwrapping, however a detailed discussion of the effects of an algorithm itself on the extracted wrapped phase is out of the scope of this tutorial and will not be covered here.

2. The effect of noise on two-dimensional phase unwrapping

A phase unwrapper detects the existence of a phase wrap in an image by calculating the difference between two successive samples. If this difference is larger than $+\pi$, then the phase unwrapper considers there to be a wrap at this location. This could either be a genuine phase wrap, or it could actually be a fake wrap due to the presence of noise. To study the effect of noise on 2D phase unwrapping, let us add noise to the simulated continuous phase image that was shown previously in Figure 1(a). Then we shall wrap the noisy phase image. After that, we will attempt to phase unwrap the simulated object. This process is implemented in the Matlab program that is shown below. The noise variance is set here to a value of 0.4. As we can see from Figure 4, such a low level of added noise does not adversely affect the operation of the Itoh unwrapping algorithm.

```
%This program shows the problems encountered when unwrapping a noisy 2D phase
%image by using computer simulation
clc; close all; clear
N = 512;
[x,y]=meshgrid(1:N);
noise_variance = 0.4;
image1 = 2*peaks(N) + 0.1*x + 0.01*y + noise_variance*randn(N,N);
figure, colormap(gray(256)), imagesc(image1)
title('Noisy continuous phase image displayed as visual intensity array')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1,'FaceColor','interp', 'EdgeColor','none', 'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Noisy continuous phase image displayed as a surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

figure, plot(image1(410,:))
title('Row 410 of the original noisy continuous phase image')
xlabel('Pixels'), ylabel('Phase in radians')

%wrap the 2D image
image1_wrapped = atan2(sin(image1), cos(image1));
figure, colormap(gray(256)), imagesc(image1_wrapped)
title('Noisy wrapped phase image displayed as visual intensity array')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1_wrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,70), camlight left, axis tight
title('Noisy wrapped phase image plotted as a surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

figure, plot(image1_wrapped(410,:))
title('Row 410 of the wrapped noisy image')
xlabel('Pixels'), ylabel('Phase in radians')

%Unwrap the image using the Itoh algorithm: the first method
%Unwrap the image first by sequentially unwrapping the rows one at a time.
image1_unwrapped =  image1_wrapped;
for i=1:N
```

```
    image1_unwrapped(i,:) = unwrap(image1_unwrapped(i,:));
end
%Then unwrap all the columns one-by-one
for i=1:N
    image1_unwrapped(:,i) = unwrap(image1_unwrapped(:,i));
end
figure, colormap(gray(256)), imagesc(image1_unwrapped)
title('Unwrapped noisy phase image using the Itoh algorithm: the first method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped noisy phase image using the Itoh unwrapper: the first method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%Unwrap the image using the Itoh algorithm: the second method
%Unwrap the image by first sequentially unwrapping all the columns.
image2_unwrapped =  image1_wrapped;
for i=1:N
    image2_unwrapped(:,i) = unwrap(image2_unwrapped(:,i));
end
%Then unwrap all the a rows one-by-one
for i=1:N
    image2_unwrapped(i,:) = unwrap(image2_unwrapped(i,:));
end
figure, colormap(gray(256)), imagesc(image2_unwrapped)
title('Unwrapped noisy image using the Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image2_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped noisy phase image using the Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')
```

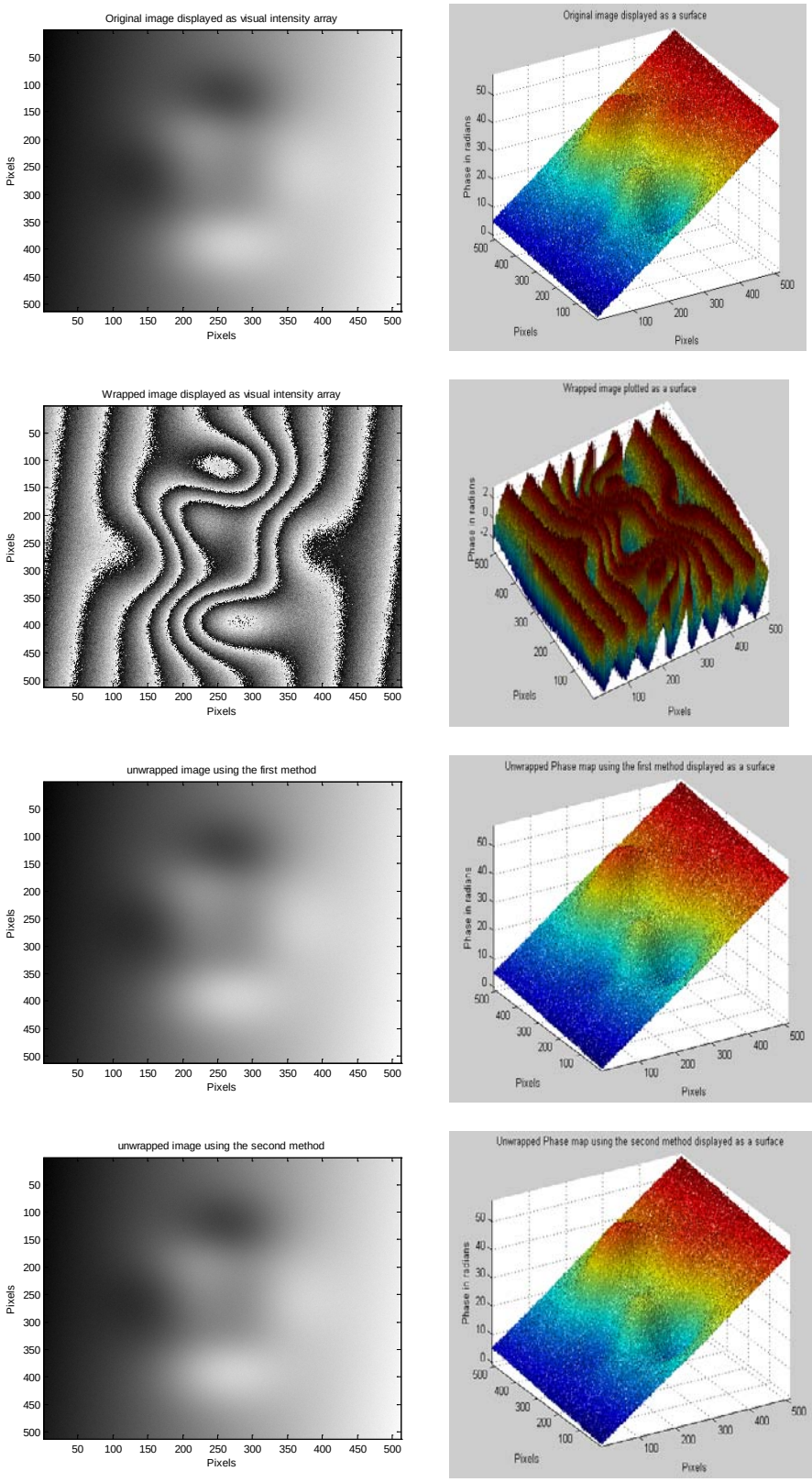
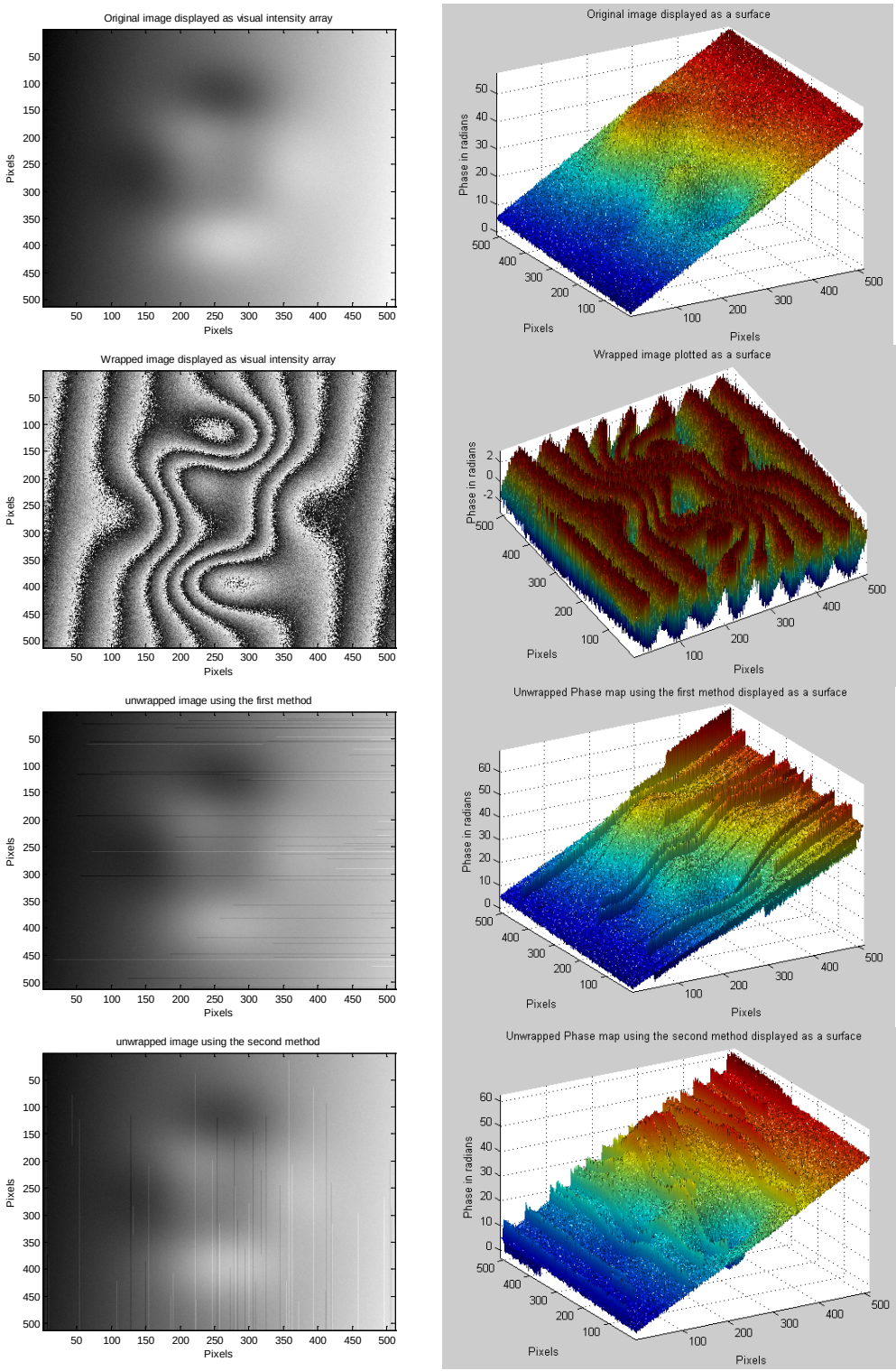


Figure 4: (a) & (b) A noisy computer-generated continuous phase image. (c) & (d) The noisy phase image is wrapped. (e) & (f) Phase unwrapping using the Itoh algorithm: first method. (g) & (h) Phase unwrapping using the Itoh algorithm: second method. The noise variance is set here to a value of 0.4.



(a)	*(b)*
(c)	*(d)*
(e)	*(f)*
(g)	*(h)*

Figure 5: (a) & (b) A noisy computer-generated continuous phase image. (c) & (d) The noisy phase image is wrapped. (e) & (f) Phase unwrapping using the Itoh algorithm: first method. (g) & (h) Phase unwrapping using the Itoh algorithm: second method. The noise variance is set here to a higher value of 0.6.

When we increase the noise variance to a value of 0.6 there are problems. In this case the Itoh phase unwrapping algorithm fails to successfully unwrap this image. Notice that there are 2π discontinuities still present in the unwrapped phase images. Also notice that this time, the first and the second methods of implementing the Itoh algorithm, now produce different results.

As explained in the separate 1D phase unwrapping tutorial that you should have studied previously, error accumulation occurs during the phase unwrapping process, and this is the reason that complicates the process of unwrapping noisy 2D wrapped phase images. Figure 5(e) shows an image that has been processed using the Itoh algorithm: implemented using the first method. This algorithm unwraps the image by firstly sequentially phase unwrapping all the rows one at a time, and then when the unwrapping of all the rows is complete, it subsequently moves on to unwrap all the columns, one at a time. Close inspection of Figure 5(e) reveals some information about the error accumulation problem. For example, row 455 in Figure 5(e) contains a fake wrap. This fake wrap produces a 2π error which propagates throughout the row, from the location of the fake wrap right through until the end of the row. Similar errors also occur during processing for a number of other rows in this wrapped phase image. The Itoh phase unwrapping algorithm: implemented using the the first method, here produces 2π errors that appear as horizontal lines in the resultant unwrapped phase image.

Figure 5(g) shows an image processed using the Itoh algorithm: implemented using the second method. This algorithm changes the order of phase unwrapping the rows and columns when compared to the first implementation. In other words, it unwraps the image by firstly sequentially phase unwrapping all the columns, one at a time. Then, once all the columns are unwrapped, the algorithm moves on to sequentially unwrap all the rows, one at a time. Close inspection of Figure 5(g) reveals some information about the error accumulation problem. For example, column 360 in Figure 5(g) contains a fake wrap. This fake wrap produces a 2π error which propagates throughout the column, from the location of the fake wrap right through until the end of the column. Similar errors also occur during processing for a number of other columns in the wrapped phase image. The Itoh phase unwrapping algorithm: implemented using the second method, here produces 2π errors that appear as vertical lines in the unwrapped phase image.

Researchers have developed many phase unwrapping algorithms that attempt to prevent error propagation occurring. A number of these algorithms are explained in [2]. Also, here at the General Engineering Research Institute (GERI) at LJMU we have developed a robust 2D phase unwrapped algorithm called the 2D-SRNCP phase unwrapper [3]. Our algorithm is based on sorting by reliability, following a non-continuous path and exhibits excellent performance in coping with the noise that corrupts real wrapped phase images. Don't worry about the detail of how it works, just regard it as a very advanced and robust unwrapping algorithm and use it as a tool. You can download the 2D-SRNCP phase unwrapper in Matlab by following the link http://www.ljmu.ac.uk/GERI/90225.htm.

The wrapped phase image shown in Figure 5(c) is processed using the 2D-SRNCP phase unwrapper. The resultant image is displayed as a visual intensity array in Figure 6(a) and also as a 3D surface plot as shown in Figure 6(b). Comparing Figures 6(a) & (b) with Figures 5(a) & (b) respectively reveals that here our algorithm has succeeded in correctly processing the wrapped phase image and has prevented error propagation.

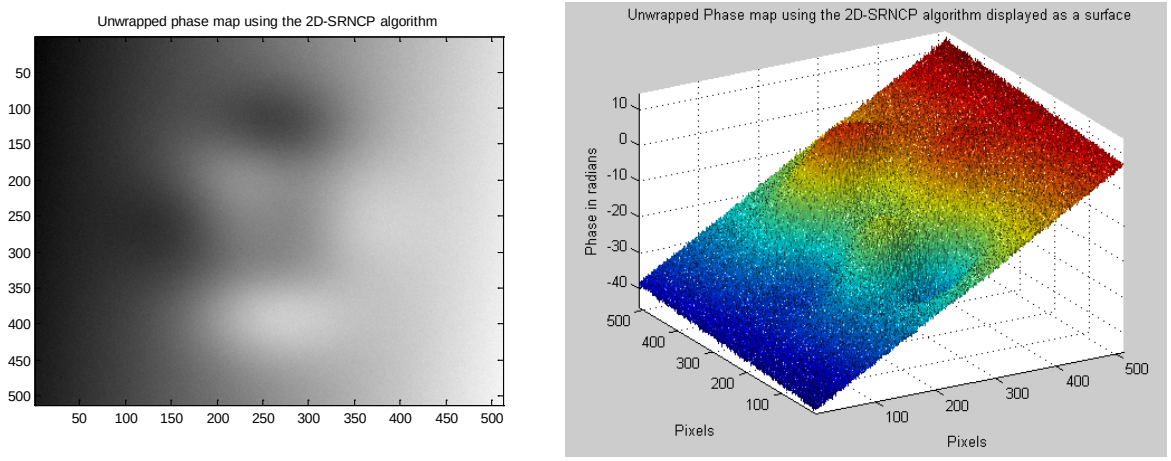
Please note that the 2D-SRNCP phase unwrapper is written in the C programming language. This C program is callable from Matlab using the *Mex* 'Matlab Executable' dynamically linked subroutine functionality. The C code must be compiled in Matlab first, before it is called. To compile the C code in Matlab, at the Matlab prompt, type the following; `mex Miguel_2D_unwrapper.cpp`

The Matlab code that may be used to unwrap the image is given below.

```
%How to call the 2D-SRNCP phase unwrapper from the C language

%You should have already compiled the phase unwrapper's C code first
%If you haven't, to compile the C code: in the Matlab Command Window type
%         mex Miguel_2D_unwrapper.cpp
%The wrapped phase that you present as an input to the compiled C function
%should have the single data type (float in C)
WrappedPhase = single(image1_wrapped);
UnwrappedPhase = Miguel_2D_unwrapper(WrappedPhase);
figure, colormap(gray(256))
imagesc(UnwrappedPhase);
xlabel('Pixels'), ylabel('Pixels')
title('Unwrapped phase image using the 2D-SRNCP algorithm')

figure
surf(double(UnwrappedPhase),'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase image using the 2D-SRNCP displayed as a surface')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')
```



(a) *(b)*

Figure 6: (a) & (b) Unwrapped phase image using the 2D-SRNCP algorithm.

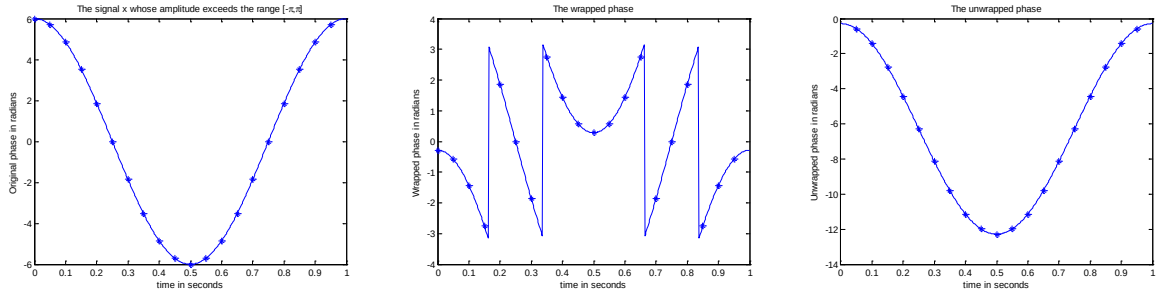
3. The effect of under sampling on two-dimensional phase unwrapping

As has been explained previously, a phase unwrapper detects the existence of a wrap in an image by calculating the difference between two successive samples. If this difference is larger than a value of $+\pi$ or smaller than $-\pi$, then the phase unwrapper considers that there is a wrap in existence at this location. This might be a genuine phase wrap, or it might also be a fake wrap that is caused by noise, or under-sampling.

Phase unwrapping of phase images that are under sampled can be difficult, or in some cases even impossible. This occurs when the difference between two successive samples is larger than $+\pi$, or is less than $-\pi$. This large difference between adjacent samples is present merely due to the fact that the phase image does not contain enough samples and is not because of the existence of a real phase wrap. Such a situation automatically generates an incorrect 'fake wrap'.

Let us first review the effect of under-sampling on the 1D phase unwrapping process. According to Nyquist sampling theory, if a function $f(x)$ contains no frequencies higher than *B* Hertz, then it may be completely determined by sampling it at the rate of 2*B* or greater. In the case where $f(x)$ is a pure sinusoidal signal, then every period in $f(x)$ must be sampled by at least with two samples. This principle also applies to a wrapped phase signal.

Suppose that we consider the 1D continuous phase signal that is shown in Figure 7(a). This signal contains 20 samples and covers exactly one period of the cyclic waveform. This signal is phase wrapped as shown in Figure 7(b). This wrapped signal is sampled at a sufficiently high rate and it contains four genuine wraps. This wrapped signal may be phase unwrapped using the 1D Itoh algorithm and the unwrapped result is shown in Figure 7(c). Notice that in this case the relatively simple 1D Itoh algorithm correctly unwraps the wrapped phase signal. Also note here that whilst the shape of the unwrapped signal is the same as the original, that the actual phase values for each point on the graph is now different, i.e. the original signal in Figure 7(a) ranges from +6 to -6 radians, whereas the unwrapped signal in Figure 7(c) ranges from 0 to -12 radians. You should be aware that most phase unwrappers only produce such 'relative' rather than 'absolute' phase values as their output. Some advanced unwrappers will produce the same shaped relative phase output, but with different numbers in terms of the absolute phase values, every time the code is executed. You should be aware that it is possible to adopt certain measurement strategies which actually do measure absolute phase, rather than relative phase.



(a) *(b)* *(c)*

Figure 7: (a) A continuous phase signal that contains 20 samples. (b) The phase wrapped signal. (c) The phase unwrapped signal.

Now let us reduce the number of samples in the same 1D phase signal that appears in Figure 7(a), halving the sampling rate so that now only 10 samples are taken for this signal, as shown in Figure 8(a). This signal is then wrapped as is shown below in Figure 8(b). This wrapped signal now contains four genuine wraps and also two fake wraps. These two fake wraps occur due to the under sampling of the signal and their positions are highlighted in Figure 8(b). The difference between the third and the fourth samples is smaller than $-\pi$. A phase unwrapper would consider this large difference to be a wrap and would add a value of 2π to the fourth sample and also to all the samples to the right of it, as shown in Figure 8(c). This would have the effect of corrupting the whole 1D phase unwrapped signal. Similarly, the difference between the eighth and the ninth samples is larger than $+\pi$ and once again a phase unwrapper would consider this to be a wrap and hence would subtract a value of 2π from the ninth and tenth samples, as shown in Figure 8(c). This would also have the effect of corrupting the rest of the 1D phase unwrapped signal. Notice that the phase unwrapped signal is now completely different to the original continuous phase signal that was shown in Figure 8(a). Note that this signal has been processed here using the 1D Itoh algorithm.

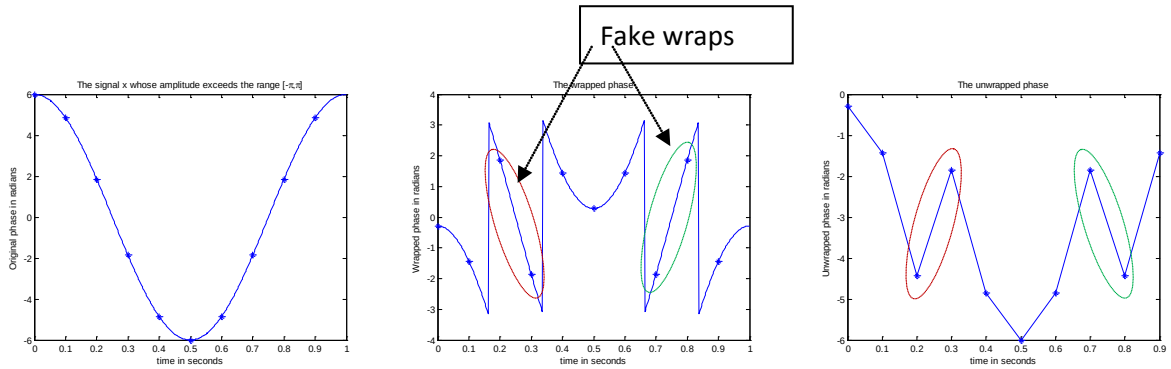


Figure 8: (a) A continuous phase signal that now contains only 10 samples. (b) The wrapped signal. (c) The unwrapped signal

Next we will use computer-generated under sampled phase images to explain the effects of undersampling on 2D phase unwrapping algorithms. First we will create artificial under-sampled phase images. Then we shall theoretically analyse the computer generated phase images to investigate the maximum permissible sampling rates in the x and y directions for the specific datasets, according to sampling theory. Next we will wrap these images. After that we will process these images using two different phase unwrapping algorithms: namely the Itoh algorithm and the 2D-SRNCP algorithm. Finally we will compare the images produced by these two unwrappers with the original continuous phase map.

Suppose that we have the computer-generated continuous phase image $f(x,y)$, which is shown as both a visual intensity array and also as a 3D surface in Figures 9(a) & (b) and is represented by the equation;

$$f(x,y) = 20e^{-\frac{1}{4}(x^2+y^2)} + 2x + y, \qquad -3 \le x \le 3, -3 \le y \le 3$$

Now we shall analyse this simulated phase image in terms of sampling theory. You should have previously completed the 1D phase unwrapping tutorial and you may wish to review the 1D discussion on under-sampling within that document, which will help you understand the discussion on under-sampling in 2D that follows.

First let us find the maximum rate of change of phase in the x direction for this dataset. The rate of phase change in the phase image in the x direction is shown in Figure 9(c) and this is given by the following equation, which has been produced by partial differentiation of the equation above, with respect to x;

$$\frac{\partial f(x,y)}{\partial x} = -10xe^{-\frac{1}{4}(x^2+y^2)} + 2$$

The location of the maximum phase change in the x direction actually occurs at the point in this continuous phase image where x=-1.4172 and y=-0.0015. The value of this maximum phase change at this location is 10.5776 radians.

Therefore, if successive samples in the x direction must change by a value of less than pi, so that they are not incorrectly flagged up as wraps, then the sampling rate (the number of samples in a period, denoted N_x) will be given by;

$$N_x > \frac{10.5776}{\pi} = 3.367$$

The number of actual data points sampled in the x direction for this example N_{Rx} is given by the following equation;

$$N_{Rx} = R_x N_x$$

Where N_x is the sampling rate in the x direction and R_x is the data range for the x axis, and for the example given here the range $R_x = 6$ (6= 3 - -3).

Then, according to the above equation;

$$N_{Rx} = 6.0 \text{ x } 3.367 = 20.2020$$

Therefore the continuous phase image that has been generated here must be sampled with at least 21 samples in the x direction, or it will be under-sampled. If it is sampled with less than 21 samples in the x direction, then such an under-sampling may lead to two successive samples having a difference of more than pi between them, which would erroneously produce a fake wrap at this location when being processed by an unwrapping algorithm.

Similarly, the rate of phase change in the phase image in the y direction is shown in Figure 9(d) and is given by the following equation, which has been produced by partial differentiation of the equation for $f(x,y)$ presented at the start of this section, this time with respect to y;

$$\frac{\partial f(x,y)}{\partial y} = -10ye^{-\frac{1}{4}(x^2+y^2)} + 1$$

The location of the maximum phase change in the y direction occurs at the point in this continuous phase image where x=-0.0044 and y=-1.4143. The value of this maximum phase change in y at this location is 9.5776 radians.

Therefore, if successive samples in the y direction must change by a value of less than pi, so that they are not incorrectly flagged up as wraps, then the sampling rate (the number of samples in a period, denoted N_y) will be given by;

$$N_y > \frac{9.5776}{\pi} = 3.0486$$

The number of actual data points sampled in the y direction for this example N_{Ry} is given by the following equation;

$$N_{Ry} = R_y N_y$$

Where N_y is the sampling rate in the y direction and R_y is the data range for the y axis, and for the example given here the range $R_y = 6$ (6= 3 - -3).

Then, according to the above equation;

$$N_{Ry} = 6.0 \text{ x } 3.0486 = 18.2916$$

Therefore the continuous phase image that has been generated here must be sampled with at least 19 samples in the y direction in order to avoid under-sampling and the erroneous production of fake wraps.

The code to create and plot this simulated continuous phase dataset, along with the x and y rates of change of the phase, is given below;

```
clc; close all; clear
NRx = 512; NRy=512;
tx = linspace(-3,3,NRx);
ty = linspace(-3,3,NRy);
[x,y]=meshgrid(tx,ty);
image1 = 20*exp(-0.25*(x.^2 + y.^2)) + 2*x + y;
figure, colormap(gray(256)), imagesc(tx,ty,image1)
title('Original image displayed as a visual intensity array')
xlabel('x axis'), ylabel('y axis')

figure
surf(x,y,image1,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Original image displayed as a surface plot')
xlabel('x axis'), ylabel('y axis'), zlabel('Phase in radians')

xdiff = diff(image1')';
figure, colormap(gray(256)), imagesc(tx(1:end-1),ty,xdiff)
title('Phase change in the x direction')
xlabel('x axis'), ylabel('y axis')

ydiff = diff(image1);
figure, colormap(gray(256)), imagesc(tx,ty(1:end-1),ydiff)
title('Phase change in the y direction')
xlabel('x axis'), ylabel('y axis')
```

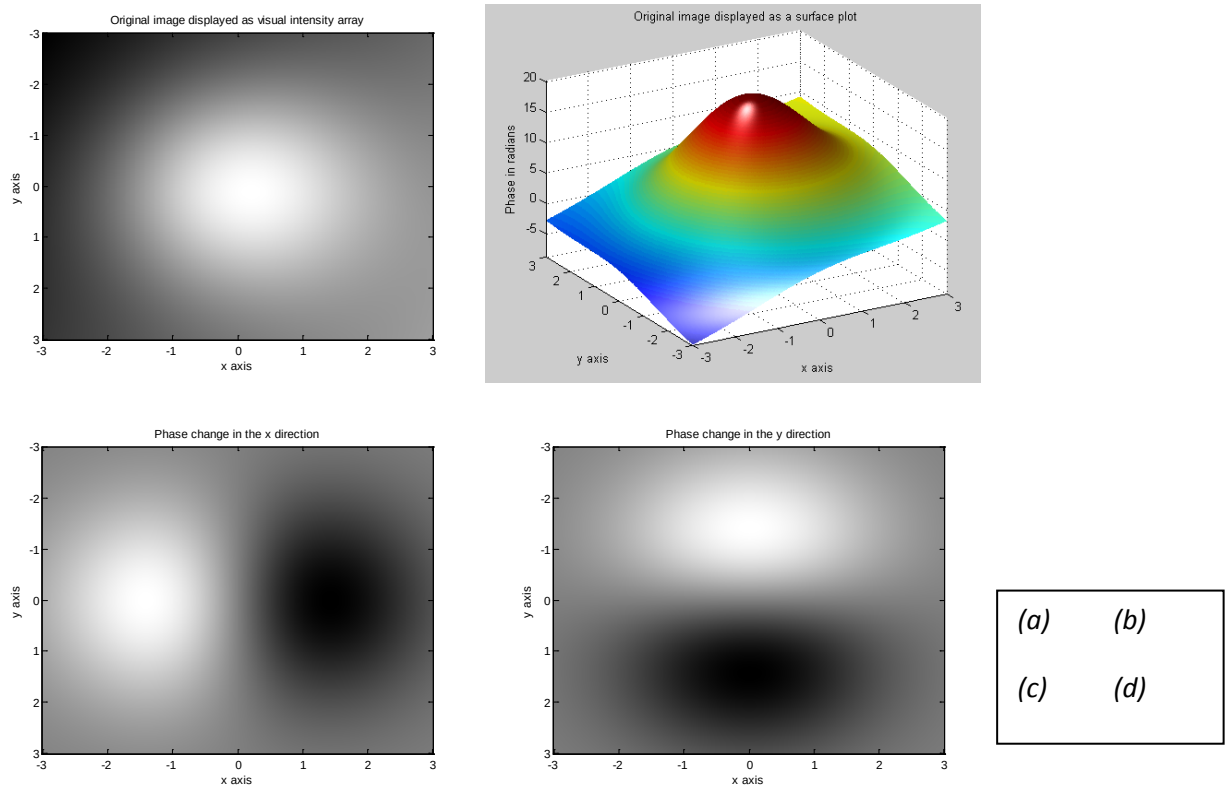
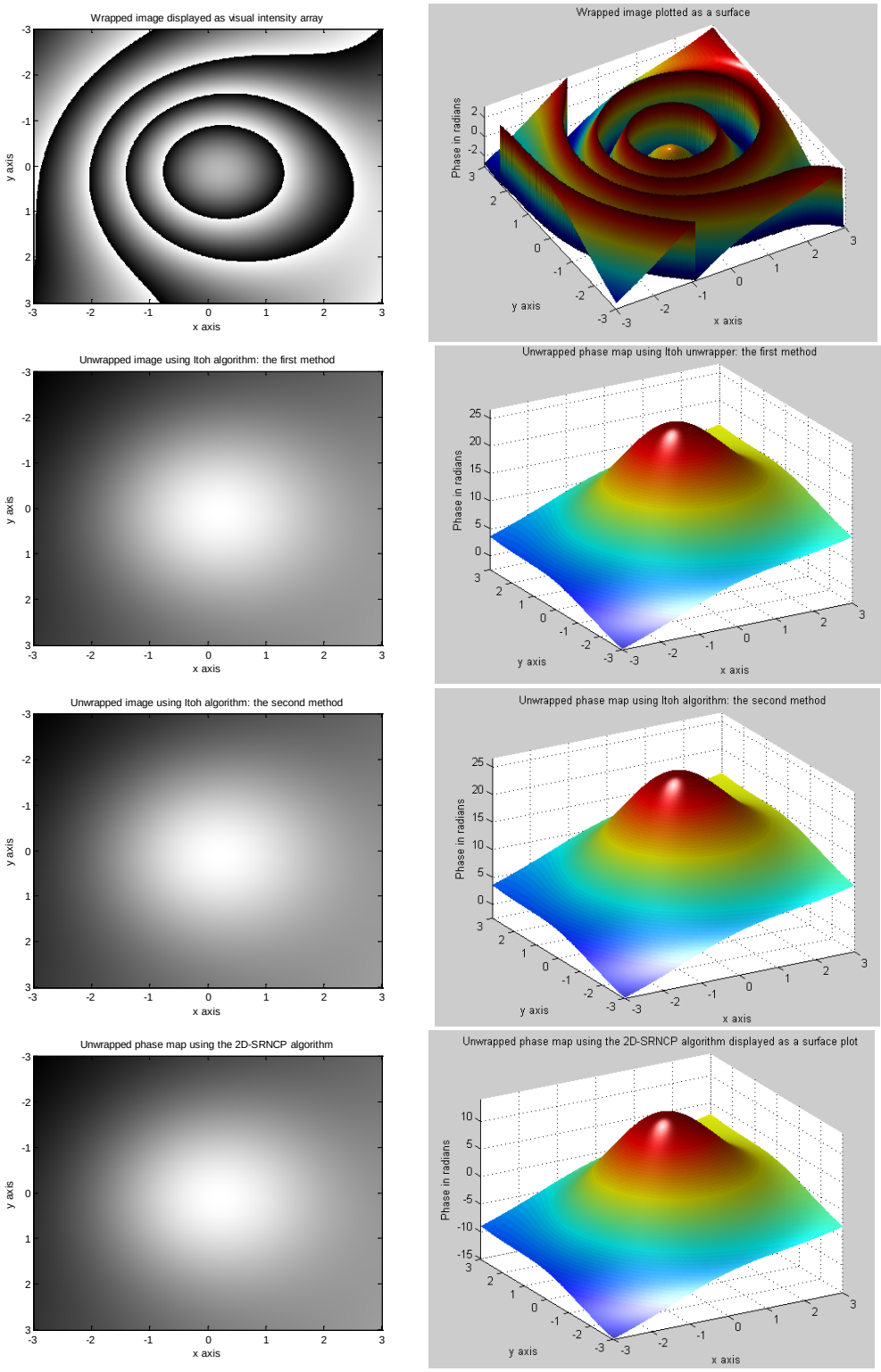


Figure 9: (a) & (b) A computer-generated continuous phase image. (c) & (d) Phase changes in both the x and y directions, respectively, for the computer-generated phase image. Here, NRx=512, NRy=512. Continued...

The computer-generated continuous phase image may now be artificially wrapped and the result of this is shown in Figures 9(e) & (f). The wrapped phase image was unwrapped using the Itoh algorithm: implemented using the first method, as is shown in Figures 9(g) & (h). Next the wrapped phase image was unwrapped using the Itoh algorithm: implemented using the second method, and the results are shown in Figures 9(i) & (j). Finally, the wrapped phase image was unwrapped using the 2D-SRNCP algorithm, as is shown in Figures 9(k) & (l).



(e)	*(f)*
(g)	*(h)*
(i)	*(j)*
(k)	*(l)*

Figure 9: Continued. (e) & (f) Wrapped image. Image unwrapper using (g) & (h) Itoh algorithm: the first method, (i) & (j) Itoh algorithm: the second method, (k) & (l) 2D-SRNCP algorithm. Here, the sampling rates are high, with NRx=512 and NRy=512.

The Matlab code that was used to generate all the images that are shown in Figure 9 is given below. The value for NRx is set to 512 here, and the corresponding value for NRy is set to 512. Note that all three phase unwrapping algorithms succeeded in correctly unwrapping the wrapped phase image that is shown in Figure 9(e).

```
%wrap the 2D image
image1_wrapped = atan2(sin(image1), cos(image1));
figure, colormap(gray(256)), imagesc(tx,ty,image1_wrapped)
title('Wrapped image displayed as a visual intensity array')
xlabel('x axis'), ylabel('y axis')

figure
surf(x,y,image1_wrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,70), camlight left, axis tight
title('Wrapped image plotted as a surface')
xlabel('x axis'), ylabel('y axis'), zlabel('Phase in radians')

%Unwrap the image using the Itoh algorithm: implemented using the first method
%Unwrap the image by firstly unwrapping all the rows, one at a time.
image1_unwrapped =  image1_wrapped;
for i=1:NRy
    image1_unwrapped(i,:) = unwrap(image1_unwrapped(i,:));
end
%Then unwrap all the columns, one at a time
for i=1:NRx
    image1_unwrapped(:,i) = unwrap(image1_unwrapped(:,i));
end
figure, colormap(gray(256)), imagesc(tx,ty,image1_unwrapped)
title('Unwrapped image using the Itoh algorithm: the first method')
xlabel('x axis'), ylabel('y axis')

figure
surf(x,y,image1_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase map using the Itoh unwrapper: the first method')
xlabel('x axis'), ylabel('y axis'), zlabel('Phase in radians')

%Unwrap the image using the Itoh algorithm: implemented using the second method
%Unwrap the image by firstly unwrapping all the columns one at a time.
image2_unwrapped =  image1_wrapped;
for i=1:NRx
    image2_unwrapped(:,i) = unwrap(image2_unwrapped(:,i));
end
%Then unwrap all the a rows one at a time
for i=1:NRy
    image2_unwrapped(i,:) = unwrap(image2_unwrapped(i,:));
end
figure, colormap(gray(256)), imagesc(tx,ty,image2_unwrapped)
title('Unwrapped image using the Itoh algorithm: the second method')
xlabel('x axis'), ylabel('y axis')

figure
surf(x,y,image2_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase map using the Itoh algorithm: the second method')
xlabel('x axis'), ylabel('y axis'), zlabel('Phase in radians')
```

```
%call the 2D phase unwrapper from the C language
%To compile the C code: in Matlab Command Window type
%         mex Miguel_2D_unwrapper.cpp
%The wrapped phase should have the single data type (float in C)
WrappedPhase = single(image1_wrapped);
UnwrappedPhase = Miguel_2D_unwrapper(WrappedPhase);
figure, colormap(gray(256))
imagesc(tx,ty,UnwrappedPhase);
xlabel('x axis'), ylabel('y axis')
title('Unwrapped phase map using the 2D-SRNCP algorithm')

figure
surf(x,y,double(UnwrappedPhase),'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase map using the 2D-SRNCP displayed as a surface plot')
xlabel('x axis'), ylabel('y axis'), zlabel('Phase in radians')
```

The above procedures are now repeated, but this time the sampling rates are lowered, with NRx set to a value of 23 and NRy set to a value of 21. Note from the results that all the phase unwrapping algorithms succeeded in unwrapping the wrapped phase image that is shown in Figure 10(c). The resulting unwrapped phase images are shown in Figure 10.

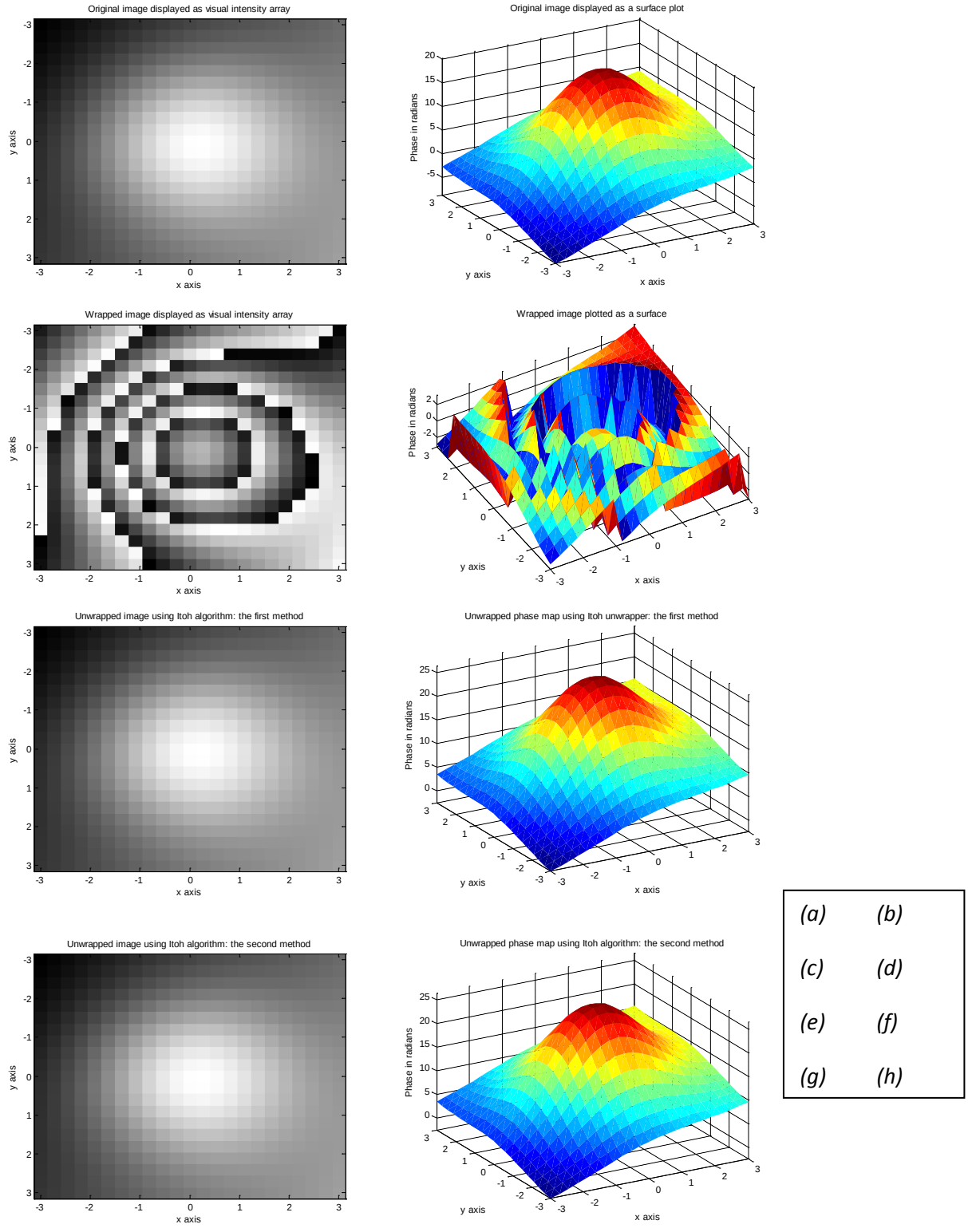
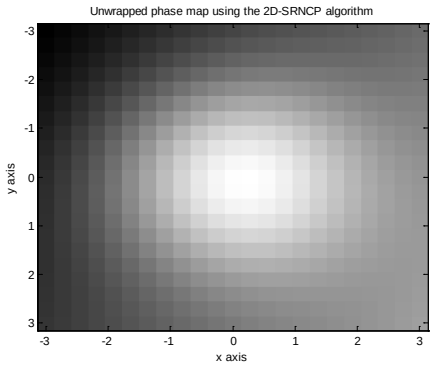
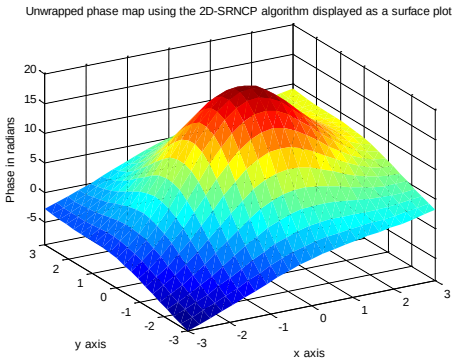


Figure 10: (a) & (b) A computer-generated continuous phase image. (c) & (d) Wrapped image. Image unwrapped using (e) & (f) the Itoh algorithm: the first method, (g) & (h) the Itoh algorithm: the second method. Continued...





(i) *(j)*

Figure 10: Continued (i) & (j) the 2D-SRNCP algorithm. Here, the sampling rates have been reduced, but are still above the theoretical minimum permissible limit, with NRx=23 and NRy=21.

The above procedures are now repeated once again, but this time with NRx set to a value of 20 and NRy set to a value of 18 (i.e. below the theoretical minimum sampling rates). By setting NRx and NRy to these values, the computer-generated continuous phase image is now under sampled. It can be seen from the results, as shown in Figures 11(e)-(h), that the Itoh algorithm fails to correctly unwrap the wrapped phase image. Here the errors produced by fake wraps that are caused by undersampling propagate throughout the image. On the other hand, the 2D-SRNCP phase unwrapper, although it cannot prevent the errors occurring and they are visibly present in the unwrapped phase map, does prevent these errors from propagating through the image and corrupting good data, as is shown in Figures 11 (i) & (j).

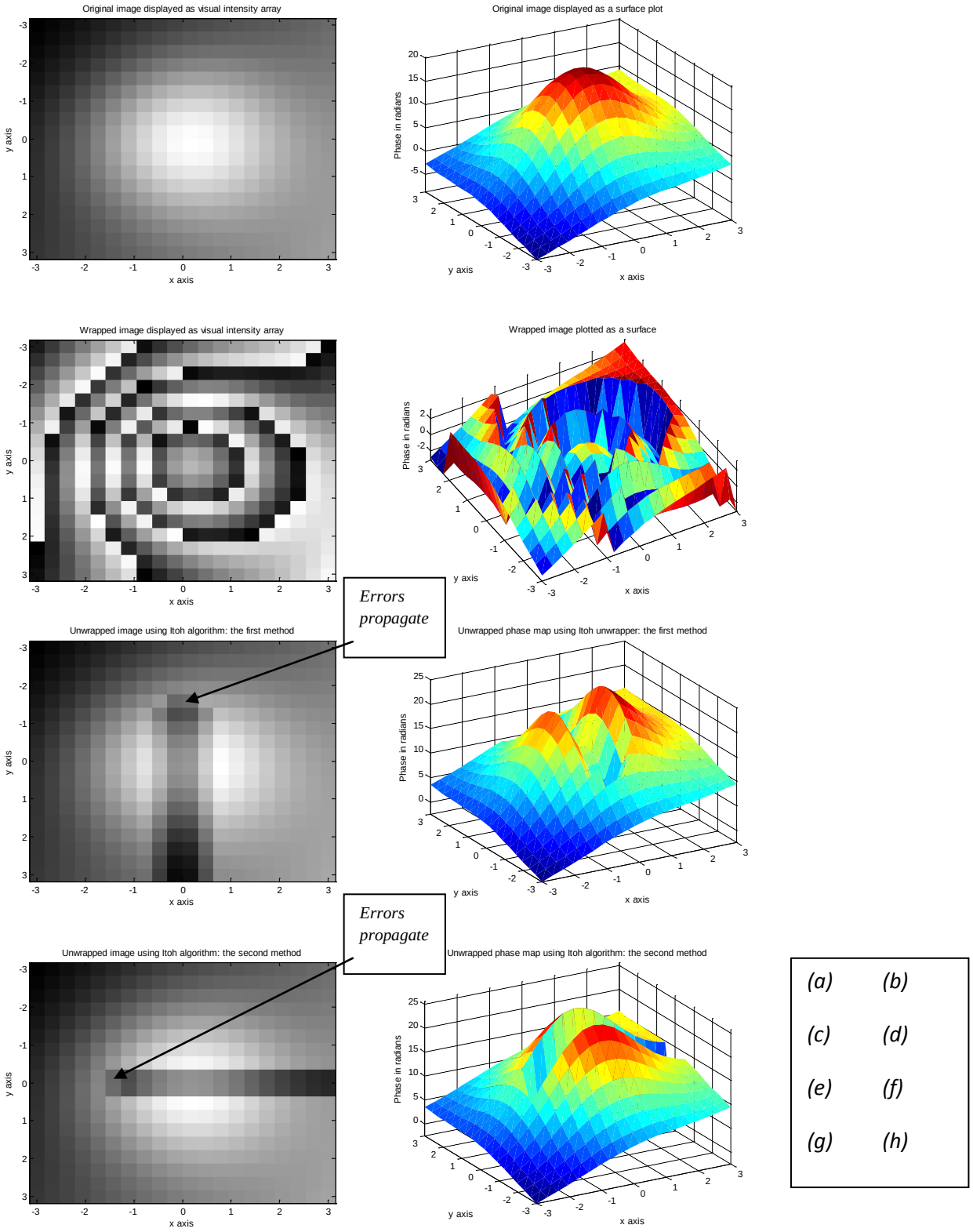


Figure 11: (a) & (b) A computer-generated continuous phase image. (c) & (d) Wrapped phase image. Image unwrapped using (e) & (f) The Itoh algorithm: the first method, (g) & (h) The Itoh algorithm: the second method. Continued...

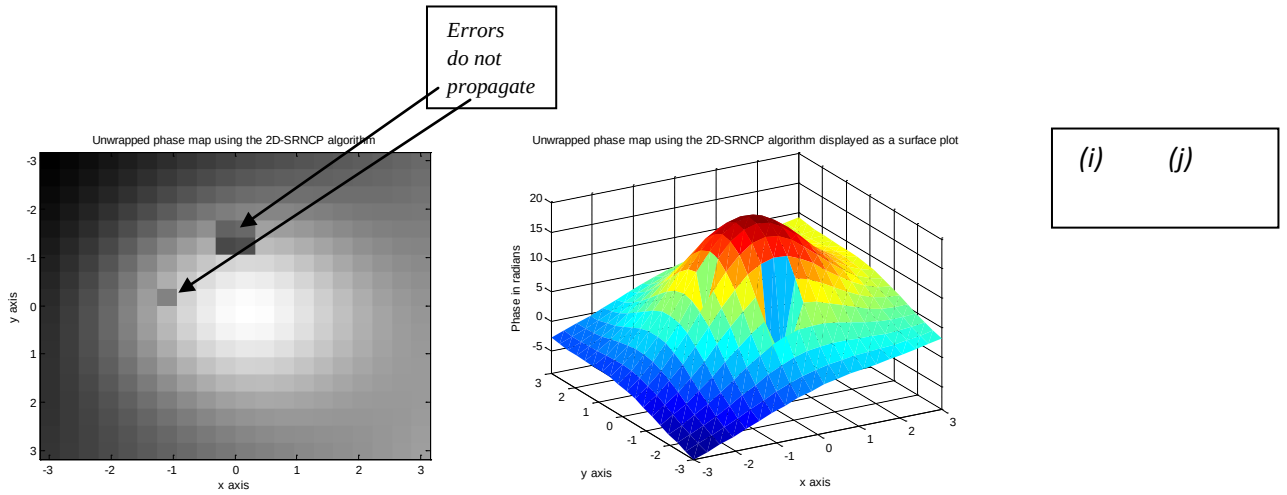


Figure 11: Continued (i) & (j) The 2D-SRNCP algorithm. Here, we are using the lowest sampling rates, below the theoretically minimum permissible limit, with NRx=20 and NRy=18.

4. Effect of phase discontinuities on two-dimensional phase unwrapping

The phase unwrapping of a continuous phase image that contains sudden changes in its phase can be difficult, or even impossible. This occurs when these phase changes are larger than $+\pi$, or less than $-\pi$. Such large and abrupt phase changes may seem to be real wraps, but are actually fake wraps that exist merely due to these sudden phase changes. Alternatively they may also act to mask real phase wraps, causing them to disappear. We will use computer-generated phase images here to explain the effects of these phase discontinuities on 2D phase unwrapping algorithms. Then we will wrap these images. After that we will process these images using two different phase unwrapping algorithms: namely the Itoh and the 2D-SRNCP algorithms. Finally we will compare the images that are produced by these unwrappers with the original continuous phase images.

4.1 Example 1

Suppose that we have the computer-generated continuous phase image that is shown both as a visual intensity array, and a surface plot, in Figures 12(a) & 10(b). This phase image contains sudden changes in its phase within a rectangular region that runs from row 100 up until row 412, and from column 200 until column 300 inclusive. The sudden phase change is of a magnitude of 10 radians. The Matlab implementation of this phase map is shown in the Matlab code given below.

The phase image is first wrapped and then it is unwrapped using the two different phase unwrapping algorithms that were mentioned previously. As you can deduce from examining Figure 12, the Itoh algorithm fails to successfully unwrap this image, whereas the 2D-SRNCP algorithm succeeds in unwrapping this phase image.

A closer inspection of Figure 12(c) reveals that a number of fake wraps exist. For example, a number of fake wraps are located in rows 100 and 412. These fake wraps disrupt the proper operation of the Itoh algorithm: the first method, as shown in Figure 12(f). A very large number of fake wraps (there are actually 312 of them!) are located in column 300. These fake wraps are produced due to the sudden 5 radian phase change that occurs at this location. These fake wraps disrupt the operation of the Itoh algorithm: the second method, as shown in Figure 12(h).

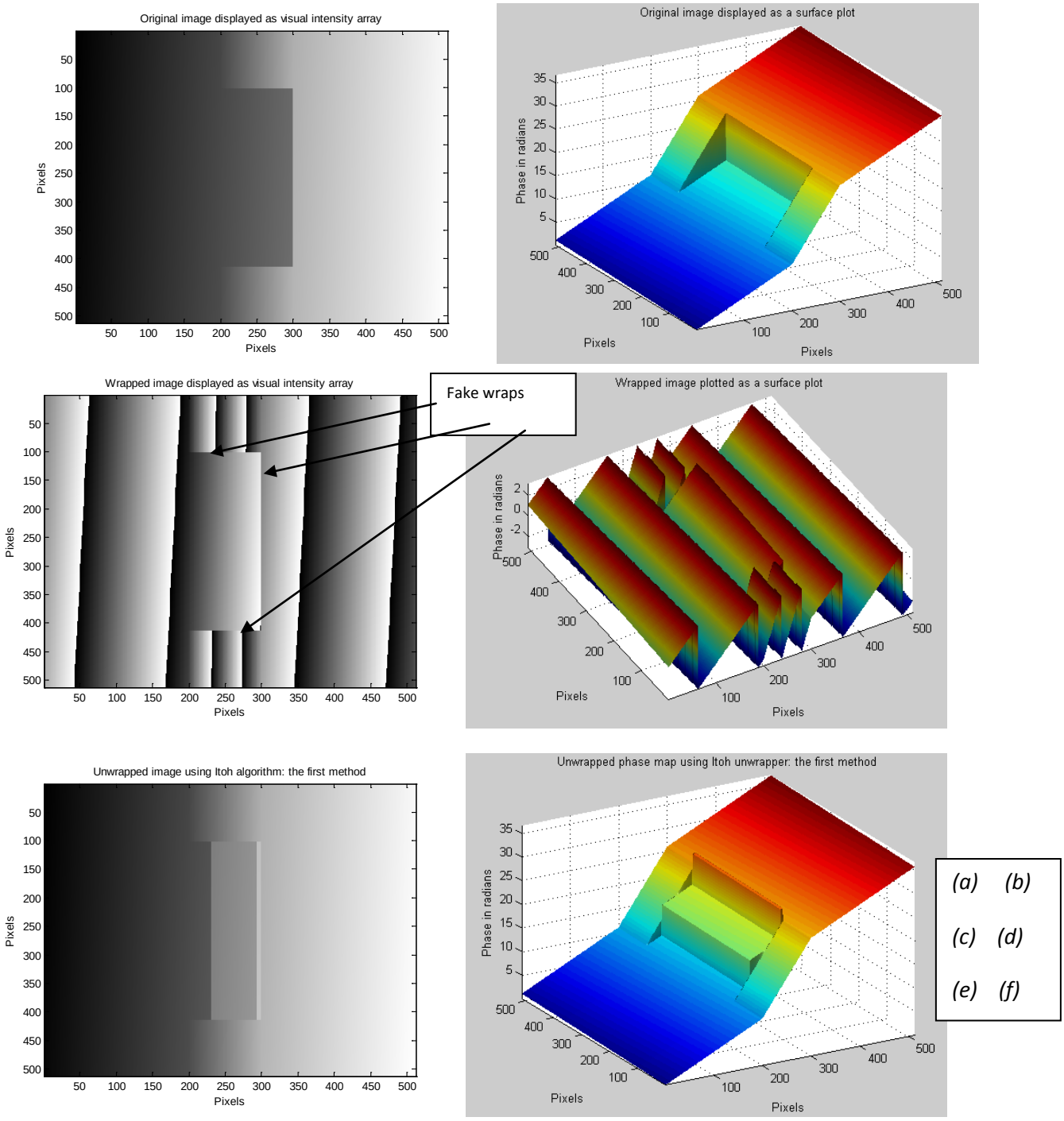


Figure 12: (a) & (b) A computer-generated phase image. (c) & (d) Wrapped image. Image unwrapped using (e) & (f) the Itoh algorithm: the first method. Continued...

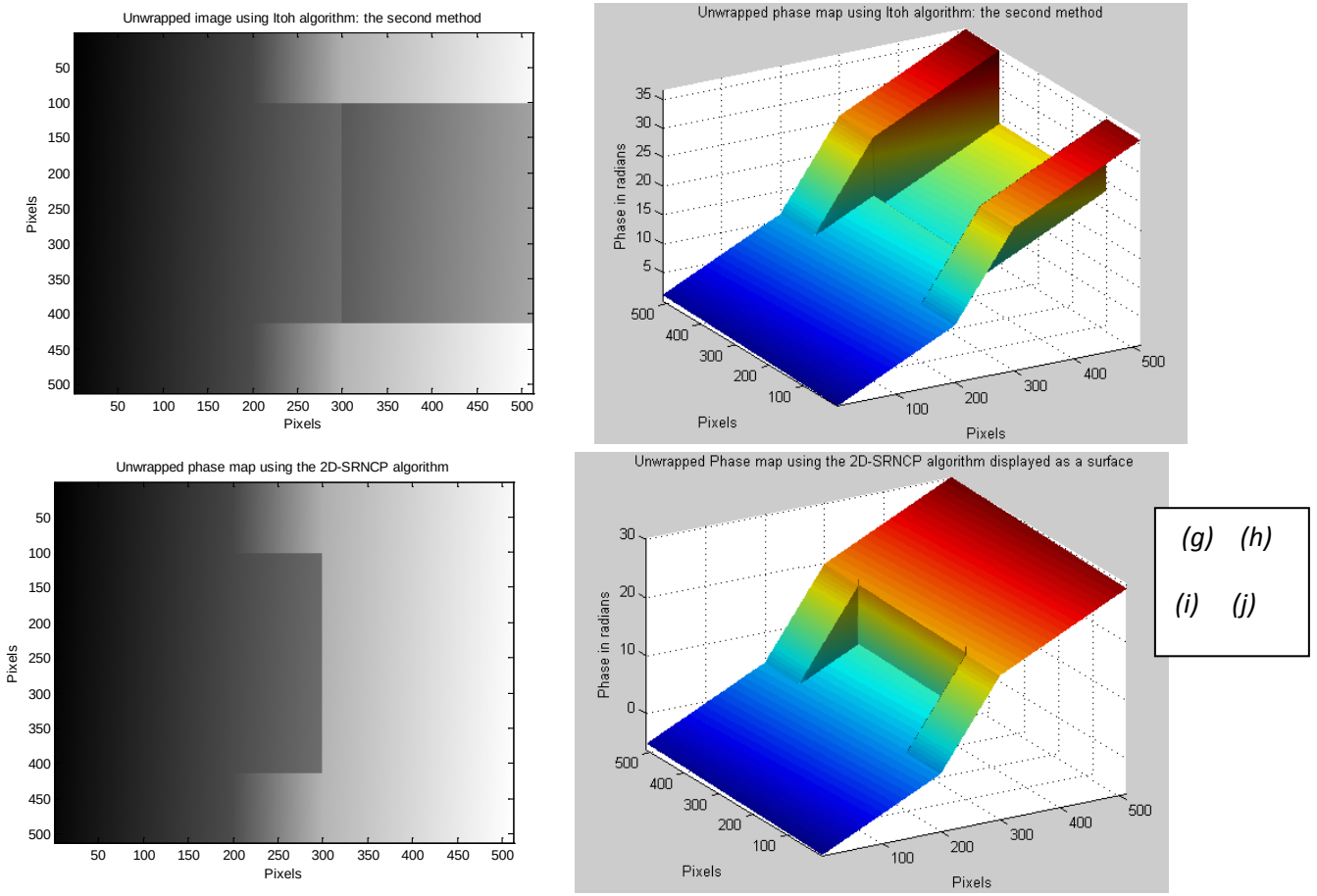


Figure 12: Continued. (g) & (h) the Itoh algorithm: the second method, (i) & (j) the 2D-SRNCP algorithm.

The Matlab code that was used to generate the images that are shown in Figure 12 is given below;

```
%This program is to simulate the 2D phase unwrapping problem
clc; close all; clear
N = 512;
[x,y] = meshgrid(1:N);
shape = zeros(N,N);
phaseChange = 10;
shape(:,300:end) = phaseChange;
x1=meshgrid(1:100);
shape(1:100,200:299) = x1*phaseChange/100;
shape(413:512,200:299) = x1*phaseChange/100;
image1 = shape + 0.05*x + 0.002*y;
figure, colormap(gray(256)), imagesc(image1)
title('Original image displayed as a visual intensity array')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1,'FaceColor','interp', 'EdgeColor','none', 'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Original image displayed as a surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%wrap the 2D image
image1_wrapped = atan2(sin(image1), cos(image1));
figure, colormap(gray(256)), imagesc(image1_wrapped)
title('Wrapped image displayed as a visual intensity array')
```

```
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1_wrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,70), camlight left, axis tight
title('Wrapped image plotted as a surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%Unwrap the image using Itoh the algorithm: the first method
%Unwrap the image by firstly unwrapping the rows one at a time.
image1_unwrapped =  image1_wrapped;
for i=1:N
    image1_unwrapped(i,:) = unwrap(image1_unwrapped(i,:));
end
%Then unwrap all the columns one at a time
for i=1:N
    image1_unwrapped(:,i) = unwrap(image1_unwrapped(:,i));
end
figure, colormap(gray(256)), imagesc(image1_unwrapped)
title('Unwrapped image using the Itoh algorithm: the first method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase map using the Itoh unwrapper: the first method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%Unwrap the image using the Itoh algorithm: the second method
%Unwrap the image by firstly unwrapping all the columns one at a time.
image2_unwrapped =  image1_wrapped;
for i=1:N
    image2_unwrapped(:,i) = unwrap(image2_unwrapped(:,i));
end
%Then unwrap all the a rows one at a time
for i=1:N
    image2_unwrapped(i,:) = unwrap(image2_unwrapped(i,:));
end
figure, colormap(gray(256)), imagesc(image2_unwrapped)
title('Unwrapped image using the Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image2_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase map using the Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%Call the 2D-SRNCP phase unwrapper from the C language code
%To compile the C code: in Matlab Command Window type
%         mex Miguel_2D_unwrapper.cpp
%The wrapped phase should have the single data type (float in C)
WrappedPhase = single(image1_wrapped);
UnwrappedPhase = Miguel_2D_unwrapper(WrappedPhase);
figure, colormap(gray(256))
imagesc(UnwrappedPhase);
xlabel('Pixels'), ylabel('Pixels')
```

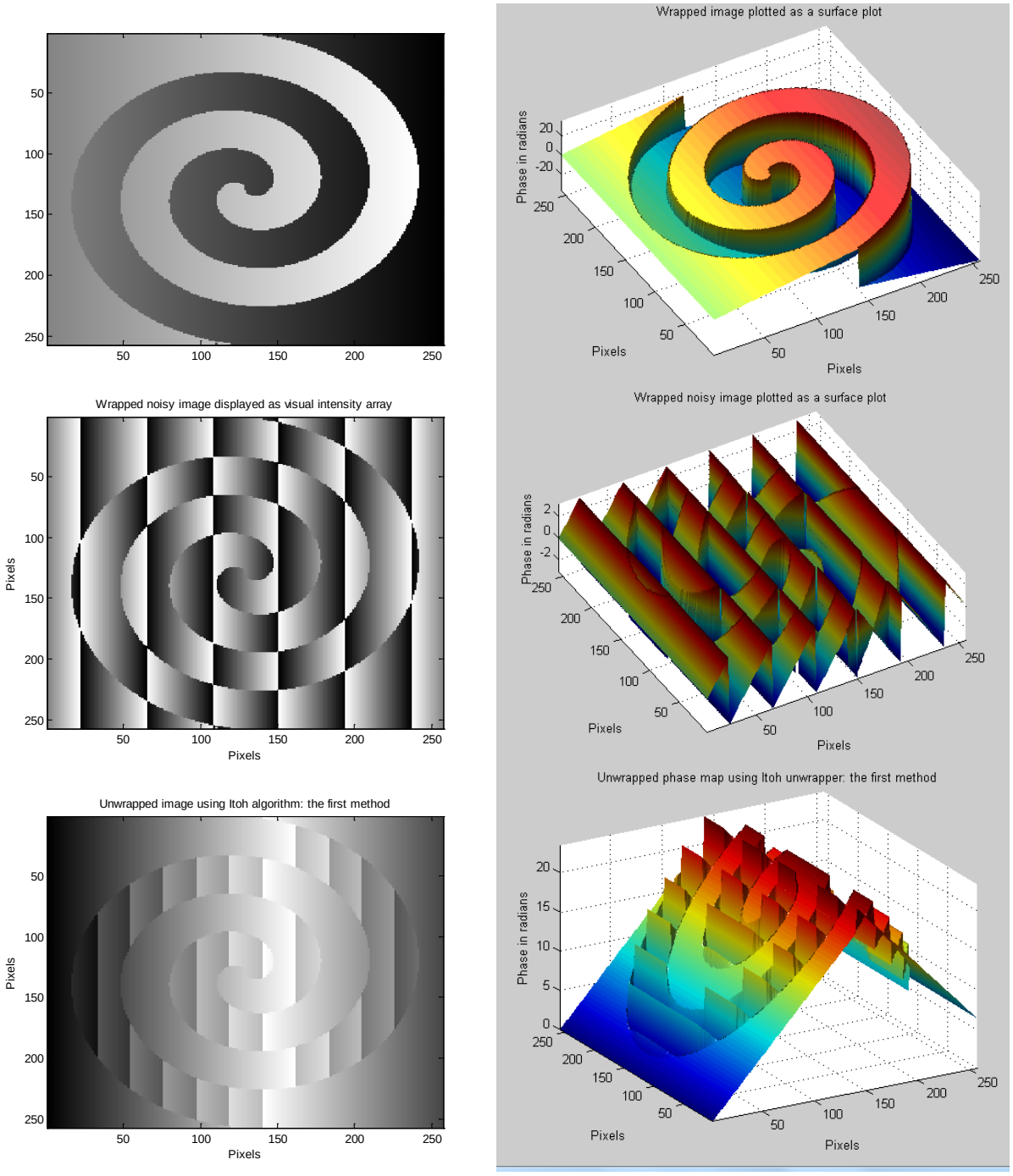
```
title('Unwrapped phase map using the 2D-SRNCP algorithm')

figure
surf(double(UnwrappedPhase),'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped Phase map using the 2D-SRNCP algorithm displayed as a surface')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')
```

4.2 Example 2

Another continuous phase image can be downloaded from the following URL; ftp://ftp.wiley.com/public/sci_tech_med/phase_unwrapping/.

Double click on the file *data.zip*. Download the file and uncompress it. Extract the phase image '*spiral.257x257.surf*' and store it on your computer's hard disk. This continuous phase image is shown in Figures 13(a) & 13(b). Let us first wrap this phase image as shown Figures 13(c) & 13(d). This particular wrapped phase image is a very challenging example and it is very difficult to unwrap [2]. The Itoh algorithm: implemented using the first method fails to unwrap this image. The resultant image that is produced using this algorithm is shown Figures 13(e) & 13(f). The Itoh algorithm: implemented using the second method also fails to unwrap this image. The resultant image for the second version of this algorithm is shown Figures 13(g) & 13(h). Even the 2D-SRNCP algorithm fails to unwrap this image. The resultant image for the 2D-SRNCP algorithm is shown Figures 13(i) & 13(j).



(a) *(b)*

(c) *(d)*

(e) *(f)*

Figure 13: (a) & (b) A computer-generated phase image. (c) & (d) Wrapped image. Image unwrapped using (e) & (f) Itoh algorithm: the first method. Continued...

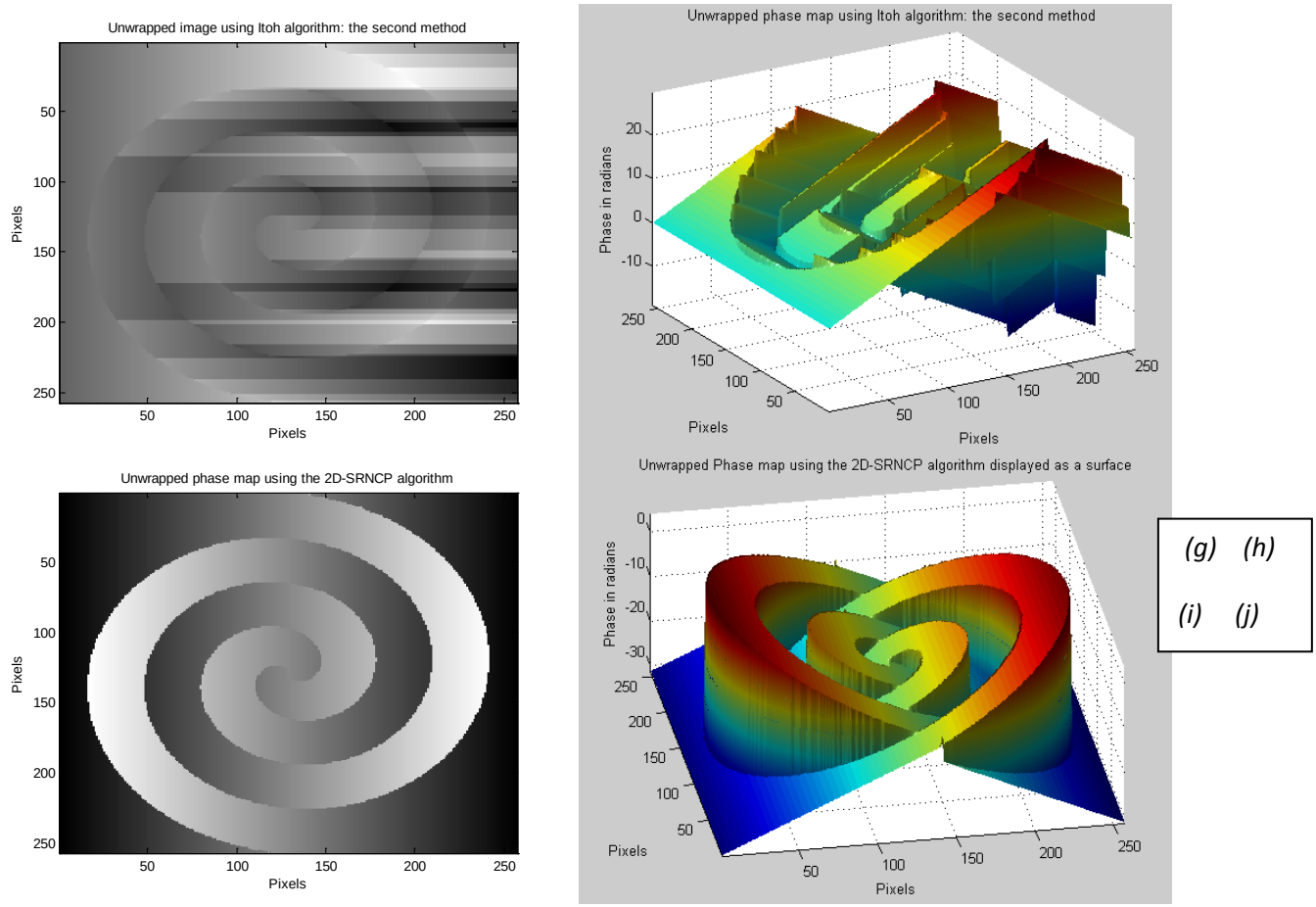


Figure 13: Continued. (g) & (h) Itoh algorithm: the second method, (i) & (j) 2D-SRNCP algorithm.

The Matlab code that may be used to generate the images shown in Figure 13 is given below.

```
%This program is to simulate the 2D phase unwrapping problem
%load the continuous phase image
clc; close all; clear
N = 257;
fid = fopen('spiral.257x257.surf','r','b');
image1 = fread(fid, N * N, 'float');
fclose(fid);
image1 = reshape(image1, N, N);
image1 = double(image1);
figure, colormap(gray(256)), imagesc(image1);

figure
surf(image1,'FaceColor','interp', 'EdgeColor','none', 'FaceLighting','phong')
view(-30,70), camlight left, axis tight
title('Wrapped image plotted as a surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%wrap the continuous phase image
image1_wrapped = atan2(sin(image1), cos(image1));
figure, colormap(gray(256)), imagesc(image1_wrapped)
title('Wrapped noisy image displayed as a visual intensity array')
xlabel('Pixels'), ylabel('Pixels')

figure
```

```
surf(image1_wrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,70), camlight left, axis tight
title('Wrapped noisy image plotted as a surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%Unwrap the image using the Itoh algorithm: implemented using the first method
%Unwrap the image by firstly unwrapping the rows one at a time.
image1_unwrapped =  image1_wrapped;
for i=1:N
    image1_unwrapped(i,:) = unwrap(image1_unwrapped(i,:));
end
%Then unwrap all the columns one at a time
for i=1:N
    image1_unwrapped(:,i) = unwrap(image1_unwrapped(:,i));
end
figure, colormap(gray(256)), imagesc(image1_unwrapped)
title('Unwrapped image using the Itoh algorithm: the first method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image1_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase map using the Itoh unwrapper: the first method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%Unwrap the image using Itoh algorithm: implemented using the second method
%Unwrap the image by firstly unwrapping all the columns one at a time.
image2_unwrapped =  image1_wrapped;
for i=1:N
    image2_unwrapped(:,i) = unwrap(image2_unwrapped(:,i));
end
%Then unwrap all the a rows one at a time
for i=1:N
    image2_unwrapped(i,:) = unwrap(image2_unwrapped(i,:));
end
figure, colormap(gray(256)), imagesc(image2_unwrapped)
title('Unwrapped image using the Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels')

figure
surf(image2_unwrapped,'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('Unwrapped phase map using the Itoh algorithm: the second method')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')

%Call the 2D-SRNCP phase unwrapper from the compiled C language code
%To compile the C code: in Matlab Command Window type
%         mex Miguel_2D_unwrapper.cpp
%The wrapped phase should have the single data type (float in C)
WrappedPhase = single(image1_wrapped);
UnwrappedPhase = Miguel_2D_unwrapper(WrappedPhase);
figure, colormap(gray(256))
imagesc(UnwrappedPhase);
xlabel('Pixels'), ylabel('Pixels')
title('Unwrapped phase map using the 2D-SRNCP algorithm')

figure
```

```
surf(double(UnwrappedPhase),'FaceColor','interp', 'EdgeColor','none',
'FaceLighting','phong')
view(-10,50), camlight left, axis tight
title('Unwrapped Phase map using the 2D-SRNCP algorithm displayed as a surface')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')
```

5. Conclusions

The phase unwrapping of images that are sampled at sufficiently high sampling rates, and that do not contain noise, or sudden phase changes, is relatively simple and this process can be carried out using the 2D Itoh Phase unwrapping algorithm. However, as soon as any single one of these three conditions is violated, then phase unwrapping can become a very difficult task to accomplish. This is due to the fact that an error in processing one pixel in a wrapped phase image affects other pixels within the image because of error propagation. A wrapped phase image may contain hundreds of thousands, or even millions of pixels. A phase unwrapping algorithm that processes one single pixel incorrectly may make all the phase information in the image unusable. This makes the 2D phase unwrapping process such a difficult task to perform correctly.

Researchers have developed many algorithms to phase unwrap real images that contain sources of errors. A number of these algorithms are explained in detail in [2]. We have experimentally investigated many of these algorithms and have found that the Flynn algorithm that uses a quality map, and the weighted L^p algorithm both produce very robust results. You can find the C language source code of both algorithms in [2, 5]. The execution times of both algorithms are high, as unsurprisingly this robustness comes at the cost of slow speeds. Both of these algorithms are capable of phase unwrapping the extremely difficult example image that is shown in Figure 13(c).

At the General Engineering Research Institute at LJMU we have developed a robust phase unwrapping algorithm with a very low execution time, called the 2D-SRNCP algorithm [3]. We have found that this algorithm is very suitable for use in many practical applications.

References

1. K. Itoh, “Analysis of the phase unwrapping problem,” Applied Optics, Vol. 21, No. 14, p. 2470, July 15, 1982.
2. Dennis Ghiglia and Mark Pritt, Two-dimensional phase unwrapping theory, algorithms and applications, John Wiley & Sons, 1998.
3. M. Arevalillo Herráez, D. R. Burton, M. J. Lalor and M. A. Gdeisat, "A Fast two dimensional phase unwrapping algorithm based on sorting by reliability following a non-continuous path," Applied Optics, Vol. 41, No. 35, pp 7437-7444, 2001.
4. http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem
5. ftp://ftp.wiley.com/public/sci_tech_med/phase_unwrapping/