Multiplexing and Demultiplexing Signals for Radiography Application Using the Discrete Fourier Transform

Grant T. Gullberg,[1,2] Michael Fuller,[3] and Youngho Seo,[2]

[1]Lawrence Berkeley National Laboratory,

Molecular Biophysics and Integrated Bioimaging Division,

[2]UCSF Physics Research Laboratory,

Department of Radiology and Biomedical Imaging,

University of California San Francisco, CA, USA

[3]TF Instruments, Salinas, CA, USA

Technical Report LBNL - 2001642

Lawrence Berkeley National Laboratory

February 7, 2025

Abstract

Our goal is to develop an X-ray phase-contrast imaging system that can provide excellent soft tissue contrast of phase, attenuation, and small-angle scatter. We propose to replace the common system of G0, G1, and G2 gradings with a biprism array to replace the G1 grading and introduce a novel X-ray tube designed to replace the motion of the phase stepping grading G2. The proposed X-ray tube uses temporal multiplexing to provide simultaneous virtual “electronic phase stepping.” In this work the discrete Fourier transform is used to separate from the composite measurement individual X-ray phase contrast measurements sampled at different frequencies. The method performs a discrete Fourier transform of a composite refence sequence to obtain using the frequency amplitudes **calibration factors** needed to extract the X-ray phase contrast measurement amplitudes from the composite image. The composite reference sequence is the sum of the individual sequences, at different frequencies, with amplitudes of one. The method takes the discrete Fourier transform of this composite reference sequence; whereby, the amplitude of each frequency component is compared with the total sum of its stand-alone sequence amplitude. A **calibration factor** is determined so that the amplitude of this composite reference frequency times the **calibration factor** must equal the total sum of the sequence amplitude—the zero-frequency amplitude of the discrete Fourier transform of its stand-alone sequence. To demultiplex the composite measured signal these **calibration factors** are multiplied by the amplitudes of the frequency components of the discrete Fourier transform of the composite X-phase-contrast measurement to obtain the amplitude of each frequency encoded measurement. Using these **calibration factors**, we demonstrate with the discrete Fourier transform in Mathematica the extraction of individual images from a composite image that one would expect obtaining from our proposed new X-ray phase contrast imaging system. We then demonstrate as an example how using images from X-ray phase contrast data one can calculate phase, attenuation and the dark field images using grading phase step data supplied to use from Microworks, GmbH in Karlsruhe, Germany.

Key words: Multiplexing, demultiplexing, X-ray phase contrast imaging, telecommunications, discrete Fourier transform, attenuation, dark field, small angle scatter, visibility, differential phase

Preface

This report describes the method of programing in Mathematica the multiplexing and demultiplexing images that Mike Fuller proposed to acquire with his new X-ray tube design. I am primarily doing this because it took me some effort to figure out how to demultiplex images using the discrete Fourier transform, and felt necessary to write it down so it is not forgotten. While reading several papers in the literature, I had a lot of difficulty trying to figure out how to demultiplex using the discrete Fourier transform. Therefore, I have tried to include details of the method as I understand them into this report. I also included as an example what one would expect to obtain if one could demultiplex images acquired at different frequencies providing acquisitions at different phase steps. I show this by calculating the attenuation, dark field, and phase images from a X-ray phase stepping acquisition supplied by Microworks, GmbH, Karlsruhe, Germany. I have also included in appendices Mathematica code for demultiplexing and code for processing these X-ray phase stepping data.

I. INTRODUCTION

There is significant interest to develop interferometry-based X-ray phase contrast imaging systems that can provide highly resolved X-rays with spatially-modulated intensity to enable the full potential of X-ray optics to image phase, attenuation, and small angle scatter properties of soft tissue [1]. Common to these systems shown in Figure 1 is a source grading G0 for the production of multiple coherent X-ray sources, a G1 phase grading to generate a Talbot-Lau interference pattern, and a phase stepping grading G2 to provide images at different frequencies from which the Fourier transform of the image and reference intensity image can be decomposed into attenuation, phase, and small angle scatter images of the tissue.

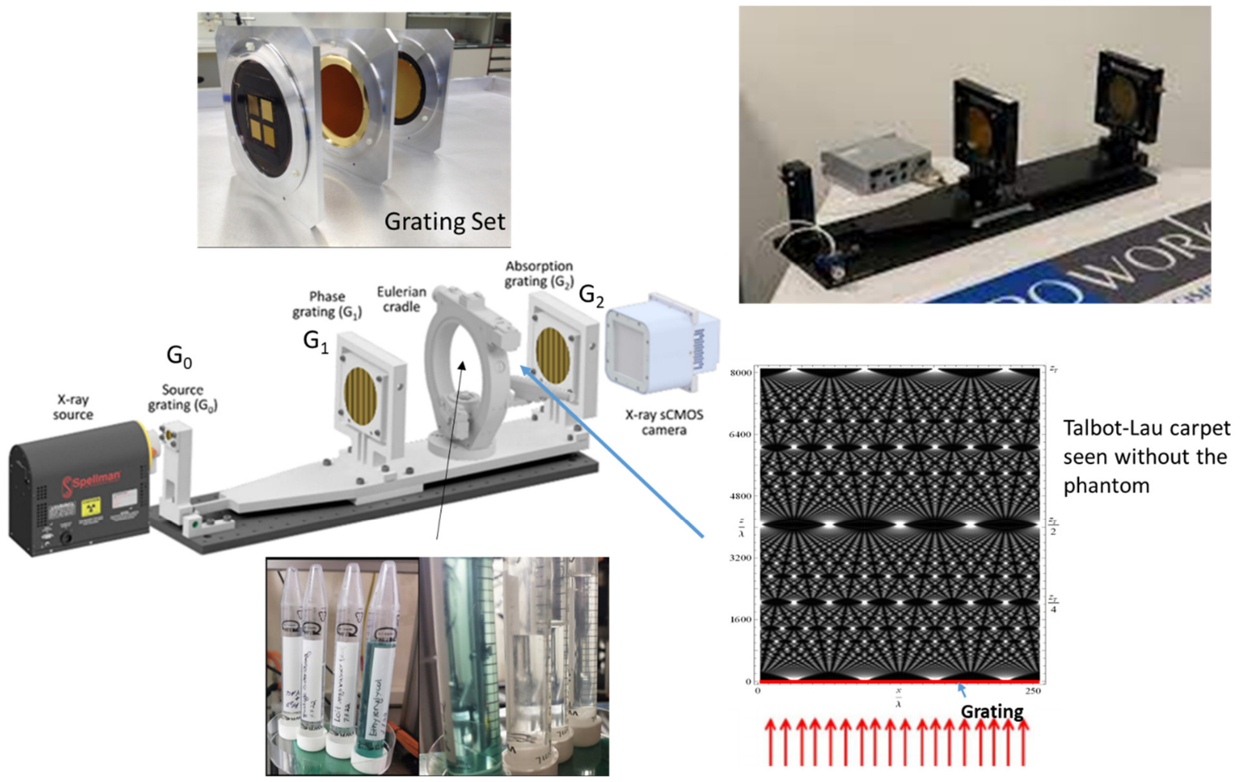


Figure 1. The X-ray phase contrast system with phantom and gradings at Microworks in Karlsruhe, Germany.

In our work [2, 3], we propose a new hardware concept shown in Figure 2 for X-ray phase contrast imaging wherein the phase grading is replaced with an array of Fresnel biprisms and the phase stepping grading is eliminated by using a specially designed X-ray tube that provides pulses of X-rays at different frequencies that can be acquired simultaneously to provide a multiplexed phase encoded image. The goal is to develop biprism interferometry imaging systems with excellent polychromatic performance that produce high-contrast fringes with spatially incoherent X-ray illumination.

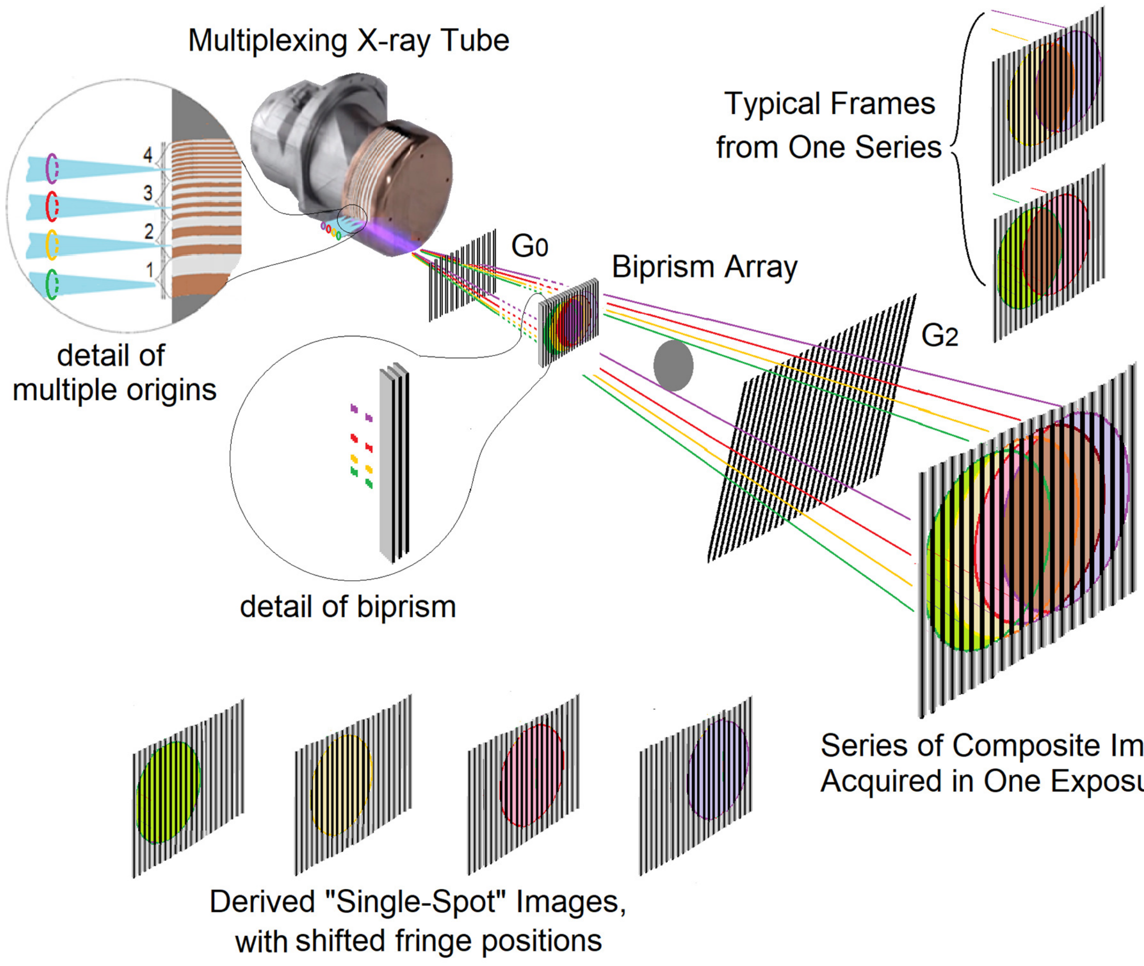

Multiplexing X-ray Tube
Typical Frames from One Series
4
3
2
1
G0
Biprism Array
G2
detail of multiple origins
detail of biprism
Series of Composite Images Acquired in One Exposure
Derived "Single-Spot" Images, with shifted fringe positions

Figure 2. Schematic diagram of a new hardware concept for X-ray phase contrast imaging. The phase grading (G_1) is replaced with an array of Fresnel biprisms and the motion of the phase stepping grading (G_2) is eliminated by using a specially designed X-ray tube with multiple spot origins from a patterned cylindrical rotating anode with oscillating translation for single-shot data acquisition [3]. (Image of X-ray tube courtesy of Rigaku Corp., Akishima-shi, Tokyo, Japan)

Multiplexing and demultiplexing signal transmission has been an active area of research due to many applications of telecommunications and especially so with the advent of the internet. Multiplexing is the process of combining multiple signals into a signal stream to be sent over a shared medium. In communication applications [4], multiplexing is a way of sending multiple signals over a communications network at the same time in the form of a single, complex signal. The signal can be digitally transported over a fiber optic cable or transported by analogue as a radio wave. When the signal reaches its destination, a process called demultiplexing recovers the separate signals.

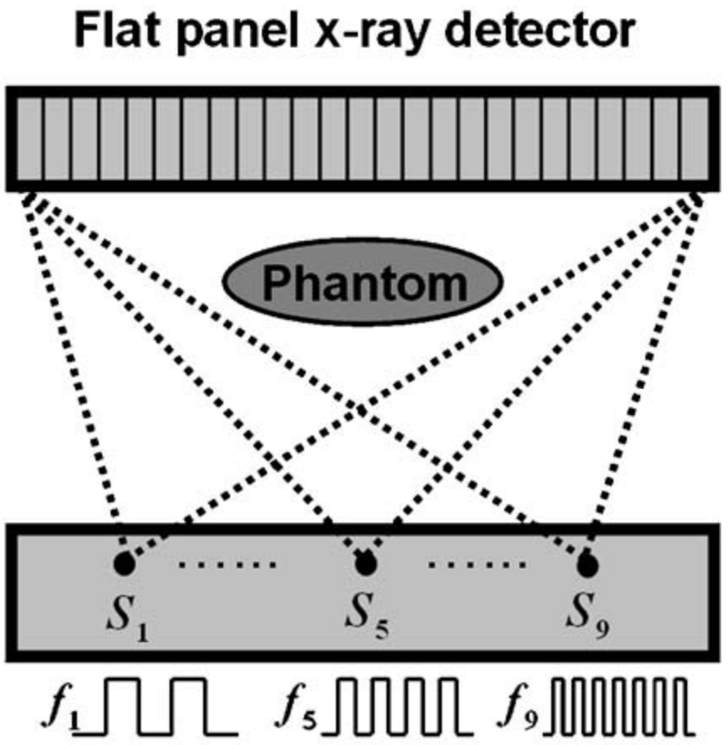


Figure 3. Multiplexing the X-ray Signal (Modified from [5])

In our context we illustrate in Figure 3 the idea of multiplexing using an example from Zhang et al. [5-8] where a detector acquires multiple signals from several X-ray tubes. The X-ray sources could be placed longitudinally relative to the detector (longitudinal tomography) as shown in Figure 3 or arranged at different angles for computed tomography applications. Each one of the X-ray tubes are pulsed at different frequencies and the images are acquired at a particular frame rate. The problem becomes that of demultiplexing the data into separate images as if they were acquired separately by each X-ray tube. In the following we provide the methods for demultiplexing the individual images using the discrete Fourier transform.

II. METHODS

Our method for demultiplexing images acquired simultaneously is demonstrated assuming we have four images shown in Figure 4 that are formed digitally by X-ray sources at different frequencies and acquired simultaneously at a frame rate of 32 frames per second. The discrete Fourier transform is used to extract the images from the composite by demultiplexing.

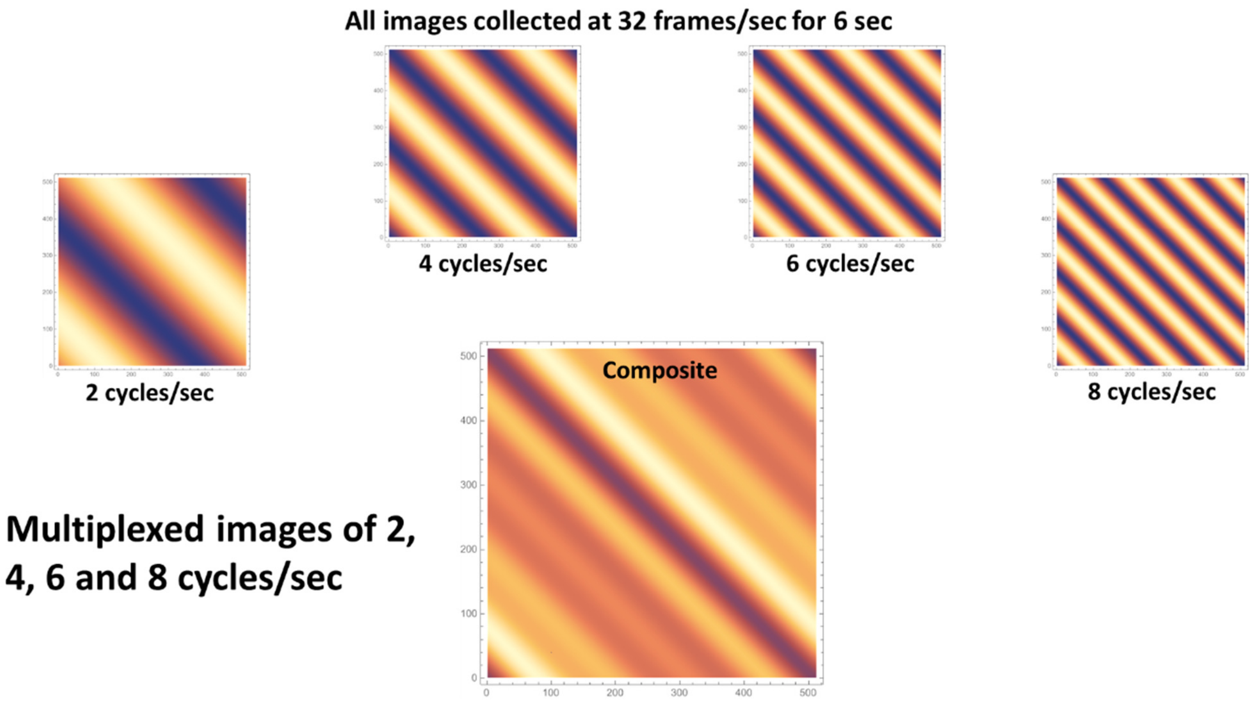


Figure 4. Simulation of images multiplexed to form a composite image by acquiring at 32 frames per second four images simultaneously pulsed at 2, 4, 6, 8 cycles/sec.

The first task is to run a reference time sequence for the desired acquisition protocol. In our example, we acquire 32 frames of data per second for 6 seconds. We want to multiplex the data so that we can acquire 4 images at the same time. We pulse one X-ray tube at 2 cycles per sec, another at 4 cycles per second, another at 6 cycles per second and a fourth X-ray tube at 8 cycles per second.

This means as shown in the Figure 5 that if a pulse of 2 cycles per second is applied, the X-ray tube is on 0.5 seconds and off 0.5 seconds. In the plot, 32-time segments correspond to one second, which amounts at 32 frames per second for an acquisition of 16 counts (1 count for each frame when X-ray is on) when the X-ray tube is on; and 16 counts are not acquired when the X-ray tube is off. This means 16 frames of data are acquired in one second. Likewise in Figure 6, for a 4 cycle per second pulse, the X-ray tube is on for 0.25 seconds and off for 0.25 seconds, which amounts to the acquisition of the same amount of 16 counts of data in one second.

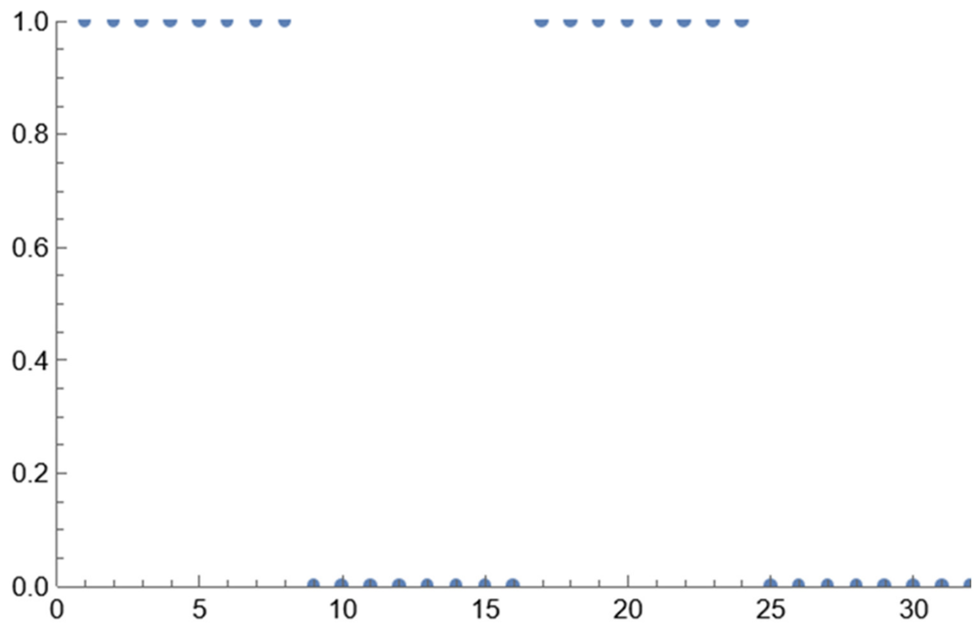


Figure 5. X-ray tube is on 0.5 seconds and off 0.5 seconds providing a pulse of 2 cycles per second If 32 frames of data are acquired/second, then 16 counts are acquired in one second.

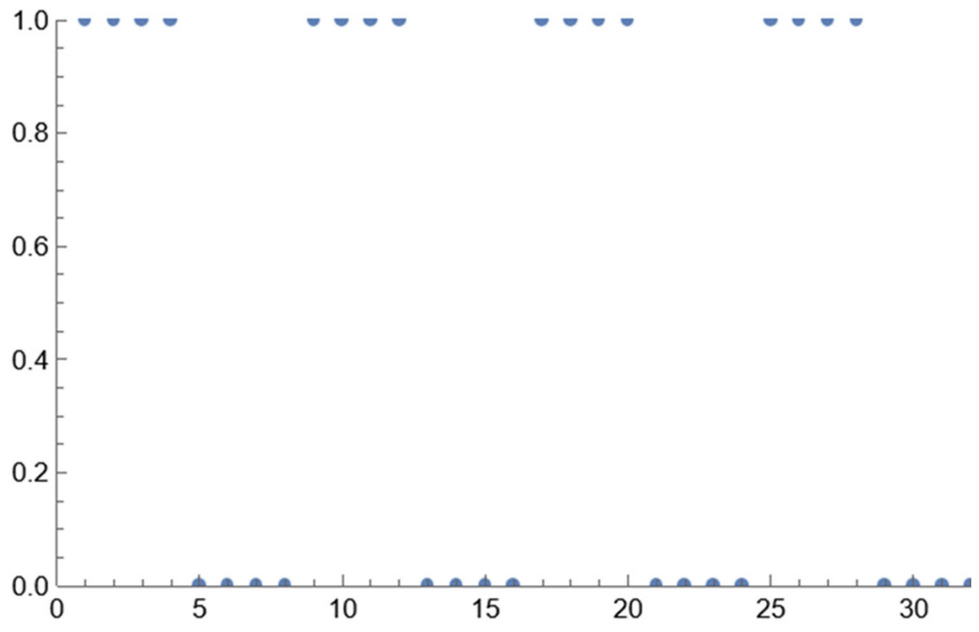


Figure 6. X-ray tube is on for 0.25 seconds and off for 0.25 seconds providing a pulse of 4 cycles/second. If 32 frames of data are acquired /second, then the same number as in Figure 5 of 16 counts are acquired in one second.

For our calibration we assume that data are collected for 6 seconds. This means that for a 2 cycle per second pulse, the X-ray tube is on for 3 seconds while data are being acquired at 32 frames per second and off for 3 seconds. Assuming one count is acquired for each frame of acquisition, this amounts to a total of 96 counts in 3 seconds when the X-ray tube is on. In our plot in Figure 7, we have 192-time points for 6 seconds, which amounts to 96 counts that are acquired when the X-ray tube is turned on for 6 seconds. For 4 cycles per second pulse, the X-ray tube is again on for 3 seconds and off for 3 seconds in 6 seconds, dividing on and off of the X-ray tube into 24 on-off segments that amounts to the same number of counts of 96 counts acquired at 32 frames per second. Likewise for 6 cycles per second, the X-ray tube is on for 3 seconds and off for 3 seconds in 6 seconds, dividing the on and off of the X-ray tube into 36 on-off segments that amounts to the same number of counts of 96 acquired at 32 frames per second. Likewise for 8 cycles per second, the X-ray tube is on for 3 seconds and off for 3 seconds in 6 seconds, dividing the on and off of the X-ray tube into 48 on-off segments that amounts to the same number of counts of 96 acquired at 32 frames per second.

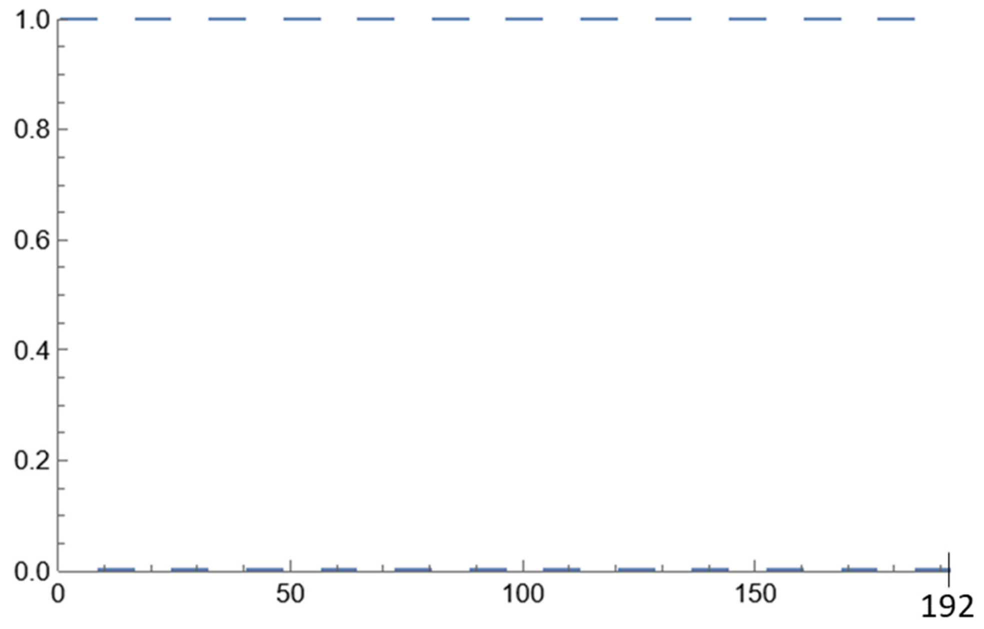


Figure 7. Data collected at 32 frames/second for 6 seconds for a 2 cycles per second pulse shown with 192-time units. Assuming one count is acquired for each frame of acquisition this amounts to a total of 96 counts in 3 seconds when the X-ray tube is on.

Next let's create discrete time sequences of 2, 4, 6, 8 cycles/sec. We program the time sequences for 192 points using the following Mathematica code (Notice that the amplitude is 1 when the sequence is on.):

For 2 hz: For[I=1, I12, I++, For[J=1, J8, J++, wcn1[[(I-1)(16)+J]]=1.]]

For 4 hz: For[I=1, I24, I++, For[J=1, J, J++, wcn1[[(I-1)(8)+J]]=1.]]

For 6 hz: For[I=1, I32, I++, For[J=1, J3, J++, wcn1[[(I-1)(6)+J]]=1.]]

For 8 hz: For[I=1, I48, I++, For[J=1, J, J++, wcn1[[(I-1)(4)+J]]=1.]]

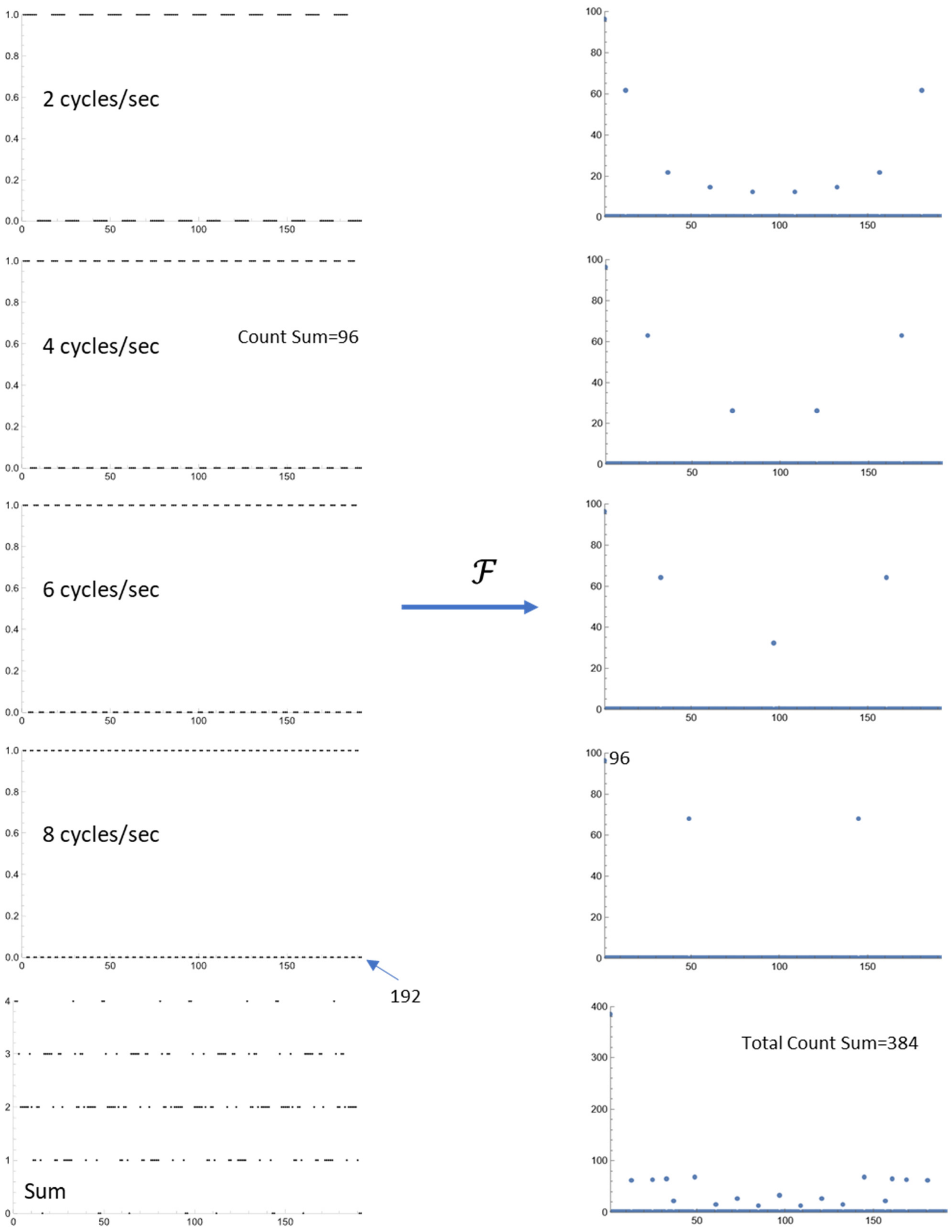


Figure 8. Fourier transform of sequences 2, 4, 6, 8 pulses/second and sum of the sequences all acquired at 32 frames/second.

Next, we take discrete Fourier transforms of these discrete sequences showed in Figure 8. We see in all cases that the maximum value of the discrete Fourier transform is 96, equal to the sum of the number of counts in each sequence with an amplitude of 1. Notice for 2 cycles/sec that the amplitude of the frequence is at 13, for 4 cycles/sec the amplitude is at 25, for 6 cycles/sec the amplitude is at 33, and for 8 cycles/sec the amplitude is at 49. When we sum these 4-time sequences together at each time point we obtain the sum of the time sequences and its discrete Fourier transform shown in Figures 8 and 9. The discrete Fourier transform has an amplitude for the zero frequency of 384 which is the total counts for the four-time sequences, each with total counts of 96 counts. (Note: the amplitude after taking the discrete Fourier transform must be multiplied by the $\sqrt{192}$ – the square root of the number of time points in the time sequence.)

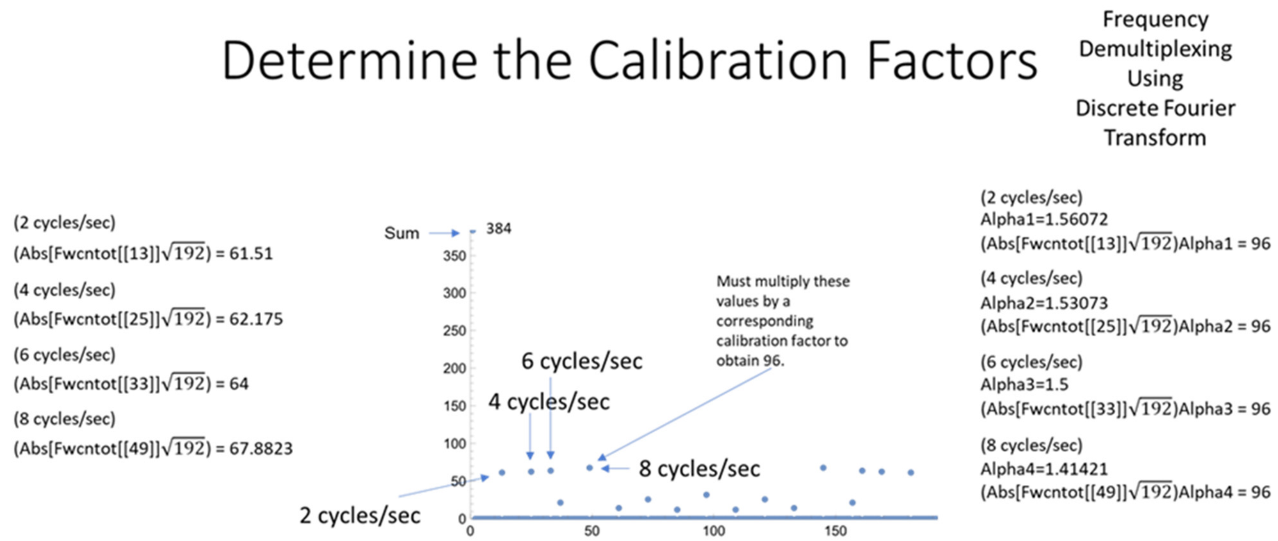


Figure 9. Calibration factors are calculated for each frequency for demultiplexing each sequence from the composite sequence. Notice, for example, Alpha1 × 61.51 = 1.56072 × 61.51 = 96.

If we are to extract (demultiplex) a particular time sequence from the sum of the sequences, we must calculate a calibration factor. For example, the frequency spectrum for 2 cycles/sec has an amplitude of 61.51. We need to multiply this by Alpha1=1.56072 to obtain a value of 96 (In our code: (Abs[Fwcntot[[13]])Alpha1=96) which is the total sum of the sequence acquired at 2 cycles/sec. To demultiplex the 2 cycles/sec sequence from the composite image, we want to multiply the amplitude value at frequency index equal to 13 (2 cycle/sec) of the discrete Fourier transform of the composite sequence by the calibration factor Alpha1. This gives the sum of the sequence values if it were to acquire separately the sequence at a frequency of 2 cycles/sec.

Let's look at this more closely for the example of our images in Figure 10 that are multiplexed (summed together) with simultaneous pulses of 2, 4, 6, and 8 cycles/sec with an acquisition of 32 frames/sec. The amplitude of the images is sinusoidal and formed with different frequencies of amplitudes across the image so they could be distinguished as different images. It just happens that the frequency of the amplitude across the image is similar to the frequency of the acquired amplitude for each pixel in the image. So, the spatial functions for the 2D (512×512) images are

$$h(x,y) = \cos\left[\frac{(x+y)\,\pi}{256} - \frac{\pi}{2}\right] + 1 \quad for\ 2\ cycles/sec$$

$$h(x,y) = \cos\left[\frac{(x+y)\,\pi}{128} - \pi\right] + 1 \quad for\ 4\ cycles/sec$$

$$h(x,y) = \cos\left[\frac{(x+y)\,\pi}{85} - \pi\right] + 1 \quad for\ 6\ cycles/sec$$

$$h(x,y) = \cos\left[\frac{(x+y)\,\pi}{64} - \pi\right] + 1\ for\ 8\ cycles/sec$$

Suppose we want to demultiplex from the composite image in Figure 10 the image value acquired at 2 cycles/sec for pixel (100,40). We know from our test sequence that our calibration factor is Alpha1=1.56072. In Figure 11 we see the Fourier spectrum of the total image sequence acquired at (100,40). The amplitude at the frequency index 13 is 122.354. If we multiply this by our calibration factor Alpha1 we get a value of 190.961. This should be the total sum of the sequence for pixel (100,40) acquired at 2 cycles/sec.

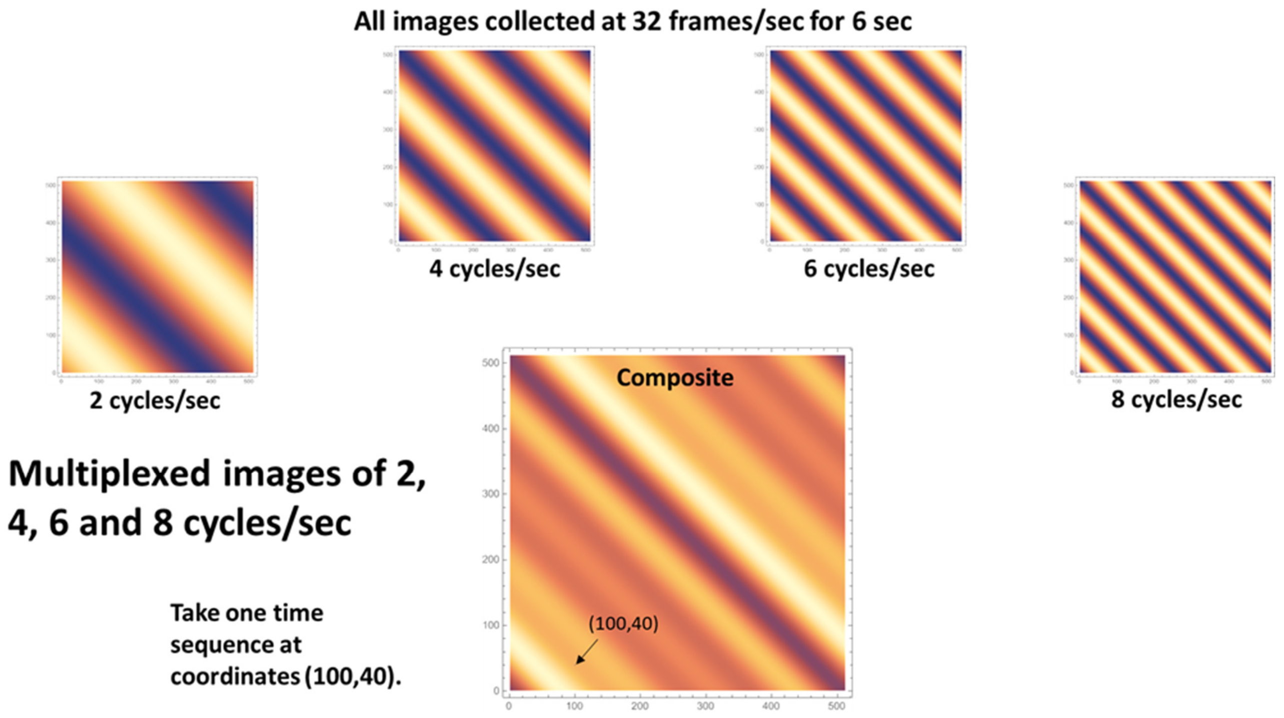


Figure 10. Composite image acquired at 32 frames/sec from individual image sequences of 2, 4, 6, 8 cycles/sec.

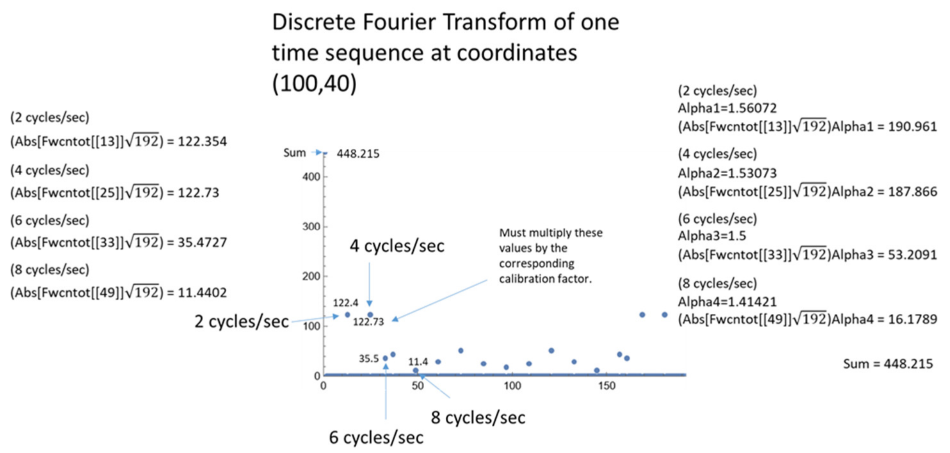


Figure 11. The Fourier spectrum of the total acquired sequence for the image pixel (100,40).

Therefore, to extract the image value at (100,40) from the composite at 2 cycles/sec we multiply the amplitude of the discrete Fourier transform at the frequency index equal to 13 by the calibration factor Alpha1=1.56072. Likewise, to extract the image values at (100,40) for 4, 6, and 8 cycles/sec, we multiply the frequency spectrum at the index 25 by Alpha2=1.53073, the frequency spectrum at the index 33 by Alpha3=1.5, and the frequency spectrum at the index 49 by Alpha4=1.41421, respectively. This is summarized in Figure 12 for the pixel (100,40). In Mathematica, we use the following code to demultiplex image values for all pixels from the composite (multiplexed) image:

```
wcn1ref =Table[
For [ IX= 1, IX 192, IX++, wcnx1 [IX]= wcntot[[ IX, JX, KX]]] ;
Fwcnx1= Fourier[ wcnx1] ;
(Abs [Fwcnx1[[13]]] Sqrt[192] Alpha1, {JX, 1, 512} ,{ KX, 1, 512}] ;
```

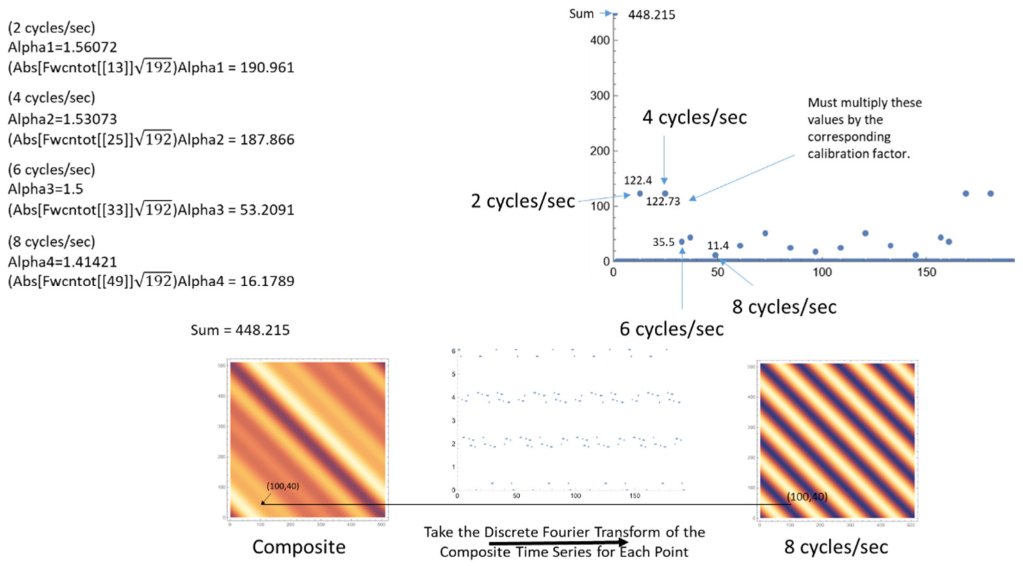

(2 cycles/sec)
Alpha1=1.56072
(Abs[Fwcntot[[13]]√192)Alpha1 = 190.961
(4 cycles/sec)
Alpha2=1.53073
(Abs[Fwcntot[[25]]√192)Alpha2 = 187.866
(6 cycles/sec)
Alpha3=1.5
(Abs[Fwcntot[[33]]√192)Alpha3 = 53.2091
(8 cycles/sec)
Alpha4=1.41421
(Abs[Fwcntot[[49]]√192)Alpha4 = 16.1789
Sum = 448.215
Sum 448.215
4 cycles/sec
2 cycles/sec
Must multiply these values by the corresponding calibration factor.
122.4
122.73
35.5
11.4
8 cycles/sec
6 cycles/sec
(100,40)
Composite
Take the Discrete Fourier Transform of the Composite Time Series for Each Point
(100,40)
8 cycles/sec

Figure 12. Frequency value for 8 cycles/sec of the discrete Fourier transform of the composite sequence at (100,40) is multiplied by the calibration factor Alpha4 to obtain the image pixel value at (100,40). This process is repeated for all pixels to form the image at 8 cycles/sec.

III. Results

Mathematica (Wolfram Research, Champaign, Illinois) code was used to demonstrate the demultiplexing of four images acquired simultaneously at 32 frames per second for 6 seconds. The data were multiplexed to obtain a composite of the 4 images. One image was sequenced at 2 cycles per sec, another at 4 cycles per second, another at 6 cycles per second, and a fourth at 8 cycles per second. The results of the composite image and demultiplexed images extracted from the composite are shown in Figure 13.

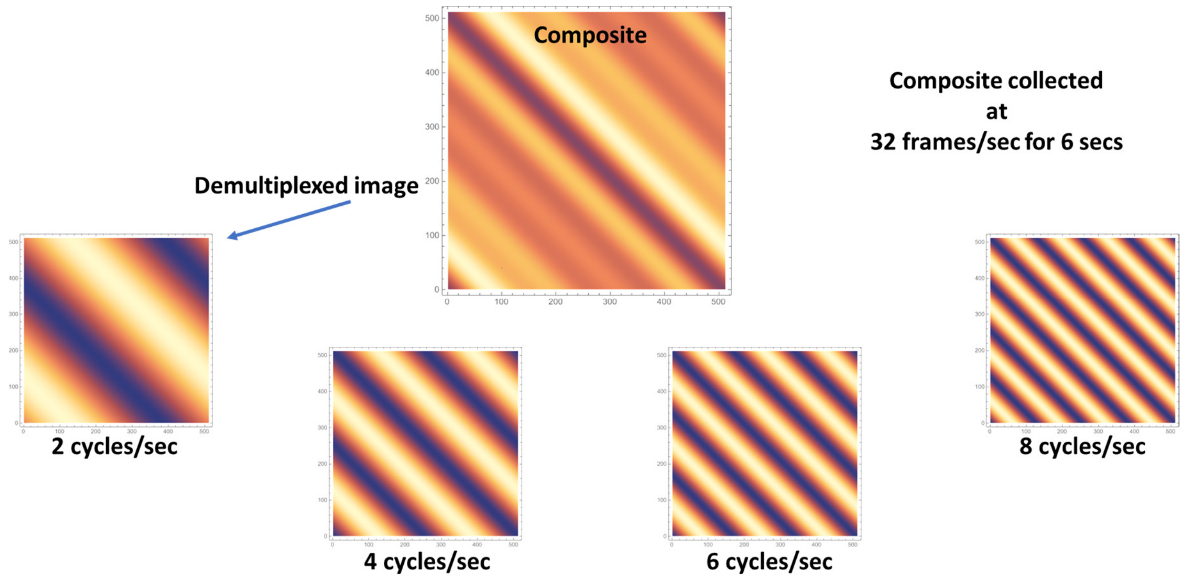


Figure 13. Images sequenced at 2, 4, 6, 8 cycles/sec were demultiplexed from a composite image collected at 32 frames /sec.

IV. Discussion

Here we provide the method with code (see Appendix I) for multiplexing and demultiplexing images acquired simultaneously. The method formulated in the context of four images acquired with an X-ray tube being pulsed at different frequencies and the data acquired simultaneously at 32 frames per second. Using the discrete Fourie transform allows us a straight forward method of extracting the images from a composite by demultiplexing. Our example was presented without assuming any noise in our acquisition. The effects of noise need to be further analyzed.

In the following we show how we can use the demultiplexed images at different phase steps (phase encodings) to obtain the attenuation, small angle scatter and the phase image of the projected image. For our example, we use data obtained from Microworks, GmbH, Karlsruhe, Germany. Note, these data were not multiplexed but are used here to show as an example what X-ray phase contrast data can provide, whether obtained by multiplexing or not. In Figure 14 we show projection images obtained from Microworks of a phantom obtained at four different phase steps. At each phase step there is a reference image without phantom and an image with phantom. A picture of the phantom is shown in Figure 1.

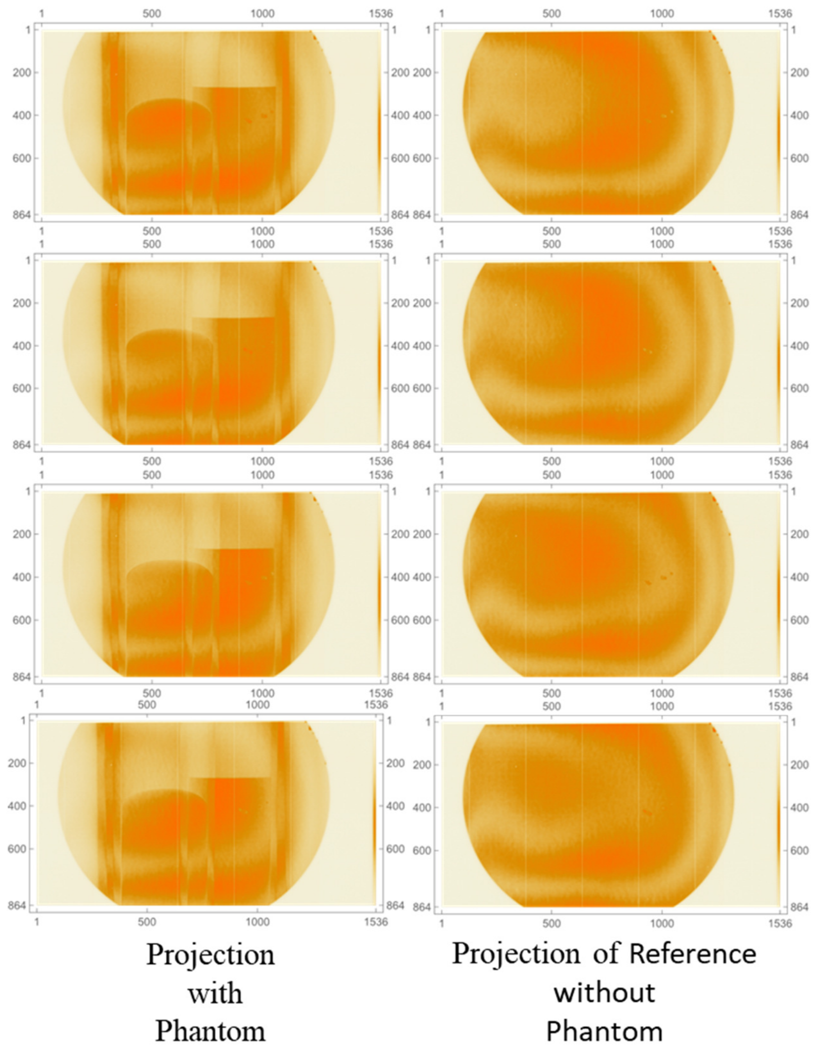


Figure 14. Projection images at four grading phase steps. These 4, increments of 8, of 32 phase step images were used to calculate attenuation, dark field, and phase images using Mathematica.

The projection of the irradiance distribution onto the detector surface is approximated by fitting four phase stepping projections to an approximated Fourier expansion [2]:

$$I_5(i,j,\theta_q,\phi_r,x_g) \approx a_0(i,j,\theta_q,\phi_r) + a_1(i,j,\theta_q,\phi_r) cos\left[\frac{2\pi(i-(I+1)/2)}{x_p}x_g - \Phi\left(i,j,\theta_q,\phi_r\right)\right] ,$$

where (i,j) are coordinates of the detector pixel $(0 \le i \le I), (0 \le j \le J)$; x_g is the spatial sampling in the direction of the phase grating; x_p is the period in *x*; (θ_q,ϕ_r) is the rotation angle of the sample around the optical axis; and a_0, a_1, and Φ are the mean, amplitude, and phase of the sinusoidal curve, respectively. In Figure 15 we see for one pixel of the image intensity matrix, the Fourier transform of the values taken over four phase steps provide the frequency coefficients a_0, a_1. The values for a_0 and a_1 in Figure 15 provide us with the Fourier expansion at the pixel (500,700).

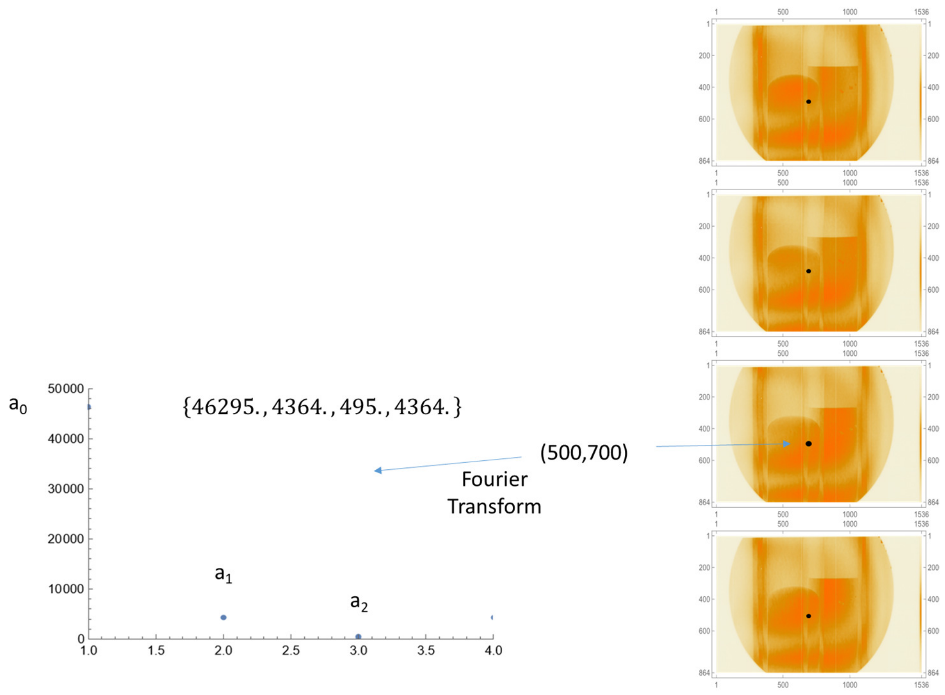


Figure 15. Inverse discrete Fourier transform of the values taken over four phase steps provide the frequency coefficients a_0, a_1 and a_2, for the pixel (500,700).

To obtain the attenuation image we use the following equation

$$p(i,j,\theta_q,\phi_r) = -\ln\left[\frac{a_0^{obj}(i,j,\theta_q,\phi_r)}{a_0^{ref}(i,j,\theta_q,\phi_r)}\right] = -\ln\left[\frac{I_{0,obj}(i,j,\theta_q,\phi_r)}{I_{0,ref}(i,j)}\right] ,$$

where $I_{0,obj}$ is the object intensity and $I_{0,ref}$ is the reference image intensity. In Figure 15, $I_{0,obj}$ is the zero component a_0 at pixel (500, 700) of the inverse Fourier transform of the sequence for the object. In our Mathematica code we take the inverse Fourier transform of every pixel value Datax[[k]] at the phase steps k=1,…, Nstep:

FDatx=Sqrt[Nsteps] × InverseFourier[Datax].

If we form FData[[I,J.k]]=FDatx[[k]], then the attenuation image DataAtn[[I,J]] is

DataAtn[[I,J]] = − Log[Abs[FData[[I,J,1]]/Abs[FRData[[I,J,1]] ,

where FRData[[I,J,1]] is the reference image and FData[[I,J,1]] is the image with phantom. Note that FData[[I,J,1] corresponds to the zero frequency FDatx[[1]] at coordinates (I,J), of the inverse Fourier transform of Datax. Figure 16 shows the resultant attenuation image for four phase steps.

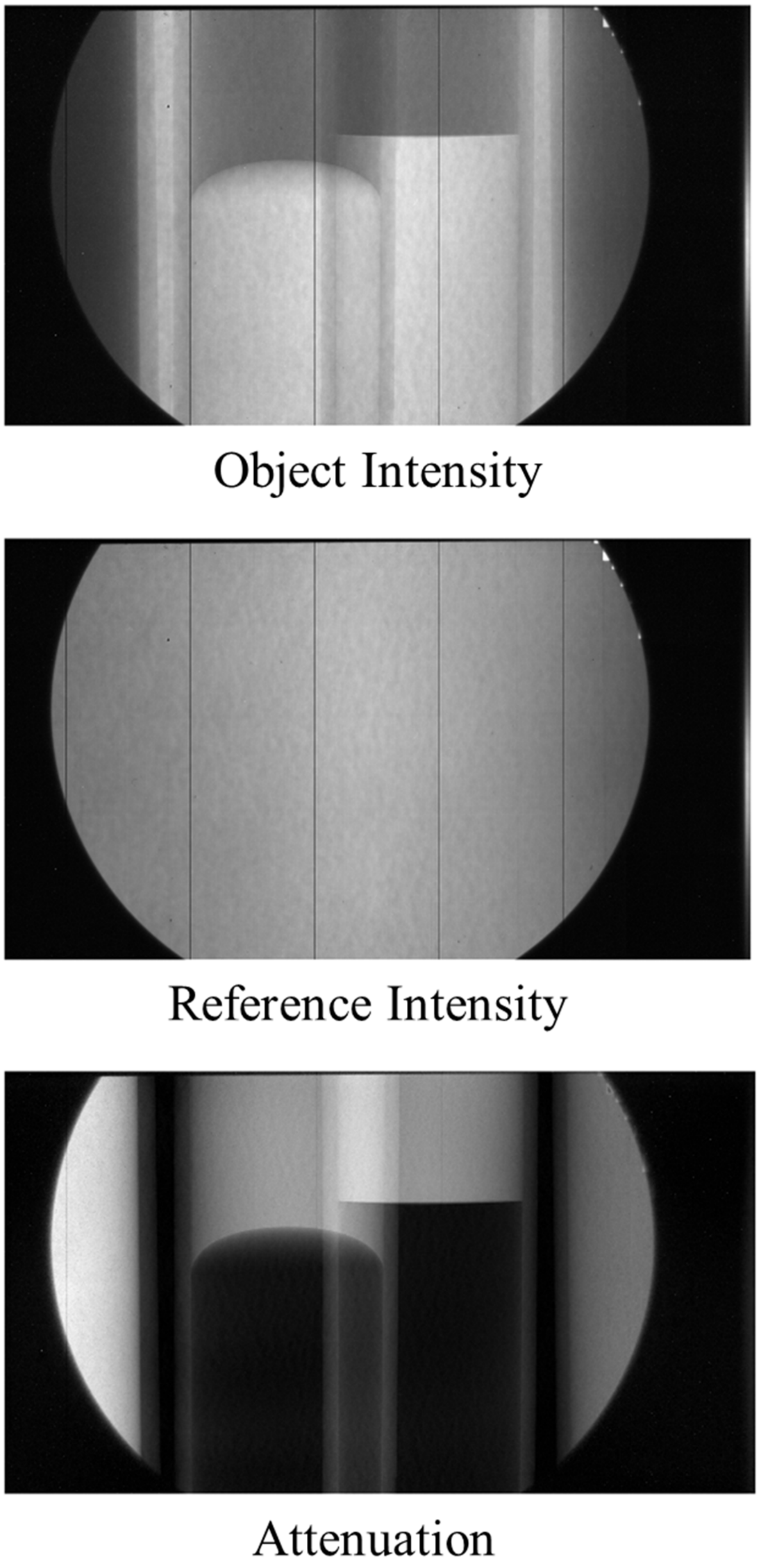


Figure 16. The object intensity, the reference intensity, and the attenuation image obtained from four phase steps. The object intensity and the reference intensity are the zero components of the inverse Fourier transform of the object and reference intensity for four phase steps.

To obtain the small angle scatter image we need to calculate the visibility image of the object and the reference image:

$$V_{obj}(i,j,\theta_q,\phi_r) = \frac{a_1^{obj}(i,j,\theta_q,\phi_r)}{a_0^{obj}(i,j,\theta_q,\phi_r)} \quad ,$$

$$V_{ref}(i,j,\theta_q,\phi_r) = \frac{a_1^{ref}(i,j,\theta_q,\phi_r)}{a_0^{ref}(\theta_q,\phi_r)} \quad ,$$

to obtain the ratio of the visibility image of the object and the reference shown in Figure 17. In our Mathematica code we obtain the ratios:

```
VisibilityObj[[I,J]]=FData[[I,J,2]]/FData[[I,J,1]]  ,
```

```
VisibilityRef[[I,J]]=FRData[[I,J,2]]/FRData[[I,J,1]]  .
```

In the matrices, the index 1 refers to the zero component of the inverse discrete Fourier transform and index 2 refers to the first order frequency of the inverse discrete Fourier transform. The small angle scatter image (dark field image) shown in Figure 18 is obtained from

$$m(i,j,\theta_q,\phi_r) = -\ln\left[\frac{V_{obj}(i,j,\theta_q,\phi_r)}{V_{ref}(i,j,\theta_q,\phi_r)}\right] \quad .$$

In our Mathematica code

```
SmallAng[[I,J]] = − Log[(FData[[I,J,2] × FRData[[I,J,1]])/(FData[[I,J,1]] × FRData[[I,J,2]])]  .
```

This gives the result for the small angle scatter image (dark field image) in Figure 18. Note that FData[[I,J,1]] is the zero frequency for the pixel at (I,J) and FData[[I,J,2]] is first frequency component for the pixel (I,J).

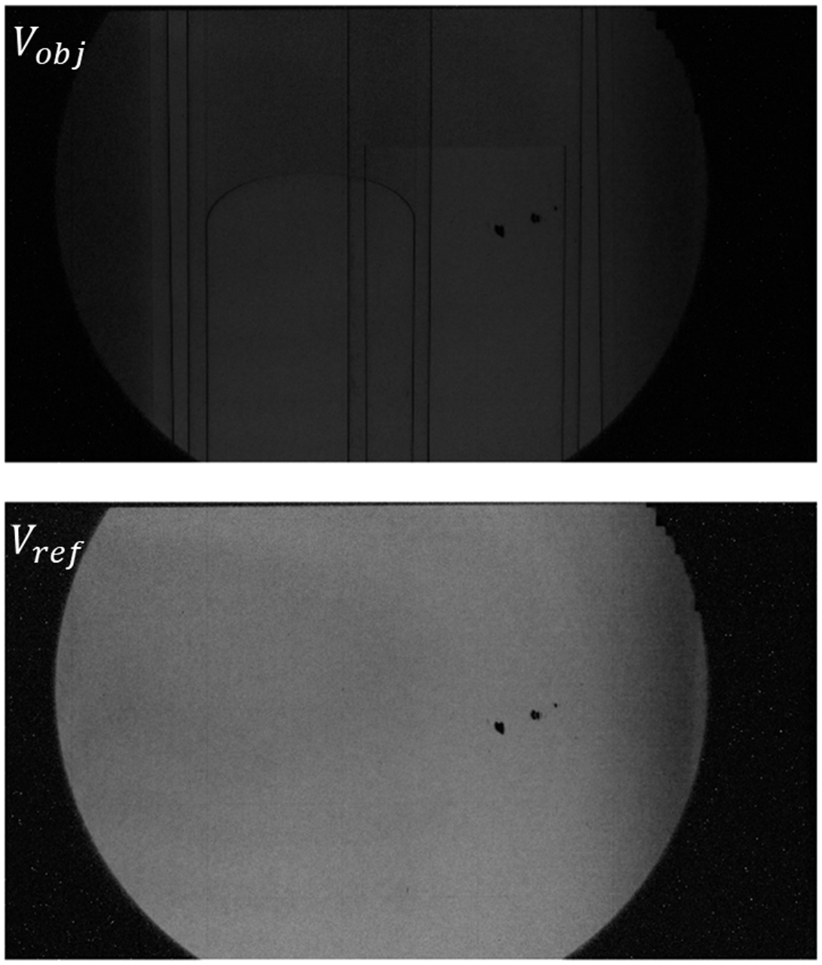


Figure 17. Visibility images of the object (phantom) and the reference without phantom.

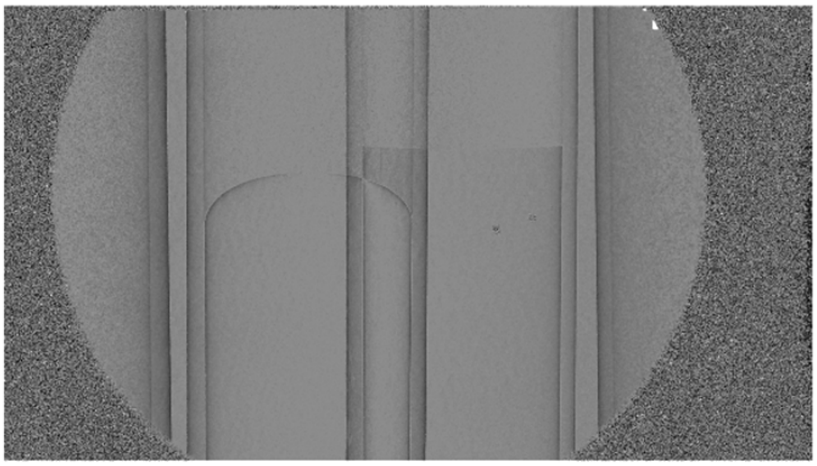


Figure 18. Dark field image calculated as the minus the log of the ratio of the visibility images of the object and the reference in Figure 17.

For the phase image we calculate the differential phase:

$$d_{\Phi_x}(i,j,\theta_q,\phi_r) = \Delta\Phi(i,j,\theta_q,\phi_r) = \Phi_{obj}(i,j,\theta_q,\phi_r) - \Phi_{ref}(i,j,\theta_q,\phi_r) \ ,$$

$$= \arg\,[\mathcal{F}_1^{-1}]_{obj}(i,j,\theta_q,\phi_r) - \arg[\mathcal{F}_1^{-1}]_{ref}\,(i,j,\theta_q,\phi_r) \ ,$$

where arg is the argument of the complex number. In our Mathematica code

DataPhase1[[I,J]] = If[Arg[FData[[I,J,2]]] <0, Arg[FData[[I,J,2]]] + 2, Arg[FData[[I,J,2]]]] ,

DataPhase2[[I,J]] = If[Arg[FRData[[I,J,2]]] <0, Arg[FRData[[I,J,2]]] + 2, Arg[FData[[I,J,2]]]] .

Note Arg[FRData[[I,J,2]]] is the phase Φ_{ref} without phantom and Arg[FData[[I,J,2]]] is the phase Φ_{obj} with phantom. The operation Arg[FData[[I,J,2]]] amounts to calculating the argument of the complex number FData[[I,J,2]], which is the arctan[y/x] = arctan[imaginary/real]. The test for less than zero is performed to avoid phase wrap while calculating the arctan operation in Mathematica. In Figure 19 we see the result for the reference phase Φ_{ref} and the phase with phantom Φ_{obj} and the differential phase $\Delta\Phi$ calculated using:

DataPhase[[I,J]] = DataPhase1[[I,J]] − DataPhase2[[I,J]].

Microworks, GmbH, Karlsruhe, Germany supplied us with 32 phase steps for both image with the phantom shown in Figure 1 and images without the phantom for reference in our calculations. In our previous examples we showed results using just four phase steps equally divided among the 32 phase steps. In Figure 20 we show the results using all 32 phase steps. Our results are compared with the results processed by Microworks using a Python code. One sees differences in contrast and brightness, but the structure is very much the same. At this time, we still have not resolved why there are differences in contrast.

Multiplexing is a method that has the potential to advance X-ray phase contrast imaging by deleting the cumbersome operation of phase stepping with mechanical gradings; and biprism arrays [3] have the potential to eliminate another grading to better utilize X-ray flux in the production of high-contrast fringes. X-ray phase contrast imaging also has the potential to measure directly the tensor field representing small angle scatter [9]. The development of new technology such as the proposed X-ray tube will further this development of X-ray phase contrast imaging and has significant potential to advance the technology of X-ray CT which might be envisioned in the next generation of X-ray CT scanners.

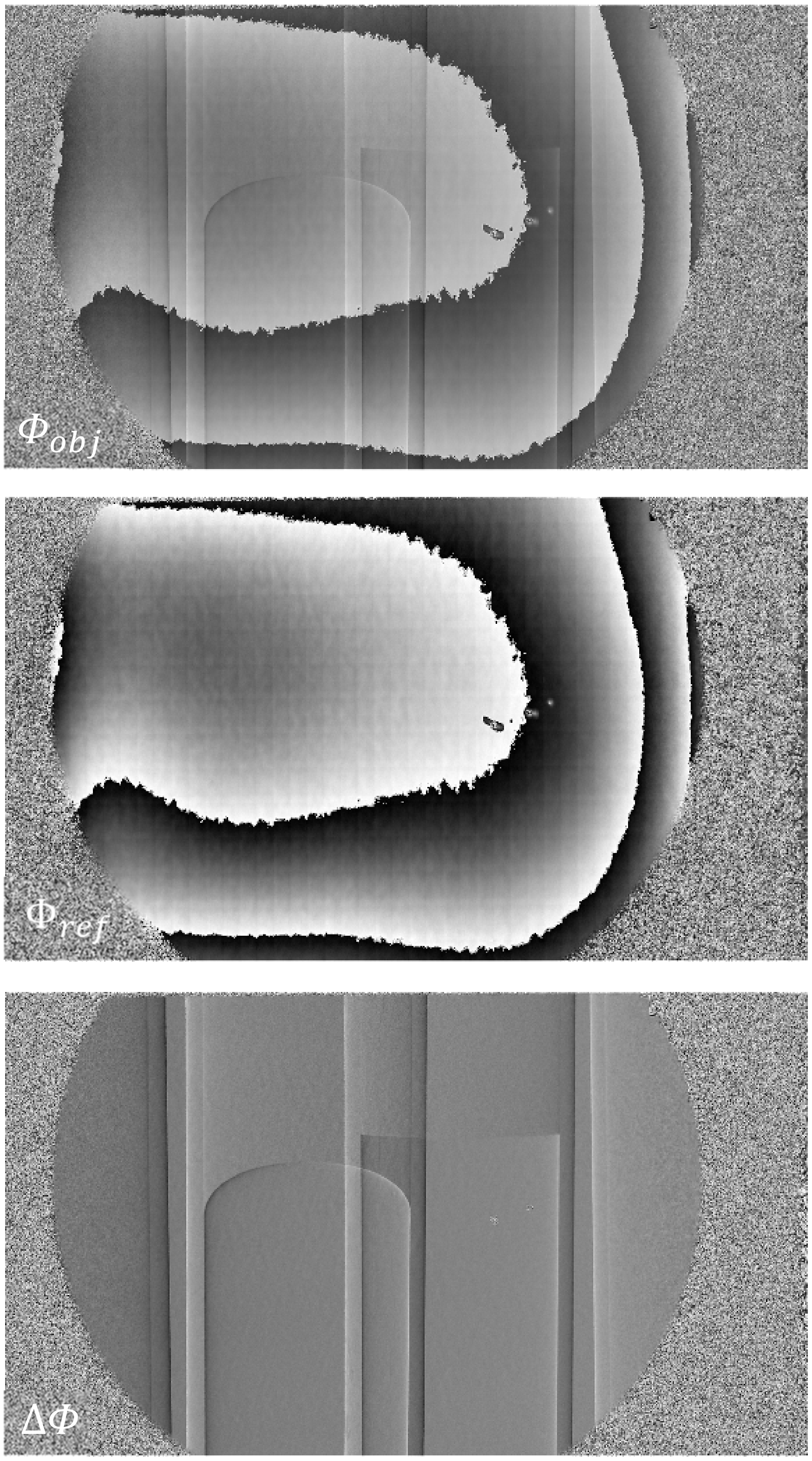


Figure 19. Differential phase image $\Delta\Phi$ of the phantom using the calculated phase Φ_{obj} of the inverse Fourier transform of the phantom image and the calculated phase Φ_{ref} of the inverse Fourier transform of the reference image; image without phantom.

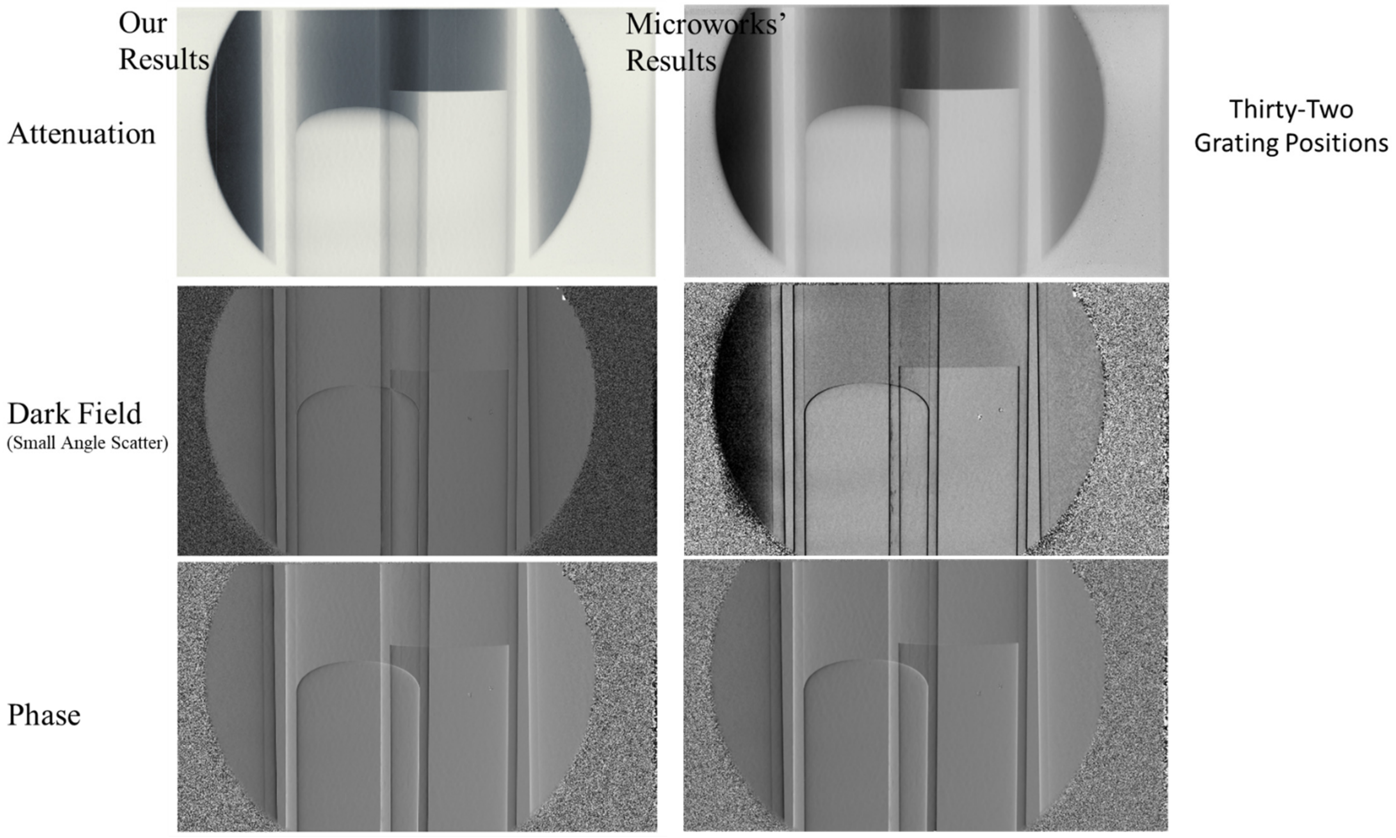


Figure 20. Comparison of our results with Microworks' results using data from Microworks of 32 phase steps collected for the phantom study illustrated in Figure 1. Note, the intensity for the attenuation image is reversed the intensity of that in Figure 16.

In addition to communication applications multiplexing is an active area of research in the application of single photon sources in quantum metrology, which is the study of making high-resolution and highly sensitive measurements of physical parameters using quantum theory to describe physical systems. A review of progress in single-photon sources based on multiplexing multiple probabilistic photon-creation events is presented in [10]. Such multiplexing allows higher single-photon probabilities and lower contamination from higher-order photon states. Multiplexing in such sources parallelizes the spontaneous photon creation in a number of different modes and then actively switches the photons into a single output mode based on feedback from the foreshadowing of the detection of events. The review presents several applications of single photon sources and the requirements for multiplexed sources and compares various approaches to multiplexing using different degrees of freedom. It is possible that source multiplexing can provide a quantum advantage over traditional computers.

References

[1] F. Pfeiffer, O. Bunk, C. David, M. Bech, G. L. Duc, A. Bravin. P. Cloetens, "High resolution brain tumor visualization using three-dimensional x-ray phase contrast tomography," Phys Med Biol, vol. 52, pp. 6923-30, 2007.

[2] W. Tao, Y. Sung, S. J. W. Kim, Q. Huang, G. T. Gullberg, Y. Seo, M. Fuller, "Tomography of dark field scatter including single-exposure Moiré fringe analysis with X-ray bi-prism interferometry – A simulation study," Med Phys, vol. 48, pp. 6293–11, 2021.

[3] G. T. Gullberg, U. Shrestha, S. J. W. Kim, Y. Seo, M. Fuller, "X-ray bi-prism interferometry--- A design study of proposed novel hardware," Med Phys, vol. 48, pp. 6508-23, 2021.

[4] A. V. Oppenheim, A. S. Willsky S. H. Nawab, "Signals & Systems [2nd Edition]," Pearson Education Limited, Edinburg Gate, Harlow, Essex CM20 2JE, England, 2014.

[5] J. Zhang et al., "Multiplexing radiography using a carbon nanotube-based x-ray source," Appl Phys Lett, vol. 89, pp. 1-3, 2006.

[6] J. Zhang, G. Yang, Y. Lee, S. Chang, J. P. Lu, O. Zhou, "Evaluation of frequency multiplexing radiography based on multi-pixel x-ray technology," Proc. SPIE 6510, Medical Imaging 2007: Physics of Medical Imaging, 65103Q, 19 March 2007.

[7] J. Zhang, G. Yang, Y. Lee, S. Chang, J. P. Lu, O. Zhou, "Multiplexing radiography based on carbon nanotube field emission x-ray technology," Proc. SPIE 6510, Medical Imaging 2007: Physics of Medical Imaging, 65100W, 14 March 2007.

[8] J Zhang, G Yang, Y Lee, S Chang, J Lu, O Zhou, "MO-E-L100J-07: Multiplexing Radiography for Ultra-Fast Computed Tomography: A Feasibility Study," Medical Physics, Volume 34, Issue 6Part16, pp. 2523-38, June 2007.

[9] W. Tao, L. Lyu, Y. Sung, G. T. Gullberg, M. Fuller, Y. Seo, Q. Huang, "X-ray small angle tensor tomography. Proceedings," 17th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine; pp. 386-9, 2023, Long Islang, NY, United States.

[10] E. Meyer-Scott, C. Silberhorn, A. Migdall, "Single-photon sources: Approaching the ideal through multiplexing," Rev Sci Instrum, Vol. 91(4), pp. 041101, 2020.

```
In[1]:= (***************Appendix I ***************)
        (*Mathematica Code for Multiplexing and Demultiplexing*)

        Clear[h1]; Clear[hcn1]; Clear[wcn1]; Clear[fcn1];

        Print["Collection at 32 frames/sec of a 2 cycle/sec pulse for 6 sec. "]
        Print["Hz1"]
        Hz1 = 2

        hcn1[x_, y_] := Cos[(x + y) π / 256 - π / 2] + 1.

        (*fcn1 is transpose of hcn1*) fcn1 = Table[hcn1[x, y], {x, 1, 512, 1}, {y, 1, 512, 1}];
        Print["sumwcn1=Sum[fcn1[[i,1]],{i,512}]"]
        sumwcn1 = Sum[fcn1⟦i, 1⟧, {i, 512}]

        (*N=192
          For[II=1,II≤12,II++,For[JJ=1,JJ≤8,JJ++,w1⟦(II-1)(16)+JJ⟧=1.]]2 hz;
        For[II=1,II≤24,II++,For[JJ=1,JJ≤4,JJ++,w2⟦(II-1)(8)+JJ⟧=1.]]4 hz;
        For[II=1,II≤32,II++,For[JJ=1,JJ≤3,JJ++,w3⟦(II-1)(6)+JJ⟧=1.]]6 hz;
        For[II=1,II≤48,II++,For[JJ=1,JJ≤2,JJ++,w4⟦(II-1)(4)+JJ⟧=1.]]8 hz;*)

        (*w1 is the reference time sequence with
         32 frames/sec of a 2 cycle/sec pulse for 6 sec.*)
        w1 = Table[0., {Ix, 1, 192}];
        For[II = 1, II ≤ 12, II++, For[JJ = 1, JJ ≤ 8, JJ++, w1⟦(II - 1) 16 + JJ⟧ = 1.]]
        Print["Reference time sequence w1 for 32 frames/sec of a 2 cycle/sec pulse for 6 sec"]
        ListPlot[w1, PlotRange → {{0, 192}, {0, 1}}, PlotStyle → PointSize[Small]]
        w1sum = 0.;
        For[IX = 1, IX ≤ 192, IX++, w1sum = w1sum + w1⟦IX⟧];
        Print["Sum of frames in 6 sec: w1sum"]; w1sum
        Fw1 = Table[0., {Ix, 1, 192}];
        Fw1 = Fourier[w1];
        Print[
         "Fourier spectrun Abs[Fw1]Sqrt[192]: 32 frames/sec of a 2 cycle/sec pulse for 6 sec"]
        ListPlot[Abs[Fw1] Sqrt[192],
         PlotRange → {{1, 192}, {0, 100}}, PlotStyle → PointSize[Medium]]
        Print["Fw1[[1]]Sqrt[192]"]; Fw1⟦1⟧ Sqrt[192]

        (*wcn1 is 3D array of time sequence and image
         for 32 frames/sec of a 2 cycle/sec pulse for 6 sec.*)
        wcn1 = Table[0., {IX, 1, 192}, {JX, 1, 512}, {KX, 1, 512}];
        For[JX = 1, JX ≤ 512, JX++, For[KX = 1, KX ≤ 512, KX++, For[II = 1, II ≤ 12, II++,
            For[JJ = 1, JJ ≤ 8, JJ++, wcn1⟦(II - 1) (16) + JJ, JX, KX⟧ = fcn1⟦JX, KX⟧]]]];

        (*wcnx1 is the image time sequence at corrdinates
         (40,100) with 32 frames/sec of a 2 cycle/sec pulse for 6 sec.*)
```

```
Print["fcn1[[40,100]]"]
fcn1⟦40, 100⟧
Print["Clear[wcnx1]"]
Clear[wcnx1];
wcnx1 = Table[0., {II, 1, 192}];
For[II = 1, II ≤ 192, II++, wcnx1⟦II⟧ = wcn1⟦II, 40, 100⟧]
Print["Image time squence at coordinates(40,100)with
   32 frames/sec of a 2 cycle/sec pulse for 6 sec"]
ListPlot[wcnx1, PlotRange → {{0, 192}, {0, fcn1⟦40, 100⟧}}, PlotStyle → PointSize[Small]]

(*wcn1ref is the image where each voxel is acquired
 for 6 sec at 32 frames/sec with a 2 cycle/sec pulse*)
wcn1ref = Table[0., {IX, 1, 512}, {JX, 1, 512}];
For[II = 1, II ≤ 512, II++,
 For[JJ = 1, JJ ≤ 512, JJ++, wcn1ref⟦II, JJ⟧ = Sum[wcn1⟦i, II, JJ⟧, {i, 192}]]]
Print["wcn1ref[[40,100]]"]
wcn1ref⟦40, 100⟧ = 0.
Print["Image wcn1ref where each voxel is
   acquired for 6 sec at 32 frames/sec with a 2 cycle/sec pulse"]
ListDensityPlot[wcn1ref, PlotRange → All]

Print["*****************************************************"]

Clear[h2]; Clear[hcn2]; Clear[wcn2]; Clear[fcn2];

Print["Collection at 32 frames/sec of a 4 cycle/sec pulse for 6 sec. "]
Print["Hz2"]
Hz2 = 4

hcn2[x_, y_] := Cos[(x + y) π / 128 - π] + 1.
fcn2 = Table[hcn2[x, y], {x, 1, 1024, 1}, {y, 1, 1024, 1}];
Print["sumwcn2=Sum[fcn2[[i,1]],{i,1024}]"]
sumwcn2 = Sum[fcn2⟦i, 1⟧, {i, 1024}]

(*N=192
  For[II=1,II≤12,II++,For[JJ=1,JJ≤8,JJ++,wcn1⟦(II-1)(16)+JJ⟧=1.]]2 hz;
For[II=1,II≤24,II++,For[JJ=1,JJ≤4,JJ++,wcn2⟦(II-1)(8)+JJ⟧=1.]]4 hz;
For[II=1,II≤32,II++,For[JJ=1,JJ≤3,JJ++,wcn3⟦(II-1)(6)+JJ⟧=1.]]6 hz;
For[II=1,II≤48,II++,For[JJ=1,JJ≤2,JJ++,wcn4⟦(II-1)(4)+JJ⟧=1.]]8 hz;*)

(*"w2 is the reference time sequence with 32 frames/sec
  of a 4 cycle/sec pulse for 6 sec."*)w2 = Table[0., {Ix, 1, 192}];
For[II = 1, II ≤ 24, II++, For[JJ = 1, JJ ≤ 4, JJ++, w2⟦(II - 1) 8 + JJ⟧ = 1.]]
Print["Reference time sequence w2 for 32 frames/sec of a 4 cycle/sec pulse for 6 sec"]
ListPlot[w2, PlotRange → {{0, 192}, {0, 1}}, PlotStyle → PointSize[Small]]
w2sum = 0.;
For[IX = 1, IX ≤ 192, IX++, w2sum = w2sum + w2⟦IX⟧];
```

```
Print["Sum of frames in 6 sec: w2sum"]; w2sum
Fw2 = Table[0., {Ix, 1, 192}];
Fw2 = Fourier[w2];
Print[
 "Fourier spectrum AbspFw2]Sqrt[192]: 32 frames/sec of 4 cycle/sec pulse for 6 sec"]
ListPlot[Abs[Fw2] Sqrt[192],
 PlotRange → {{1, 192}, {0, 100}}, PlotStyle → PointSize[Medium]]
Print["Fw2[[1]]Sqrt[192]"]; Fw2⟦1⟧ Sqrt[192]

(*wcn2 is the 3D array of time sequence and image
 for 32 frames/sec of a 4 cycle/sec pulse for 6 sec.*)
Print["wcn2=Table[0.,{IX,1,192},{JX,1,512},{KX,1,512}"]
wcn2 = Table[0., {IX, 1, 192}, {JX, 1, 512}, {KX, 1, 512}];
For[JX = 1, JX ≤ 512, JX++, For[KX = 1, KX ≤ 512, KX++, For[II = 1, II ≤ 24, II++,
    For[JJ = 1, JJ ≤ 4, JJ++, wcn2⟦(II - 1) (8) + JJ, JX, KX⟧ = fcn2⟦JX, KX⟧]]]];

(*wcnx2 is the image time sequence at coordinated
 (40,100) with 32 frames/ of a 4 cycle/sec pulse for 6 sec.*)
Print["fcn2[[40,100]]"]
fcn2⟦40, 100⟧
Print["Clear[wcnx2]"]
Clear[wcnx2];
wcnx2 = Table[0., {II, 1, 192}];
For[II = 1, II ≤ 192, II++, wcnx2⟦II⟧ = wcn2⟦II, 40, 100⟧]
Print["Image time squence at coordinates(40,100)with
   32 frames/sec of a 4 cycle/sec pulse for 6 sec"]
ListPlot[wcnx2, PlotRange → {{0, 192}, {0, fcn2⟦40, 100⟧}}, PlotStyle → PointSize[Small]]

(*wcn2ref is the image where each voxel is acquired
 for 6 sec at 32 frames/sec with a 4 cycle /sec pulse*)
wcn2ref = Table[0., {IX, 1, 512}, {JX, 1, 512}];
For[II = 1, II ≤ 512, II++,
 For[JJ = 1, JJ ≤ 512, JJ++, wcn2ref⟦II, JJ⟧ = Sum[wcn2⟦i, II, JJ⟧, {i, 192}]]]
Print["wcn2ref[[40,100]]"]
wcn2ref⟦40, 100⟧ = 0.
Print["Image wcn2ref where each voxel is
   acquired for 6 sec at 32 frames/sec with a 4 cycle/sec pulse"]
Print["ListDensityPlot[wcn2ref,PlotRange→All]"]
ListDensityPlot[wcn2ref, PlotRange → All]

Print["********************************************************"]

Clear[h3]; Clear[hcn3]; Clear[wcn3]; Clear[fcn3];

Print["Collection at 32 frames/sec of a 6 cycle/sec pulse for  6 sec. "]
Print["Hz3"]
Hz3 = 6
```

```
hcn3[x_, y_] := Cos[(x + y) π / 85 - π] + 1.
fcn3 = Table[hcn3[x, y], {x, 1, 1024, 1}, {y, 1, 1024, 1}];
Print["sumwcn3=Sum[fcn3[[i,1]],{i,1024}]"]
sumwcn3 = Sum[fcn3⟦i, 1⟧, {i, 1024}]

(*N=192
  For[II=1,II≤12,II++,For[JJ=1,JJ≤8,JJ++,wcn1⟦(II-1)(16)+JJ⟧=1.]]2 hz;
For[II=1,II≤24,II++,For[JJ=1,JJ≤4,JJ++,wcn2⟦(II-1)(8)+JJ⟧=1.]]4 hz;
For[II=1,II≤32,II++,For[JJ=1,JJ≤3,JJ++,wcn3⟦(II-1)(6)+JJ⟧=1.]]6 hz;
For[II=1,II≤48,II++,For[JJ=1,JJ≤2,JJ++,wcn4⟦(II-1)(4)+JJ⟧=1.]]8 hz;*)

(*"w3 is the reference time sequence with 32 frames/sec
  of a 6 cycle/sec pulse for 6 sec."*)w3 = Table[0., {Ix, 1, 192}];
For[II = 1, II ≤ 32, II++, For[JJ = 1, JJ ≤ 3, JJ++, w3⟦(II - 1) 6 + JJ⟧ = 1.]]
Print["Reference time sequence w3 for 32 frames/sec of a 6 cycle/sec pulse for 6 sec"]
ListPlot[w3, PlotRange → {{0, 192}, {0, 1}}, PlotStyle → PointSize[Small]]
w3sum = 0.;
For[IX = 1, IX ≤ 192, IX++, w3sum = w3sum + w3⟦IX⟧];
Print["Sum of frames in 6 sec: w3sum"]; w3sum
Fw3 = Table[0., {Ix, 1, 192}];
Fw3 = Fourier[w3];
Print[
 "Fourier spectrum Abs[Fw3]Sqrt[192]: 32 frames/sec of a 6 cycle/sec pulse for 6 sec"]
ListPlot[Abs[Fw3] Sqrt[192],
 PlotRange → {{1, 192}, {0, 100}}, PlotStyle → PointSize[Medium]]
Print["Fw3[[1]]Sqrt[192]"]; Fw3⟦1⟧ Sqrt[192]

(*wcn3 is 3D array of time sequence and image
 for 32 frames/sec of a 6 cycle/sec pulse for 6 sec.*)
wcn3 = Table[0., {IX, 1, 192}, {JX, 1, 512}, {KX, 1, 512}];
For[JX = 1, JX ≤ 512, JX++, For[KX = 1, KX ≤ 512, KX++, For[II = 1, II ≤ 32, II++,
    For[JJ = 1, JJ ≤ 3, JJ++, wcn3⟦(II - 1) (6) + JJ, JX, KX⟧ = fcn3⟦JX, KX⟧]]]];

(*wcnx3 is the image time sequence at corrdinates
 (40,100) with 32 frames/sec of a 6 cycle/sec pulse for 6 sec.*)
Print["fcn3[[40,100]]"]
fcn3⟦40, 100⟧
Print["Clear[wcnx3]"]
Clear[wcnx3];
wcnx3 = Table[0., {II, 1, 192}];
For[II = 1, II ≤ 192, II++, wcnx3⟦II⟧ = wcn3⟦II, 40, 100⟧]
Print["Image time squence at coordinates(40,100)with
   32 frames/sec of a 6 cycle/sec pulse for 6 sec"]
ListPlot[wcnx3, PlotRange → {{0, 192}, {0, fcn3⟦40, 100⟧}}, PlotStyle → PointSize[Small]]

(*wcn3ref is the image where each voxel is acquired
 for 6 sec at 32 frames/sec with a 6 cycle/sec pulse*)
wcn3ref = Table[0., {IX, 1, 512}, {JX, 1, 512}];
For[II = 1, II ≤ 512, II++,
 For[JJ = 1, JJ ≤ 512, JJ++, wcn3ref⟦II, JJ⟧ = Sum[wcn3⟦i, II, JJ⟧, {i, 192}]]]
Print["wcn3ref[[40,100]]"]
wcn3ref⟦40, 100⟧ = 0.
Print["Image wcn3ref where each voxel is
   acquired for 6 sec at 32 frames/sec with a 6 cycle/sec pulse"]
ListDensityPlot[wcn3ref, PlotRange → All]

Print["*******************************************************"]

Clear[h4]; Clear[hcn4]; Clear[wcn4]; Clear[fcn4];

Print["Collection at 32 frames/sec of a 8 cycle/sec pulse for 6 sec. "]
Print["Hz4"]
Hz4 = 8

hcn4[x_, y_] := Cos[(x + y) π / 64 - π] + 1.
fcn4 = Table[hcn4[x, y], {x, 1, 1024, 1}, {y, 1, 1024, 1}];
Print["sumwcn4=Sum[fcn4[[i,1]],{i,1024}]"]
sumwcn4 = Sum[fcn4⟦i, 1⟧, {i, 1024}]

(*N=192
  For[II=1,II≤12,II++,For[JJ=1,JJ≤8,JJ++,wcn1⟦(II-1)(16)+JJ⟧=1.]]2 hz;
For[II=1,II≤24,II++,For[JJ=1,JJ≤4,JJ++,wcn2⟦(II-1)(8)+JJ⟧=1.]]4 hz;
For[II=1,II≤32,II++,For[JJ=1,JJ≤3,JJ++,wcn3⟦(II-1)(6)+JJ⟧=1.]]6 hz;
For[II=1,II≤48,II++,For[JJ=1,JJ≤2,JJ++,wcn4⟦(II-1)(4)+JJ⟧=1.]]8 hz;*)

(*"w4 is the reference time sequence with 32 frames/sec
  of a 8 cycle/sec pulse for 6 sec."*)w4 = Table[0., {Ix, 1, 192}];
For[II = 1, II ≤ 48, II++, For[JJ = 1, JJ ≤ 2, JJ++, w4⟦(II - 1) 4 + JJ⟧ = 1.]]
Print["Reference time sequence w4 for 32 frames/sec of a 8 cycle/sec pulse for 6 sec"]
ListPlot[w4, PlotRange → {{0, 192}, {0, 1}}, PlotStyle → PointSize[Small]]
w4sum = 0.;
For[IX = 1, IX ≤ 192, IX++, w4sum = w4sum + w4⟦IX⟧];
Print["Sum of frames in 6 sec: w4sum"]; w4sum
Fw4 = Table[0., {Ix, 1, 192}];
Fw4 = Fourier[w4];
Print[
 "Fourier spectrum Abs[Fw4]Sqrt[192]: 32 frames/sec of a 8 cycle/sec pulse for 6 sec"]
ListPlot[Abs[Fw4] Sqrt[192],
 PlotRange → {{1, 192}, {0, 100}}, PlotStyle → PointSize[Medium]]
Print["Fw4[[1]]Sqrt[192]"]; Fw4⟦1⟧ Sqrt[192]

(*wcn4 is 3D array of time sequence and image
 for 32 frames/sec of a 8 cycle/sec pulse for 6 sec.*)
```

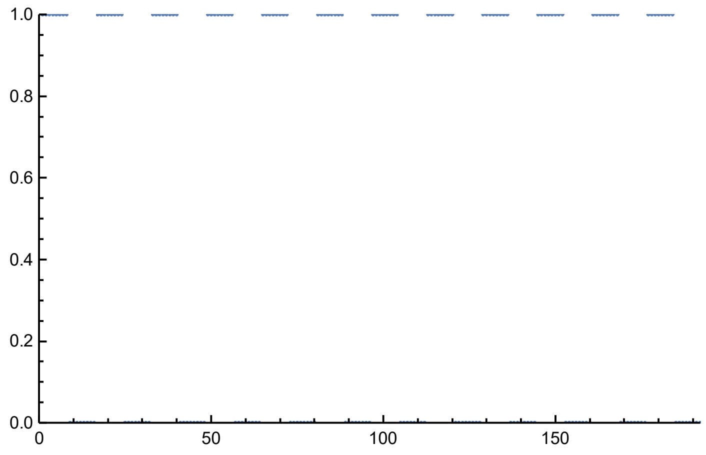
```
wcn4 = Table[0., {IX, 1, 192}, {JX, 1, 512}, {KX, 1, 512}];
For[JX = 1, JX ≤ 512, JX++, For[KX = 1, KX ≤ 512, KX++, For[II = 1, II ≤ 48, II++,
    For[JJ = 1, JJ ≤ 2, JJ++, wcn4[[(II - 1) (4) + JJ, JX, KX]] = fcn4[[JX, KX]]]]]];

(*wcnx4 is the image time sequence at corrdinates
 (40,100) with 32 frames/sec of a 6 cycle/sec pulse for 6 sec.*)
Print["fcn4[[40,100]]"]
fcn4[[40, 100]]
Print["Clear[wcnx4]"]
Clear[wcnx4];
wcnx4 = Table[0., {II, 1, 192}];
For[II = 1, II ≤ 192, II++, wcnx4[[II]] = wcn4[[II, 40, 100]]]
Print["Image time squence at coordinates(40,100)with
   32 frames/sec of a 8 cycle/sec pulse for 6 sec"]
ListPlot[wcnx4, PlotRange → {{0, 192}, {0, fcn4[[40, 100]]}}, PlotStyle → PointSize[Small]]

(*wcn4ref is the image where each voxel is acquired for 6 sec at 32 frames/sec
 with a 6 cycle/sec pulse*)wcn4ref = Table[0., {IX, 1, 512}, {JX, 1, 512}];
For[II = 1, II ≤ 512, II++, For[JJ = 1, JJ ≤ 512, JJ++,
  wcn4ref[[II, JJ]] = Sum[wcn4[[i, II, JJ]], {i, 192}]]]
Print["wcn4ref[[40,100]]"]
wcn4ref[[40, 100]] = 0.
Print["Image wcn3ref where each voxel is
   acquired for 6 sec at 32 frames/sec with a 6 cycle/sec pulse"]
ListDensityPlot[wcn4ref, PlotRange → All]

Print["*******************************************************"]

(*"wtot is total of the reference time sequences of
  32 frames/sec with 2, 4, 6, & 8 cycle/sec pulses for 6 sec."*)
wtot = Table[0., {Ix, 1, 192}];
For[IX = 1, IX < 192, IX++, wtot[[IX]] = w1[[IX]] + w2[[IX]] + w3[[IX]] + w4[[IX]]];
wtotsum = 0.;
For[IX = 1, IX ≤ 192, IX++, wtotsum = wtotsum + wtot[[IX]]];
Print["wtotsum"]; wtotsum

Fwtot = Table[0., {Ix, 1, 192}];
Fwtot = Fourier[wtot];
Print[
 "Fourier spectrum Abs[Fwtot]Sqrt[192] of the total of the reference time sequences"]
ListPlot[Abs[Fwtot] Sqrt[192],
 PlotRange → {{1, 192}, {0, 400}}, PlotStyle → PointSize[Medium]]

Print["Abs[Fwtot[[1]]] √192 "]
Abs[Fwtot[[1]]] √192
Print["Abs[Fwtot[[13]]] √192 "]
```

```
Abs[Fwtot⟦13⟧] √192
Print["Abs[Fwtot[[25]]] √192 "]
Abs[Fwtot⟦25⟧] √192
Print["Abs[Fwtot[[33]]] √192 "]
Abs[Fwtot⟦33⟧] √192
Print["Abs[Fwtot[[49]]] √192 "]
Abs[Fwtot⟦49⟧] √192

Print["Alpha1"];
Alpha1 = 96/ (Abs[Fwtot⟦13⟧] √192)
(*Alpha1=1.560722576129026`*)
Print["Alpha2"];
Alpha2 = 96/ (Abs[Fwtot⟦25⟧] √192) (*Alpha2=1.5307337294603593`*)
Print["Alpha3"];
Alpha3 = 96/ (Abs[Fwtot⟦33⟧] √192)
(*Alpha3=1.414213562373095`*)
Print["Alpha4"];
Alpha4 = 96/ (Abs[Fwtot⟦49⟧] √192)
(*Alpha4=1.414213562373095`*)

Print["**********************************"]

(*wcntot is 3D array of the sum of the 3D images of 2, 4, 6, 8 cycle/sec*)
Clear[wcntot];
wcntot = Table[0., {IX, 1, 192}, {JX, 1, 512}, {KX, 1, 512}];
For[JX = 1, JX ≤ 512, JX++,
  For[KX = 1, KX ≤ 512, KX++, For[IX = 1, IX ≤ 192, IX++, wcntot⟦IX, JX, KX⟧ =
     wcn1⟦IX, JX, KX⟧ + wcn2⟦IX, JX, KX⟧ + wcn3⟦IX, JX, KX⟧ + wcn4⟦IX, JX, KX⟧]]];

(*fcntot is the total sum of the image values at (40,100)*)
fcntot = Table[0., {IX, 1, 512}, {JX, 1, 512}];
Print["total of the image squence amplitude at fcntot[[40,100]]"]
fcntot⟦40, 100⟧ = fcn1⟦40, 100⟧ + fcn2⟦40, 100⟧ + fcn3⟦40, 100⟧ + fcn4⟦40, 100⟧
(*wcnxtot is the total of the image time sequences at corrdinates(40,100)*)
Print[
 "wcnxtot is the total image value from the time sequences at coordinates(40,100)"]
Clear[wcnxtot];
wcnxtot = Table[0., {II, 1, 192}];
For[II = 1, II ≤ 192, II++, wcnxtot⟦II⟧ = wcntot⟦II, 40, 100⟧]
ListPlot[wcnxtot, PlotRange → {{0, 192}, {0, fcntot⟦40, 100⟧}},
 PlotStyle → PointSize[Small]]

(*wcntotref is the image where each voxel is the sum of the time
 sequences for 6 sec at 32 frames/sec with 2, 4, 6, 8  cycle/sec pulses*)
Clear[wcntotref];
wcntotref = Table[0., {JX, 1, 512}, {KX, 1, 512}];
For[II = 1, II ≤ 512, II++, For[JJ = 1, JJ ≤ 512, JJ++, wcntotref⟦II, JJ⟧ =
   wcn1ref⟦II, JJ⟧ + wcn2ref⟦II, JJ⟧ + wcn3ref⟦II, JJ⟧ + wcn4ref⟦II, JJ⟧]]

Print["Total image acquired for 6 secs
   at 32 frames/sec for pulses of 2, 4, 6, and 8 cycle/sec. "]
Print["ListDensityPlot[wcntotref,PlotRange→All]"]
Print["wcntotref[[40,100]]"]
wcntotref⟦40, 100⟧
wcntotref⟦40, 100⟧ = 0
ListDensityPlot[wcntotref, PlotRange → All]

Clear[Fwcnxtot];
Fwcnxtot = Fourier[wcnxtot];
Print["Plot of Abs[Fwcnxtot] √192 "]
ListPlot[Abs[Fwcnxtot] √192,
 PlotRange → {{0, 192}, {0, Abs[Fwcnxtot⟦1⟧] √192}}, PlotStyle → PointSize[Medium]]
Print["Abs[Fwcnxtot[[1]]] √192 "]
Abs[Fwcnxtot⟦1⟧] √192
Print["Abs[Fwcnxtot[[13]]] √192 "]
Abs[Fwcnxtot⟦13⟧] √192
Print["Abs[Fwcnxtot[[25]]] √192 "]
Abs[Fwcnxtot⟦25⟧] √192
Print["Abs[Fwcnxtot[[33]]] √192 "]
Abs[Fwcnxtot⟦33⟧] √192
Print["Abs[Fwcnxtot[[49]]] √192 "]
Abs[Fwcnxtot⟦49⟧] √192

Print["(Abs[Fwcnxtot[[13]]] √192 )Alpha1"]
(Abs[Fwcnxtot⟦13⟧] √192) Alpha1
Print["(Abs[Fwcnxtot[[25]]] √192 )Alpha2"]
(Abs[Fwcnxtot⟦25⟧] √192) Alpha2
Print["(Abs[Fwcnxtot[[33]]] √192 )Alpha3"]
(Abs[Fwcnxtot⟦33⟧] √192) Alpha3
Print["(Abs[Fwcnxtot[[49]]] √192 )Alpha4"]
(Abs[Fwcnxtot⟦49⟧] √192) Alpha4
```

```
wcn1ref = Table[0., {JX, 1, 512}, {KX, 1, 512}];
wcnx1 = Table[0., {II, 1, 192}];
Fwcnx1 = Table[0., {II, 1, 192}];

wcn1ref = Table[
   For[IX = 1, IX ≤ 192, IX++, wcnx1⟦IX⟧ = wcntot⟦IX, JX, KX⟧];
   Fwcnx1 = Fourier[wcnx1];
   (Abs[Fwcnx1⟦13⟧] Sqrt[192]) Alpha1, {JX, 1, 512}, {KX, 1, 512}];

Print[
 "Demultiplexed 2 cycle/sec image: Composite collected at 32 frames/sec for 6 sec. "]
Print["ListDensityPlot[wcn1ref,PlotRange→All]"]
ListDensityPlot[wcn1ref, PlotRange → All]

Print["**********************************"]

wcn2ref = Table[0., {JX, 1, 512}, {KX, 1, 512}];
wcnx2 = Table[0., {II, 1, 192}];
Fwcnx2 = Table[0., {II, 1, 192}];

wcn2ref = Table[
   For[IX = 1, IX ≤ 192, IX++, wcnx2⟦IX⟧ = wcntot⟦IX, JX, KX⟧];
   Fwcnx2 = Fourier[wcnx2];
   (Abs[Fwcnx2⟦25⟧] Sqrt[192]) Alpha2, {JX, 1, 512}, {KX, 1, 512}];

Print[
 "Demultiplexed 4 cycle/sec image: Composite collected at 32 frames/sec for 6 sec. "]
Print["ListDensityPlot[wcn2ref,PlotRange→All]"]
ListDensityPlot[wcn2ref, PlotRange → All]

Print["**********************************"]

wcn3ref = Table[0., {JX, 1, 512}, {KX, 1, 512}];
wcnx3 = Table[0., {II, 1, 192}];
Fwcnx3 = Table[0., {II, 1, 192}];

wcn3ref = Table[
   For[IX = 1, IX ≤ 192, IX++, wcnx3⟦IX⟧ = wcntot⟦IX, JX, KX⟧];
   Fwcnx3 = Fourier[wcnx3];
   (Abs[Fwcnx3⟦33⟧] Sqrt[192]) Alpha3, {JX, 1, 512}, {KX, 1, 512}];

Print[
 "Demultiplexed 6 cycle/sec image: Composite collected at 32 frames/sec for 6 sec. "]
Print["ListDensityPlot[wcn3ref,PlotRange→All]"]
ListDensityPlot[wcn3ref, PlotRange → All]

Print["**********************************"]
```

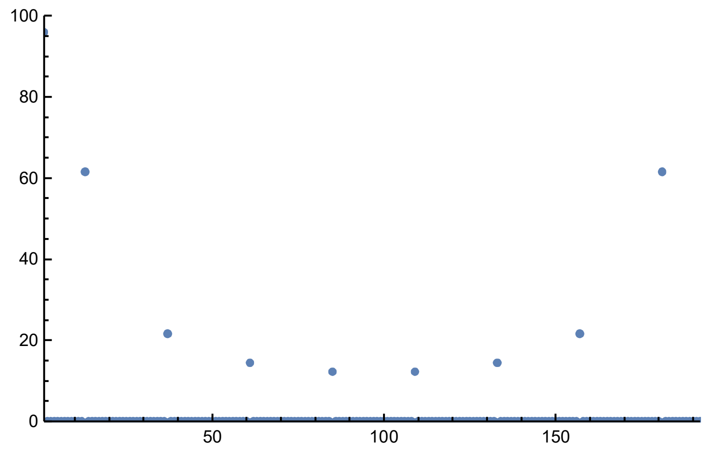
```
wcn4ref = Table[0., {JX, 1, 512}, {KX, 1, 512}];
wcnx4 = Table[0., {II, 1, 192}];
Fwcnx4 = Table[0., {II, 1, 192}];

wcn4ref = Table[
   For[IX = 1, IX ≤ 192, IX++, wcnx4⟦IX⟧ = wcntot⟦IX, JX, KX⟧];
   Fwcnx4 = Fourier[wcnx4];
   (Abs[Fwcnx4⟦49⟧] Sqrt[192]) Alpha4, {JX, 1, 512}, {KX, 1, 512}];

Print[
 "Demultiplexed 8 cycle/sec image: Composite collected at 32 frames/sec for 6 sec. "]
Print["ListDensityPlot[wcn4ref,PlotRange→All]"]
ListDensityPlot[wcn4ref, PlotRange → All]
```

```
Collection at 32 frames/sec of a 2 cycle/sec pulse for 6 sec.

Hz1

Out[4]= 2

sumwcn1=Sum[fcn1[[i,1]],{i,512}]

Out[8]= 512.

Reference time sequence w1 for 32 frames/sec of a 2 cycle/sec pulse for 6 sec

Out[12]=
```



```
Sum of frames in 6 sec: w1sum

Out[15]=
96.

Fourier spectrun Abs[Fw1]Sqrt[192]: 32 frames/sec of a 2 cycle/sec pulse for 6 sec
```

Out[19]=



```
Fw1[[1]]Sqrt[192]
```

Out[20]=

```
96. + 0. i
```

```
fcn1[[40,100]]
```

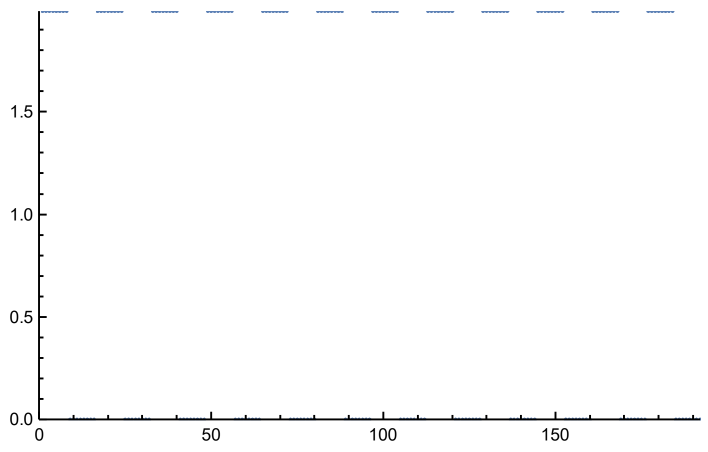
Out[24]=

```
1.98918
```

```
Clear[wcnx1]
```

```
Image time squence at coordinates(40,100)with 32 frames/sec of a 2 cycle/sec pulse for 6 sec
```

Out[30]=



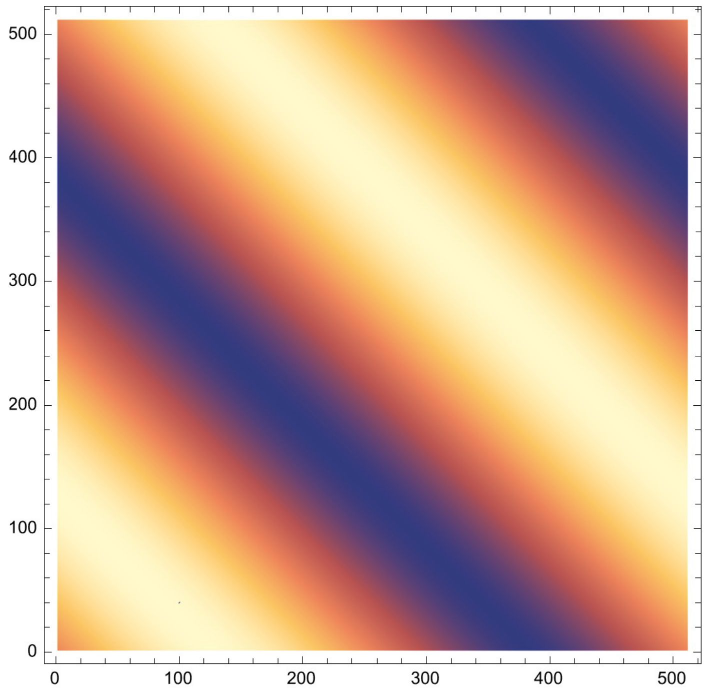
```
wcn1ref[[40,100]]
```

Out[33]=

```
0.
```

```
Image wcn1ref where each voxel is acquired for 6 sec at 32 frames/sec with a 2 cycle/sec pulse
```

Out[35]=



```
*********************************************************
Collection at 32 frames/sec of a 4 cycle/sec pulse for 6 sec.
Hz2
```

Out[40]=

```
4
```

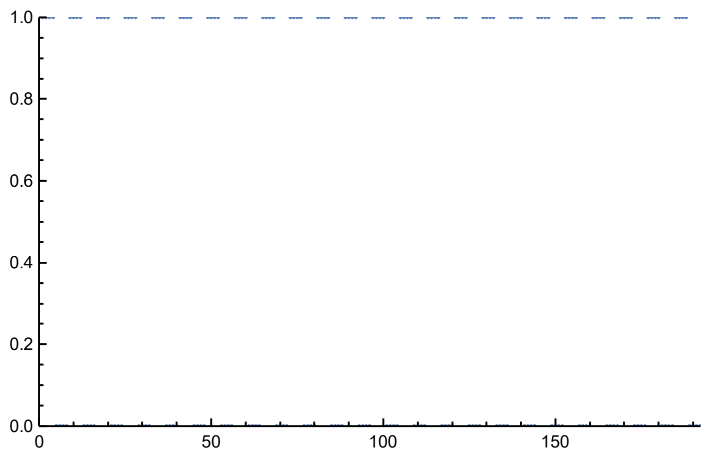
```
sumwcn2=Sum[fcn2[[i,1]],{i,1024}]
```

Out[44]=

```
1024.
```

```
Reference time sequence w2 for 32 frames/sec of a 4 cycle/sec pulse for 6 sec
```

Out[48]=

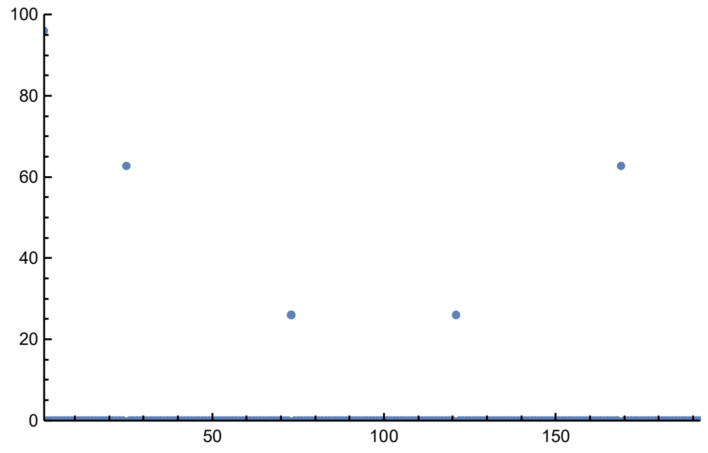


```
Sum of frames in 6 sec: w2sum
```

```
Out[51]=
96.
```

```
Fourier spectrum AbspFw2]Sqrt[192]: 32 frames/sec of 4 cycle/sec pulse for 6 sec
```

```
Out[55]=
```



```
Fw2[[1]]Sqrt[192]
```

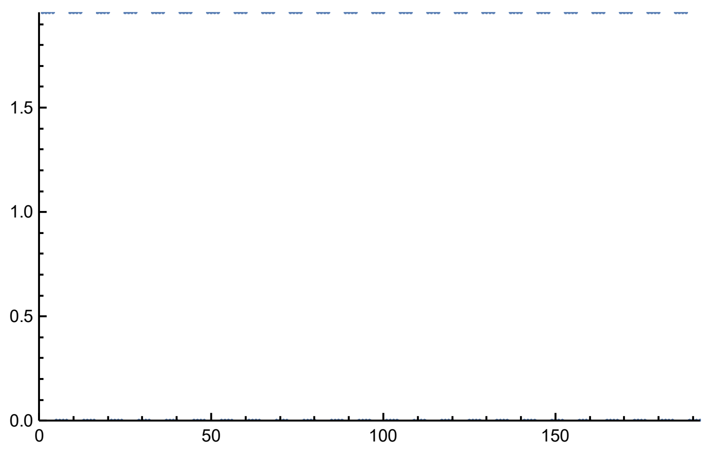
```
Out[56]=
96. + 0. i
```

```
wcn2=Table[0.,{IX,1,192},{JX,1,512},{KX,1,512}

fcn2[[40,100]]
```

```
Out[61]=
1.95694
```

```
Clear[wcnx2]

Image time squence at coordinates(40,100)with 32 frames/sec of a 4 cycle/sec pulse for 6 sec
```

```
Out[67]=
```

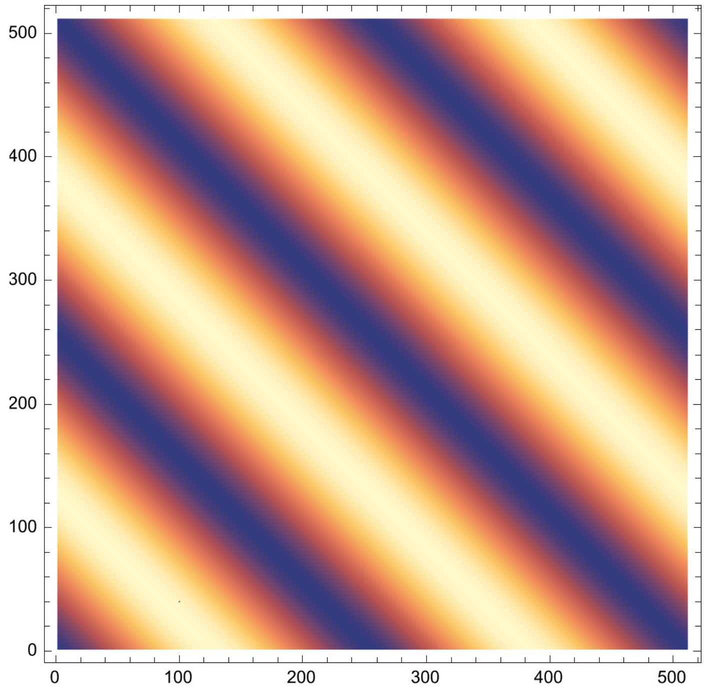


```
wcn2ref[[40,100]]
```

```
Out[70]=
0.
```

```
Image wcn2ref where each voxel is acquired for 6 sec at 32 frames/sec with a 4 cycle/sec pulse

ListDensityPlot[wcn2ref,PlotRange→All]
```

Out[73]=



```
*****************************************************

Collection at 32 frames/sec of a 6 cycle/sec pulse for  6 sec.

Hz3
```

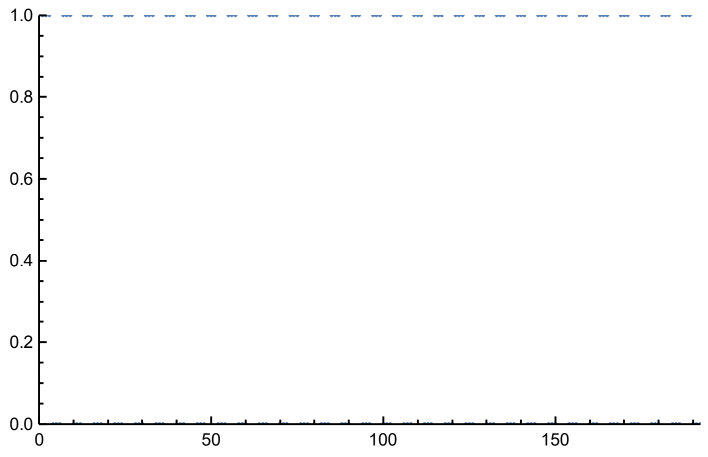
Out[78]=

```
6

sumwcn3=Sum[fcn3[[i,1]],{i,1024}]
```

Out[82]=

```
1020.04

Reference time sequence w3 for 32 frames/sec of a 6 cycle/sec pulse for 6 sec
```

Out[86]=

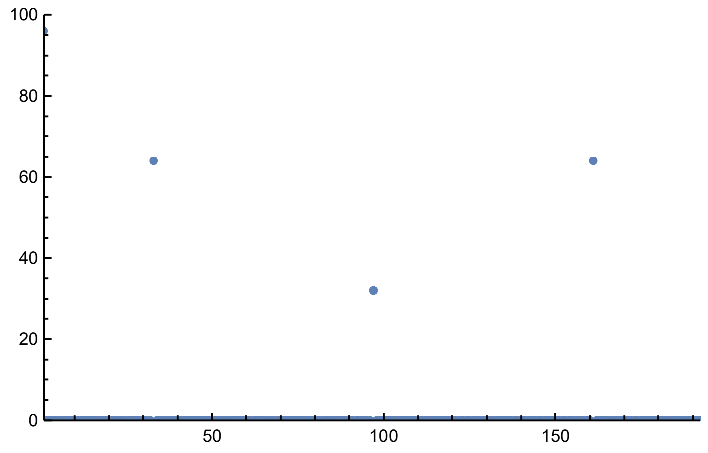


```
Sum of frames in 6 sec: w3sum
```

```
Out[89]=
96.
```

Fourier spectrum Abs[Fw3]Sqrt[192]: 32 frames/sec of a 6 cycle/sec pulse for 6 sec

Out[93]=



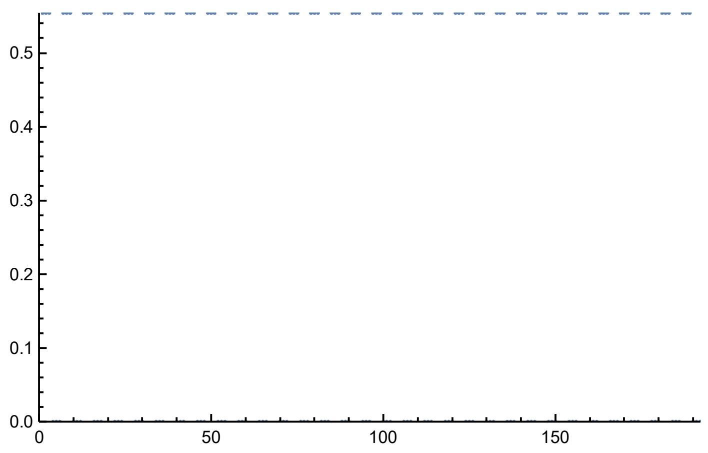
```
Fw3[[1]]Sqrt[192]
Out[94]=
96. + 0. i
```

```
fcn3[[40,100]]
Out[98]=
0.554262
```

```
Clear[wcnx3]
```

Image time squence at coordinates(40,100)with 32 frames/sec of a 6 cycle/sec pulse for 6 sec

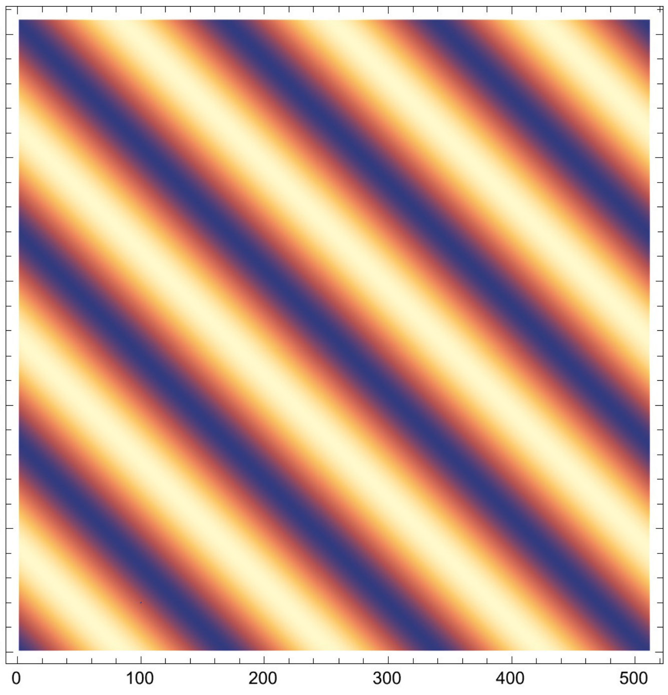
Out[104]=



```
wcn3ref[[40,100]]
Out[107]=
0.
```

Image wcn3ref where each voxel is acquired for 6 sec at 32 frames/sec with a 6 cycle/sec pulse

Out[109]=



```
*******************************************************
Collection at 32 frames/sec of a 8 cycle/sec pulse for 6 sec.
Hz4
```

Out[114]=

```
8
```

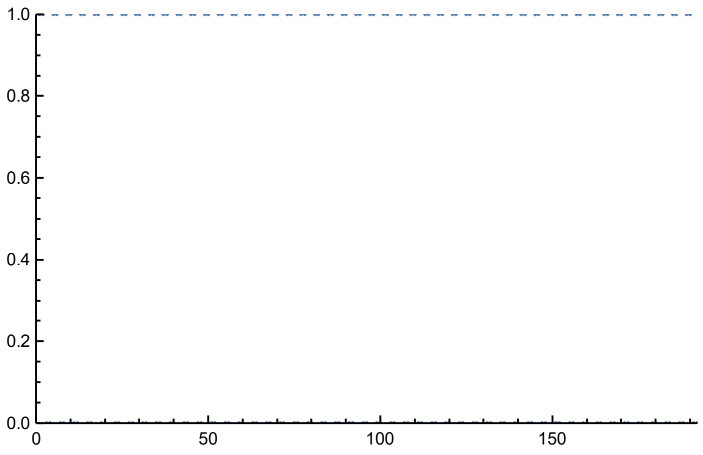
```
sumwcn4=Sum[fcn4[[i,1]],{i,1024}]
```

Out[118]=

```
1024.
```

```
Reference time sequence w4 for 32 frames/sec of a 8 cycle/sec pulse for 6 sec
```

Out[122]=

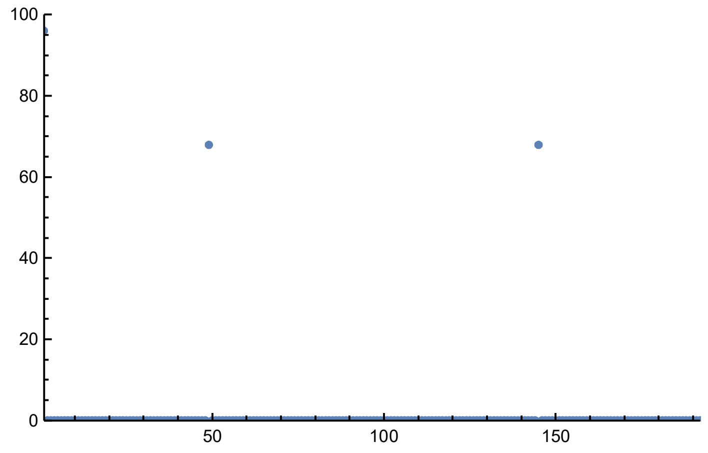


```
Sum of frames in 6 sec: w4sum
```

```
Out[125]=
96.
```

Fourier spectrum Abs[Fw4]Sqrt[192]: 32 frames/sec of a 8 cycle/sec pulse for 6 sec

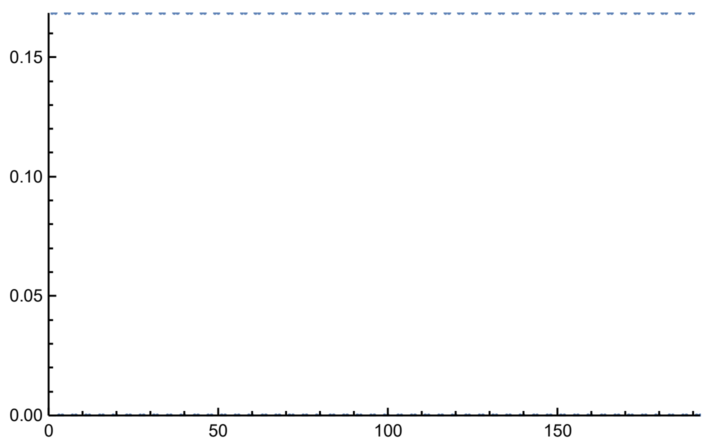
```
Out[129]=
```



```
Fw4[[1]]Sqrt[192]
Out[130]=
96. + 0. i
fcn4[[40,100]]
Out[134]=
0.16853
Clear[wcnx4]
```

Image time squence at coordinates(40,100)with 32 frames/sec of a 8 cycle/sec pulse for 6 sec

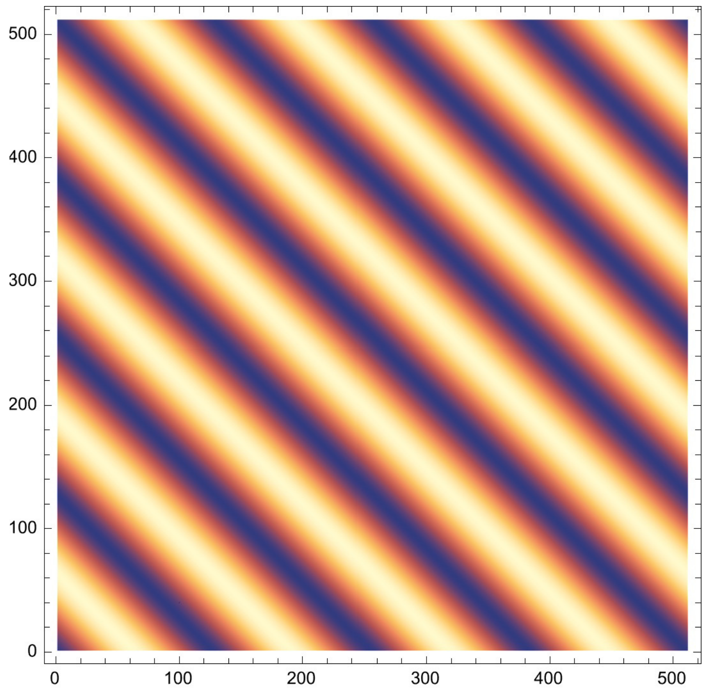
```
Out[140]=
```



```
wcn4ref[[40,100]]
Out[143]=
0.
```

Image wcn3ref where each voxel is acquired for 6 sec at 32 frames/sec with a 6 cycle/sec pulse

Out[145]=



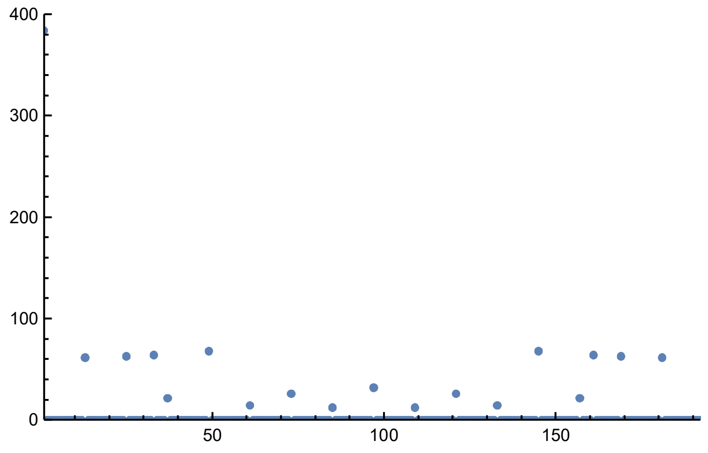
```
*********************************************************

wtotsum
```

Out[151]=

```
384.
```

```
Fourier spectrum Abs[Fwtot]Sqrt[192] of the total of the reference time sequences
```

Out[155]=



```
Abs[Fwtot[[1]]] √192
```

Out[157]=

```
384.
```

```
Abs[Fwtot[[13]]] √192
```

Out[159]=

```
61.51
```

```
Abs[Fwtot[[25]]] √192
```

Out[161]=

```
62.715
```

```
Abs[Fwtot[[33]]] √192
```

Out[163]=

```
64.
```

```
Abs[Fwtot[[49]]] √192
```

Out[165]=

```
67.8823
```

```
Alpha1
```

Out[167]=

```
1.56072
```

```
Alpha2
```

Out[169]=

```
1.53073
```

```
Alpha3
```

Out[171]=

```
1.5
```

```
Alpha4
```

Out[173]=

```
1.41421
```

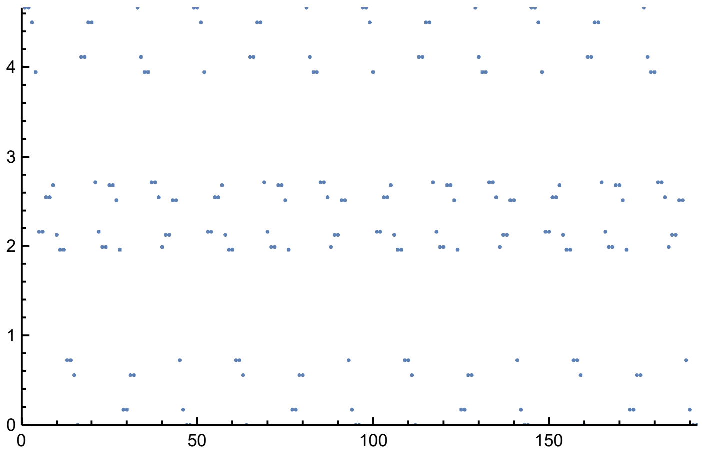
```
**********************************

total of the image squence amplitude at fcntot[[40,100]]
```

Out[180]=

```
4.66891
```

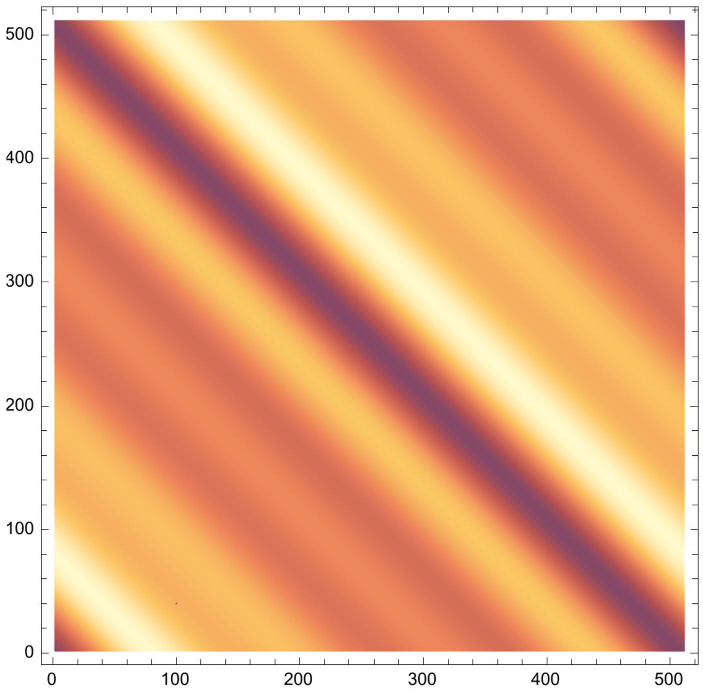
```
wcnxtot is the total image value from the time sequences at coordinates(40,100)
```

Out[185]=



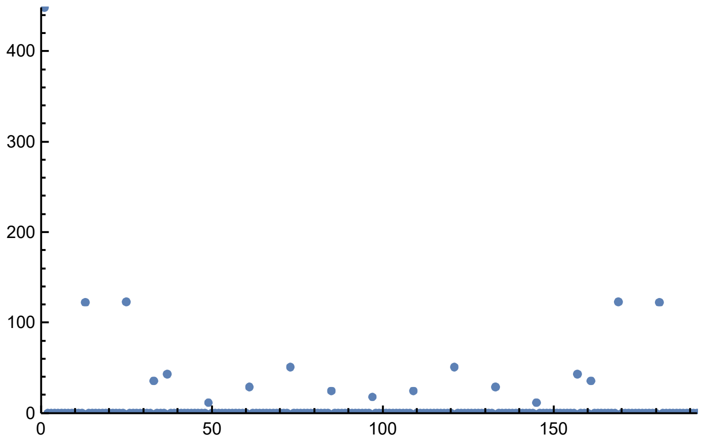
```
Total image acquired for 6 secs at 32 frames/sec for pulses of 2, 4, 6, and 8 cycle/sec.

ListDensityPlot[wcntotref,PlotRange→All]

wcntotref[[40,100]]
```

```
Out[192]=
0.
Out[193]=
0
Out[194]=
```



Plot of Abs[Fwcnxtot] $\sqrt{192}$

```
Out[198]=
```



Abs[Fwcnxtot[[1]]] $\sqrt{192}$

```
Out[200]=
448.215
```

Abs[Fwcnxtot[[13]]] $\sqrt{192}$

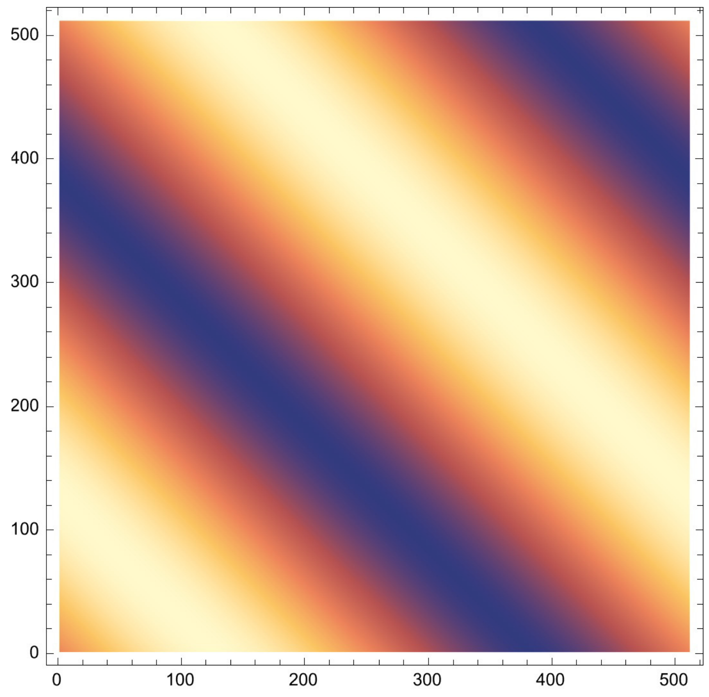
```
Out[202]=
122.354
```

Abs[Fwcnxtot[[25]]] $\sqrt{192}$

```
Out[204]=
122.73

Abs[Fwcnxtot[[33]]] √192
Out[206]=
35.4727

Abs[Fwcnxtot[[49]]] √192
Out[208]=
11.4402

(Abs[Fwcnxtot[[13]]] √192) Alpha1
Out[210]=
190.961

(Abs[Fwcnxtot[[25]]] √192) Alpha2
Out[212]=
187.866

(Abs[Fwcnxtot[[33]]] √192) Alpha3
Out[214]=
53.2091

(Abs[Fwcnxtot[[49]]] √192) Alpha4
Out[216]=
16.1789
```

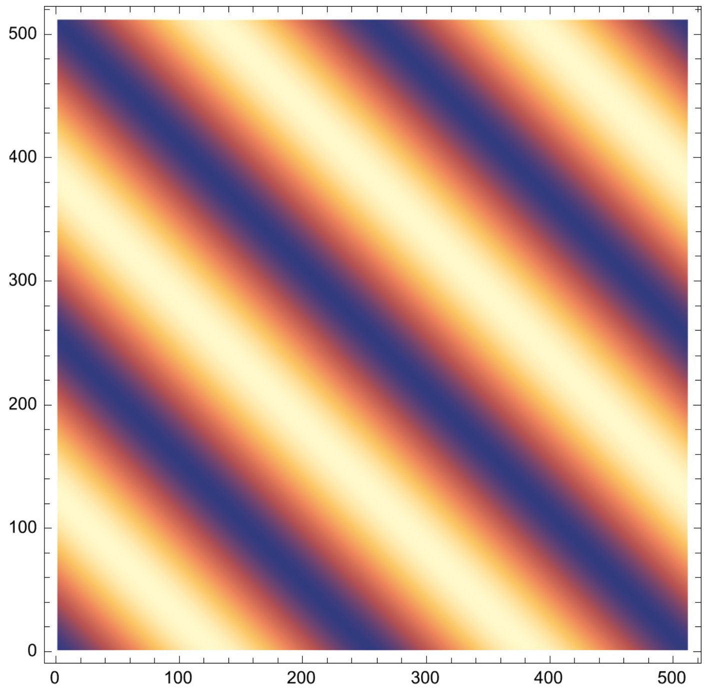
Demultiplexed 2 cycle/sec image: Composite collected at 32 frames/sec for 6 sec.

```
ListDensityPlot[wcn1ref,PlotRange→All]
Out[223]=
```



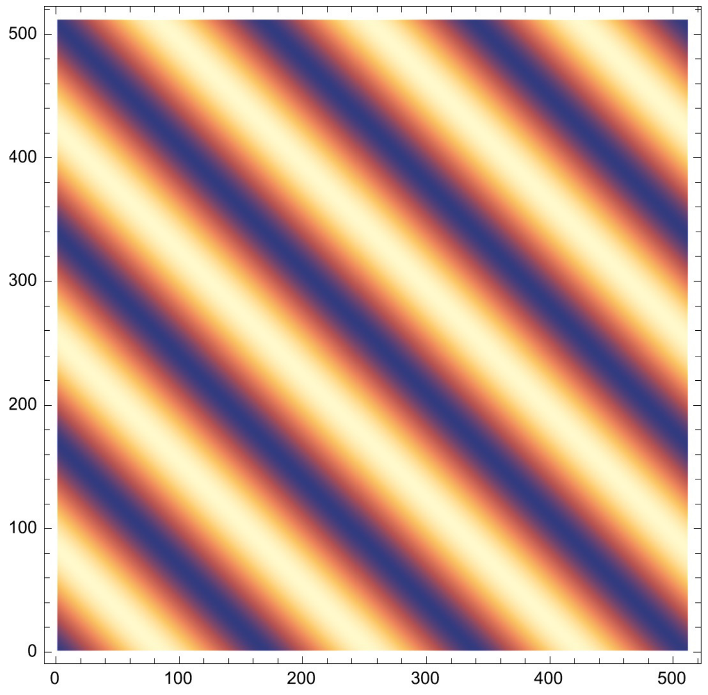
```
**********************************

Demultiplexed 4 cycle/sec image: Composite collected at 32 frames/sec for 6 sec.

ListDensityPlot[wcn2ref,PlotRange→All]
```

Out[231]=



```
**********************************

Demultiplexed 6 cycle/sec image: Composite collected at 32 frames/sec for 6 sec.

ListDensityPlot[wcn3ref,PlotRange→All]
```

Out[239]=

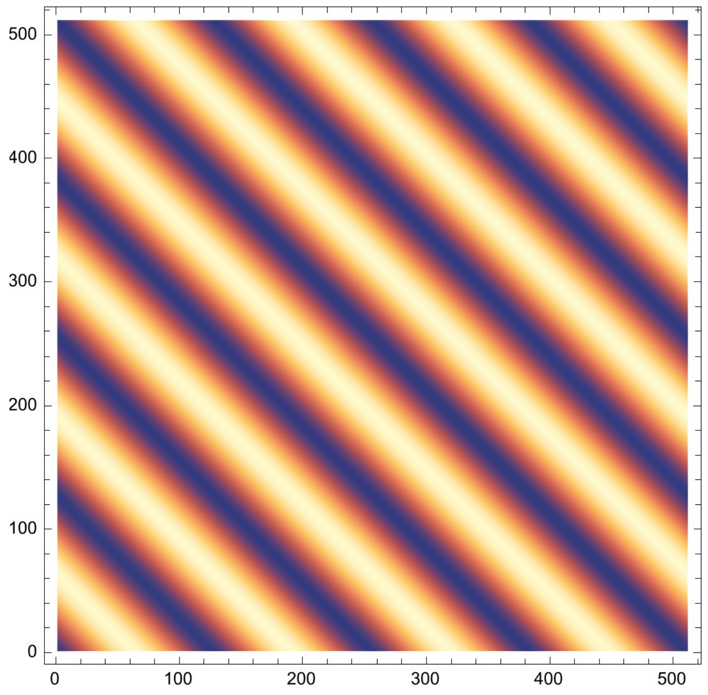


```
*********************************
```

```
Demultiplexed 8 cycle/sec image: Composite collected at 32 frames/sec for 6 sec.
```

```
ListDensityPlot[wcn4ref,PlotRange→All]
```

Out[247]=



In[1537]:=

```
(***************Appendix II ***************)
(*Mathematica Code for Extracting Attenuation,Dark Field,
and Phase Images from Phase Stepping Acquisitions*)

data1 = Import["Desktop/Fuller/New Data/raw data/20220803_00321.tif", "Data"];
data1⟦1, 1⟧
data1⟦864, 1536⟧
Print["Data with Phantom"]
MatrixPlot[data1, AspectRatio → 0.5625]
rdata1 = Import["Desktop/Fuller/New Data/raw data/20220803_00441.tif", "Data"];
rdata1⟦1, 1⟧
rdata1⟦864, 1536⟧
Print["Data without Phantom (Reference Scan)"]
MatrixPlot[rdata1, AspectRatio → 0.5625]
data2 = Import["Desktop/Fuller/New Data/raw data/20220803_00329.tif", "Data"];
data2⟦1, 1⟧
data2⟦864, 1536⟧
Print["Data with Phantom"]
MatrixPlot[data2, AspectRatio → 0.5625]
rdata2 = Import["Desktop/Fuller/New Data/raw data/20220803_00449.tif", "Data"];
Print["Data without Phantom (Reference Scan)"]
MatrixPlot[rdata2, AspectRatio → 0.5625]
data3 = Import["Desktop/Fuller/New Data/raw data/20220803_00337.tif", "Data"];
data3⟦1, 1⟧
data3⟦864, 1536⟧
Print["Data with Phantom"]
MatrixPlot[data3, AspectRatio → 0.5625]
rdata3 = Import["Desktop/Fuller/New Data/raw data/20220803_00457.tif", "Data"];
Print["Data without Phantom (Reference Scan)"]
MatrixPlot[rdata3, AspectRatio → 0.5625]
data4 = Import["Desktop/Fuller/New Data/raw data/20220803_00345.tif", "Data"];
data4⟦1, 1⟧
data4⟦864, 1536⟧
Print["Data with Phantom"]
MatrixPlot[data4, AspectRatio → 0.5625]
rdata4 = Import["Desktop/Fuller/New Data/raw data/20220803_00465.tif", "Data"];
Print["Data without Phantom (Reference Scan)"]
MatrixPlot[rdata4, AspectRatio → 0.5625]
Data = Table[0., {i, 1, 864}, {j, 1, 1536}, {k, 1, 4}];
RData = Table[0., {i, 1, 864}, {j, 1, 1536}, {k, 1, 4}];
For[II = 1, II ≤ 864, II++, For[JJ = 1, JJ ≤ 1536, JJ++, Data⟦II, JJ, 1⟧ = data1⟦II, JJ⟧;
  RData⟦II, JJ, 1⟧ = rdata1⟦II, JJ⟧]]
For[II = 1, II ≤ 864, II++, For[JJ = 1, JJ ≤ 1536, JJ++, Data⟦II, JJ, 2⟧ = data2⟦II, JJ⟧;
  RData⟦II, JJ, 2⟧ = rdata2⟦II, JJ⟧]]
For[II = 1, II ≤ 864, II++, For[JJ = 1, JJ ≤ 1536, JJ++, Data⟦II, JJ, 3⟧ = data3⟦II, JJ⟧;
  RData⟦II, JJ, 3⟧ = rdata3⟦II, JJ⟧]]
```

```
For[II = 1, II ≤ 864, II++, For[JJ = 1, JJ ≤ 1536, JJ++, Data⟦II, JJ, 4⟧ = data4⟦II, JJ⟧;
  RData⟦II, JJ, 4⟧ = rdata4⟦II, JJ⟧]]
(****************************)
FData = Table[0., {i, 1, 864}, {j, 1, 1536}, {k, 1, 4}];
FRData = Table[0., {i, 1, 864}, {j, 1, 1536}, {k, 1, 4}];
DataAtn = Table[0., {i, 1, 864}, {j, 1, 1536}];
DataPhase1 = Table[0., {i, 1, 864}, {j, 1, 1536}];
DataPhase2 = Table[0., {i, 1, 864}, {j, 1, 1536}];
DataPhase = Table[0., {i, 1, 864}, {j, 1, 1536}];
VisibilityObj = Table[0., {i, 1, 864}, {j, 1, 1536}];
VisibilityRef = Table[0., {i, 1, 864}, {j, 1, 1536}];
SmallAng = Table[0., {i, 1, 864}, {j, 1, 1536}];
FRData1 = Table[0., {i, 1, 864}, {j, 1, 1536}];
FRData2 = Table[0., {i, 1, 864}, {j, 1, 1536}];
Datax = Table[0., {JX, 1, 4}];
FDatax = Table[0., {JX, 1, 4}];
RDatax = Table[0., {JX, 1, 4}];
FRDatax = Table[0., {JX, 1, 4}];
ObjI = Table[0., {i, 1, 864}, {j, 1, 1536}];
RefI = Table[0., {i, 1, 864}, {j, 1, 1536}];

DataAtn = Table[For[KX = 1, KX ≤ 4, KX++, Datax⟦KX⟧ = Data⟦IX, JX, KX⟧];
   For[KX = 1, KX ≤ 4, KX++, RDatax⟦KX⟧ = RData⟦IX, JX, KX⟧];
   FDatx = Sqrt[4] InverseFourier[Datax];
   FRDatx = Sqrt[4] InverseFourier[RDatax];
   For[KX = 1, KX ≤ 4, KX++, FData⟦IX, JX, KX⟧ = FDatx⟦KX⟧];
   For[KX = 1, KX ≤ 4, KX++, FRData⟦IX, JX, KX⟧ = FRDatx⟦KX⟧];
   ObjI⟦IX, JX⟧ = FData⟦IX, JX, 1⟧;
   RefI⟦IX, JX⟧ = FRData⟦IX, JX, 1⟧;
   DataAtn⟦IX, JX⟧ = -Log[Abs[FData⟦IX, JX, 1⟧] / Abs[FRData⟦IX, JX, 1⟧]],
   {IX, 1, 864}, {JX, 1, 1536}];

Print["Object Intensity"]
ImageReflect[ListDensityPlot[Abs[ObjI], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]
Print["Reference Intensity"]
ImageReflect[ListDensityPlot[Abs[RefI], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]
DataAtn⟦1, 1⟧
DataAtn⟦864, 1536⟧
Print["Attenuation Image"]
ImageReflect[ListDensityPlot[Abs[DataAtn], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]

Print["Fourier Spectrum at (500,700)"]
For[KX = 1, KX ≤ 4, KX++, FRDatax⟦KX⟧ = FRData⟦500, 700, KX⟧];
Table[Abs[FRDatax⟦i⟧], {i, 1, 4}]
```

```
ListPlot[Abs[FRDatax], PlotRange → {{1, 4}, {0., 50000}}, PlotStyle → PointSize[Medium]]

Arg[FData⟦700, 925, 2⟧]
Arg[FRData⟦700, 925, 2⟧]
Arg[FData⟦500, 925, 2⟧]
Arg[FRData⟦500, 925, 2⟧]
DataPhase1 =
  Table[DataPhase1⟦IX, JX⟧ = If[Arg[FData⟦IX, JX, 2⟧] < 0,
     Arg[FData⟦IX, JX, 2⟧] + 2 π, Arg[FData⟦IX, JX, 2⟧]], {IX, 1, 864}, {JX, 1, 1536}];
DataPhase1⟦700, 925⟧
DataPhase1⟦500, 925⟧
Print["DataPhase1 with Object"]
ImageReflect[ListDensityPlot[Abs[DataPhase1], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]
DataPhase2 =
  Table[DataPhase2⟦IX, JX⟧ = If[Arg[FRData⟦IX, JX, 2⟧] < 0,
     Arg[FRData⟦IX, JX, 2⟧] + 2 π, Arg[FRData⟦IX, JX, 2⟧]], {IX, 1, 864}, {JX, 1, 1536}];
DataPhase2⟦700, 925⟧
DataPhase2⟦500, 925⟧
Print["DataPhase2 without Phantom (Reference)"]
ImageReflect[ListDensityPlot[Abs[DataPhase2], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]
DataPhase1 =
  Table[DataPhase1⟦IX, JX⟧ = If[DataPhase2⟦IX, JX⟧ > DataPhase1⟦IX, JX⟧,
     DataPhase1⟦IX, JX⟧ + 2 π, DataPhase1⟦IX, JX⟧], {IX, 1, 864}, {JX, 1, 1536}];
DataPhase1⟦700, 925⟧
DataPhase1⟦500, 925⟧
Print["DataPhase1 Phase with Object Corrected for Phase Wrap"]
ImageReflect[ListDensityPlot[Abs[DataPhase1], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]
DataPhase =
  Table[DataPhase⟦IX, JX⟧ = DataPhase1⟦IX, JX⟧ - DataPhase2⟦IX, JX⟧,
   {IX, 1, 864}, {JX, 1, 1536}];
DataPhase⟦700, 925⟧
DataPhase⟦500, 925⟧
Print["Differential Phase"]
ImageReflect[ListDensityPlot[Abs[DataPhase], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]

VisibilityObj = Table[
   VisibilityObj⟦IX, JX⟧ = FData⟦IX, JX, 2⟧ / FData⟦IX, JX, 1⟧, {IX, 1, 864}, {JX, 1, 1536}];
VisibilityRef = Table[VisibilityRef⟦IX, JX⟧ = FRData⟦IX, JX, 2⟧ / FRData⟦IX, JX, 1⟧,
   {IX, 1, 864}, {JX, 1, 1536}];
SmallAng = Table[SmallAng⟦IX, JX⟧ =
    -Log[(FData⟦IX, JX, 2⟧ × FRData⟦IX, JX, 1⟧) / (FData⟦IX, JX, 1⟧ × FRData⟦IX, JX, 2⟧)],
   {IX, 1, 864}, {JX, 1, 1536}];
```

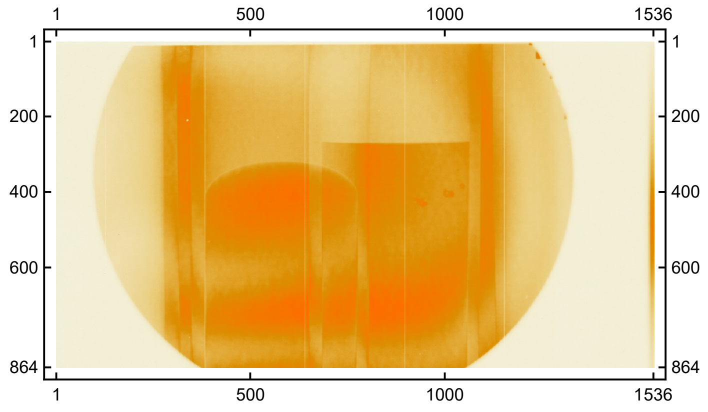
```
FRData1 = Table[FRData1[[IX, JX]] = Abs[FRData[[IX, JX, 1]]], {IX, 1, 864}, {JX, 1, 1536}];
FRData2 = Table[FRData2[[IX, JX]] = Abs[FRData[[IX, JX, 2]]], {IX, 1, 864}, {JX, 1, 1536}];

VisibilityObj[[700, 925]]
VisibilityObj[[500, 925]]
Print["VisibilityObj"]
ImageReflect[ListDensityPlot[Abs[VisibilityObj], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]
VisibilityRef[[700, 925]]
VisibilityRef[[500, 925]]
Print["VisibilityRef"]
ImageReflect[ListDensityPlot[Abs[VisibilityRef], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]

SmallAng[[1, 1]]
SmallAng[[864, 1536]]
SmallAng[[432, 768]]
Print["Small Angle - Dark Field Image"]
ImageReflect[ListDensityPlot[Abs[SmallAng], PlotRange → All,
  ColorFunction → GrayLevel, AspectRatio → 0.5625, Frame → False, Axes → False]]
```

Out[1538]=
```
627
```

Out[1539]=
```
1914
```

```
Data with Phantom
```

Out[1541]=

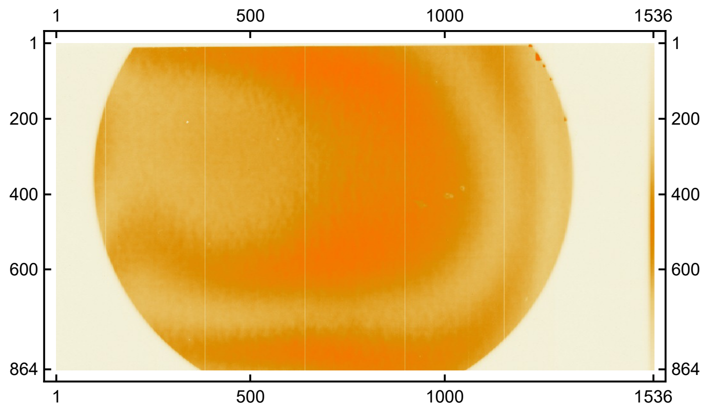


Out[1543]=
```
621
```

Out[1544]=
```
2176
```

```
Data without Phantom (Reference Scan)
```

Out[1546]=



Out[1547]=

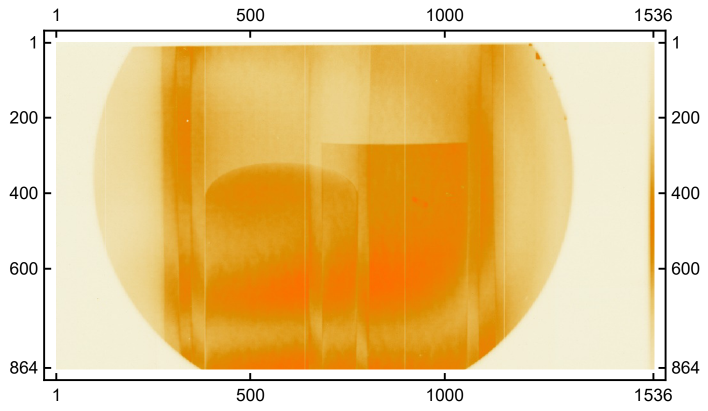
```
617
```

Out[1548]=

```
1948
```

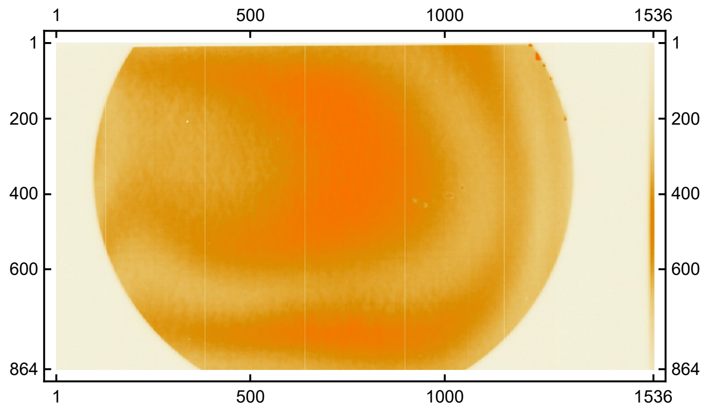
```
Data with Phantom
```

Out[1550]=



```
Data without Phantom (Reference Scan)
```

Out[1553]=

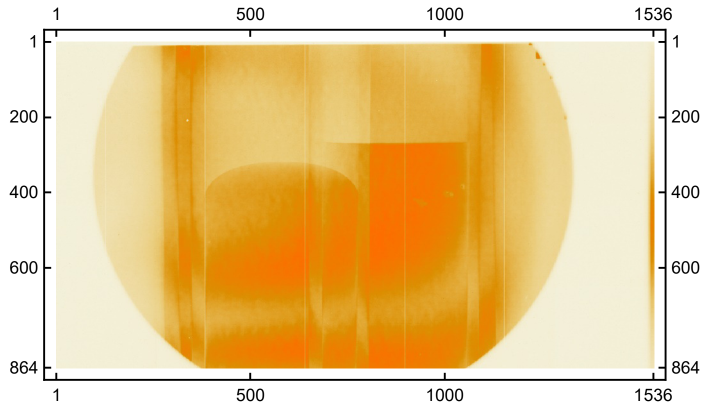


Out[1554]=

```
620
```

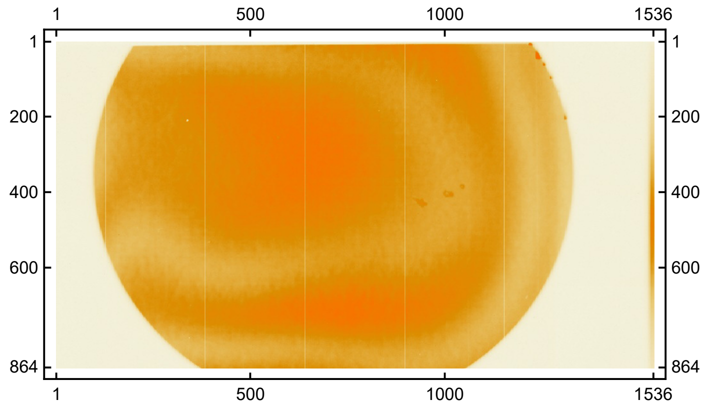
```
Out[1555]=
        1961

        Data with Phantom
```

Out[1557]=



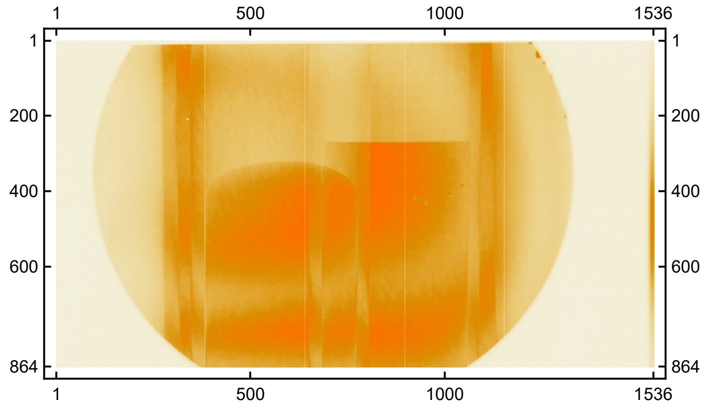
```
        Data without Phantom (Reference Scan)
```

Out[1560]=



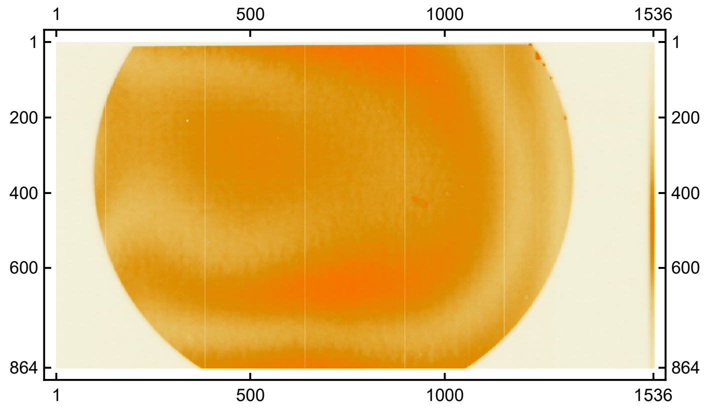
```
Out[1561]=
        609

Out[1562]=
        1977

        Data with Phantom
```

Out[1564]=



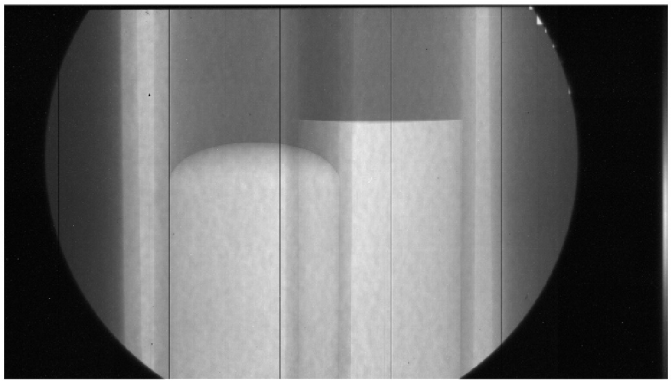
Data without Phantom (Reference Scan)

Out[1567]=



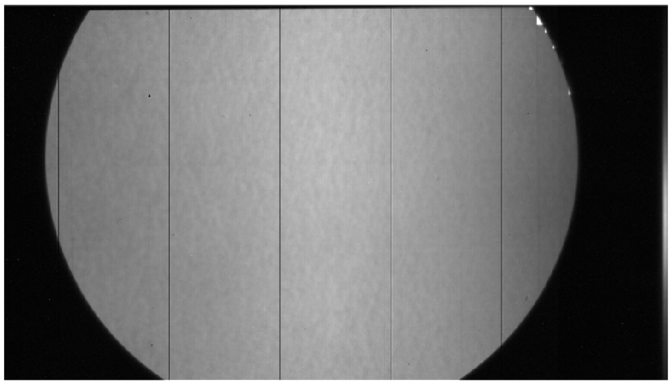
Object Intensity

Out[1587]=



Reference Intensity

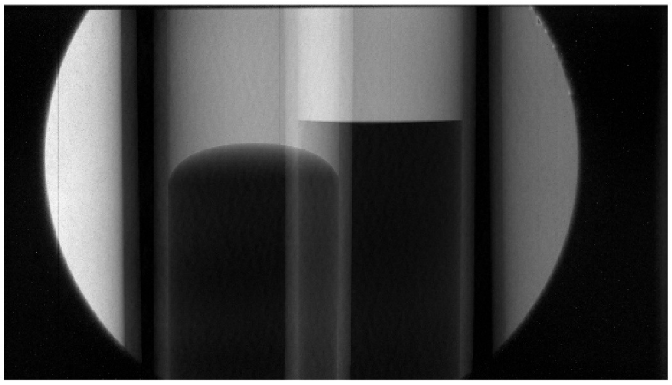
Out[1589]=



```
Out[1590]=
        0.0192237

Out[1591]=
        0.0915736
```

Attenuation Image

Out[1593]=

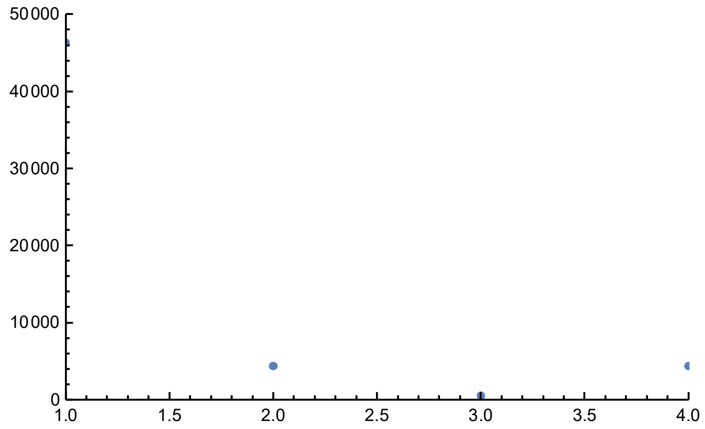


```
Fourier Spectrum at (500,700)
```

Out[1596]=

```
{46 295., 4364.99, 495., 4364.99}
```

Out[1597]=



Out[1598]=

```
-0.0761916
```

Out[1599]=

```
2.896
```

Out[1600]=

```
-3.00697
```

Out[1601]=

```
0.0302726
```

Out[1603]=

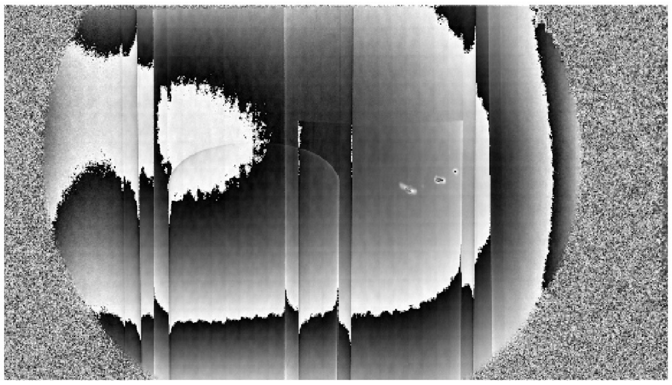
```
6.20699
```

Out[1604]=

```
3.27622
```

```
DataPhase1 with Object
```

Out[1606]=



Out[1608]=

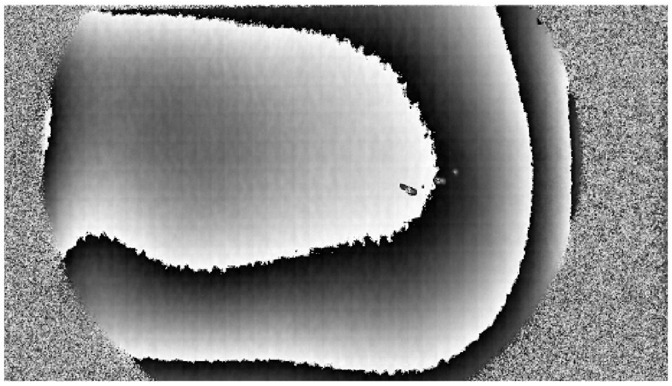
```
2.896
```

Out[1609]=

```
0.0302726
```

DataPhase2 without Phantom (Reference)

Out[1611]=



Out[1613]=

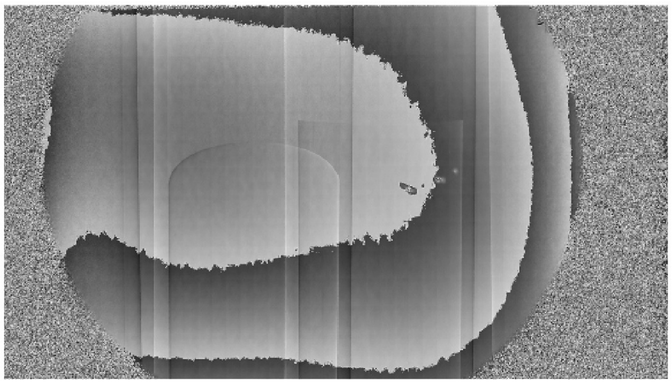
```
6.20699
```

Out[1614]=

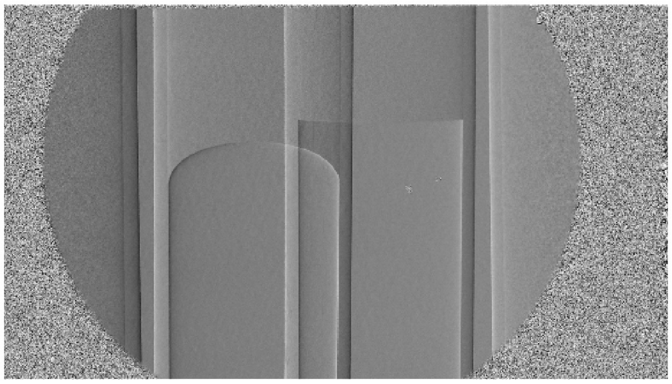
```
3.27622
```

DataPhase1 Phase with Object Corrected for Phase Wrap

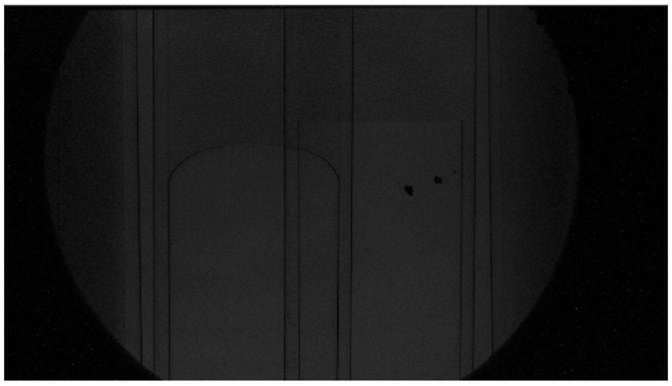
Out[1616]=



```
Out[1618]=
        3.311

Out[1619]=
        3.24594

        Differential Phase

Out[1621]=
```



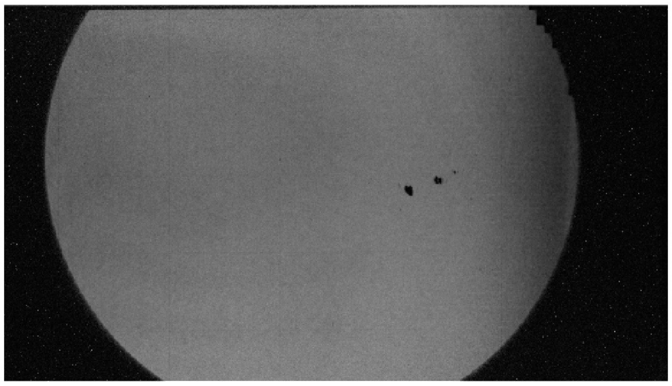
```
        Power: Infinite expression 1/(0. + 0. ⅈ) encountered.

        Power: Infinite expression 1/(0. + 0. ⅈ) encountered.

        Power: Infinite expression 1/(0. + 0. ⅈ) encountered.

        General: Further output of Power::infy will be suppressed during this calculation.

Out[1627]=
        0.109776 - 0.00838024 ⅈ

Out[1628]=
        -0.0954764 - 0.0129317 ⅈ

        VisibilityObj

Out[1630]=
```



```
Out[1631]=
        -0.107922 + 0.0270512 ⅈ

Out[1632]=
        0.100054 + 0.00302981 ⅈ
```

```
VisibilityRef
```

Out[1634]=



Out[1635]=

```
0.248428 - 2.8177 i
```

Out[1636]=

```
0.783098 - 2.84838 i
```

Out[1637]=

```
0.0166914 + 2.60921 i
```

```
Small Angle - Dark Field Image
```

Out[1639]=