FREEFORM MEASUREMENT WITH STITCHING TALBOT INTERFEROMETER

by

Yasunori Furukawa

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Optical Science and Engineering

Charlotte

2018

Approved by:

_____
Dr. Angela Davies

_____
Dr. Christopher J. Evans

_____
Dr. Thomas Suleski

ABSTRACT

YASUNORI FURUKAWA. Freeform measurement with stitching Talbot interferometer.
(Under the direction of DR. ANGELA DAVIES)

Freeform optics are being used in many fields to realize the outstanding performance due to their high degree of freedom. To realize sufficiently high performance, it is necessary to manufacture the freeform surface with high accuracy, which requires a high accurate measurement of the freeform surface. Several methods to measure the freeform surface have been suggested. However, they are time-consuming, expensive or their dynamic range is limited. Therefore, we propose a new system that realizes high dynamic range and rapid freeform measurements of freeform surfaces using a combination of the Talbot interferometer and the stitching technique. In this thesis, we introduced the theory and simulation of the Talbot interferometer, and verified that the large reflected wavefront from a freeform (50 μmPV) can be retrieved from the Talbot image. For the stitching, we introduce the algorithm and confirm by simulation that alignment errors of the surface under test can be estimated and reduced. In addition, we estimate the measurement error of both the Talbot interferometer and stitching. In the experiment, the Talbot interferometer was assembled using Fizeau interferometer and a displacement sensor. Furthermore, we suggest a way to assemble the optical system for freeform surface measurement.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Angela Davies for her inspirational guidance toward the advancement and completion of the research project. I have learned a lot from our numerous technical discussions. In addition, I would like to thank Dr. Christopher Evans not only for his contribution as my co advisor and committee member but also for his unconditional assistance. I am also thankful to Dr. Jimmie Miller for his thoughtful contributions and Dr. Thomas J. Suleski for help in understanding diffraction effects of the grating. I would like to acknowledge the other members of the research team: Marashi Cameron and Porter Mitchell who contributed significantly to the mechanical design and assembly of our optical system. I would like to thank the University of North Carolina at Charlotte, the Department of Physics and Optical Science, and the Department of Mechanical Engineering.

I would like to thank Canon Inc. and Canon USA Inc. for giving me an opportunity to study optics in the University of North Carolina at Charlotte. Lastly, I am especially grateful to my wife, family and my parents for their constant support and encouragement, and for their genuine interest in my work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1. Freeform optics

Freeform optics are optical parts composed of a surface or surfaces that lack translational or rotational symmetry. Compared with spherical or aspherical optics, freeform optics offer more design degrees of freedom [1], and this enables reduced system size [2], low aberration, increased field of view and larger spectral bandwidth. However, since it is difficult to design, manufacture and measure a freeform surface, they are not yet commonly incorporated into optical designs. In recent years, with advances in computing and fabrication [3], freeform optics are being used in various advanced optical products such as head mount displays [4], laser printers [5], astronomy-based system [6] and extreme-ultraviolet (EUV) lithography [7].

The Alvarez lens [8] [9]is an example of freeform optics. Its surface is the so-called 'monkey saddle' and can be expressed by

$$z(x,y) = A\left(\frac{1}{3}x^3 + xy^2\right), \qquad \textbf{(1)}$$

where (x, y) denote Cartesian coordinates in the plane, and $\boldsymbol{A}$ is the coefficient whose unit is mm^{-2}. Figure 1 shows the Alvarez lens surface when the coefficient $\boldsymbol{A}$ is 0.0012 mm^{-2}, and the aperture size is 10 mm.

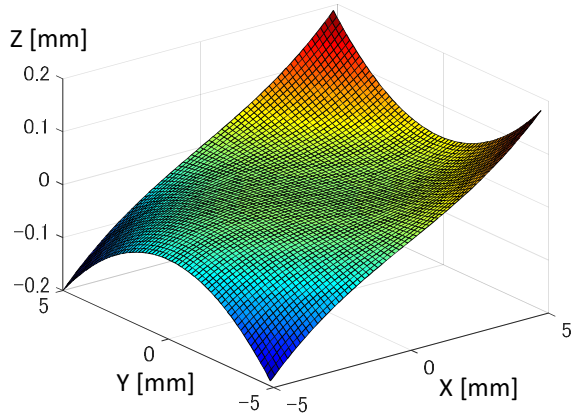


Figure 1 Alvarez lens surface.

The Alvarez lens is normally used in pairs as is shown in Figure 2.

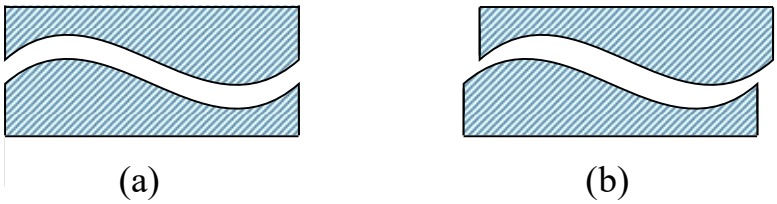


Figure 2 Cross-section illustrations of the Alvarez lens. (a) with no offset and (b) lateral displacement.

With a lateral shift of the Alvarez lens by x_0 in the x-direction, the surface geometry is expressed by

$$z(x - x_0, y) = A\left\{\frac{1}{3}(x - x_0)^3 + (x - x_0)y^2\right\}$$

$$= z(x, y) - Ax_0(x^2 + y^2) + Ax_0^2\left(x - \frac{1}{3}x_0\right). \quad \textbf{(2)}$$

The first term of Equation (2) is the same as the Alvarez lens without shifting. The second term is a spherical component. The third term shows tilt and piston. Therefore, when a wavefront passes through two sheared Alvarez lenses, the optical path impact leads to a change in the wavefront curvature by having a lateral relative shift between the two Alvarez surfaces in the x-direction. This means the Alvarez lens can control the focus distance with a compact overall physical system size. For this reason, the Alvarez lens is used for glasses [10].

The Alvarez lens surface cannot be measured by a conventional interferometer because of the extreme deviation from a sphere or a plain surface, and the corresponding steep slopes. Therefore, we chose the measurement of an Alvarez lens to define measurement specifications of the measurement system proposed in this thesis.

1.2. Freeform surface metrology

The quality of any optical surface effects performance, therefore measuring the surface is one of the most important steps in the manufacturing process. However,

compared with spherical and aspherical surfaces, which are used in conventional products and relatively easy to measure, it is much more difficult to measure a freeform surface. Therefore, the measurement of freeform surfaces is still very challenging. Several methods have been introduced for measuring freeform surfaces.

1.2.1. Profilometer

A profilometer is widely used to measure the freeform surface. In particular, the Talysurf series [11], an Ultrahigh accurate 3-D profilometer (UA3P) [12], and MahrSurf [13] are commonly used in industry. The profilometer uses the stylus to drag along the test surface and measures its deflection. The profilometer can measure almost arbitrary surface with high spatial resolution because the height of the surface is measured point by point. However, measurement time is long, and the stylus often damages the test surface.

1.2.2. CGH interferometer

In a CGH (computer generated hologram) interferometer [14], a CGH is inserted behind the transmission sphere to form a specified wavefront to match the freeform surface. Since the incident ray and reflected ray from the test surface are almost normal to the test surface, the wavefront on the detector is almost flat, and the fringe density is not high, which realizes a high accuracy measurement. In addition, the measurement time is short. However, the CGH must be designed and fabricated for each specific freeform surface. It is time-consuming, and the cost is high. Moreover, it is sensitive to alignment errors of the CGH and of the test sample [15].

1.2.3. Deflectometry

A deflectometry uses structured light and measures the slope profile of the test surface by detecting the distortion of a pattern (often a sinusoidal pattern) reflected from the test surface. As examples of the deflectometry, the technique known as Software Configurable Optical Test System (SCOTS) [16] [17] was developed at the University of

Arizona, and Phase Measuring Deflectometry (PMD) [18] was developed at the University of Erlangen-Nuremberg. Deflectometry has a relatively high dynamic range, but the measurement accuracy is not high. In addition, it requires calibration of the projection screen and, camera, in addition to a geometric calibration of the entire system for improving the accuracy.

1.2.4. Shearing interferometer

In a shearing interferometer [19], the test wavefront is divided by a beam splitter, shear plate or grating [20], and an interferogram is formed by overlapping the wavefront with a small laterally-shifted duplicate of itself. The interferogram is thus a measure of the sheared wavefront. In this way, the interference signal is self-referencing and therefore a shearing interferometer does not require a reference wavefront as in conventional interferometry. Also, it has a large dynamic range due to a low fringe density. Therefore, shearing interferometry is used extensively in diverse applications such as inspection of a beam wavefront [21], testing of optical components and systems [22], the beam collimation check [23] [24] [25] and the study of flow and diffusion phenomena in gases [26]. Low spatial resolution is one drawback compared to conventional interferometry, and 2-dimensional integration is necessary, which is sensitive to random noise such as CCD (charge-coupled device) noise [27]. In addition, when the tilt of a wavefront is to be measured, the tilt leads to a piston component of the retrieved sheared wavefront, which is equivalent to the lateral shift of the whole grating in the grating implementation of a shearing interferometer. Therefore, in the case of applying the FT method (discussed in Chapter 3.3.1) to retrieve the tilt in the incident wavefront, drift of the grating must be considered. Also, in the case of using the phase shift method described in Chapter 3.3.2, prominent position repeatability of the grating is required to measure the tilt in the wavefront.

1.2.5. Stitching interferometer

In conventional interferometry, it is difficult to measure a large test surface because optics in the interferometer must be similarly large. In addition, it is difficult to measure a surface with a large deviation from a flat or spherical surface because the fringe density becomes too high to detect. To overcome these limitations, in stitching interferometry [28] [29] [30] [31], sub-aperture measurements are taken and the test sample is translated and tilted between measurements, usually with a common overlap region between adjacent measurements. The measurements are then connected (stitched together) by estimating and eliminating alignment errors of the test sample from the difference in the overlap area of the measurements. Thus, this method can measure large sized flat [32] or spherical surfaces [33]. However, if we measure the test surface whose deviation from a flat or sphere is large with a stitching interferometer, the sub-apertures must be small, meaning the number of measurements must be large. A large number of measurements increase measurement time and usually measurement error, effectively reducing the dynamic range. The uncertainty in the data for the overlap regions is reduced by averaging, but there is an added uncertainty related to the estimation of the alignment error of the test sample.

1.2.6. Tilted wave interferometer

The tilted wave interferometer (TWI) [34] has been discussed in the literature over the last few years. The TWI is a Twyman–Green interferometer with a two-dimensional extended source. An extended source is well modeled as a collection of incoherent point sources over an array in a plane perpendicular to the optical axis. The light from each point source gives rise to a tilted wavefront incident to the test surface. The TWI measures only rays which pass through the beam stop in the Fourier plane of the imaging optics to limit the fringe density. In other words, the TWI measures only the beam reflecting from the test surface which shows no tilt with respect to the reference. The measurement is rapid;

however, the dynamic range is limited by a maximum slope and maximum aperture size of the test surface. In addition, calibration of the TWI is very difficult and requires careful consideration of retrace errors, which occurs when the optical path of the incident ray and reflected ray from the test surface become significantly different [35] [36].

1.3. Motivation

Compared with these methods, our system provides rapid and flexible (high dynamic range) freeform measurements. Our method is a combination of the Talbot interferometry and the stitching technique. A Talbot interferometer is a type of shearing interferometer that consists of only a grating and a CCD and is thus particularly compact. The small aperture of our Talbot interferometer and its shearing-based principal mean that local measurements can be made of relatively extreme freeform surfaces. Our instrument combines this advantage with stitching to enable full measurements of large extreme freeform optics. In addition, the compact size of our instrument is compatible with on-machine metrology and this is highly desirable for freeform surface fabrication where the loss of coordinate system registration with removal of the part is particularly detrimental. The stitching aspect adds two important capabilities. First, a maximum measurable wavefront gradient when restricted to a sub-aperture region means larger freeforms with more extreme gradients can be measured. Second, high spatial resolution measurements of large freeforms are possible with a small sensor and a small optical system.

As shown in Figure 3, our measurement system is characterized by high accuracy and high dynamic range for large components. In addition, the instrument is low cost, compact, non-contact, and relatively easy to calibrate and characterize. Therefore, our system is an important contribution to the field of freeform surface metrology.

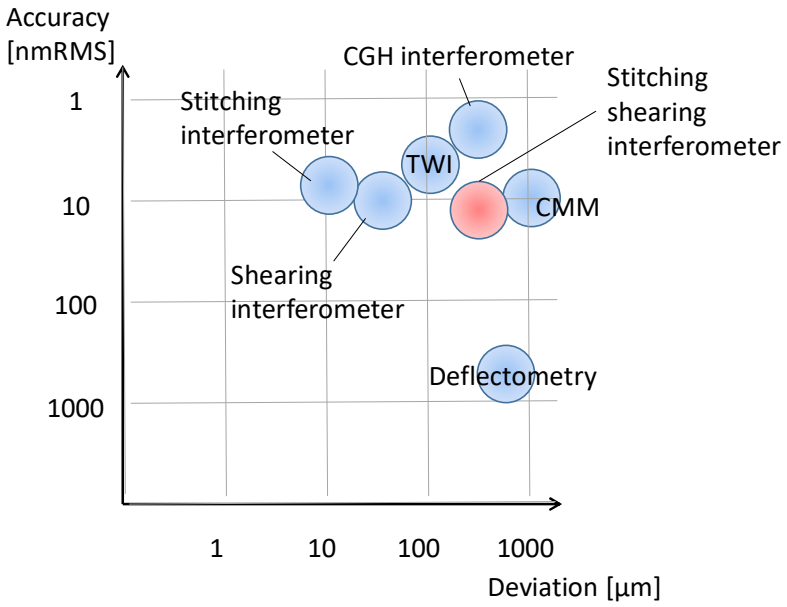


Figure 3 Specification about dynamic range and accuracy.

1.4. Thesis overview

This thesis consists of seven Chapters. Chapter 2 describes the principle of our measurement system. Chapter 3 describes the theory, sheared wavefront retrieval methodology, specification, and simulation of the Talbot interferometer. Chapter 4 describes the principle and simulation of the stitching technique. Chapter 5 describes the uncertainty estimation of the Talbot interferometer, and the stitching process by simulation with the mathematical software, MATLAB®. The retrace error is also investigated using an optical design software, Zemax. Chapter 6 describes the assembly and alignment of both the Talbot interferometer and the full optical system. Finally, conclusions and future works are given in Chapter 7.

2. Freeform measurement system

2.1. Principle

The schematic of the freeform measurement system is shown in Figure 4. The light source is a He-Ne laser whose wavelength is 632.8 nm. A collimated beam is formed by a collimator lens which is then transferred to the test surface by two relay lenses. The beam reflected from the test surface passes back through the relay lens pair and is imaged onto the Talbot interferometer [37]. Here, the test surface and the Talbot interferometer are conjugates of each other by the two relay lenses. The Talbot interferometer is a wavefront measurement device which consists of a cross grating and an image detector array such as a charge-coupled device (CCD). The wavefront can be calculated from the distortion of the self-image which occurs at constant Talbot distances from the grating. Since the sheared wavefront is detected, the Talbot interferometer can measure a more steeply sloped wavefront than a Fizeau interferometer. However, measuring the whole test surface in one-shot is not possible for freeforms with large slope variation because very large wavefront gradients lead to dense fringes that become unresolvable on the sensor. Therefore, small local areas of the test surface are measured with motion between the sampled areas realized with a 5-axis stage. When the test sample is moved to a new area, it is adjusted with potentially all 5-axes of the stage so that the wavefront gradients on the Talbot interferometer are as small as possible. The sub-aperture slope profile of the test surface at each location is obtained by reverse raytracing with the measured wavefront. In reverse raytracing, ray tracing is performed from the Talbot interferometer to the test surface using design values and calibration values of the optical system, and the ray coordinates and the ray angle or wavefront on the test surface are calculated. The sub-aperture shape is calculated by integrating the sub-aperture slope profile two-dimensionally. Finally, the full aperture test shape profile is obtained by connecting the multiple sub-aperture measurements by estimating and removing the alignment degrees of freedom between sub-

aperture measurements.

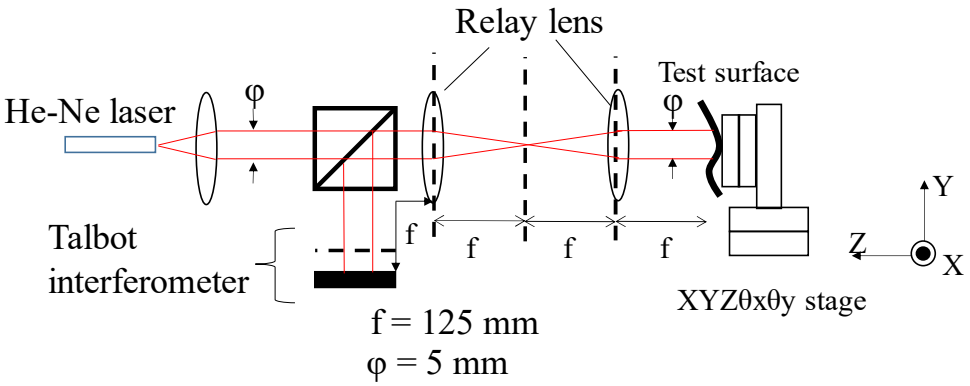


Figure 4 The schematic diagram of the freeform measurement.

2.2. Measurement procedure

The measurement procedure is shown in Figure 5. The steps are the following:

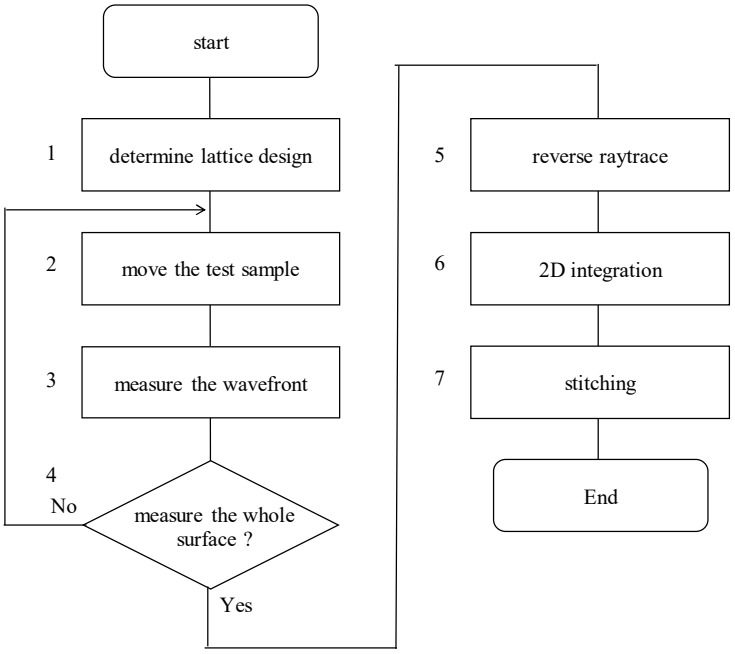


Figure 5 Measurement flow.

1. Determine sub-aperture measurement layout

   Determine the number of measurements, the test sample position, and orientations for each sub-aperture measurement. The detail is described in Chapter 4.2

2. Move the test sample

Move the test sample with the 5-axis stage following the sub-aperture measurement layout. For each sub-aperture measurement, tip and tilt the test sample so that the difference between the measured wavefront and the nominal expected wavefront (based on ideal surface shape) becomes minimal.

3. Measure the wavefront

   Measure the wavefront of the beam reflected from the test surface with the Talbot interferometer. The detail is described in Chapter 3.

4. Measure the whole surface?

   If all sub-apertures have been measured, proceed to the next step. If not, go back to step 2.

5. Reverse raytrace

   Reverse raytrace from the Talbot interferometer to test surface is carried out with the measured wavefront and the optical design of the measurement system using an optical design software package (Zemax), obtaining the ray coordinate and slope on the test sample.

6. Two-dimensional integration

   At this point, the measured data is the surface slope profile. The shape profile is calculated by 2-dimensional integration. The detail is described in Chapter 3.4.

7. Stitching

   The whole test surface is obtained by stitching together the sub-aperture measurements with removal of the alignment degrees of freedom in each measurement. Details are described in Chapter 4.

3. Talbot interferometer

3.1. Background

A Talbot interferometer is one kind of shearing interferometers. As is shown in Figure 6, it is composed of a grating and a photo detector such as a charge-coupled device (CCD). The incoming wavefront is divided mainly into the 0 and $\pm 1^{st}$ order diffraction beams by the grating, and they form an interferogram on a detector array positioned in the space beyond the grating. In particular, the self-image of the grating occurs at regular distances called the Talbot distance or the Talbot length behind the grating, which is called the Talbot effect [38]. If the incoming wave has aberration, the image is distorted from an exact grating pattern. Therefore, we can obtain the wavefront aberration by analyzing the distortion of the image. Compared with other shearing interferometers, the Talbot interferometer is compact, and therefore suitable for sub-aperature measurements of extreme wavefronts with large slopes.

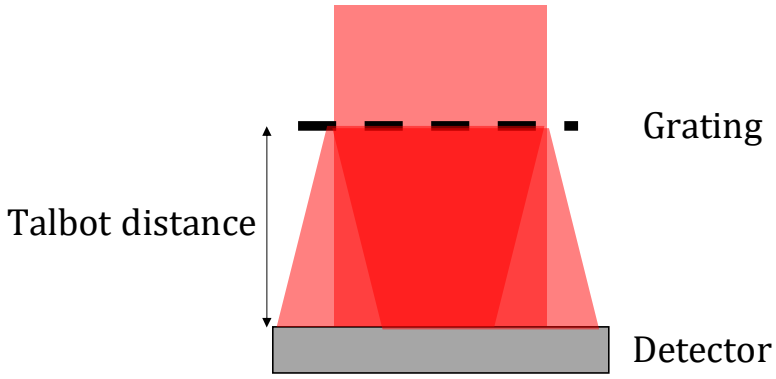


Figure 6 Talbot interferometer.

3.2. Principle

There are three ways to explain the Talbot interferometer: Fresnel approximation theory, angular spectrum propagation, and five beams interference model. The Fresnel approximation theory is the simplest way of theoretically understanding the Talbot phenomenon. The angular spectrum propagation calculation is not approximate; therefore

it is useful for accurate simulation. However, it requires a lot of computational memory. The five beams interference model is approximate and neglects higher order diffraction effects. Since it is not computationally challenging, it is suitable for simulating the 2-dimensional Talbot image. The details of the three theories are discussed below.

3.2.1. Fresnel approximation theory

For simplicity, we assume that the grating has a cosine amplitude transmittance characteristic with pitch ***p*** in the horizontal direction. Assuming the lateral coordinates on the grating are ξ, η, the amplitude transmittance t_0 is expressed by

$$t_0(\xi,\eta) = \frac{1}{2}\left[1 + m\cos\left(\frac{2\pi}{p}\xi\right)\right], \quad \textbf{(3)}$$

where ***m*** indicates contrast. According to the Fresnel approximation theory [38], the electric field ***E*** which propagates distance ***L*** is calculated by

$$E(x,y) = \frac{1}{i\lambda L}e^{i\frac{2\pi}{\lambda}L}\int t_0(\xi,\eta)exp\left\{i\frac{\pi}{\lambda L}[(x-\xi)^2+(y-\eta)^2]\right\}d\xi d\eta$$

$$= \frac{1}{i\lambda L}e^{i\frac{2\pi}{\lambda}L}t_0(\xi,\eta)\otimes exp\left[i\frac{\pi}{\lambda L}(\xi^2+\eta^2)\right]$$

$$= \frac{1}{i\lambda L}e^{i\frac{2\pi}{\lambda}L}\mathcal{F}^{-1}\mathcal{F}\left\{t_0(\xi,\eta)\otimes exp\left[i\frac{\pi}{\lambda L}(\xi^2+\eta^2)\right]\right\}$$

$$= \frac{1}{i\lambda L}e^{i\frac{2\pi}{\lambda}L}\mathcal{F}^{-1}\left\{\mathcal{F}\left(t_0(\xi,\eta)\right)\mathcal{F}\left(exp\left[i\frac{\pi}{\lambda L}(\xi^2+\eta^2)\right]\right)\right\}$$

$$= \frac{1}{i\lambda L}e^{i\frac{2\pi}{\lambda}L}\mathcal{F}^{-1}\left\{\left[\frac{1}{2}\delta(f_x,f_y) + \frac{1}{4}m\delta\left(f_x-\frac{1}{p},f_y\right)\right.\right.$$

$$\left.\left. + \frac{1}{4}m\delta\left(f_x+\frac{1}{p},f_y\right)\right] i\frac{\lambda L}{\pi}exp\left[-i\pi\lambda L\left(f_x^2+f_y^2\right)\right]\right\}$$

$$= \frac{1}{\pi}e^{i\frac{2\pi}{\lambda}L}\left[\frac{1}{2} + \frac{1}{4}mexp\left(\frac{-i\pi\lambda L}{p^2}\right)exp\left(\frac{i2\pi x}{p}\right) + \frac{1}{4}mexp\left(\frac{-i\pi\lambda L}{p^2}\right)exp\left(\frac{-i2\pi x}{p}\right)\right]$$

$$= \frac{1}{2\pi} e^{i\frac{2\pi}{\lambda}L} \left\{ \left[1 + mexp\left(\frac{-i\pi\lambda L}{p^2}\right) cos\left(\frac{2\pi x}{p}\right) \right] \right\}, \quad \textbf{(4)}$$

where $\boldsymbol{\lambda}$ is the wavelength, $\boldsymbol{f_x}$, $\boldsymbol{f_y}$ are coordinates of the Fourier domain, $\boldsymbol{\mathcal{F}}$ is the Fourier transform operator, $\boldsymbol{\mathcal{F}^{-1}}$ is the inverse Fourier transform operator, $\otimes$ expresses convolution. The irradiance is expressed by

$$I(x,y) = \left(\frac{1}{2\pi}\right)^2 \left[1 + 2mcos\left(\frac{\pi\lambda L}{p^2}\right) cos\left(\frac{2\pi x}{p}\right) + m^2 cos^2\left(\frac{2\pi x}{p}\right) \right]. \quad \textbf{(5)}$$

When the distance $\boldsymbol{L}$ is $2np^2/\lambda$ (n = 1,2,3 ...), the irradiance is expressed by

$$I(x,y) = \left(\frac{1}{2\pi}\right)^2 \left[1 + mcos\left(\frac{2\pi x}{p}\right) \right]^2. \quad \textbf{(6)}$$

Equation (6) shows the self-image of the grating. When the distance $\boldsymbol{L}$ is $(2n-1)p^2/\lambda$ (n = 1,2,3 ...), the intensity is expressed by

$$I(x,y) = \left(\frac{1}{2\pi}\right)^2 \left[1 - mcos\left(\frac{2\pi x}{p}\right) \right]^2. \quad \textbf{(7)}$$

Equation (7) shows the reversed self-image of the grating. As mentioned above, the phenomena by which the self-image appears at a constant interval is called the Talbot effect. In addition, since any transmittance functions are a linear combination of harmonic functions, the Talbot distance does not depend on the spatial frequency of the transmittance function.

3.2.2. Angular spectrum propagation

When we know the electric field ***E(x,y,0)*** at z = 0, we can calculate the electric field ***E(x,y,z)*** at any $\boldsymbol{z}$ without approximation by the angular spectrum propagation [39]. The theory is described below.

Using the Fourier transform, the electric field ***E(x,y,z)*** is expressed by

$$E(x,y,z) = \iint \Gamma(f_x, f_y; z) exp\left[-i2\pi\left(f_x x + f_y y\right)\right] df_x df_y. \quad \textbf{(8)}$$

where $\boldsymbol{f_x}$, $\boldsymbol{f_y}$ are coordinates in the Fourier domain, $\boldsymbol{\Gamma}$ is a Fourier component of the electric field $\boldsymbol{E}$. The electric field $\boldsymbol{E}$ must satisfy the Helmholtz equation,

$$\nabla^2 E + \mathrm{k}^2 E = 0. \quad \textbf{(9)}$$

Substituting Equation (8) into Equation (9), we obtain

$$\frac{\partial^2 \Gamma(f_x,f_y;z)}{\partial z^2} + \left\{\mathrm{k}^2 - (2\pi f_x)^2 - \left(2\pi f_y\right)^2\right\}\Gamma\left(f_x, f_y; z\right) = 0. \quad \textbf{(10)}$$

When $1 - (\lambda f_x)^2 - \left(\lambda f_y\right)^2 < 0$, $\boldsymbol{\Gamma}$ which satisfies Equation (10) can be written in the form

$$\Gamma\left(f_x, f_y; z\right) = \Gamma\left(f_x, f_y; 0\right) exp\left[-kz\sqrt{(\lambda f_x)^2 + \left(\lambda f_y\right)^2 - 1}\right]. \quad \textbf{(11)}$$

Equation (11) indicates that $\boldsymbol{\Gamma}$ decays exponentially in the z direction, and this is called the evanescent field. When $1 - (\lambda f_x)^2 - \left(\lambda f_y\right)^2 > 0$, $\boldsymbol{\Gamma}$ can be written in the form

$$\Gamma\left(f_x, f_y; z\right) = \Gamma\left(f_x, f_y; 0\right) exp\left[ikz\sqrt{1 - (\lambda f_x)^2 - \left(\lambda f_y\right)^2}\right]. \quad \textbf{(12)}$$

$\boldsymbol{\Gamma\left(f_x, f_y; 0\right)}$ is expressed by

$$\Gamma\left(f_x, f_y; 0\right) = \iint \mathrm{E}(x, y; 0) exp\left[i2\pi\left(f_x x + f_y y\right)\right] dxdy. \quad \textbf{(13)}$$

Therefore, substituting Equations (12) - (13) into Equation (8), the electric field $\boldsymbol{E}$ is calculated by

$$E(x,y,z) = \iint \Gamma\left(f_x, f_y; 0\right) exp\left[ikz\sqrt{1 - (\lambda f_x)^2 - \left(\lambda f_y\right)^2}\right] exp\left[-i2\pi\left(f_x x + f_y y\right)\right] df_x df_y$$

$$= \iint \iint E(x,y,0) exp\left[i2\pi\left(f_x x + f_y y\right)\right] dxdy$$

$$exp\left[ikz\sqrt{1 - (\lambda f_x)^2 - \left(\lambda f_y\right)^2}\right] exp\left[-i2\pi\left(f_x x + f_y y\right)\right] df_x df_y. \quad \textbf{(14)}$$

Equation (14) shows that the electric field ***E(x,y,z)*** can be calculated by Fourier transforming ***E(x,y,0)***, multiplying by $exp\left[ikz\sqrt{1 - (\lambda f_x)^2 - \left(\lambda f_y\right)^2}\right]$ and inverse Fourier transforming.

We assume that the light comes to the CCD through the grating. The electric field $\boldsymbol{E_0}$ on the grating is expressed by

$$E_0(x,y) = A(x,y)t(x,y)exp[ikW(x,y)], \quad \textbf{(15)}$$

where $\boldsymbol{A}$ is the amplitude of the light, $\boldsymbol{t}$ is the transmittance of the grating, $\boldsymbol{k}$ is a wavenumber, and $\boldsymbol{W}$ is the wavefront of the light. According to Equation (14), the electric field on the

plane of distance $\boldsymbol{L}$ from the grating is written as

$$E(x,y,L) = \mathcal{F}^{-1}\left\{\mathcal{F}\left(E_0(x,y)\right)exp\left[-ikL\sqrt{1-(\lambda f_x)^2-\left(\lambda f_y\right)^2}\right]\right\}, \quad \textbf{(16)}$$

In Equation (16), the expression inside the square root is physically meaningful for positive values. A negative value, meaning

$$1-(\lambda f_x)^2-\left(\lambda f_y\right)^2 < 0. \quad \textbf{(17)}$$

leads to an amplitude that grows exponentially and this is not physical. Therefore, the value which satisfies Equation (17) must be zero for simulation before the inverse Fourier transform is calculated. When the distance $\boldsymbol{L}$ is long, a lot of computational memory is necessary because the exponential term changes quickly with the Fourier domain coordinates $\boldsymbol{f_x}$, $\boldsymbol{f_y}$, and high-density sampling is required to capture this detail. Otherwise, an error due to aliasing occurs in the calculation result. The irradiance $\boldsymbol{I}$ on the plane of $\boldsymbol{L}$ distance from the grating is calculated by

$$I(x,y,L) = |E(x,y,L)|^2. \quad \textbf{(18)}$$

3.2.3. Five beams interference model

Talbot effect using a cross grating can be calculated from the interference of five-beam (0, ±1 in the x-direction, ±1 in the y-direction order diffraction beams). As first order diffraction beams are added to a linear phase $\mp 2\pi x/p$ or $\mp 2\pi y/p$, the electric fields of 0, ±1 order diffraction beams are respectively expressed by

$$E_0(x,y) = A_0 exp\left(ikW(x,y)\right), \quad \textbf{(19)}$$

$$E_{1,0,j_x}(x,y) = A_1 exp\left[ik\left\{W(x-a,y)+\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)\right\}\right], \quad \textbf{(20)}$$

$$E_{-1,0,j_x}(x,y) = A_1 exp\left[ik\left\{W(x+a,y)-\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)\right\}\right], \quad \textbf{(21)}$$

$$E_{0,1,j_y}(x,y) = A_1 exp\left[ik\left\{W(x,y-a)+\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right], \quad \textbf{(22)}$$

$$E_{0,-1,j_y}(x,y) = A_1 exp\left[ik\left\{W(x,y+a)-\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right], \quad \textbf{(23)}$$

where j_x, j_y is an integer which indicates the phase step in the x-direction and y-direction, respectively, A_0, and A_1 are the amplitudes of the 0 and ±1st order diffracted beams, respectively, k is the wave number, W is the wavefront, a is the amount of shear, p is the pitch of the grating, N is the number of phase shift. Referring to Appendix A, the irradiance of the five-beam interference is expressed by

$$I_{j_x,j_y}(x,y) = \left|E_0(x,y) + E_{1,0,j_x}(x,y) + E_{-1,0,j_x}(x,y) + E_{0,1,j_x}(x,y) + E_{0,-1,j_x}(x,y)\right|^2$$

$$= A_0^2 + 4A_1^2 + 4B_x(x,y)cos\left[\delta W_x(x,y) + \frac{2\pi}{N}j_x\right] + 4B_y(x,y)cos\left[\delta W_y(x,y) + \frac{2\pi}{N}j_y\right]$$

$$+ 2A_1^2 cos\left[2\left\{\delta W_x(x,y) + \frac{2\pi}{N}j_x\right\}\right] + 2A_1^2 cos\left[2\left\{\delta W_y(x,y) + \frac{2\pi}{N}j_y\right\}\right]$$

$$+8A_1^2 cos\left(\delta W_x(x,y) + \frac{2\pi}{N}j_x\right) cos\left(\delta W_y(x,y) + \frac{2\pi}{N}j_y\right) cos\left[\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2} - \frac{d^2W(x,y)}{dy^2}\right\}\right] \quad , \tag{24}$$

where

$$B_x(x,y) = A_0A_1 \cos\left[\frac{ka^2}{2}\frac{d^2W(x,y)}{dx^2}\right], \tag{25}$$

$$B_y(x,y) = A_0A_1 \cos\left[\frac{ka^2}{2}\frac{d^2W(x,y)}{dy^2}\right], \tag{26}$$

$$\delta W_x(x,y) = ka\left\{\frac{dW(x,y)}{dx} + \frac{a^2}{6}\frac{d^3W(x,y)}{dx^3}\right\} - \frac{2\pi}{p}x, \tag{27}$$

$$\delta W_y(x,y) = ka\left\{\frac{dW(x,y)}{dy} + \frac{a^2}{6}\frac{d^3W(x,y)}{dy^3}\right\} - \frac{2\pi}{p}y. \tag{28}$$

The first and second terms of Equation (24) are constant, and the coefficients of the third and fourth term, $4B_x$, $4B_y$ include the variables x and y, but they are assumed to be constant because the spatial change is very small. The fifth and sixth terms are twice the frequency of the third and fourth terms, and unnecessary for the wavefront retrieval. These terms are removed in the process of the sheared wavefront retrieval described in Chapter 3.3. Focusing on the x-shear wavefront, the intensity is expressed by

$$I_{j_x,j_y}(x,y) = 4cos\left[\delta W_x(x,y) + \frac{2\pi}{N}j_x\right]\{B_x(x,y) + C(x,y)\}$$

$$+2A_1^2 cos\left[2\left\{\delta W_x(x,y) + \frac{2\pi}{N}j_x\right\}\right] + I_Y(x,y), \quad \textbf{(29)}$$

.

where

$$C(x,y) = 2A_1^2 cos\left(\delta W_y(x,y) + \frac{2\pi}{N}j_y\right) cos\left[\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2} - \frac{d^2W(x,y)}{dy^2}\right\}\right], \quad \textbf{(30)}$$

$$I_Y(x,y) = A_0^2 + 4A_1^2 + 4B_y(x,y)cos\left[\delta W_y(x,y) + \frac{2\pi}{N}j_y\right]$$

$$+2A_1^2 cos\left[2\left\{\delta W_y(x,y) + \frac{2\pi}{N}j_y\right\}\right]. \quad \textbf{(31)}$$

The $\boldsymbol{\delta W_x(x,y)}$ in the argument of the first cosine term in Equation (29) is obtained by the sheared wavefront retrieval method described in Chapter 3.3.

3.3. Sheared wavefront retrieval

There are three methods to retrieve the sheared wavefront from the Talbot image: the Fourier transform method [40] [41], the x y phase shift method [42] [43] and the diagonal phase shift method. As is shown in Table 1, the Fourier transform method retrieves the sheared wavefront from only one image. Therefore, it is fast, robust against vibration and drift, and an actuator is not necessary. However, the high spatial frequency component of the wavefront cannot be obtained because it is filtered in the process of the wavefront retrieval. The x y phase shift method enables us to obtain the high frequency component of the wavefront, but the grating must be moved tens of microns in the x and y direction, respectively. Thus, it requires two actuators, is time-consuming, and is sensitive to external vibration and drift. In the diagonal phase shift method, the grating is moved in the diagonal direction (45 degree from the x-direction). Compared with the x y phase shift method, it requires only one actuator, and the wavefront is retrieved from fewer images, but the spatial resolution is a little worse. Other features are similar to the x y phase shift method. Table

1 shows the pros and cons of each method.

Table 1 Comparison between Fourier transform, phase shift and diagonal phase shift method.

	Fourier transform method	X y phase shift method	Diagonal phase shift method
Pros	- Fast - Simple design - Not sensitive to vibration	Can obtain high frequency component	Can obtain high frequency component
Cons	High frequency component is filtered	- Time-consuming - Sensitive to vibration - Requires two actuators	- Time-consuming - Sensitive to vibration - Requires an actuator

These three methods are described below. In addition, the data obtained by these methods must be unwrapped because it is wrapped 2π. Unwrapping is described in Chapter 3.3.4.

3.3.1. Fourier transform method

The Fourier transform (FT) method was published by Takeda [40] [41], and enables us to retrieve the wavefront from a single fringe pattern. Therefore, the configuration of the measurement system becomes simpler because the actuator to move the reference part or grating is not necessary. The FT method is widely applied to fringe analysis such as ultrashort pulse analysis [44], thin film thickness measurements [45] and evaluation of imaging optics for EUV lithography [46]. On the other hand, the Fourier transform method has a drawback in that the high frequency component of the wavefront is filtered.

When $\Delta W_x(x,y)$ is expressed by

$$\Delta W_x(x,y) = \frac{dW(x,y)}{dx} + \frac{a^2}{6}\frac{d^3W(x,y)}{dx^3}, \quad \textbf{(32)}$$

from Equation(29), the interferogram including aberration is expressed by

$$I_{0,0}(x,y)\sim 2\mathrm{A}_0\mathrm{A}_1\left[e^{i\mathrm{k}\left\{\Delta W_x(x,y)\mathrm{a}-\frac{\lambda}{\mathrm{p}}\mathrm{x}\right\}}+e^{-i\mathrm{k}\left\{\Delta W_x(x,y)\mathrm{a}-\frac{\lambda}{\mathrm{p}}\mathrm{x}\right\}}\right]\{B_x(x,y)+C(x,y)\}$$
$$+\mathrm{A}_1^2\left[e^{i2\mathrm{k}\left\{\Delta W_x(x,y)\mathrm{a}-\frac{\lambda}{\mathrm{p}}\mathrm{x}\right\}}+e^{-i2\mathrm{k}\left\{\Delta W_x(x,y)\mathrm{a}-\frac{\lambda}{\mathrm{p}}\mathrm{x}\right\}}\right]+I_Y(x,y). \quad \textbf{(33)}$$

Taking the Fourier transform of both side, we have

$$\mathcal{F}\left\{I_{0,0}(\mathrm{x},\mathrm{y})\right\}\sim \mathrm{I}_0\delta(f_x)+2\mathrm{A}_0\mathrm{A}_1B_x(x,y)\int e^{i2\pi\left\{\frac{1}{\lambda}\Delta W_x(x,y)\mathrm{a}-\left(\frac{1}{\mathrm{p}}+f_x\right)\mathrm{x}\right\}}+$$
$$e^{-i2\pi\left\{\frac{1}{\lambda}\Delta W_x(x,y)\mathrm{a}-\left(\frac{1}{\mathrm{p}}-f_x\right)\mathrm{x}\right\}}\mathrm{dx}+2\mathrm{A}_0\mathrm{A}_1\mathcal{F}\left\{C(x,y)\left[e^{i\mathrm{k}\left\{\Delta W_x(x,y)\mathrm{a}-\frac{\lambda}{\mathrm{p}}\mathrm{x}\right\}}+\right.\right.$$
$$\left.\left.e^{-i\mathrm{k}\left\{\Delta W_x(x,y)\mathrm{a}-\frac{\lambda}{\mathrm{p}}\mathrm{x}\right\}}\right]\right\}+\mathrm{A}_1^2\int e^{i2\pi\left\{\frac{2}{\lambda}\Delta W_x(x,y)\mathrm{a}-\left(\frac{2}{\mathrm{p}}+f_x\right)\mathrm{x}\right\}}+e^{-i2\pi\left\{\frac{2}{\lambda}\Delta W_x(x,y)\mathrm{a}-\left(\frac{2}{\mathrm{p}}-f_x\right)\mathrm{x}\right\}}\mathrm{dx}+$$
$$\mathcal{F}\{I_Y(x,y)\}. \quad \textbf{(34)}$$

The data near the frequency ***x/p*** is filtered and then shifted to the center of the Fourier plane. We then obtain $\Gamma(f_x,f_y)$ as

$$\Gamma\left(f_x,f_y\right)=2\mathrm{A}_0\mathrm{A}_1B_x(x,y)\int e^{i2\pi\left\{\frac{a}{\lambda}\Delta W_x(x,y)-f_x\mathrm{x}\right\}}dx. \quad \textbf{(35)}$$

The inverse Fourier-transformed signal of $\Gamma(f_x,f_y)$ can be expressed as

$$\mathcal{F}^{-1}\left\{\Gamma\left(f_x,f_y\right)\right\}=2\mathrm{A}_0\mathrm{A}_1B_x(x,y)\,e^{i2\pi\left\{\frac{a}{\lambda}\Delta W_x(x,y)\right\}}. \quad \textbf{(36)}$$

Next, $\Delta W_x(x,y)$ is calculated by

$$\Delta W_x(x,y)=\frac{\lambda}{2\pi a}atan\left(\frac{Im\left[\mathcal{F}^{-1}\{\Gamma(f_x,f_y)\}\right]}{Re\left[\mathcal{F}^{-1}\{\Gamma(f_x,f_y)\}\right]}\right), \quad \textbf{(37)}$$

where ***Im[]*** indicates the imaginary part, ***Re[]*** indicates the real part. Assuming the following equation is true,

$$\frac{dW(x,y)}{dx}\gg\frac{a^2}{6}\frac{d^3W(x,y)}{dx^3}, \quad \textbf{(38)}$$

the sheared wavefront is obtained by

$$\frac{\mathrm{dW(x,y)}}{\mathrm{dx}}=\frac{\lambda}{2\pi a}atan\left(\frac{Im\left[\mathcal{F}^{-1}\{\Gamma(f_x,f_y)\}\right]}{Re\left[\mathcal{F}^{-1}\{\Gamma(f_x,f_y)\}\right]}\right), \quad \textbf{(39) Figure 7}\text{Figure 7 shows the}$$

above procedure of the sheared wavefrront retrieval using simulation data.

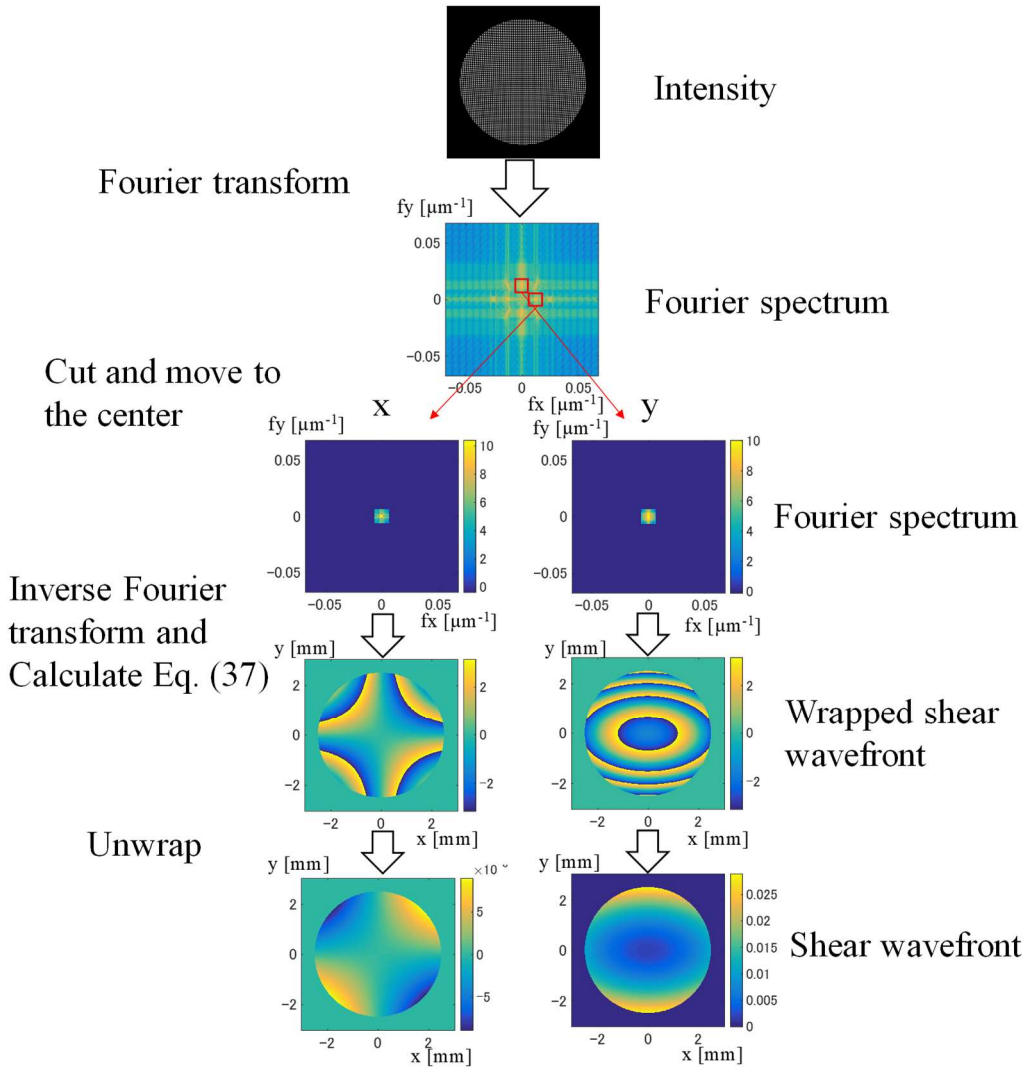


Figure 7 Sheared wavefront retrieval flow.

3.3.2. X y phase shift method

The x y phase shift can be implemented by shifting the grating in both the x and y directions. Although the Talbot image includes twice the frequency of the self-image as is shown in the second term of Equation (29), it cancels in the calculation process of the wavefront retrieval. For example, the four step wavefront retrieval is calculated by

$$I_{3,0} - I_{1,0} = 4\left\{cos\left(\delta W_x(x,y) + \frac{3}{4}2\pi\right) - cos\left(\delta W_x(x,y) + \frac{1}{4}2\pi\right)\right\}\{B_x(x,y) + C(x,y)\} + 2A_1^2\left\{cos\left[2\left\{\delta W_x(x,y) + \frac{3}{4}2\pi\right\}\right] - cos\left[2\left\{\delta W_x(x,y) + \frac{1}{4}2\pi\right\}\right]\right\} = 8sin\big(\delta W_x(x,y)\big)\{B_x(x,y) + C(x,y)\}. \quad \textbf{(40)}$$

$$I_{2,0} - I_{0,0} = 4\left\{cos\left(\delta W_x(x,y) + \frac{2}{4}2\pi\right) - cos\big(\delta W_x(x,y)\big)\right\}\{B_x(x,y) + C(x,y)\} + 2A_1^2\left\{cos\left[2\left\{\delta W_x(x,y) + \frac{2}{4}2\pi\right\}\right] - cos[2\{\delta W_x(x,y)\}]\right\} = -8cos\big(\delta W_x(x,y)\big)\{B_x(x,y) + C(x,y)\}. \quad \textbf{(41)}$$

$$-\frac{I_{3,0}-I_{1,0}}{I_{2,0}-I_{0,0}} = tan\{\delta W_x(x,y)\}. \quad \textbf{(42)}$$

From Equations (40) - (42), we can see that the twice frequency terms of the self-image cancel. Since δW_x is expressed by Equation (27), the sheared wavefront is calculated by

$$\frac{dW(x,y)}{dx} \sim \frac{dW(x,y)}{dx} + \frac{a^2}{6}\frac{d^3W(x,y)}{dx^3} = \frac{1}{ak}\text{atan}\left(\frac{I_{3,0}-I_{1,0}}{I_{2,0}-I_{0,0}}\right) + \frac{\lambda x}{ap}. \quad \textbf{(43)}$$

Except for N = 3, the double–frequency term disappears. Therefore, the error due to it is small. Similarly, the y-shear wavefront is also retrieved.

3.3.3. Diagonal phase shift method

When the grating is shifted in the diagonal direction, the irradiance is approximately expressed by

$$I_{j,j}(x,y) \sim A_0^2 + 4A_1^2 + 4B_x(x,y)cos\left(kp\frac{dW(x,y)}{dx} - \frac{2\pi}{p}x + \frac{2\pi}{N}j\right) + 4B_y(x,y)cos\left(kp\frac{dW(x,y)}{dy} - \frac{2\pi}{p}y + \frac{2\pi}{N}j\right), \quad \textbf{(44)}$$

where ***j*** is an integer that represents the phase step. A Fourier transform of the irradiance

gives the result shown in Figure 8. The Fourier spectrum except for the signal regarding the x-shear wavefront is substituted zero as is shown in Figure 9. When it is inverse Fourier transformed, the cosine of x is obtained as shown in Figure 10..

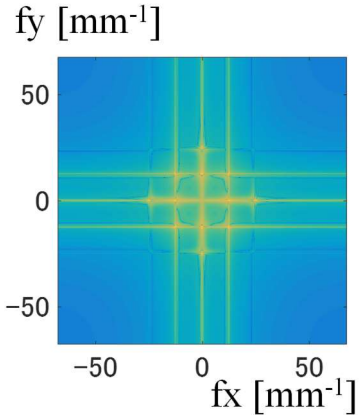


Figure 8 Fourier spectrum of irradiance (log of absolute).

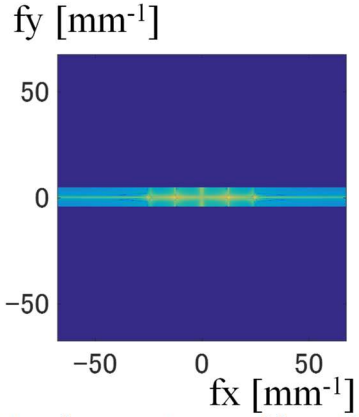


Figure 9 Filtered Fourier spectrum of irradiance (log of absolute).

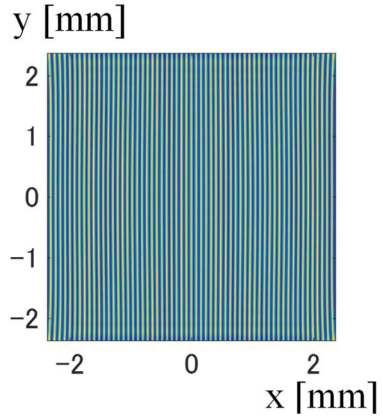


Figure 10 Irradiance whose y frequency component is filtered.

When the same processing is applied to all the phase shifted irradiance images, the x-sheared wavefront is obtained using the same equation as the phase shift method in

Chapter 3.3.2. Also, y-shear wavefront is obtained in the analogous way. Compared with the Fourier transform method, the diagonal phase shift method enables us to obtain higher frequency component of the wavefront. Compared with the phase shift method, the diagonal phase shift method requires fewer irradiance images, and only one direction shift of the grating.

3.3.4. Unwrapping

Many unwrapping algorithms have been developed [47] [48]. In our research, a complicated algorithm is not necessary because the measured wavefront will be smooth, meaning there are no vortices or discontinuities. Therefore, we apply the simple unwrapping method called the flood-fill algorithm [49].

3.4. Two-dimensional integration

What we directly obtain from the Talbot interferometer is the sheared wavefront or sheared shape. Therefore, 2-dimensional integration is necessary to obtain the wavefront or shape of the test surface. As is shown in Table 2, three methods of integration have been explored in this work: a Fourier transform method, a fitting method with the sheared wavefront and a path integral method. In the Fourier transform method [50], integration is done by Fourier transforming twice. It can retrieve high frequency wavefront, but the boundary condition affects the result because the sheared wavefront is often non-contiguous near the boundary. The fitting method [51] [52] [53] is called modal method. The sheared wavefront is integrated by making many functions and fitting the measured sheared wavefront with functions. This method is fast, and robust to noise. But it has the drawback that the wavefront except for functions cannot be retrieved. In addition, it requires a lot of computational memories. The path integral method [54] [55] is called zonal method. The integration is implemented by keeps adding adjacent data. This method takes

time, and is sensitive to the noise, but the high frequency component of wavefront can be retrieved. Table 2 shows the pros and cons of three methods.

Table 2 Integration methods.

	Fourier transform method	Fitting method (modal method)	Path integral method (zonal method)
Pros	- Fast - High spatial resolution	- Fast - Robust to noise	High spatial resolution
Cons	Boundary condition affects the result	- Low spatial resolution - Requires a lot of memories	- Time-consuming - sensitive to noise

Three methods are explained below.

3.4.1. Fourier transform method

For simplicity, consider one dimension. The sheared wavefront $\boldsymbol{W_m}$ measured by Talbot interferometer is expressed by

$$W_m(x) = W(x+a) - W(x), \tag{45}$$

where $\boldsymbol{W}$ is the incident wavefront, and $\boldsymbol{a}$ is the amount of shear. To obtain the incident wavefront $\boldsymbol{W}$, Fourier transform is implemented.

$$\begin{aligned}\mathcal{F}\{W_m(x)\} &= \int W(x+a)e^{-i2\pi f}\ dx - \mathcal{F}\{W(x)\} \\ &= \int W(x+a)e^{-i2\pi\ (x+a)}dxe^{i2\pi fa} - \mathcal{F}\{W(x)\} \\ &= \int W(x)e^{-i2\pi f}\ dxe^{i2\pi fa} - \mathcal{F}\{W(x)\} \\ &= \left(e^{i2\pi fa} - 1\right)\mathcal{F}\{W(x)\}.\end{aligned} \tag{46}$$

From Equation (46), the Fourier transform of the incident wavefront $\boldsymbol{W}$ is expressed by.

$$\mathcal{F}\{W(x)\} = \frac{\mathcal{F}\{W_m(x)\}}{\left(e^{i2\pi fa}-1\right)}. \tag{47}$$

Finally, the incident wavefront $\boldsymbol{W}$ is obtained by inverse Fourier transforming Equation

(47).

$$W(x) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{W_m(x)\}}{(e^{i2\pi fa}-1)}\right\}. \quad \textbf{(48)}$$

The Fourier transform method is fast, but the boundary condition of the data often leads to errors. This is because the wavefront is not continuous on the boundary of the data, and errors appear in the high frequency component due to the boundary discontinuity.

3.4.2. Fitting method with sheared wavefront

This is called the modal method. First, the basic functions of the sheared wavefront are chosen and calculated. For example, when using Fringe Zernike polynomials [56] as the basis set, the x-shear wavefront $\boldsymbol{Z_{xs}}$ and y-shear wavefront $\boldsymbol{Z_{ys}}$ are given by

$$\begin{aligned} Z_{xs}(x,y,j) &= Z_j(x+a,y) - Z_j(x,y), \\ Z_{ys}(x,y,j) &= Z_j(x,y+a) - Z_j(x,y), \end{aligned} \quad \textbf{(49)}$$

where $\boldsymbol{j}$ is a positive integer, $\boldsymbol{Z_j}$ is ***jth*** Fringe Zernike polynomials, $\boldsymbol{a}$ is the amount of shear. Second, the measured x-shear wavefront $\boldsymbol{W_x}$, and y-shear wavefront $\boldsymbol{W_y}$ are expressed with sheared Zernike polynomials $\boldsymbol{Z_{xs}}$ and $\boldsymbol{Z_{ys}}$ by

$$\begin{bmatrix} W_x(x,y) \\ W_y(x,y) \end{bmatrix} = \sum_{j=1}^{N} C_j \begin{bmatrix} Z_{xs}(x,y,j) \\ Z_{ys}(x,y,j) \end{bmatrix}, \quad \textbf{(50)}$$

where $\boldsymbol{C}$ is the coefficients of both the x-shear wavefront $\boldsymbol{Z_{xs}}$ and the y-shear wavefront $\boldsymbol{Z_{ys}}$, and $\boldsymbol{N}$ is the number of Zernike polynomials considered for the fit. The coefficients $\boldsymbol{C}$ of the sheared Zernike polynomials are calculated by fitting the measured x-shear wavefront $\boldsymbol{W_x}$, and y-shear wavefront $\boldsymbol{W_y}$ with the sheared Zernike polynomials $\boldsymbol{Z_{xs}}$ and $\boldsymbol{Z_{ys}}$. The coefficients $\boldsymbol{C}$ are normally obtained by least square method or singular value decomposition [57].

Finally, the wavefront $\boldsymbol{W}$ is retrieved by

$$W(x,y) = \sum_{j=1}^{N} C_j Z(x,y,j). \quad \textbf{(51)}$$

This method is robust against the irregular (spike) error and a random noise due to a CCD

noise. The drawbacks are that the high spatial frequency components of the wavefront are difficult to represent because this method would require a fit out to very high order. By the way, as the wavefront obtained from the Talbot image is expressed by Equation (32), Equation (52) will be better regarding x-shear wavefront $\boldsymbol{Z_{xs}}$ and y-shear wavefront $\boldsymbol{Z_{ys}}$ in Equation (49).

$$Z_{xs}(x,y,j) = \frac{dZ_j(x,y)}{dx} + \frac{a^2}{6}\frac{d^3Z_j(x,y)}{dx^3},$$

$$Z_{ys}(x,y,j) = \frac{dZ_j(x,y)}{dy} + \frac{a^2}{6}\frac{d^3Z_j(x,y)}{dy^3}. \quad \textbf{(52)}$$

Although the third-order term can be ignored when solving for a low-amplitude wavefront, it is necessary to take this term into account in the case of a large wavefront, which is usually the case on a freeform surface.

3.4.3. Path integral method

Path integral method is one kind of zonal methods [54] [55]. Integration starts from arbitrary point on 2-dimensional grid data and keeps adding adjacent data. As an example, let us think it starts from point A in Figure 11, and we calculate the integral value on point B.

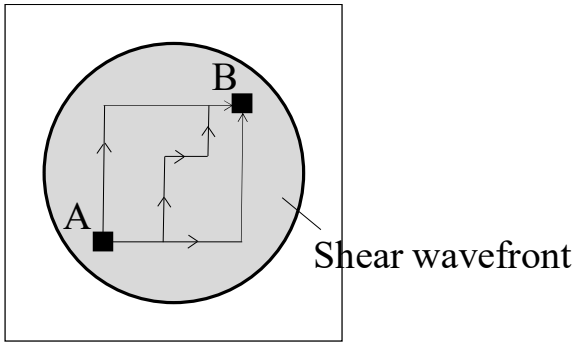


Figure 11 Integration paths.

There are different multiple paths. To reduce the integration error, it is obtained by calculating the average of integral value of different multiple paths. In addition, since this way is not enough to reduce the integration error, the start location is dispersed, and the

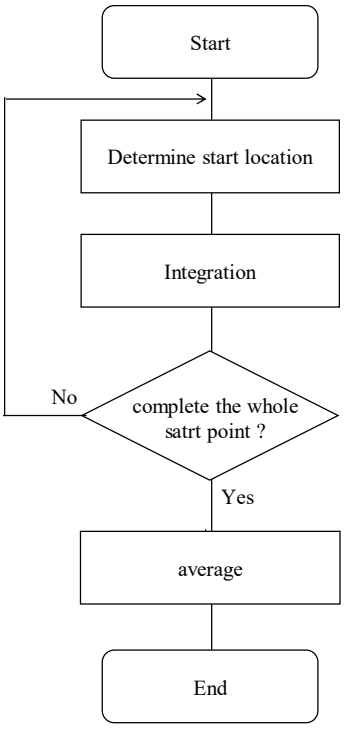


Figure 12 Integration flow.

average of their integral values is taken as the final integral value. A concrete flow is shown in Figure 12. In step 1, start location is dispersed so that they are not located on the vertical or horizontal line each other, and they separate each other as far as possible like Figure 13. In step 2, the integral value is calculated by averaging the integral value of all shortest paths. Then we repeat the integration with changing the start location. In step 4, final integral value is obtained by averaging the integral value calculated in step 2.

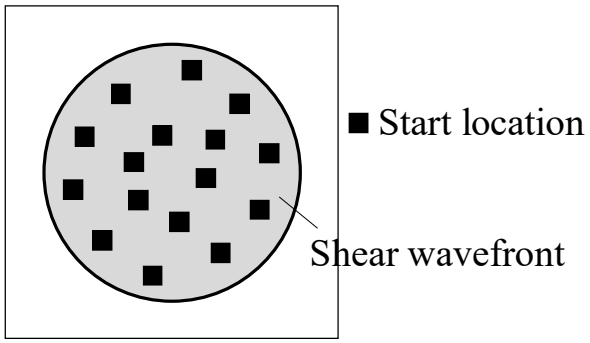


Figure 13 Start location of integration.

Local integration is implemented by calculating the difference between adjacent integral values. Specifically, first, for each of three adjacent sheared wavefronts ***dW(x₁), dW(x₂), dW(x₃)***, fitting is performed with a following quadratic polynomial ***f(x)*** of lateral coordinate ***x*** to calculate coefficients ***a₂***, **a₁**, ***a₀***.

$$f(x) = a_2x^2 + a_1x + a_0. \quad \textbf{(53)}$$

Integral value ***F(x)*** of Equation (53) is expressed by

$$F(x) = \frac{1}{3}a_2x^3 + \frac{1}{2}a_1x^2 + a_0x. \quad \textbf{(54)}$$

The difference ***D(x)*** between adjacent integral values is calculated by

$$D(x_1) = F(x_2) - F(x_1).$$
$$D(x_2) = F(x_3) - F(x_2). \quad \textbf{(55)}$$

After calculating the whole difference ***D(x)*** in measured sheared wavefront, integration is implemented by sequentially adding this difference ***D(x)*** along the path shown in Figure 11. The sheared wavefront is approximated as being a quadratic function in Equation (53) and integrated, which is called Simpson's rule [58].

3.5. Measurement range

The distortion of the fringe is proportional to the sheared wavefront. Therefore, measurement range is approximately limited by the second derivative of the wavefront. We explain it with equations as below.

The fringe of the Talbot image is expressed by Equation (29), and its phase of forth term ***ψ*** is expressed by

$$\Psi = 2k\left\{\frac{dW(x)}{dx}a + \frac{a^3}{6}\frac{d^3W(x)}{dx^3} + \frac{\lambda}{p}x\right\}$$
$$\sim 2k\left\{\frac{dW(x)}{dx}a + \frac{\lambda}{p}x\right\}, \quad \textbf{(56)}$$

According to the sampling theory, the following equation must be satisfied.

$$2\left|ka\frac{dW(x_{n+1},y)}{dx}-\frac{2\pi}{p}x_{n+1}-\left(ka\frac{dW(x_n,y)}{dx}-\frac{2\pi}{p}x_n\right)\right|<\pi,$$

$$\left|\frac{a}{\lambda}\left(\frac{dW(x_{n+1},y)}{dx}-\frac{dW(x_n,y)}{dx}\right)-\frac{q}{p}\right|<\frac{1}{4}, \quad \textbf{(57)}$$

where $\boldsymbol{q}$ is a CCD pixel size, $\boldsymbol{x_n}$ (n=1,2,3…) is x-coordinate of ***n-th*** CCD pixel. Substituting the following approximation,

$$\frac{dW(x_{n+1},y)}{dx}-\frac{dW(x_n,y)}{dx}\sim q\frac{d^2W(x_n,y)}{dx^2}, \quad \textbf{(58)}$$

the measurement range is expressed by

$$\left|\frac{dW^2(x,y)}{dx^2}\right|<\frac{\lambda}{aq}\left(\frac{1}{4}-\frac{q}{p}\right). \quad \textbf{(59)}$$

As an example, substituting a = p = 80 μm, q = 4.6 μm, λ= 0.6328 μm, the right side is 0.33 mm^{-1}. This value is bigger than the maximal second derivative wavefront reflected from the Alvarez surface shown in Figure 1. Therefore, this Talbot interferometer can measure the Alvarez surface.

3.6. Spatial resolution

In case of wavefront retrieval using the phase shift method, the spatial resolution of Talbot interferometer is equal to the amount of shear $\boldsymbol{a}$. Using FT method, the spatial resolution becomes worse because many data in the Fourier domain is filtered. Specifically, Fourier spectrum shown in Figure 14 is obtained by Fourier transforming the Talbot image. As expressed in Equation (29), there are signals at 0, 1/p (=1/80 = 0.0125 μm^{-1}) and 2/p (= 0.025 μm^{-1}).

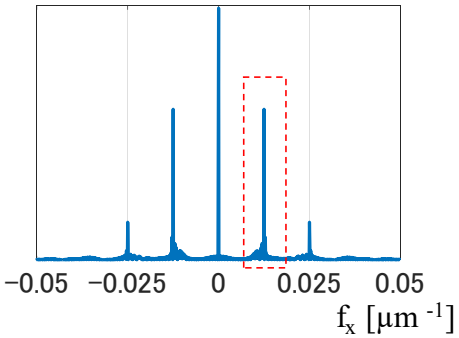


Figure 14 Fourier spectrum of the Talbot image.

In the FT method, wavefront is retrieved by cutting the Fourier spectrum data surrounded by the red dotted line in Figure 14. If the cutting area is large, wavefront retrieval error becomes large because the tails of the signal of frequency 0 and 2/p are included. Therefore, the cutting range is at most half the length between two signals, which means the maximum measurable frequency is 1/(2p). Thus, the spatial resolution with FT method is 2p (160 μm).

3.7. Dynamic range of a tilt wavefront

From Equation (24), the intensity of the Talbot interferogram is approximately expressed by

$$I(x) \sim cos\left[ka\frac{d}{dx}W(x) - \frac{2\pi}{a}x\right]. \quad \textbf{(60)}$$

Assuming the tilt wavefront is $\boldsymbol{tx}$, the intensity is expressed by

$$I(x) \sim cos\left[kat - \frac{2\pi}{a}x\right]. \quad \textbf{(61)}$$

Equation (61) shows that the tilt wavefront occurs lateral shift of the Talbot image. When the amount of the lateral shift is longer than half of the grating pitch ($\boldsymbol{a/2}$ = 40 μm), we cannot distinguish between plus tilt and minus tilt. Therefore, dynamic range of the tilt wavefront is expressed by

$$|kat| < \pi. \quad \textbf{(62)}$$

$$|t| < \frac{\lambda}{2a}. \quad \textbf{(63)}$$

Substituting λ = 632.8 nm, a = 80 μm, we obtain

$$|t| < 3.95 \times 10^{-3}. \quad \textbf{(64)}$$

This Talbot interferometer can measure the tilt wavefront less than 4 mrad.

3.8. Simulation

3.8.1. Angular spectrum propagation

We programed the angular spectrum propagation with MATLAB® to simulate the Talbot interferometer. MATLAB code is written in Appendix B. We did only one-dimension (x) because two-dimensional simulation (x, y) requires many memories and calculation time.

The electric field $\boldsymbol{E_0}$ on the grating is expressed by

$$E_0(x) = A(x)t(x)exp[ikW(x)], \quad \textbf{(65)}$$

where $\boldsymbol{W}$ is the wavefront of the light, **A** is the amplitude, and $\boldsymbol{t}$ is transmittance of the grating. In case of rectangle, the transmittance is expressed as

$$t(x) = \prod\left(\frac{x}{2p}\right) \otimes III\left(\frac{x}{p}\right), \quad \textbf{(66)}$$

where $\otimes$ indicates convolution, and it is shown in Figure 15 (a). In case of sinusoidal, the transmittance is expressed as

$$t(x) = \frac{1}{2} + \frac{1}{2}cos\left(\frac{2\pi x}{p}\right), \quad \textbf{(67)}$$

and shown in Figure 15 (b).

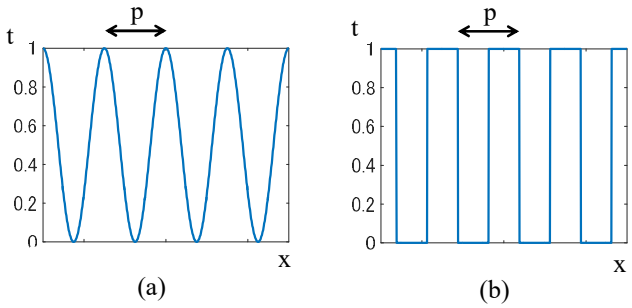


Figure 15 Transmittance amplitude of the grating. (a) sinusoidal and (b) rectangle.

Figure 16Figure 16 shows the intensity on the x-z plane calculated using Equation (16). We assumed that the grating pitch $\boldsymbol{p}$ was 80 μm, wavelength $\boldsymbol{\lambda}$ was 632.8 nm and wavefront

was constant (plane wave). As can be seen in Figure 16, the self-imaging takes place at the Talbot distance

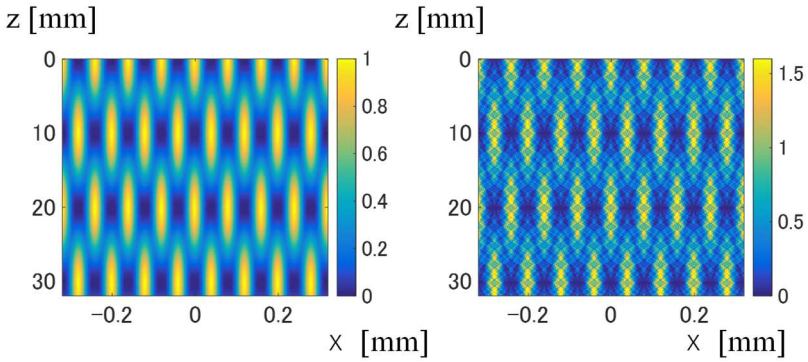


Figure 16 Irradiance on the x-z plane. (a) sinusoidal transmittance and (b) rectangle transmittance

$L = \frac{2np^2}{\lambda}, n = 1,2,3, \ldots$, and reversed self-image takes place at the distance $L = \frac{(2n-1)p^2}{\lambda}, n = 1,2,3, \ldots..$

In addition, we implemented the simulation to make sure the wavefront retrieval from Talbot image. In the same as above condition, we input the wavefront $\boldsymbol{W_{in}}$ on the grating, and calculated the intensity $\boldsymbol{I}$ on the plane (CCD) at the distance $L = p^2/\lambda = 10.114$ mm using the angular spectrum propagation. We calculated intensity four times with shifting the grating by 20 μm in the x-direction. Then, the wavefront $\boldsymbol{W_{out}}$ was retrieved from four intensities by four step phase shift method (Equations. (40) - (43)) and path integration method described in Chapter 3.4. To evaluate the retrieved wavefront, we calculated the electric field $\boldsymbol{E_{ref}}$ by propagating the wavefront to the CCD without the grating, and reference wavefront $\boldsymbol{W_{ref}}$ on the CCD was calculated by

$$W_{ref} = atan\left\{\frac{Imag(E_{ref})}{real(E_{ref})}\right\}. \quad \textbf{(68)}$$

As is shown in Figure 17, we compared the retrieved wavefront $\boldsymbol{W_{out}}$ with the reference wavefront $\boldsymbol{W_{ref}}$.

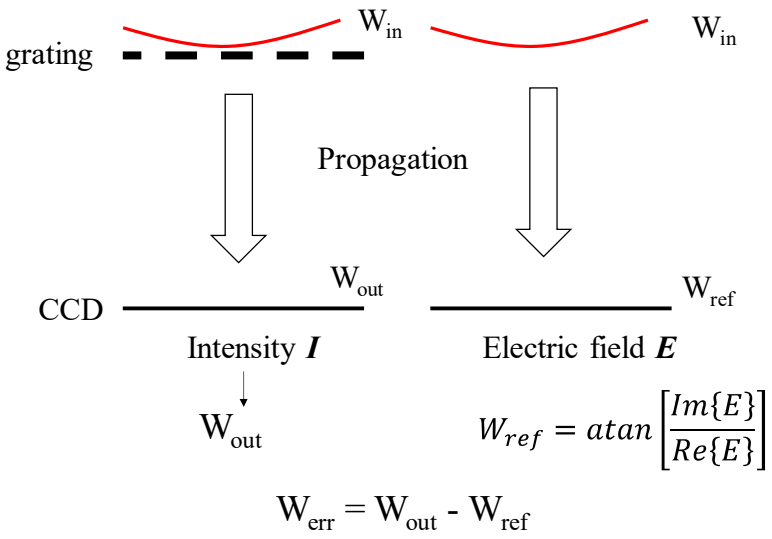


Figure 17 Schematic diagram for comparison between retrieved wavefront and reference wavefront.

The intensity is shown in Figure 18. The Talbot image is distorted a little by input wavefront $\boldsymbol{W_{in}}$.

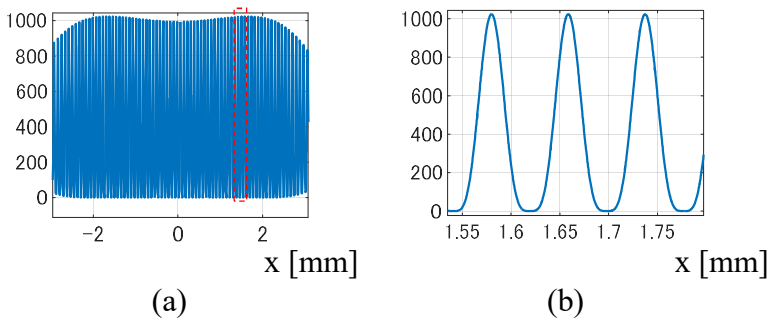


Figure 18 Intensity with sinusoidal grating calculated by angular spectrum propagation. (a) intensity and (b) magnified intensity.

Figure 19 (a) shows the reference wavefront $\boldsymbol{W_{ref}}$ and wavefront $\boldsymbol{W_{out}}$ retrieved from four intensities using phase shift method. Difference between them is shown in Figure 19 (b).

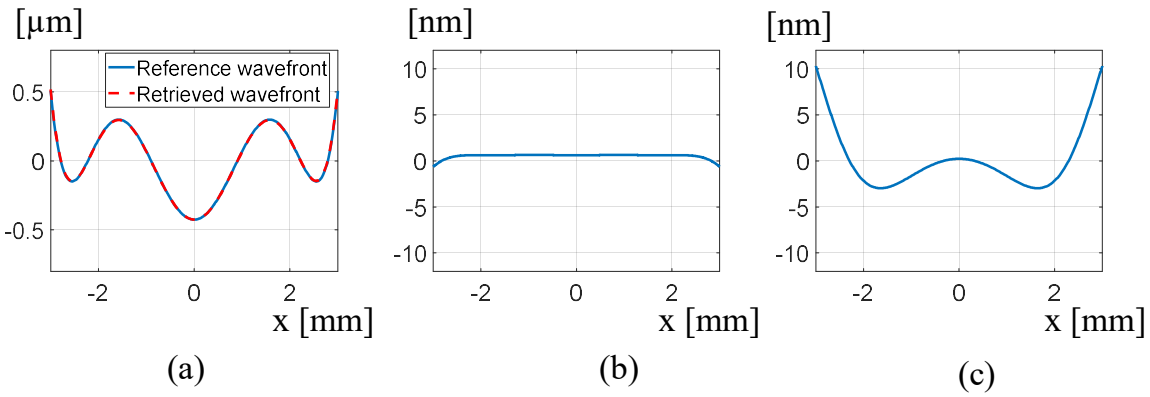


Figure 19 Comparison between retrieved wavefront and reference wavefront. (a) reference and retrieved wavefront, (b) difference between reference and retrieved wavefront and (c) difference between the reference wavefront and the retrieved wavefront with the approximation of Equation (38).

As you can see from Figure 19 (a) and (b), the wavefront can be retrieved accurately. Figure 19 (c) is the difference between the reference wavefront and the retrieved wavefront with the approximation of Equation (38). The retrieval error was 13.3 nmPV. This error is caused by ignoring the third-order differentiation term in the wavefront retrieval calculation.

Similarly, we calculated the intensity with the rectangle grating as is shown in Figure 20. High-order diffraction occurs at the edge of the rectangle grating, and the sharp rectangle irradiance boundary cannot be obtained [59]. Figure 21 (a) shows the reference wavefront $\boldsymbol{W_{ref}}$ and retrieved wavefront $\boldsymbol{W_{out}}$ from four intensities using phase shift method. Difference between them is shown in Figure 21 (b).

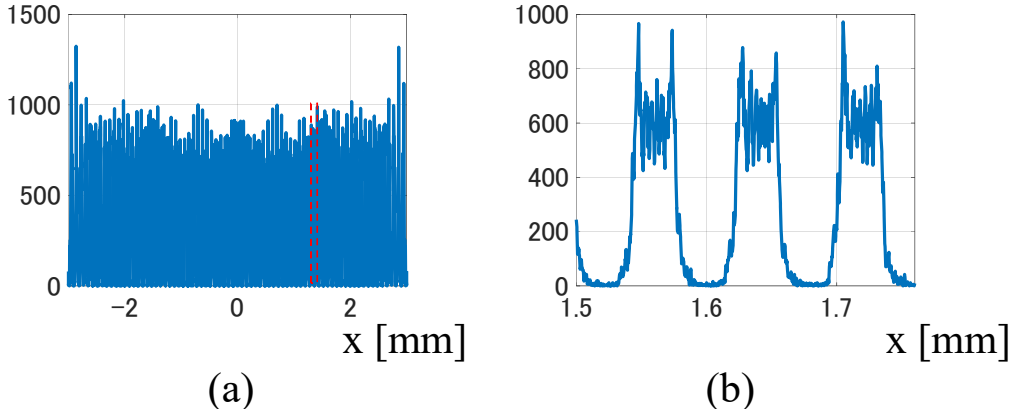


Figure 20 Intensity with rectangle grating. (a) intensity and (b) magnified intensity.

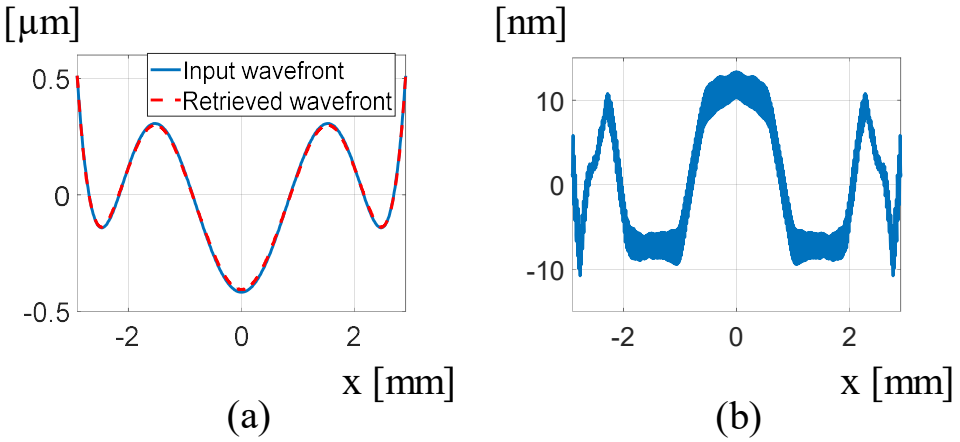


Figure 21 Comparison between retrieved wavefront with phase shift method and reference wavefront. (a) reference and retrieved wavefront and (b) difference between reference and retrieved wavefront.

The retrieved wavefront is not accurate. It will be because intensity is not sinusoidal shape, but the phase shift method assumes that the intensity is sinusoidal. Next, the results of the wavefront retrieved from the intensity shown in Figure 20 using FT method is shown in Figure 22.

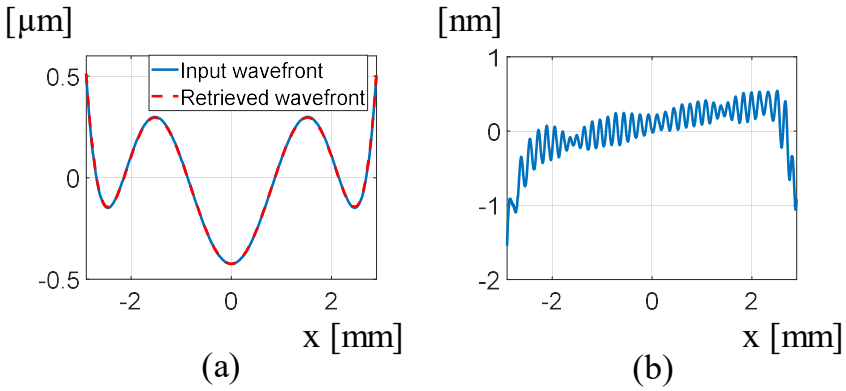


Figure 22 Comparison between retrieved wavefront and reference wavefront. (a) reference and retrieved wavefront with FT method and (b) difference between reference and retrieved wavefront.

The accuracy is better than that of the phase shift method. It is thought that the higher order diffraction from the edge of the rectangle grating occurs the intensity turbulence of high frequency, however it is filtered in the Fourier domain.

3.8.2. Five beams interference

Unlike the angular spectrum propagation, five beams interference does not consume a lot of computational memories. Therefore, it is possible to calculate the irradiance two-dimensionally. MATLAB code is written in Appendix C. Regarding the specific calculation, first, the wavefront on the CCD is set, and the electric field $\boldsymbol{E_{00}}$ is calculated with the wavefront using Equation (19). Second, we calculated four electric fields $\boldsymbol{E_{10}}$, $\boldsymbol{E_{-10}}$, $\boldsymbol{E_{01}}$, $\boldsymbol{E_{0-1}}$ with wavefront which is shifted by ±80 μm (grating pitch) in the x and y direction using Equations (20) - (23), respectively. Third, we calculated the intensity by adding five electric fields and squaring the absolute of them. As an example, the intensity when astigmatism is included is shown in Figure 23.

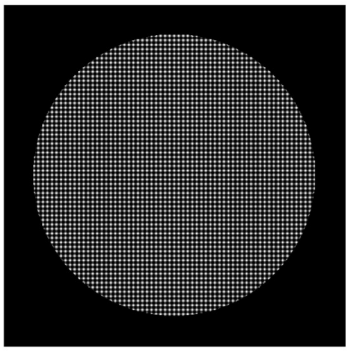


Figure 23 Irradiance of five beams interference.

Irradiance is distorted a little by input wavefront. The wavefront is retrieved from this intensity using FT method explained in Chapter 3.3.1. The results are shown in Figure 24.

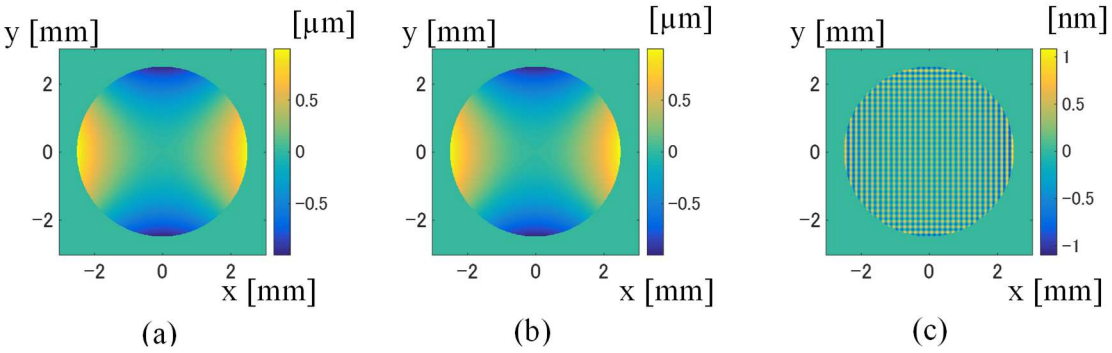


Figure 24 Wavefront retrieved from the irradiance using FT method. (a) retrieved wavefront, (b) reference wavefront and (c) difference between (a) and (b).

As can be seen in Figure 24, the wavefront is retrieved accurately. The high frequency error seen in Figure 24 (c) is considered to be caused by the fact that the tail of the signal of frequency 0 and 2/p is in the cut range of Fourier domain.

4. Stitching

4.1. Background

When the size of a test surface is larger than the sensor size or the detected wavefront is too large to measure in one-shoot, it can be measured by dividing and measuring it with the partial area of the test surface or wavefront and connecting them, which is called stitching [29]. To do that, the test sample or the sensor must be moved multiple times, and it leads to the (positional) alignment errors in measured wavefront, such as pitching, rolling, and shifting. In stitching, their alignment errors are estimated by the difference in the overlapped measurement data and removed. The principle is explained below.

4.2. Lattice design

To determine lattice design is to determine the location of the sub-aperture and tip-tilt amount of the test sample. The necessary conditions are to make sub-apertures overlap each other and cover the whole test sample with sub-apertures. The larger the overlapping area, the better the accuracy is because it is easier to estimate the alignment error of the test sample. However, this will increase the number of measurements and measurement time. After determining the location of the sub-aperture, we determine the amount of tip and tilt of the test sample so that the detected wavefront becomes minimal.

4.3. Principle

The measured shape z' including alignment errors is expressed as

$$z'_i(x - x_i, y - y_i) = z_i(x - x_i, y - y_i) + \sum_{n=1}^{5} a_{in} f_{in}(x - x_i, y - y_i), \quad \textbf{(69)}$$

where i is the number of sub-aperture, x_i, y_i are the central location of the ***i-th*** sub-aperture, z_i is the test shape of the i-th sub-aperture, f_{i1}, f_{i2}, f_{i3}, f_{i4}, f_{i5} indicate z error, x-tilt error, y-tilt error, x-shift error, y-shift error of the i-th sub-aperture measurement, respectively,

which are expressed as

$$f_{i1}(x-x_i, y-y_i) = 1, \quad \textbf{(70)}$$

$$f_{i2}(x-x_i, y-y_i) = x-x_i, \quad \textbf{(71)}$$

$$f_{i3}(x-x_i, y-y_i) = y-y_i, \quad \textbf{(72)}$$

$$f_{i4}(x-x_i, y-y_i) = z_i(x-x_i+\Delta x, y-y_i) - z_i(x-x_i, y-y_i), \quad \textbf{(73)}$$

$$f_{i5}(x-x_i, y-y_i) = z_i(x-x_i, y-y_i+\Delta y) - z_i(x-x_i, y-y_i), \quad \textbf{(74)}$$

$\boldsymbol{a_{in}}$ are coefficients of alignment errors of the ***i-th*** sub-aperture measurement. To minimize the difference between each sub-aperture measurement in the overlapped area, the following equation must be satisfied.

$$\delta = \sum_{i=1}^{N} \quad \sum_{j=1}^{N} \cap_{ij} \left\{ z_i(x-x_i, y-y_i) - z_j\left(x-x_j, y-y_j\right) \right\}^2 = min, \quad \textbf{(75)}$$

where $\cap_{ij}$ indicates the summation of the overlapped area of ***i-th*** and ***j-th*** sub-aperture. To obtain the coefficients $\boldsymbol{a_{in}}$, the following equation must be satisfied

$$\frac{\partial \delta}{\partial a_{in}} = 0. \quad \textbf{(76)}$$

Equation (76) is a matrix expressed by

$$P = FA, \quad \textbf{(77)}$$

where

$$P = \begin{bmatrix} \Delta \mathrm{P}_1 \\ \vdots \\ \Delta \mathrm{P}_N \end{bmatrix}, \quad \textbf{(78)}$$

$$\Delta \mathrm{P}_i = \sum_{j=1}^{N} \cap_{ij} \begin{bmatrix} \left\{ z'_j\left(x-x_j, y-y_j\right) - z'_i(x-x_i, y-y_i) \right\} f_{i1}(x-x_i, y-y_i) \\ \vdots \\ \left\{ z'_j\left(x-x_j, y-y_j\right) - z'_i(x-x_i, y-y_i) \right\} f_{i5}(x-x_i, y-y_i) \end{bmatrix}, \quad \textbf{(79)}$$

$$F = \begin{bmatrix} F_{1,1} & -F_{1,2} & \cdots & -F_{1,N} \\ -F_{2,1} & F_{2,2} & & \\ \vdots & & \ddots & \\ -F_{N,1} & & & F_{N,N} \end{bmatrix}, \quad \textbf{(80)}$$

where $\boldsymbol{F_{ij}}$ is 5×5 matrix, and ***s*** row ***t*** column component ($1 \leq \mathrm{s}, \mathrm{t} \leq 5$) is expressed by

$$F_{i,j}(s,t) = \cap_{ij} f_{is}(x - x_i, y - y_i) f_{jt}(x - x_j, y - y_j) \quad (\text{i} = \text{j}), \tag{81}$$

$$F_{i,i}(s,t) = \sum_{k \neq i}^{N} \cap_{ik} f_{is}(x - x_i, y - y_i) f_{it}(x - x_i, y - y_i) \quad (\text{i} \neq \text{j}), \tag{82}$$

$$A = \begin{bmatrix} \mathrm{A}_1 \\ \vdots \\ \mathrm{A}_N \end{bmatrix}, \tag{83}$$

$$\mathrm{A}_i = \begin{bmatrix} \mathrm{a}_{i1} \\ \vdots \\ \mathrm{a}_{i5} \end{bmatrix}, \tag{84}$$

The unknown coefficients $\boldsymbol{a}$ in Equation (69) can be obtained by least-squares solution as follow

$$a = (A^T A)^{-1} A^T. \tag{85}$$

We can subtract the alignment errors with coefficients $\boldsymbol{a}$ from each sub-aperture data. Finally, the whole surface is obtained by averaging the sub-aperture data in the overlapped area.

4.4. Simulation

To make sure the above principle, we implemented the stitching simulation using MATLAB®. The code is written in Appendix E. The shape of the test sample and its error shape were assumed to be the Alvarez surface and the spherical error shown in Figure 25 (a) and (b), respectively. Here, the size of the test sample is 10 × 10 mm^2.

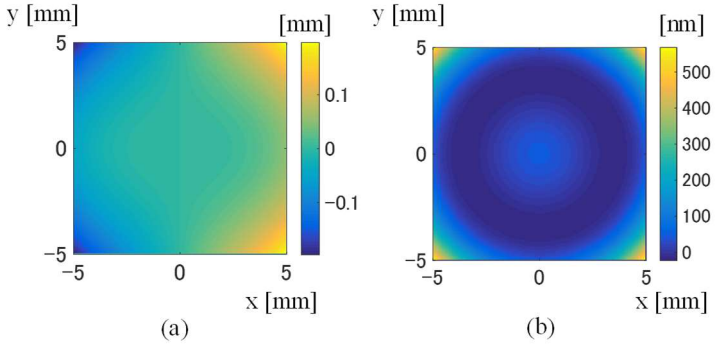


Figure 25 Test shape. (a) nominal shape and (b) error shape.

The simulation flow is shown in Figure 26.

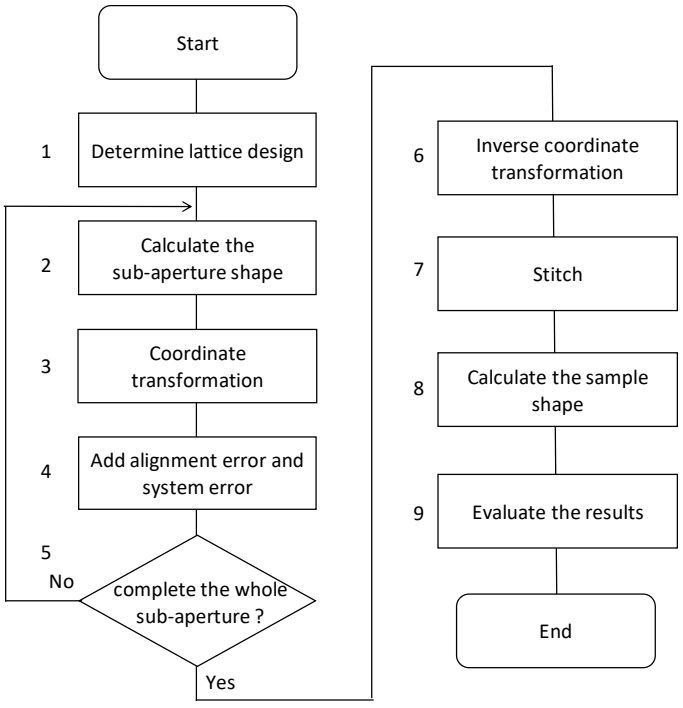


Figure 26 Stitching simulation flow.

In step 1, the lattice design is determined as is shown in Figure 27. We assumed the sub-aperture size was 4.2 mm × 3 mm rectangle, and the number of measurements was 12. The overlapping ratio of each sub-aperture is approximately 30%, and it becomes difficult to estimate the alignment error of the test sample when it is less than 30%. When the center position of the sub-aperture is displayed in Figure 27 (a), the overlap number is expressed in

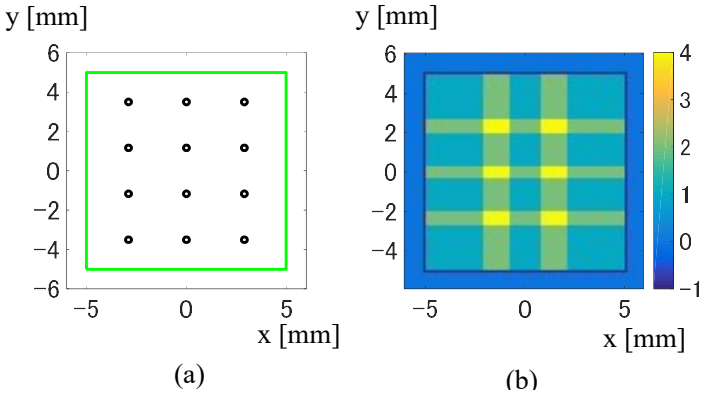


Figure 27 Lattice design. (a) center position of the sub-aperture and (b) overlap number.

Figure 27 (b). For example, since four sub-apertures are overlapped in six small yellow squares, their area is measured four times. The center coordinates of the sub-aperture and tip-tilt (θ_x, θ_y) of the test sample are shown in Table 3.

Table 3 Center coordinate of sub-aperture and tip-tilt of the test sample.

	x_0 [mm]	y_0 [mm]	z_0 [mm]	θx [rad]	θy [rad]
1	-2.9	3.5	-0.060	-0.024	-0.027
2	0	3.5	0.000	0.000	-0.017
3	2.9	3.5	0.060	0.024	-0.027
4	-2.9	1.17	-0.022	-0.008	-0.014
5	0	1.17	0.000	0.000	-0.004
6	2.9	1.17	0.022	0.008	-0.014
7	-2.9	-1.17	-0.022	0.008	-0.014
8	0	-1.17	0.000	0.000	-0.004
9	2.9	-1.17	0.022	-0.008	-0.014
10	-2.9	-3.5	-0.060	0.024	-0.027
11	0	-3.5	0.000	0.000	-0.017
12	2.9	-3.5	0.060	-0.024	-0.027

Here z_0, θ_x, θ_y are calculated by fitting the nominal shape of the test sample shown in Figure 25 (a) with Equations (70)-(72).

In step 2, according to the lattice design, we calculated the sample shape of every sub-aperture on the global coordinate (x, y, z). In step 3, we transformed it from global coordinate to measurement coordinate (x', y', z') using Equation (86).

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} \begin{bmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{bmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}. \quad \textbf{(86)}$$

In step 4, we added the alignment error (x, y, z shift, tip-tilt) of the test sample. Measurement shape of each sub-aperture including alignment error is shown in Figure 28.

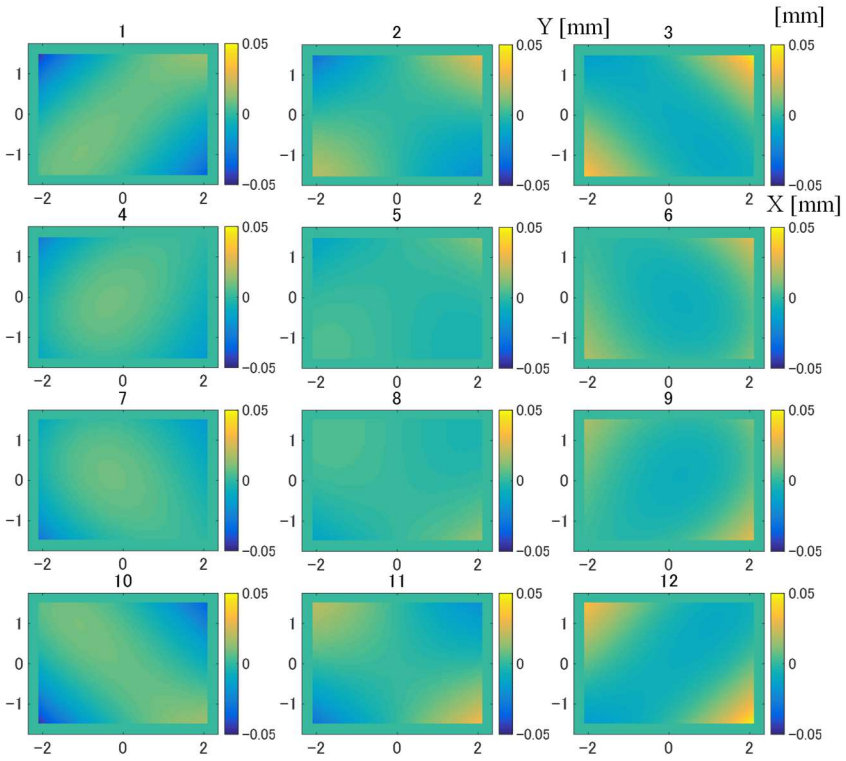


Figure 28 Sub-aperture shape.

After repeating step 2-4 until whole sub-aperture were calculated, we transformed from measurement coordinate to global coordinate using Equation (87).

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} cos\theta_y & 0 & -sin\theta_y \\ 0 & 1 & 0 \\ sin\theta_y & 0 & cos\theta_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & \sin\theta_x \\ 0 & -sin\theta_x & cos\theta_x \end{bmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}. \quad \textbf{(87)}$$

In step 6, the alignment errors are estimated by stitching described in Chapter 4.3, and removed them from sub-aperture shape. In step 7, whole sample shape was calculated by averaging the sub-aperture shape. Finally, we evaluated the stitching result by comparing it with input shape. The alignment error estimated by stitching is shown in Figure 29. x-axis is the number of the sub-aperture.

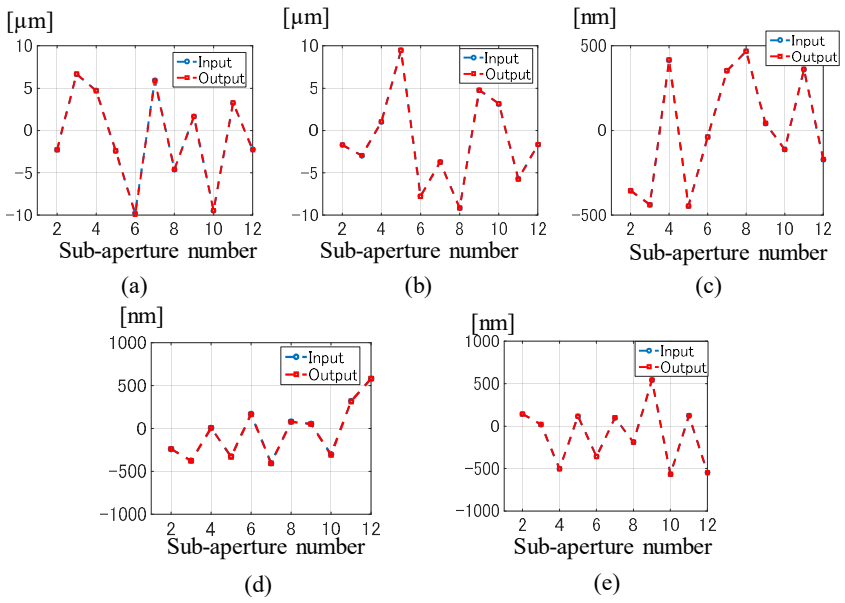


Figure 29 Estimation results of the alignment error. (a) x shift error, (b) y shift error, (c) z shift error, (d) θ_x tilt error and (e) θ_y tilt error.

As you can see, the alignment error can be estimated. The difference between the input and output alignment errors is shown in Figure 30.

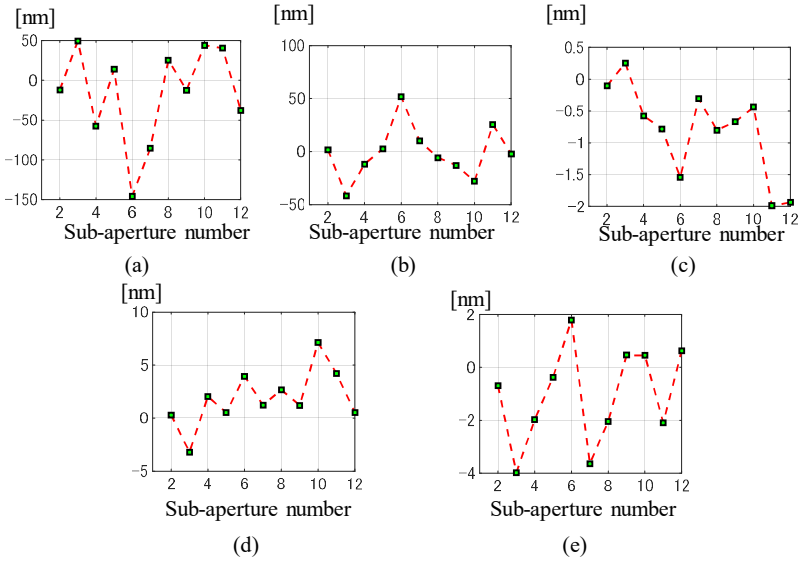


Figure 30 Estimation results of the alignment error difference. (a) x shift error, (b) y shift error, (c) z shift error, (d) θ_x tilt error and (e) θ_y tilt error.

It is found from Figure 30 that the estimation of the alignment error was not high accurate. From examining the simulation, the reasons are considered interpolation error and non-linearity. In step 6 in Figure 26, the cubic interpolation is applied to coordinate

transformation. As the shape of the submillimeter is evaluated on the nanometer order, the interpolation error cannot be ignored. As for the non-linearity, when the test sample has a large shape error or the alignment error is large, and coordinate transformation is carried out, such errors are not linear to the basic function like Zernike function. As a result, the estimation error of stitching becomes large too. Figure 31 (a), and (b) show the retrieved sample error shape with stitching and input sample error shape, respectively. The stitching error which is difference between Figure 31 (a) and (b) is shown in Figure 31 (c).

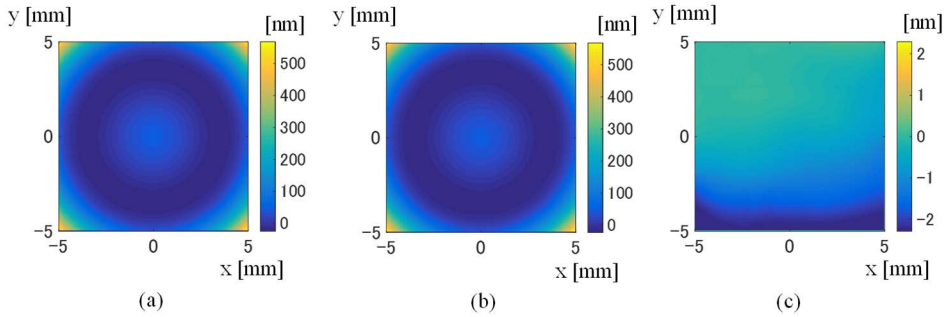


Figure 31 Retrieved shape. (a) output error shape, (b) input error shape and (c) difference between output and input error shape.

The stitching error depends on the alignment errors of the test sample, and its error shown in Figure 31 (c) is 0.76 nmRMS. This result is not bad; however, it is thought that the stitching error becomes smaller by stitching again after subtracting the alignment errors obtained by stitching because the interpolation error and the non-linear error becomes small. The estimation error of the alignment error when stitching twice is shown in Figure 32. You can see that the error is small.

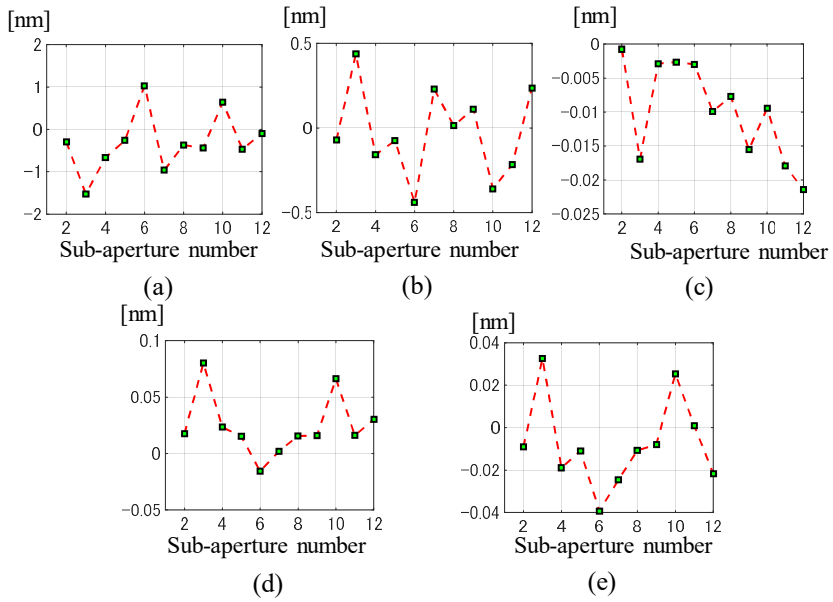


Figure 32 Estimation error of the alignment error when stitching twice. (a) x shift error, (b) y shift error, (c) z shift error, (d) θ_x tilt error and (e) θ_y tilt error.

The retrieved sample shape with stitching twice is shown in Figure 33 (a).

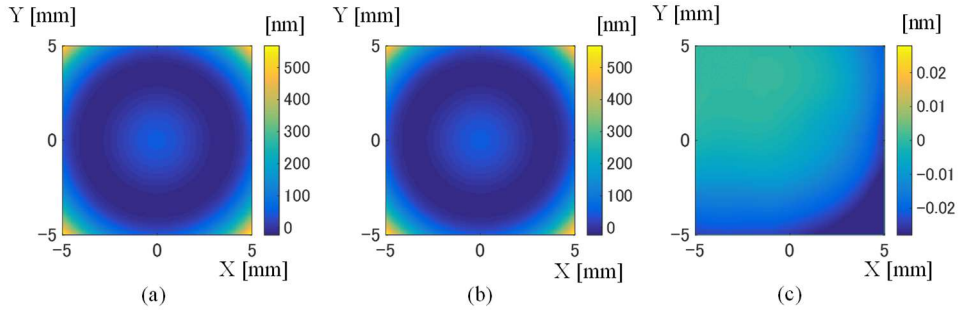


Figure 33 Retrieved shape when stitching twice. (a) output error shape, (b) input error shape and (c) difference between output and input error shape.

As you can see from Figure 33 (c), the stitching error becomes considerably small (0.01 nmRMS).

5. Error estimate

5.1. Talbot interferometer

We calculated the error sensitivity of a CCD noise, and setting error (Δz, $\Delta\theta_y$, $\Delta\theta_z$) of the grating. The error sensitivity depends on the incoming wavefront. Therefore, it is important to decide how large wavefront should be input.

To assume the actual wavefront in the experiment, we decided the input wavefront $\boldsymbol{W_{in}}$ the twice sag of the Alvarez surface on the average of the diagonal line (45 degree from x-axis) and x-axis. And we added the tilt to reduce the absolute wavefront. Specifically, the wavefront from the Alvarez surface on the x-axis, and on the diagonal line is expressed by

$$W_{x-axi}\ (x) = \frac{2A}{3}(x-5+R)^3 - a_1 x - a_0, \qquad \textbf{(88)}$$

$$W_{diag}(x) = \frac{2\sqrt{2}A}{3}\left(x-5\sqrt{2}+R\right)^3 - b_1 x - b_0, \qquad \textbf{(89)}$$

where $\boldsymbol{R}$ is a radius of a sub-aperture, $\boldsymbol{a_1}$, $\boldsymbol{b_1}$ are coefficients of tilts, $\boldsymbol{a_0}$, $\boldsymbol{b_0}$ are piston. The input wavefront $\boldsymbol{W_{in}}$ is expressed by

$$W_{in}(x) = \frac{1}{2}\left(W_{x-axi}\ (x) + W_{diag}(x)\right). \qquad \textbf{(90)}$$

Each wavefront is shown in Figure 34.

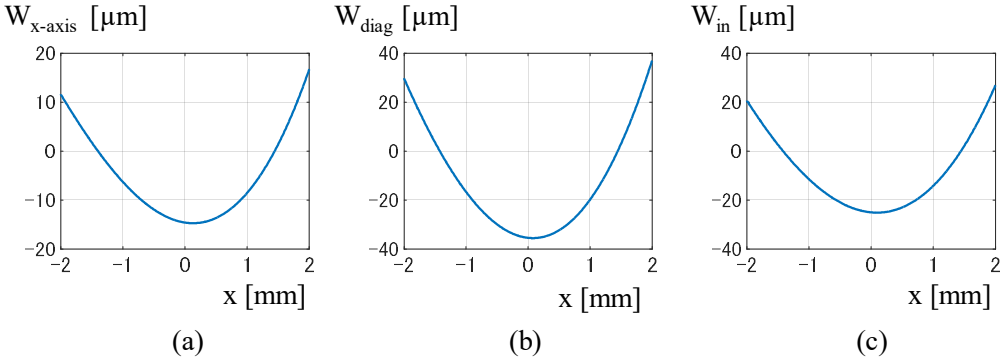


Figure 34 Wavefront from the Alvarez surface. (a) wavefront on x-axis, (b) wavefront on the diagonal line and (c) input wavefront.

The wavefront error was estimated by the same method as Chapter 3.8.1. We assumed that the grating is rectangular, and sub-aperture radius is 2 mm. First, we calculated the wavefront retrieval error by input the wavefront shown in Figure 34 (c) without additional

error. The result is shown in Figure 35.

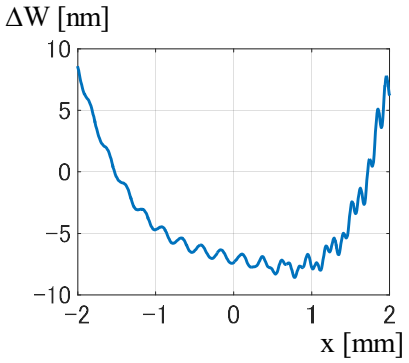


Figure 35 Wavefront retrieval error

Wavefront retrieval error is 4.34 nmPMS, and the shape error is 2.17 nmRMS.

5.1.1. CCD noise

To calculate the wavefront error sensitivity due to a CCD noise, we added the random error whose amplitude is 1, 3, and 5 % of the intensity to the intensity calculated by the angular spectrum propagation described in Chapter 3.2.2. Figure 36 shows wavefront error due to a CCD noise.

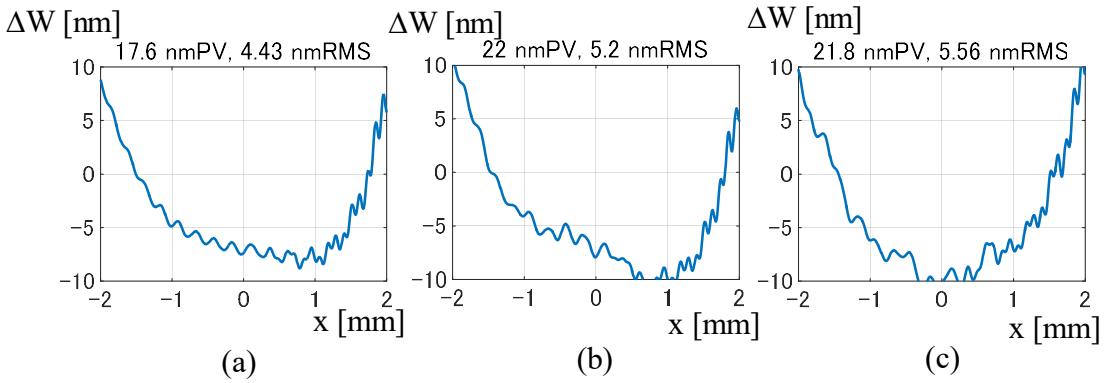


Figure 36 Wavefront error due to CCD noise. (a) 1%, (b) 3 % and (c) 5 % CCD noise.

RMS wavefront error $\boldsymbol{\sigma_{noise}}$ due to a CCD noise is calculated by

$$\sigma_{noise}(i) = \sqrt{\sigma(i)^2 - \sigma(0)^2}, \quad \textbf{(91)}$$

where $\boldsymbol{i}$ indicates the percentage of the CCD noise, $\boldsymbol{\sigma}$ is RMS wavefront error.

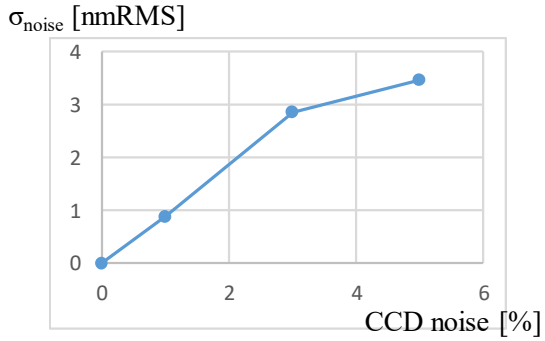


Figure 37 RMS wavefront error due to a CCD noise.

The wavefront error sensitivity due to a CCD noise was estimated to be 0.77 nmRMS / % from Figure 37, and the test shape error sensitivity was estimated to be 0.385 nmRMS / %. It is small because a CCD noise is high frequency error, which is filtered in the process of the wavefront retrieval (FT method).

5.1.2. Z position error of the grating

We estimated the wavefront error when the grating is set with shifting Δz in the z-direction from the half Talbot distance. Figure 38 shows the wavefront error when the grating is shifted -3 µm - 3 µm from the half Talbot distance.

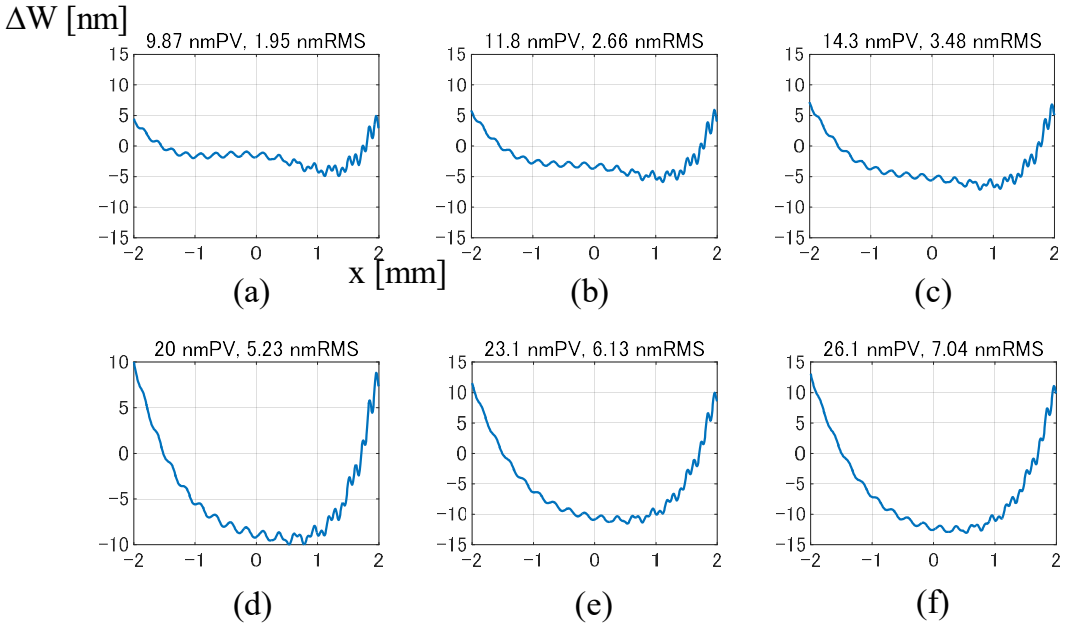


Figure 38 Wavefront error due to z error of the grating. (a) -3 µm, (b) -2 µm, (c) -1 µm, (d) 1 µm, (e) 2 µm and (f) 3 µm z error.

As the z error gets smaller, the wavefront error gets smaller. It is because there is a spherical wavefront error even if there is no z error. Therefore, when there is a z error of the grating in the minus direction, the wavefront error is canceled and it is reduced by chance. Figure 39 shows RMS wavefront error calculated with Equation (91). Here since the wavefront error with minus z error happened to be smaller than that with no z error, it was not shown.

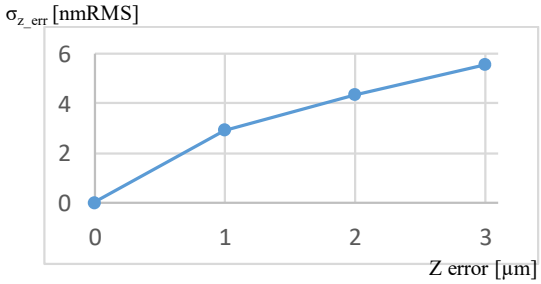


Figure 39 RMS wavefront error due to z error of the grating.

From Figure 39, the wavefront error sensitivity due to z error of the grating was estimated to be 2 nmRMS / μm, which means the test shape error sensitivity is 1 nmRMS / μm. Note that the grating position does not need to be exactly at the Talbot distance, but we need know the distance between the grating and the CCD. Assuming that the grating location from the half Talbot distance is Δz, the amount of shear a is calculated by

$$a = p\left(1 + \frac{\Delta z}{L_T}\right), \tag{92}$$

where p is a grating pitch, L_T is the half Talbot distance. The wavefront can be retrieved by the methods described in Chapter 3.3 substituting in Equation (92). Figure 40, Figure 41 and Figure 42 show the simulation result. Assuming the sinusoidal grating shown in Figure 15 (a), Figure 40 shows the calculated irradiance distributions at the position deviations from the half Talbot distance using the Angular spectrum propagation described in Chapter 3.2.2. Figure 41 shows the retrieved wavefronts from the irradiance distributions shown in Figure 40, and Figure 42 shows the difference between the reference wavefront and the retrieved wavefront.

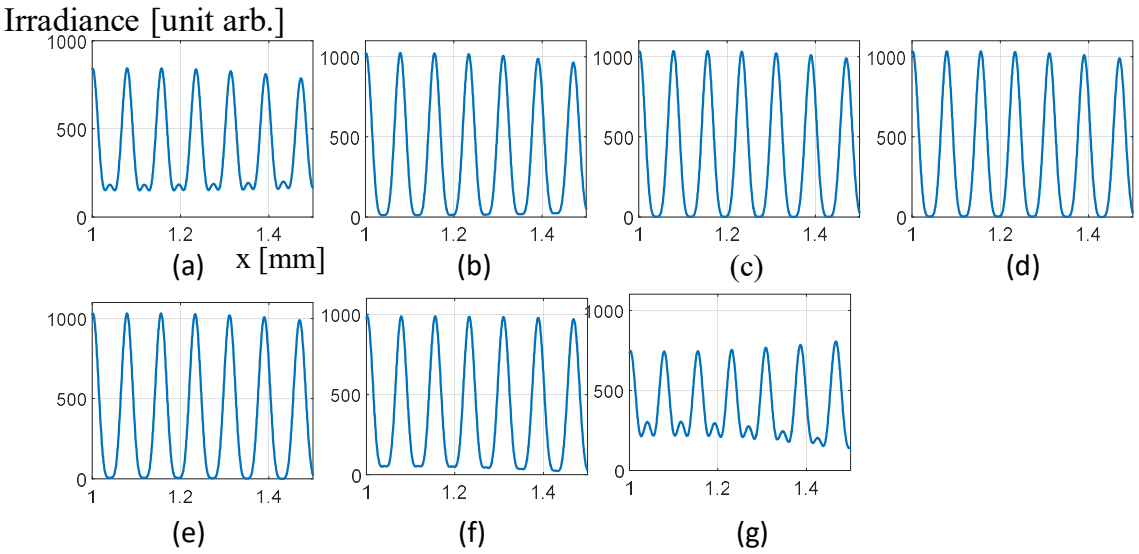


Figure 40 Irradiance distributions at different positions. The position deviation from the half Talbot distance Δz is (a) -3mm, (b) -1mm, (c) -0.1mm, (d) 0 mm, (e) 0.1mm, (f) 1mm and (g) 3mm.

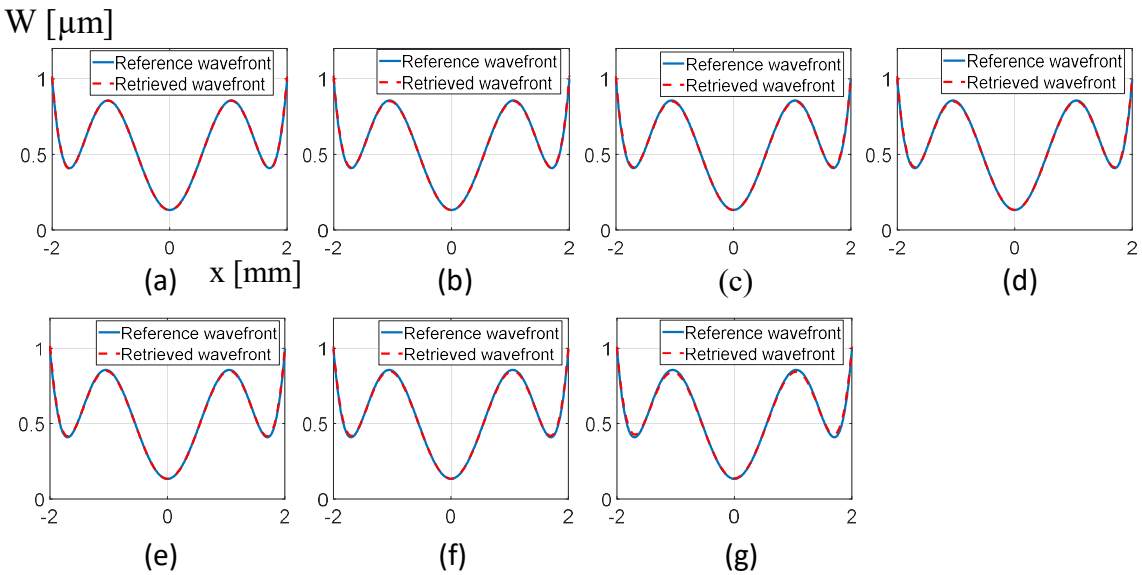


Figure 41 Retrieved wavefronts from the irradiance distributions at the different positions, Δz (a) -3mm, (b) -1mm, (c) -0.1mm, (d) 0 mm, (e) 0.1mm, (f) 1mm and (g) 3mm.

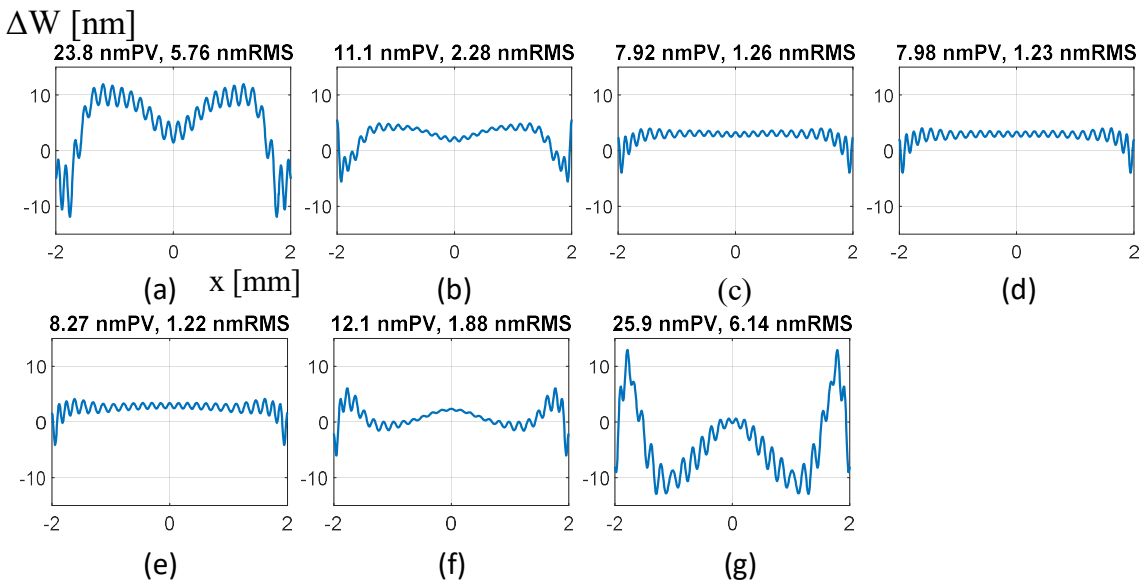


Figure 42 Differences between the reference wavefront and the retrieved wavefront at Δz (a) -3mm, (b) -1mm, (c) -0.1mm, (d) 0 mm, (e) 0.1mm, (f) 1mm and (g) 3mm.

We can see that when the position deviation from the half Talbot distance of the grating is large, the contrast of the irradiance distribution is low, and the wavefront retrieval error is large. However, when the deviation (Δz) is about 0.1 mm, the contrast is reasonable and the wavefront retrieval error is almost the same as that of no position error. From these results, we realized that it is not necessary to set the grating accurately at the half Talbot distance, but to know the distance between the grating and the CCD. We also checked that the same results were obtained when we calculated using the rectangle grating shown in Figure 15 (b).

5.1.3. Tilt error of the grating

In the Talbot interferometer with two gratings, Patorski [60] [61] and Liu and Ohba [62] [63] have shown that fringe formation is sensitive to tilt errors between the gratings. We estimated the measurement wavefront error when the grating was set inclined to the CCD. The MATLAB code is shown in Appendix D. Since it was difficult to model the tilted grating in the angular spectrum propagation, we applied the three beams interference model. To calculate the wavefront at the sensor when the wavefront is incident

on the tilted grating, it is sufficient to calculate the optical path length geometrically. We explain the specific calculation method below using Figure 43. We assumed that the grating was tilted $\boldsymbol{\theta}$, the interval between the grating and CCD was $\boldsymbol{L}$.

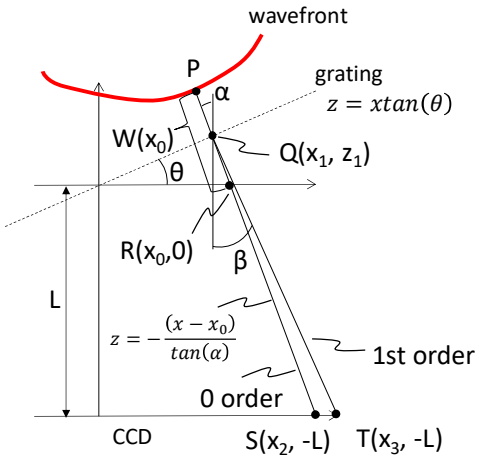


Figure 43 The schematic diagram of the optical length with tilted grating.

Wavefront $\boldsymbol{W}$ and the incident ray angle $\boldsymbol{\alpha}$ have a relation as

$$\frac{dW(x_0)}{dx_0} = sin\big(\alpha(x_0)\big), \tag{93}$$

where $\boldsymbol{x_0}$ is the x coordinate on the grating without tilt. According to Bragg's law, the diffraction angle $\boldsymbol{\beta - \theta}$ satisfies Equation (94).

$$p\{sin(\beta - \theta) - sin(\alpha - \theta)\} = m\lambda, \tag{94}$$

where $\boldsymbol{p}$ is a grating pitch, $\boldsymbol{m}$ is diffraction number. In addition, since the phase shift occurs when the ray is diffracted, the following integration value is added.

$$W_{dif} = \int sin(\alpha - \theta) - sin(\beta - \theta)dx_1/cos\,(\theta). \tag{95}$$

Assuming that the ray passes through the point $\boldsymbol{R}$, the coordinates (x_1, z_1) of the intersection $\boldsymbol{Q}$ with the grating are expressed by

$$x_1 = \frac{x_0}{1+tan(\alpha)tan(\theta)}. \tag{96}$$

$$z_1 = x_1 tan(\theta). \tag{97}$$

The x coordinate $\boldsymbol{x_2}$ of the ray on the CCD when there is no grating is calculated by

$$x_2 = x_0 + Ltan(\alpha). \tag{98}$$

Light diffracted by the grating passes through the x coordinate $\boldsymbol{x_3}$ on the CCD which is

expressed by

$$x_3 = x_1 + (L + z_1)tan(\beta). \quad \textbf{(99)}$$

The optical path lengths of PQ and QT are expressed by

$$W_{PQ} = W(x_0) - \sqrt{(x_1 - x_0)^2 + y_1^2}. \quad \textbf{(100)}$$

$$W_{QT} = \frac{(L+y_1)}{cos(\beta)}. \quad \textbf{(101)}$$

Finally, the wavefront $\boldsymbol{W_m}$ of the 0, ±1st order diffraction ray on the CCD is calculated by

$$W_m(x_3, -L) = W_{PQ} + W_{QT} + W_{dif}, \quad \textbf{(102)}$$

where $\boldsymbol{m}$ is diffraction number. We calculated the intensity of three beams interference using wavefront $\boldsymbol{W_{-1}}$, $\boldsymbol{W_0}$, $\boldsymbol{W_1}$, and retrieved the wavefront by FT method. Figure 44 shows the wavefront error when the tilts of the grating are from -3 mrad to 3 mrad,

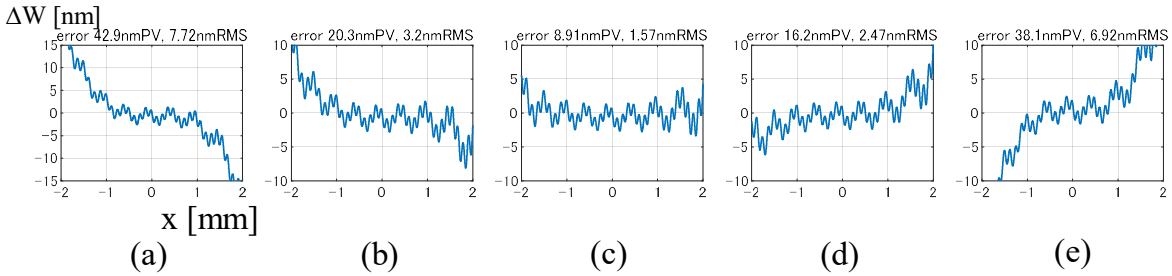


Figure 44 Wavefront error due to the grating tilt. (a) -3 mrad, (b) -1 mrad, (c) 0 mrad, (d) 1 mrad and (e) 3 mrad.

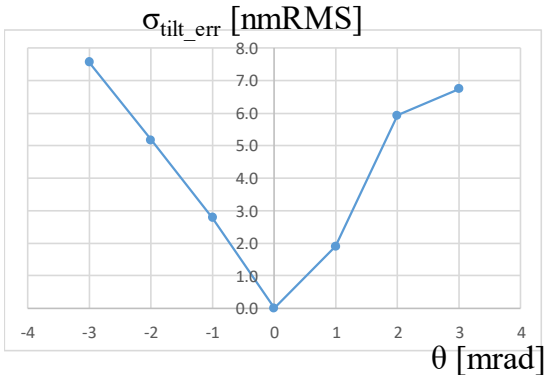


Figure 45 RMS wavefront error due to the grating tilt.

From Figure 45, the wavefront error sensitivity due to the grating tilt was estimated to be 2.4 nmRMS / mrad, which means the test shape error sensitivity is 1.2 nmRMS / mrad.

5.1.4. Rotation error of the grating

We estimated the wavefront error when the grating is rotated $\boldsymbol{\varphi}$ radian with respect to the CCD. In this case, the retrieved x-shear wavefront error $\delta W_{x,\varphi}$ is expressed by

$$\begin{aligned}\delta W_{x,\varphi}(x,y) &\approx W(x + \mathrm{a} \times \cos(\varphi), y + \mathrm{a} \times \sin(\varphi)) - W(x,y) \\ &\approx a\frac{\partial W(x,y)}{\partial x}\cos(\varphi) + a\frac{\partial W(x,y)}{\partial y}\sin(\varphi) \\ &\approx a\frac{\partial W(x,y)}{\partial x} - \frac{a}{2}\frac{\partial W(x,y)}{\partial x}\varphi^2 + a\frac{\partial W(x,y)}{\partial y}\varphi \end{aligned} \quad \textbf{(103)}$$

where $\boldsymbol{a}$ is the amount of shear, and the following approximations are applied.

$$cos(\varphi) \approx 1 - \varphi^2 \quad \textbf{(104)}$$

$$\sin(\varphi) \approx \varphi \quad \textbf{(105)}$$

The wavefront measurement error ΔW_{φ} obtained by integrating Equation (103) is expressed by

$$\Delta W_{\varphi}(x,y) = -\frac{1}{2}\varphi^2 W(x,y) + \varphi \int \frac{\partial W(x,y)}{\partial y}\, dx \quad \textbf{(106)}$$

If the rotation error $\boldsymbol{\varphi}$ is small enough, the first term is negligible. Assuming the wavefront reflected from the Alvarez surface, wavefront measurement error ΔW_{φ} is calculated by

$$\begin{aligned}\Delta W_{\varphi}(x,y) &\approx \varphi \int \frac{\partial}{\partial y}\left\{2A\left(\frac{1}{3}x^3 + xy^2\right) - c_1 x - c_2 y\right\} dx \\ &= \varphi \int 4Axy - c_2\, dx \\ &= \varphi(2Ax^2 y - c_2 x), \end{aligned} \quad \textbf{(107)}$$

where $\boldsymbol{c_1}$, $\boldsymbol{c_2}$ are the coefficients of tip-tilt. Assuming the area of x > 1 and y > 1, which is 4 × 4 mm^2 at the corner of the Alvarez surface, RMS wavefront error of Equation (107) $\Delta W_{\varphi,rms}$ becomes

$$\Delta W_{\varphi,rms} = 4.4 \times 10^4 \varphi, \quad \textbf{(108)}$$

where the unit is nmRMS, the unit of $\boldsymbol{\varphi}$ is radian. To make the RMS wavefront error less than 6 nmRMS, rotation error of the grating must be less than 0.14 mrad.

5.2. Stitching

In stitching, there are some errors due to a random error, a system error, and a spike error. Each wavefront error is estimated by simulation below.

5.2.1. Random error

The wavefront measured by the Talbot is considered to have a random error due to a CCD noise and so on. Therefore, we estimated the stitching error due to a random error by adding it to every sub-aperture data and stitching them. We calculated rms error of the test surface in stitching five times. Figure 46 shows the stitching error when 2, 6, and 10 nmPV random noise are added, respectively.

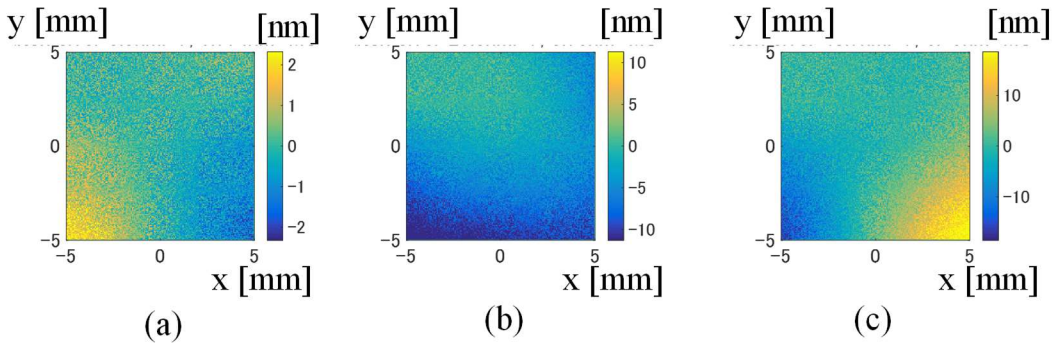


Figure 46 Stitching error due to a random error.
(a) 2 nmPV, (b) 6 nmPV and (c) 10nmPV.

Figure 47 shows stitching rms error σ_{random_err} due to a random error calculated by average of five results rms error.

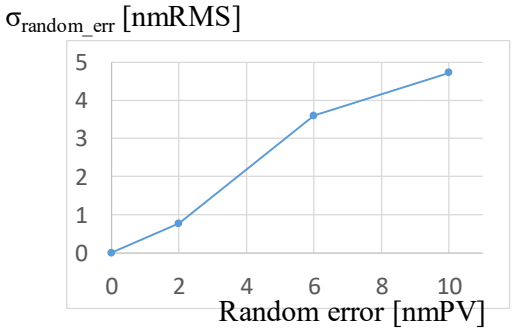


Figure 47 Stitching rms error due to a random error.

The stitching error sensitivity due to a random error was estimated to be 0.50 nmRMS / nmPV.

5.2.2. System error

Since there are errors in the Talbot interferometer and the optical system, there is also an error in the measured wavefront. Therefore, we added the error as a system error to each sub-aperture data and estimated by simulation how the system error affected the result of stitching. Figure 48 shows the system error, and Figure 49 shows stitching shape error due to the system error.

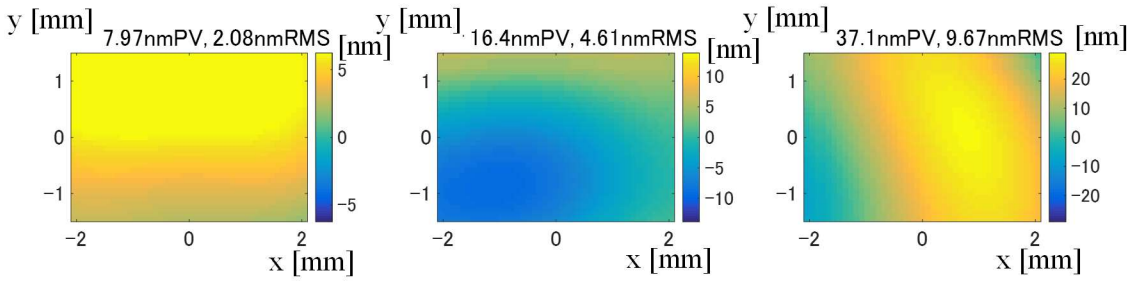


Figure 48 System error. (a) 2.1 nmRMS, (b) 4.6 nmRMS and (c) 9.7 nmRMS.

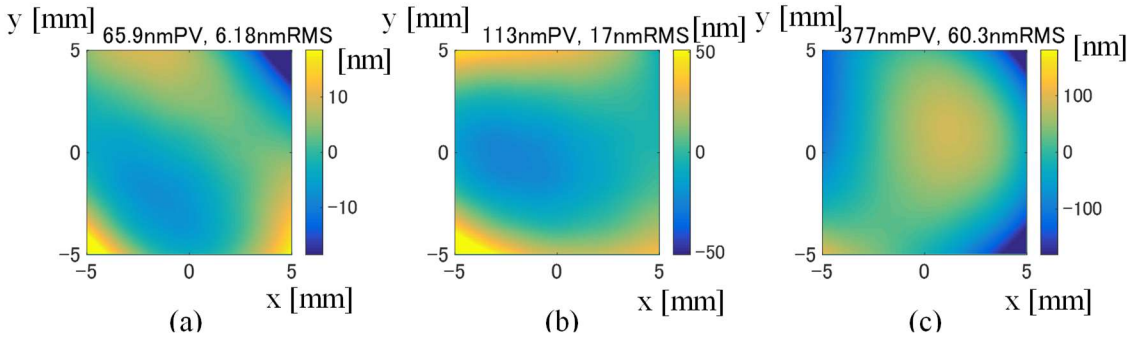


Figure 49 Stitching error due to the system error (Figure 48).

Figure 49 indicates the stitching RMS error is larger than the input system RMS error. This is thought to be because the system error is overcorrected as an alignment error, thereby increasing the stitching error. Figure 50 shows stitching RMS error σ_{sys_err} due to the system error.

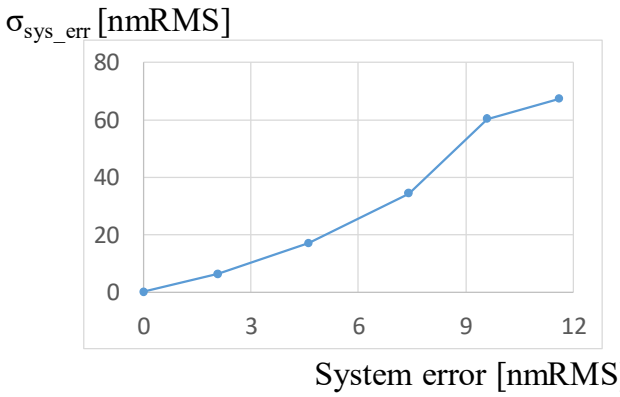


Figure 50 Stitching rms error due to a system error.

The stitching error sensitivity due to a system error was estimated to be 5.6 nmRMS / nmRMS.

5.2.3. Spike noise

Assuming that there is a spike error on the measured shape (wavefront), an error of 0.36 mm square was randomly added to each sub-aperture to estimate the stitching error. Figure 51 shows stitching shape error due to the spike error.

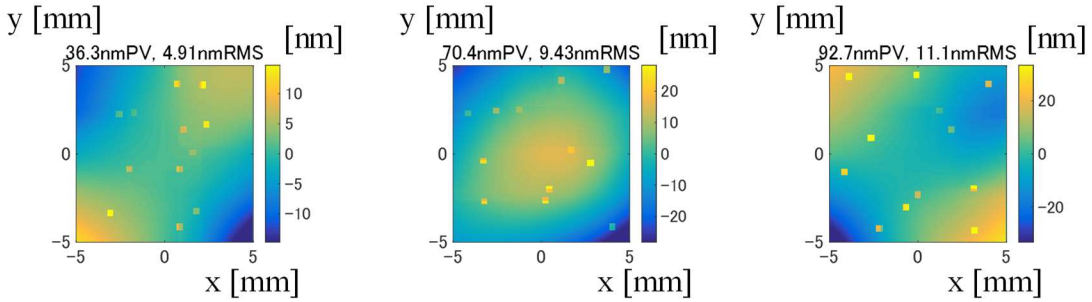


Figure 51 Stitching error due to a spike error.
(a) 10 nm, (b) 20 nm and (c) 30 nm.

Figure 52 shows stitching RMS error σ_{spike_err} due to a spike error calculated by average of five results RMS error.

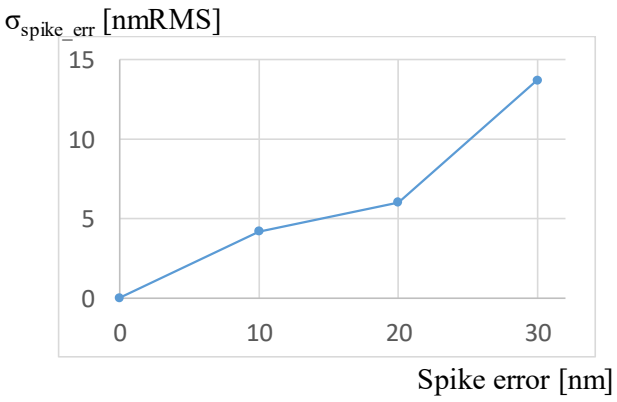


Figure 52 Stitching error due to a spike error.

The stitching error sensitivity due to a spike error was estimated to be 0.41 nmRMS / nm.

5.3. Retrace error

As shown in Figure 4, the optical system consists mainly of two achromatic lenses of the same design and a beam splitter. These optical products have wavefront aberrations due to the alignment error, homogeneity, and the surface shape error. The retrace error is an error generated due to the influence of the above error caused by the ray reflected by the test sample passing through a different optical path from the outgoing path. The retrace error is divided into a coordinate error and an angle error. The coordinate error is that the ray coordinates on the sample are different from the ray coordinates on the sensor. As the sensor and the sample are conjugate with each other, the coordinate error should be small when the test sample is not deviated largely from the flat plane. However, when the deviation of the test sample from the flat plane is large, the coordinate error is not negligible. The angle error is an error that the ray angle reflected by the test sample does not coincide with the angle incident on the sensor. Raytracing is a direct solution to correct the retrace error. The sample shape is obtained by calculating the ray angle from the wavefront measured by the sensor, raytracing from the sensor to the test sample through the optical system, calculating the ray angle (slope) and the coordinate on the sample surface, and integrating the ray slope. However, this method requires all the parameters of the optical

system including the error, which is almost impossible. Since it is difficult to know all the parameters, it is also difficult to reduce the retrace error enough. On the other hand, a calibration method of measuring the reference surface is also conceivable. By measuring the plain or spherical reference surface which has already been measured with other method and subtracting it from the measurement data of the test sample, the system error is calibrated. However, it is valid only when the test sample shape is close to the reference surface shape because the optical path is different and the wavefront error is also different if the test sample and the reference are different. Hence several calibration methods are known for these retrace errors [64]. Evans [65] proposed a correction method with multiple measurements of a tilted flat. Although this method may be simple, the coordinate error cannot be calibrated. Murphy [66] proposed a calibration method with the third-order aberration theory. It becomes difficult to characterize the optical system error when there are many lenses in the optical system. The calibration method with the perturbation theory is presented by Osten [35]. The optical system is characterized by the perturbation based on priori measurements with the reference surface at various locations. Limitation of this method is mechanical stability. Greivenkamp [67] supposed reverse optimization that the optical system was optimized by iterative raytracing calculation so that it is consistent with multiple priori measurements.

We verified how much the retrace error can be reduced by raytracing and Evans's method using the optical design software Zemax. As shown in Figure 53 and Table 4, an optical system consisting of a beam splitter and two achromatic lenses was designed. Design values of achromatic lenses, which we purchased, were downloaded from the website [68].

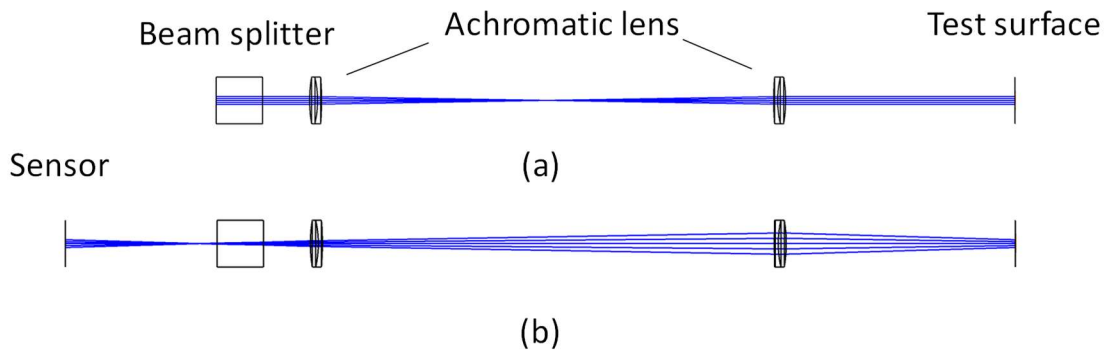


Figure 53 Optical design. (a) optical path from the light source to the test surface and (b) optical path from the test surface to the sensor.

Table 4 Lens data of the optical system.

	Surf:Type		Comment	Radius		Thickness		Material		Clear Semi-Dia		C
0	OBJECT	Standard		Infinity		Infinity				0.000		
1	STOP	Standard		Infinity		0.000				2.100		
2	(aper)	Standard		Infinity		25.040		BK7		12.700	U	
3	(aper)	Standard		Infinity		25.400				12.700	U	
4	(aper)	Standard	PAC055	75.785		4.300		BK7		12.700	U	
5	(aper)	Standard		-56.058		2.167		SF5		12.700	U	
6	(aper)	Standard		-167.954		243.966				12.700	U	
7	(aper)	Standard		167.954		2.167		SF5		12.700	U	
8	(aper)	Standard		56.058		4.300		BK7		12.700	U	
9	(aper)	Standard	PAC055	-75.785		123.804				12.700	U	
10		Standard		Infinity		0.000				12.700	U	
11	(aper and tilt:	Chebyshev Polynomial	sample	Infinity		0.000		MIRR...		10.600	U	
12		Standard		Infinity		-123.804	P			12.700	U	
13	(aper)	Standard	PAC055	-75.785	P	-4.300	P	BK7	P	12.700	P	
14	(aper)	Standard		56.058	P	-2.167	P	SF5	P	12.700	P	
15		Standard		167.954		0.000				6.136		
16	(aper)	Standard		167.954	P	-243.966	P		P	12.700	P	
17	(aper)	Standard		-167.954	P	-2.167	P	SF5	P	12.700	P	
18	(aper)	Standard		-56.058	P	-4.300	P	BK7	P	12.700	P	
19	(aper)	Standard	PAC055	75.785	P	-25.400	P		P	12.700	P	
20	(aper)	Standard		Infinity	P	-25.040	P	BK7	P	12.700	P	
21	(aper)	Standard		Infinity		-81.857				12.700	U	
22	IMAGE (aper)	Standard		Infinity	P	-			P	12.700	U	

The test shape $\boldsymbol{f}$ is the Alvarez lens, which is expressed by a white line in Figure 54. It is expressed by

$$f(x) = \frac{A\sqrt{2}}{3}(x + 4.23)^3 - xtan(\frac{2.09}{180}), \quad \textbf{(109)}$$

where $\boldsymbol{A}$ is 0.0012 mm^{-2}, the second term indicates the tilt of the test sample in stitching.

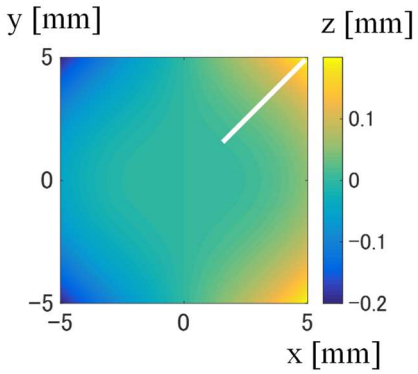


Figure 54 Test shape for simulation of calibration.

We calculated the ray coordinate $\boldsymbol{X_0}$ and angle $\boldsymbol{\theta_0}$ on the sensor by raytracing. Next, some errors as shown in Table 5 were added to this optical system, and the ray coordinate $\boldsymbol{X_a}$ and angle $\boldsymbol{\theta_a}$ on the sensor were calculated by raytracing similarly.

Table 5 Added system error.

number	Error
2	$\Delta z = 0.01$ mm
4	$\Delta x = 0.02$ mm $\Delta\theta_y = 0.005$ deg
5	$\Delta z = -0.02$ mm
6	$\Delta z = 0.4$ mm
7	$\Delta x = -0.04$ mm $\Delta\theta_y = 0.01$ deg
8	$\Delta z = 0.7$ mm
14	$\Delta z = -0.7$ mm
15	$\Delta x = -0.04$ mm $\Delta\theta_y = 0.01$ deg
16	$\Delta z = -0.4$ mm
17	$\Delta z = 0.02$ mm
18	$\Delta x = 0.02$ mm $\Delta\theta_y = 0.005$ deg
20	$\Delta z = -0.01$ mm

A plane was tilted from -3.8 degrees to 3.8 degrees in increments of 0.1 degree to implement raytracing, and a table for correcting the angle error was prepared by Equation (110) .

$$Q(X, \theta_{flat}) = \theta_{flat}(X)/2 - \varphi, \qquad \textbf{(110)}$$

where $\boldsymbol{X}$ is the ray coordinate on the detector, $\boldsymbol{\varphi}$ indicates the tilt angle of the flat plane, and

$\boldsymbol{\theta_{flat}}$ is the ray angle on the detector. The angle correction table is shown in Figure 55.

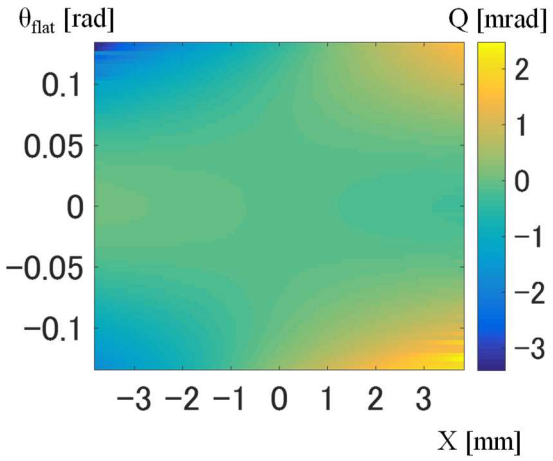


Figure 55 Angle correction table.

The coordinate error $\Delta \boldsymbol{X}$ and angle error $\Delta \boldsymbol{\theta}$ are expressed by

$$\Delta X = X_a - X_0, \quad \textbf{(111)}$$

$$\Delta\theta(X_a) = \theta(X_a) - \theta_0(X_0), \quad \textbf{(112)}$$

where $\boldsymbol{X_a}$ and $\boldsymbol{X_0}$ are ray coordinates on the detector when the optical system has no error and the error of Table 5, $\boldsymbol{\theta}$ and $\boldsymbol{\theta_0}$ are ray angles on the detector when the optical system has no error and the error of Table 5. The coordinate error $\Delta \boldsymbol{X}$ and angle error $\Delta \boldsymbol{\theta}$ are shown in Figure 56.

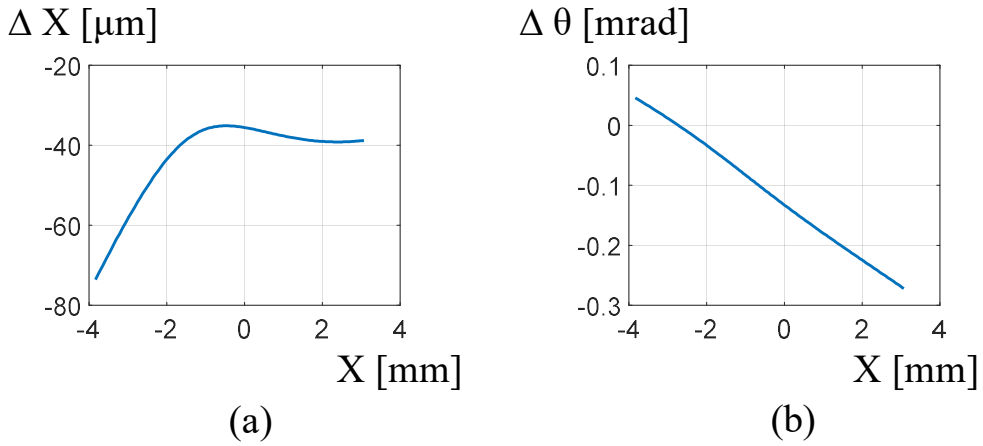


Figure 56 Retrace error. (a) coordinate error and (b) angle error.

As shown in Figure 56, the coordinate error is larger than 40 μm. We calculated the shape error Δz (difference from the nominal shape) from the ray coordinate $\boldsymbol{X_a}$ and ray angle $\boldsymbol{\theta}$ by

Equation (113) after calculating the ray angle at coordinate $\boldsymbol{X_0}$ using interpolation.

$$\Delta z(x) = \int sin[\{\theta(X_0; x) - \theta_0(X_0; x)\}/2]dx, \quad \textbf{(113)}$$

The shape error Δz is shown in Figure 57.

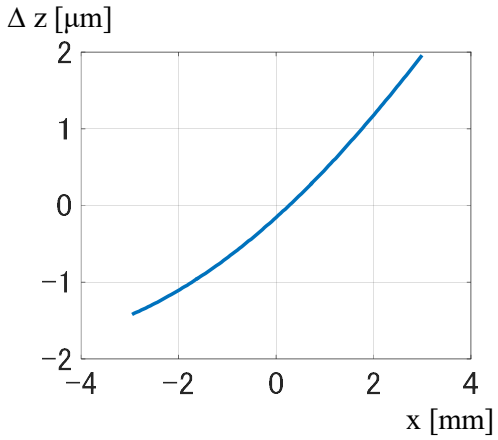


Figure 57 Shape error.

The shape error due to the retrace error is about 3.3 μmPV. Next, the angle error was corrected by Evans's method. The Figure 58 shows the shape error when the ray angle is calibrated by Equation (114) using the correction table shown in Figure 55. It is to be noted that the shape error can be calculated by Equation (115) after calculating the ray angle at coordinate $\boldsymbol{X_0}$ using interpolation in the same way as before.

$$\theta'(X_0) = \theta(X_0) - Q(X_0, \theta), \quad \textbf{(114)}$$

$$\Delta z(x) = \int sin[\{\theta'(X_0; x) - \theta(X_0; x)\}/2]dx, \quad \textbf{(115)}$$

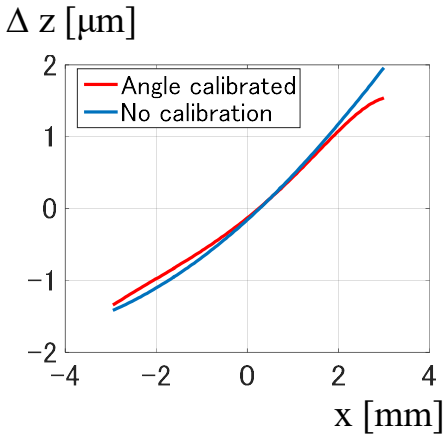


Figure 58 Shape error when Evans's method is applied.

The shape error is slightly improved. This result indicates it is not enough to calibrate only the angle error. Figure 59 shows the shape error when the coordinate error is assumed to be corrected by some method. It is calculated by

$$\Delta z(x) = \int sin[\{\theta"(x) - \theta(X_0; x)\}/2]dx, \qquad \textbf{(116)}$$

where ***θ"*** is a ray angle on the sensor calculated by interpolation at the ray coordinate ***x*** on the test sample.

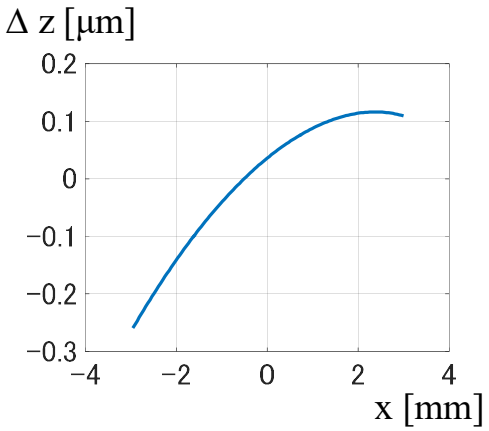


Figure 59 Shape error when the coordinate error is corrected.

The shape error was improved but not enough. From above calculation results, it was found that in this optical system, the coordinate error affects the measurement shape error more than the angle error, and a method for calibrating the coordinate error is necessary. To develop the method is a future task.

5.4. Uncertainty

Using the results obtained in Chapters 5.1 and 5.2, the measurement error estimate without the retrace error is shown in Table 6.

Table 6 Error estimation of the freeform measurement.

	Error factor	Sensitivity	Error	Measurement error
Talbot interferometer	Wavefront retrieval error			2.2 nmRMS
	CCD noise	0.39 nmRMS / %	3%	1.2 nmRMS
	Z error of the grating	1 nmRMS / μm	4 μm	4 nmRMS
	Tilt error of the grating	1.2 nmRMS / mrad	2 mrad	2.4 nmRMS
	Rotation error of the grating	22 nmRMS / mrad	0. 25 mrad	5.5 nmRMS
Stitching	Random error	0.5 nmRMS / nmPV	10 nmPV	5 nmRMS
	System error	5.6 nmRMS / nmRMS	3 nmRMS	16.8 nmRMS
	Spike error	0.41 nmRMS / nm	10 nm	4.1 nmRMS
	Total			19.6 nmRMS

The measurement error of the freeform surface excluding the retrace error was estimated to be 19.6 nmRMS when we assumed that there were errors shown in Table 6.

6. Assembly and alignment

6.1. Talbot interferometer

We requested the company whose name is 'PhotomaskPortal' to make a cross grating shown in Figure 60. The substrate is a fused silica, and the grating is made from composition of Cr and CrO_3, whose optical density is 0.2. In addition, we requested the company whose name is 'ios Optics' to coat anti-reflection (R < 0.5% at 633 nm) membrane on the other side of Chrome to avoid the stray beam. Dr. Robert A. Hudgins diced the fused silica to 2 inches square.

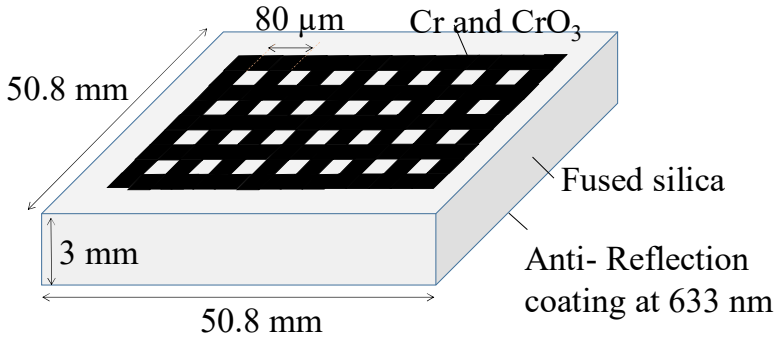
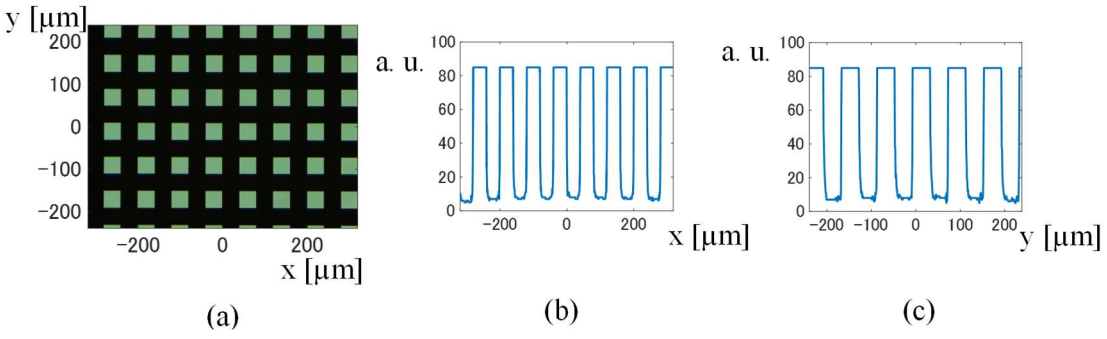


Figure 60 The schematic of a cross grating.

The image of the cross grating with microscope is shown in Figure 61. The x and y mean pitch of the grating are 80.08 µm, and 79.99 µm, respectively.



(a) (b) (c)

Figure 61 Image of the cross grating with microscope. (a) transmission image, (b) x cross section and (c) y cross section.

The picture with white light interferometer (Newview, Zygo corporation) is shown in Figure 62. As you can see, the thickness of the Chrome is 55 nm. Both Figure 61 and Figure

62 were captured by Gregory W. Caskey.

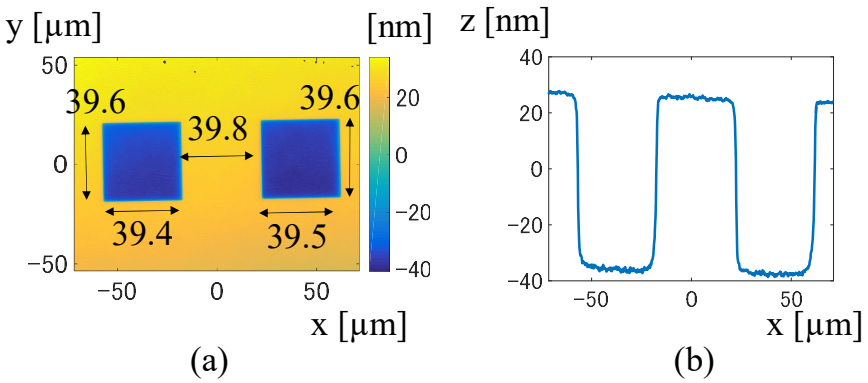


Figure 62 Image of the cross grating with white light interferometer. (a) image and (b) x cross section.

As a cover glass in front of the CCD to avoid the dust affects the measurement accuracy, we used a cover glassless 1/2-inch CCD camera whose product name is DMK 41BU02.H made by 'The Imaging Source'. A CCD chip named ICX205AL is made by 'Sony corporation'. The main specification is shown in Table 7.

Table 7 CCD specification.

Pixel number	H: 1,280 px, V: 960 px
Pixel size	H: 4.65 μm, V: 4.65 μm
Chip size	H: 7.6 mm, V: 6.2 mm
Dynamic range	8 bit

From the error estimate shown in Table 6, the grating tilt error with response to the CCD must be less than 2 mrad, and the rotation error of the grating must be less than 0.28 mrad. Furthermore, the interval between grating and CCD must be set within 10.114 mm ± 2 μm. To realize this condition, we assembled them using Fizeau interferometer (Verifire AT 1000, Zygo corporation). The assembly procedure is (1) tip-tilt adjustment of the CCD, (2) rotation adjustment of the grating, (3) tip-tilt adjustment of the grating, (4) z position adjustment of the grating. Regarding tip-tilt adjustment of the CCD, as is shown in Figure 63, we set the CCD in the collimated beam from the Fizeau interferometer and adjusted the tip-tilt of the CCD so that the fringe number of the interferogram was small.

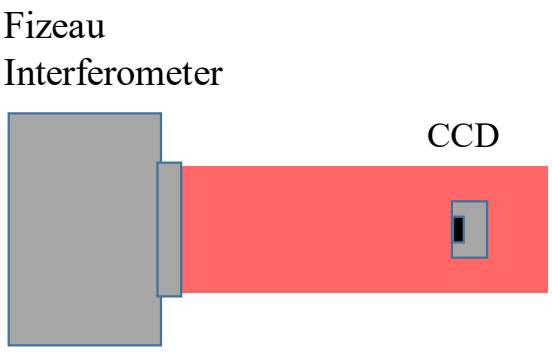


Figure 63 Setup for tip-tilt adjustment of the CCD.

The fringe and form of the CCD obtained at that time are shown in Figure 64.

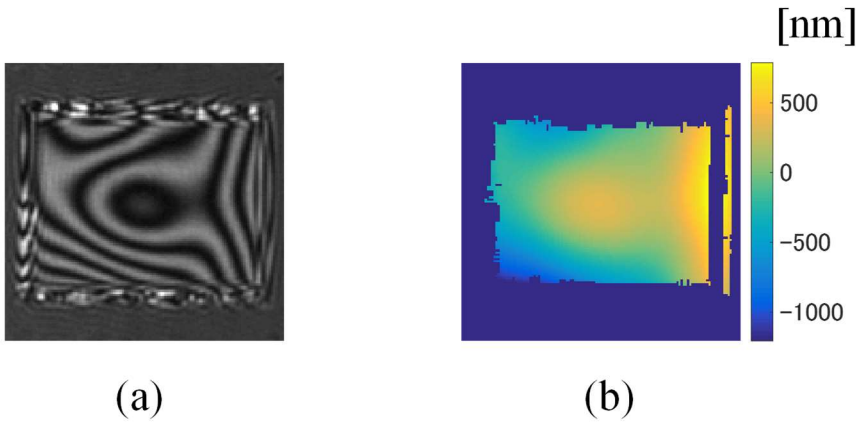


Figure 64 CCD measurement with Fizeau interferometer. (a) Fringe and (b) form.

The tilt amount of the CCD form is 800 nmPV, which is 0.14 mrad when it is converted into an angle. It is much lower than the error budget (2 mrad) shown in Table 6. You can see from Figure 64 (b) the CCD has a waviness of about 1.5 μmPV.

Regarding (2) rotation adjustment of the grating, as is shown in Figure 65. the grating mounted by 3-axis adjuster (50/50.8 mm Sq. Kinematic Mount, 3 Screws, Edmund Optics Inc) was set in front of the CCD, and the θ_z (φ) rotation angle of the grating was adjusted several times by looking at the image of the CCD so that the angular deviation with respect to the CCD became smaller.

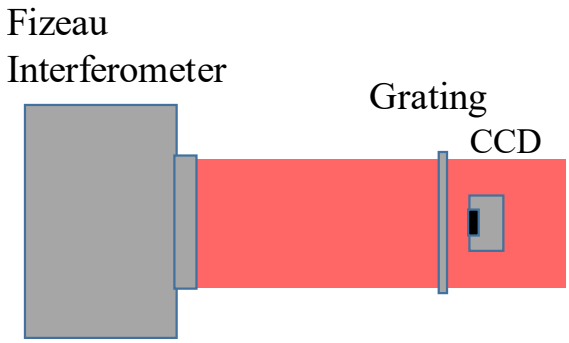


Figure 65 Setup for rotation and tip-tilt adjustment of the grating.

Figure 66 shows the magnified intensity of the CCD only in the y direction. The grating (Talbot) image is seen.

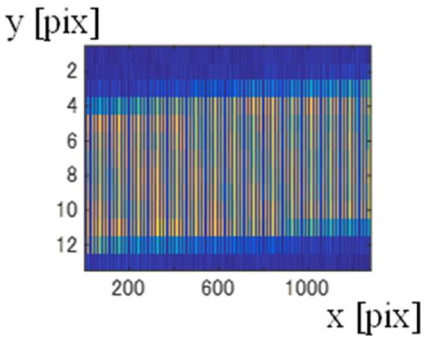


Figure 66 Magnified intensity only in the y direction.

Since the grating image is shifted by about 1 pixel from the left side to the right side, the rotation angle $\boldsymbol{\theta_z}$ is calculated by

$$\theta_z = \frac{1}{1200} = 0.83 \text{ mrad.} \quad \textbf{(117)}$$

The rotation angle $\boldsymbol{\theta_z}$ was larger than the error budget (0.14 mrad). However, we will make it smaller than 0.14 mrad by repeating more trial and error or using a rotation stage of high resolution.

Regarding (3) tip-tilt adjustment of the grating, we similarly adjusted tip–tilt of the grating by looking at the fringe of the Fizeau interferometer as is shown in Figure 65. The interference fringe and form of the grating are shown in Figure 67. The waviness of the CCD can be seen in Figure 67 (a) despite measuring a grating because the light passes

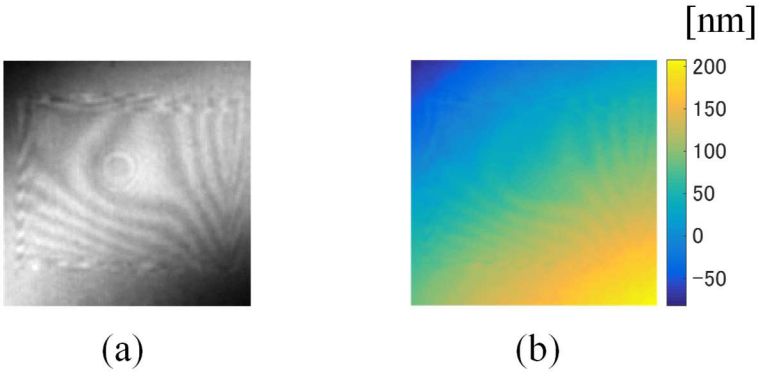


Figure 67 Grating measurement with Fizeau interferometer. (a) Fringe and (b) form.

through the grating. Since the length of the diagonal in Figure 67 is longer than 8 mm and the form is 250 nmPV, the inclination angle of the grating is calculated by 250 nm / 8 mm = 31 µrad, which is much less than error budge (2 mrad) shown in Table 6.

Regarding (4) z position adjustment of the grating, intensities when the CCD is driven in the -z direction every 1/4 inch are shown in Figure 68 together with the calculation result using the angular spectrum propagation described in Chapter 3.2.2.

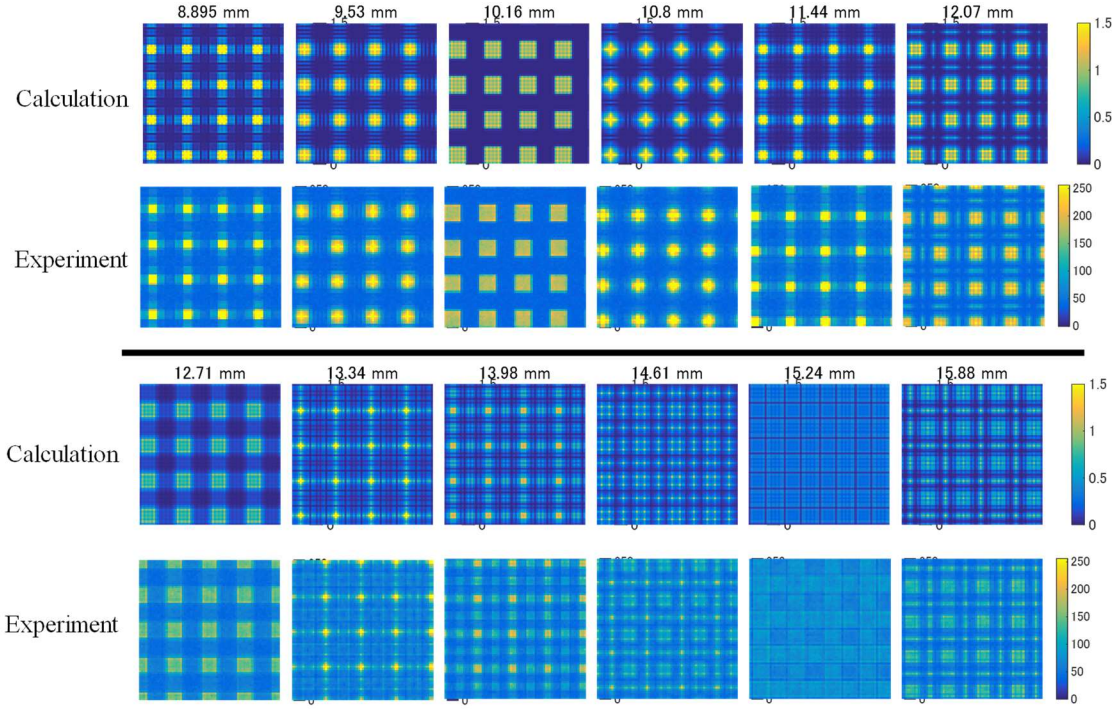


Figure 68 Intensity comparison between calculation and experiment with CCD driven in the z-direction.

We see the clear grating image when the distance between the grating and the CCD is 10.16 mm which is the half Talbot distance, and it collapses as the grating is moved away from the position. In addition, the calculation and experiment correspond to each other very well. These results show that the grating quality is reasonable and is creating the expected diffraction patterns.

We tried three methods to measure the distance between the grating and the CCD. One is the method using wavelength-shifting interferometer (Verifire^{TM} MST, Zygo corporation). In MST, by acquiring a plurality of interference data by sweeping the wavelength and Fourier transforming data on each pixel with respect to three-dimensional interference data (z is a wavelength), it is possible to simultaneously measure the intervals of a plurality of planes aligned in the optical axis direction. We thought that this method was best because it was a direct solution, but the signal from the CCD was too weak to measure the interval. We used the Fizeau interferometer in the second method. As shown in Figure 69, first, the CCD and the grating were tilted together, then the reflected light was measured with Fizeau interferometer to obtain the amount of tilt, and the Talbot image was captured with CCD. Finally, the amount of tilt of the wavefront was calculated from the Talbot image. Since the wavefront by the Talbot interferometer is proportional to the shear amount and the shear amount is proportional to the distance between the grating and CCD, the distance can be obtained by comparing the amount of tilt obtained by the Fizeau interferometer and the Talbot interferometer.

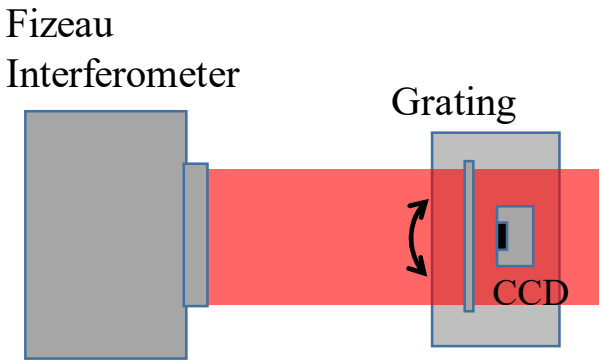
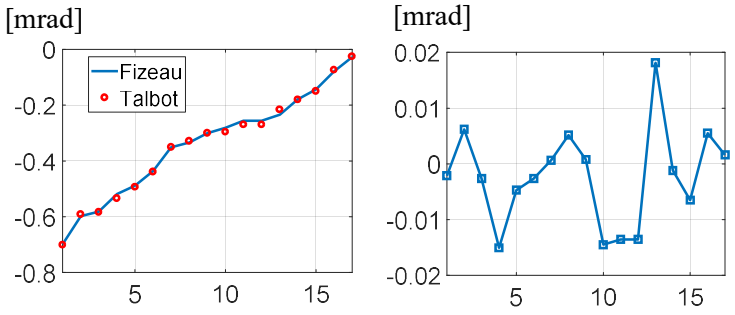


Figure 69 Setup for measuring the distance between the grating and the CCD.

Figure 70 shows the experimental results of the tilt $\boldsymbol{\theta_y}$ calculated from the measured wavefront. Tilt amount of the Talbot image is calculated assuming the distance between the grating and the CCD is half Talbot distance.



(a) (b)

Figure 70 Experimental result of the tilt wavefront. (a) tilt amount measured by Fizeau interferometer and Talbot interferomter, and (b) difference between the results of Fizeau interferometer and Talbot interferometer.

Figure 70 (a) presents that the amount of tilt of the Fizeau interferometer and the Talbot interferometer are roughly coincident, which means that the distance between the grating and the CCD is close to a half Talbot distance. Figure 70 (b) shows the difference between the tilt amount of Fizeau interferometer and the Talbot interferometer. They differ more than 1 %, which means more than 100 μm error. This error is much larger than the error budget (4 μm). Since this main cause is considered to be air fluctuation, it is expected that it can be reduced by increasing the number of measurement with Fizeau interferometer. However, as we thought the measurement error would be still large, we tried the third method.

The third method uses both Fizeau interferometer and a laser displacement sensor (Agilent 5529A Dynamic Calibrator, Agilent technology). Regarding the principle, as shown in Figure 71, it can be considered that there is a point light source at the position of the cat's eye, and the wavefront $\boldsymbol{W}$ on the CCD is expressed by Equation (118) using the distance $\boldsymbol{L_m}$ between the cat's eye and the grating, and the distance $\boldsymbol{L_T}$ +$\Delta\boldsymbol{L}$ between the grating and the CCD.

$$\mathrm{W}(x,y) = \sqrt{x^2 + y^2 + (L_m + L_T + \Delta L)^2} - (L_m + L_T + \Delta L)$$

$$\approx \frac{x^2+y^2}{2(L_m+L_T+\Delta L)}$$

$$\approx \frac{x^2+y^2}{2(L_m+L_T)}\left(1-\frac{\Delta L}{(L_m+L_T)}\right), \quad \textbf{(118)}$$

where $\boldsymbol{L_T}$ is a half Talbot distance, 10.114 mm. In the Talbot interferometer, the wavefront is retrieved by modal method described in Chapter 3.4.2. using Fringe Zernike polynomial. The fourth term $\boldsymbol{Z_4}$ of the Fringe Zernike polynomial is expressed by

$$Z_4(x,y) = 2\frac{x^2+y^2}{R^2} - 1, \quad \textbf{(119)}$$

where $\boldsymbol{R}$ is an analysis radius. When the distance between the grating and the CCD is shifted by $\Delta\boldsymbol{L}$, the shear amount becomes $\boldsymbol{p(L_t+\Delta L)/L_t}$. Therefore, assuming the obtained coefficient of the Fringe Zernike polynomial is $\boldsymbol{C_m}$, the coefficient $\boldsymbol{C_a}$ of the true value is represented by

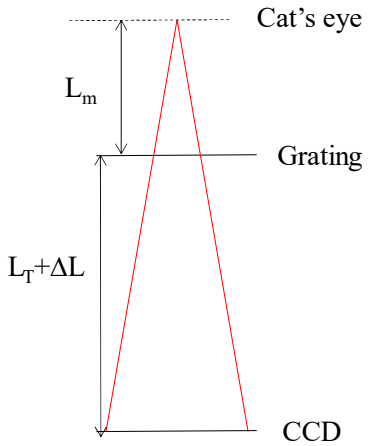


Figure 71 Model for measuring the distance between a grating and a CCD.

$$C_a = C_m\left(\frac{L_T}{L_T+\Delta L}\right)$$

$$= C_m\left(\frac{1}{1+\Delta L/L_T}\right)$$

$$\approx C_m\left(1-\frac{\Delta L}{L_T}\right). \quad \textbf{(120)}$$

If the distance $\boldsymbol{L_m}$ between the cat's eye and the grating is known, the distance $\Delta\boldsymbol{L}$ is

obtained by comparing the theory (Equation (118)) with measured coefficient (Eqs. (119) - (120)). Concretely, as we move the Talbot interferometer multiple times, which is represented by number $\boldsymbol{i}$, we can obtain $\Delta\boldsymbol{L}$ that minimizes Equation (121).

$$\sum_i \left\{ \frac{1}{2\left(L_{m,i}+L_T\right)}\left(1-\frac{\Delta L}{\left(L_{m,i}+L_T\right)}\right)-\frac{2C_{m,4,i}}{R^2}\left(1-\frac{\Delta L}{L_T}\right)\right\}^2$$
$$=\sum_i \left\{ \frac{1}{2(L_{m.i}+L_T)}-\frac{2C_{m,4,i}}{R^2}-\left(\frac{1}{2\left(L_{m,i}+L_T\right)^2}-\frac{2C_{m,4,i}}{L_T R^2}\right)\Delta L\right\}^2$$

$$\sum_i \{A_i - B_i \Delta L\}^2 \rightarrow min. \quad \textbf{(121)}$$

where

$$A_i = \frac{1}{2(L_{m.i}+L_T)}-\frac{2C_{m,4,i}}{R^2}. \quad \textbf{(122)}$$

$$B_i = \left(\frac{1}{2\left(L_{m,i}+L_T\right)^2}-\frac{2C_{m,4,i}}{L_T R^2}\right). \quad \textbf{(123)}$$

In order to minimize Equation (121), it suffices that the partial differential with $\Delta\boldsymbol{L}$ is zero, so that $\Delta\boldsymbol{L}$ can be obtained by

$$\Delta L = \frac{\sum_i A_i B_i}{\sum_i {B_i}^2}. \quad \textbf{(124)}$$

In the experiment, as is shown in Figure 72, we measured the distance $\Delta\boldsymbol{L}$ by measuring the distance $\boldsymbol{L_m}$ with the laser displacement sensor and the spherical wavefront with Talbot interferometer. Transmission sphere lens whose F number is 3.5 was used for making spherical wavefront. The grating is attached on the bread board, and CCD is attached on the z-stage on the bread board. As is shown in Figure 72 (a), the grating was set at the cat's eye position by moving the bread board so that the fringe of the Fizeau interferometer became null. Next, the corner cube reflector was attached to the Talbot interferometer (z-stage) and prepared so that the driving amount in the z-direction could be measured with a laser displacement sensor. After that, we moved the Talbot interferometer (bread board) in the z-direction multiple times and measured the driving amount with the laser displacement sensor, and the spherical wavefront with the Talbot interferometer. The measurement result

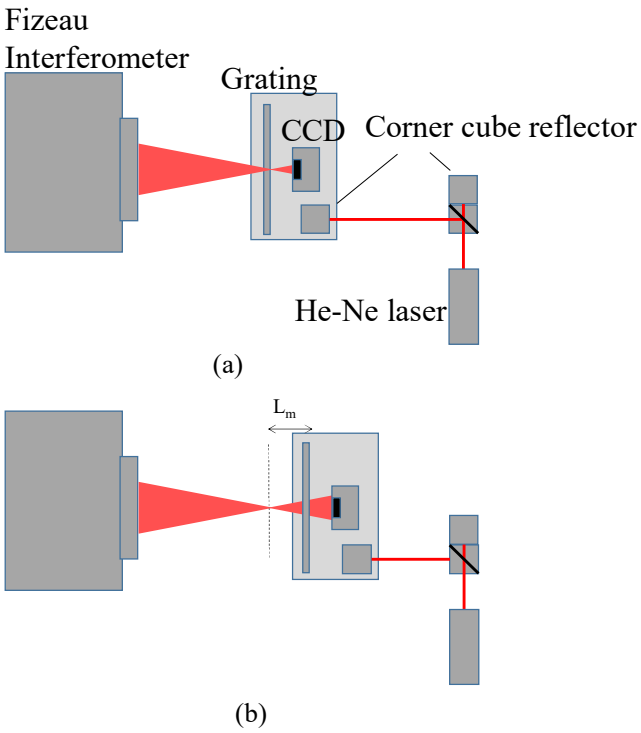


Figure 72 Setup to measure the distance between the grating and the CCD. (a) cat's eye position and (b) L_m shift in the z-direction.

is shown in Figure 73 (a). The x-axis is moving amount L_m of the Talbot interferometer, which is measured with the laser displacement sensor, and the y-axis is the coefficient of Zernike 4-th term whose analysis radius is 2 mm. The blue line is a value converted into the coefficient of Zernike 4-th term from the distance L_m.

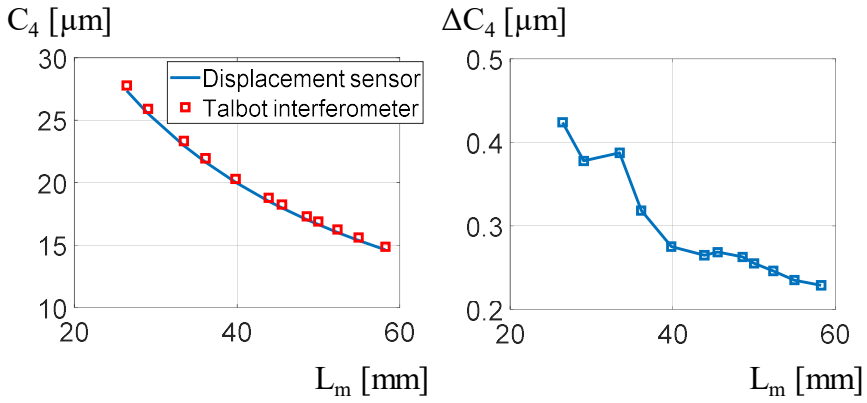


Figure 73 Experimental result of the spherical wavefront. (a) Coefficient of Zernike 4-th term and (b) Difference between the results of displacement sensor and Talbot interferometer.

Figure 73 (b) shows the difference between the results of the Talbot interferometer and the

displacement sensor. Calculating Equation (124), we obtained the result that ΔL was +190.9 μm. Therefore, we moved the CCD by 190 μm and brought it closer to the grating. We repeated the measurement and move several times, we finally obtained the results shown in Figure 74.

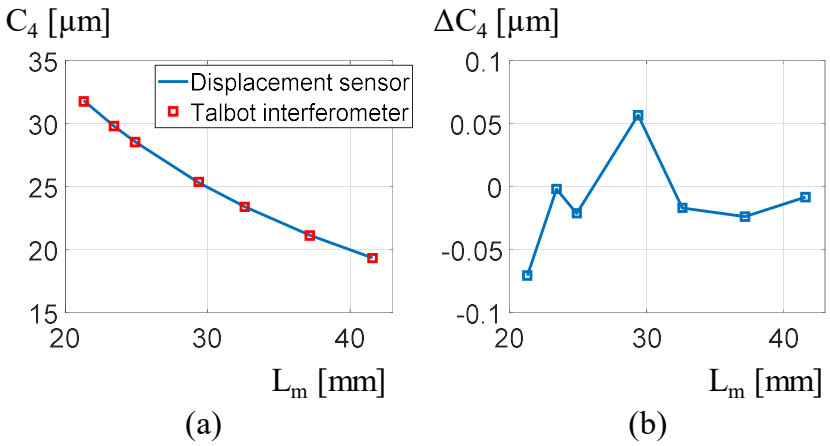


Figure 74 Experimental result of the spherical wavefront. (a) Coefficient of Zernike 4-th term and (b) Difference between the results of displacement sensor and Talbot interferometer.

Calculating Equation (124) from the results of Figure 74 (a), we obtained the result that ΔL was – 7.0 μm. This value is still larger than the error budget, but it is not necessary to set the distance between the grating and CCD to the exact half Talbot distance. If the distance ΔL is known, the wavefront can be obtained accurately by changing the amount of shear. Figure 75 shows the variation of measured distance $\delta\Delta L$ when ΔL is taken into consideration.

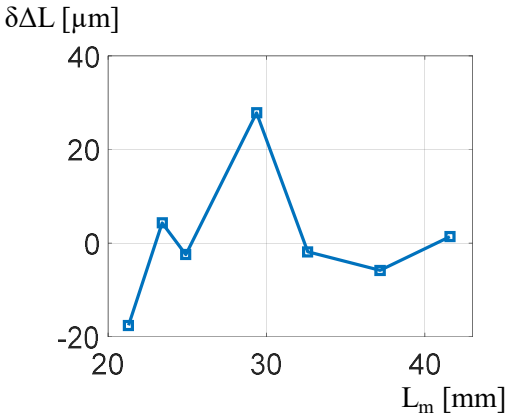


Figure 75 Variation of measured distance.

The variation of measured distance is 13.8 μmRMS, which is larger than the error budget (4 μm). This main reason is thought to be the Abbe error because we moved the Talbot interferometer manually without z-stage due to poor stroke, and the corner cube reflector could not be set on the line of the Talbot interferometer. The Abbe error can be estimated from the geometry. The corner cube reflector and the laser displacement sensor are 1 m apart, and the reflected beam can be detected even if the beam deviates by about 2.5 mm. Therefore, the maximum inclination error over the z motion range is 1.25 mrad. Since the CCD and the corner cube reflector are 20 mm apart, the Abbe error is the product of this distance and the angle, resulting in a measurement error of 25 μm at most. It will be more accurate if the Abbe error is reduced by using a long stroke stage and setting the corner cube reflector on the line of the Talbot interferometer.

The picture of the Talbot interferometer is shown in Figure 76.

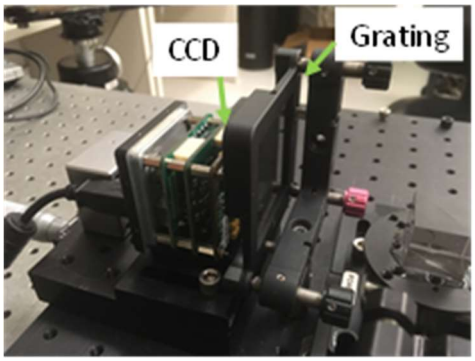


Figure 76 Talbot interferometer.

6.2. Optical system

The optical system shown in Figure 4 is mainly composed of a cube beamsplitter and two same achromatic lenses. There is an anti-reflective coating on each surface to avoid the stray light. In this Chapter, we introduce how to assemble the optical system. The procedure is described below.

First, as is shown in Figure 77, the tip-tilt of the Talbot interferometer is adjusted so that the fringe of Fizeau interferometer is null. Then, as is shown in Figure 78, the tip-

tilt of the beam splitter is adjusted so that the fringe of Fizeau interferometer is null. Here both beams from the front surface and the back surface of the beam splitter is observed. Since the ray from the test sample does not pass the front surface of the beam splitter, we can adjust it with the beam from the back surface. We can recognize which beam comes from the back surface because its power is weaker than another in the Alignment mode of the Fizeau interferometer.

Figure 77 Tip-tilt adjustment of the Talbot interferometer.

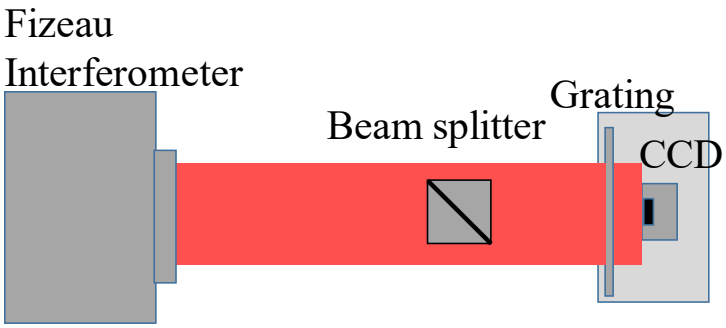


Figure 78 Tip-tilt adjustment of the beam splitter.

Next, as is shown in Figure 79, the tip-tilt of the beam splitter and the Talbot interferometer are adjusted by looking at the interference fringe between the reflected light from the grating and the reference light. And a plane mirror is set up, and tip-tilt is adjusted by looking at the interference fringes of the reflected light from the plane mirror and the reference light. Next, as shown in Figure 80, the tip-tilt of the achromatic lens is adjusted by looking at the interference fringe between the reflected light from the plain mirror and the reference light. Z position of the achromatic lens is adjusted so that the beam is focused on the CCD, and XY position is adjusted so that the beam comes to the center of the CCD.

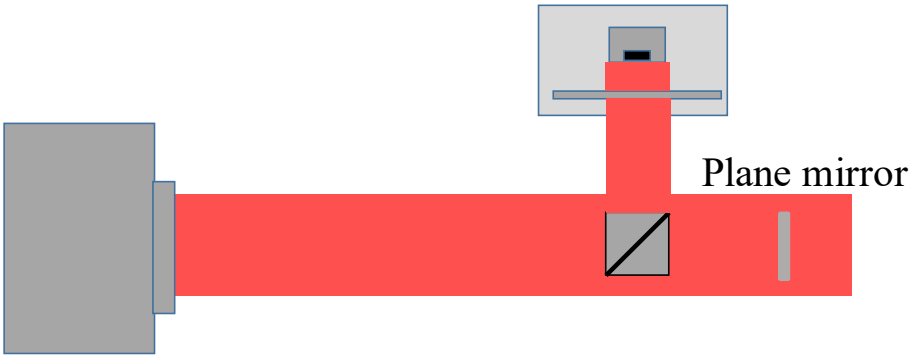


Figure 79 Tip-tilt adjustment of the plane mirror.

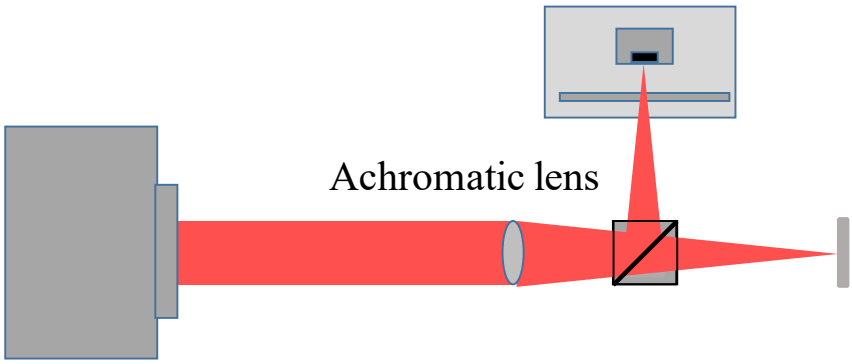


Figure 80 Alignment of the achromatic lens.

As is shown in Figure 81, a plane mirror is set up, and tip-tilt is adjusted by looking at the interference fringe between the reflected light from the plane mirror and the reference light.

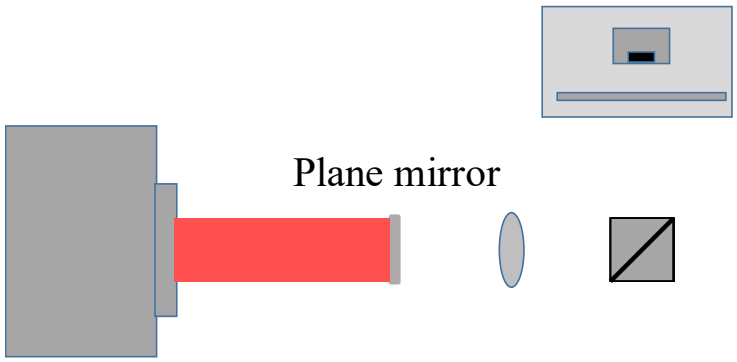


Figure 81 Tip-tilt adjustment of the plane mirror.

As is shown in Figure 82, another achromatic lens is set up and tip-tilt is adjusted by looking at the interference fringe between the reflected light from the plane mirror and the

reference light.

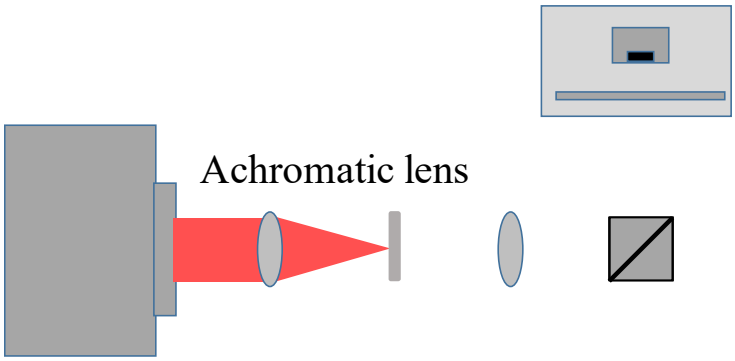


Figure 82 Tip-tilt adjustment of another achromatic lens.

As is shown in Figure 83, the plane mirror is removed, and the XYZ position of the achromatic lens is adjusted by looking at the interference fringe between the reflected light from the grating and the reference light.

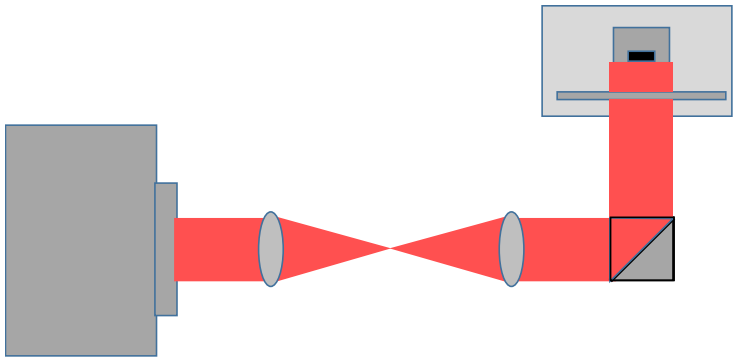


Figure 83 Alignment of another achromatic lens.

The picture of the optical system is shown in Figure 84.

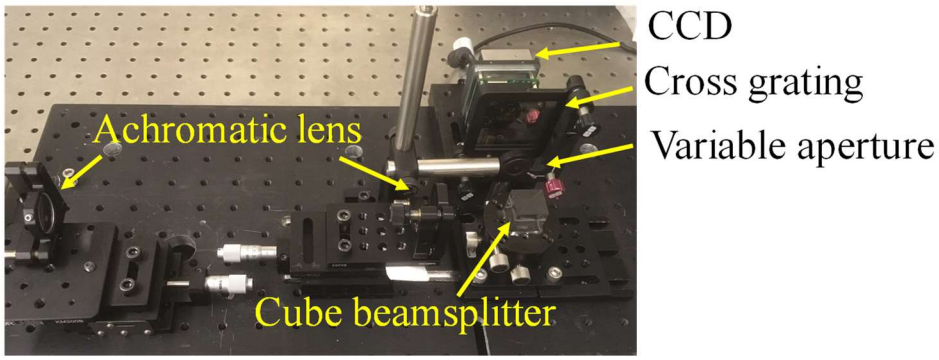


Figure 84 Optical system.

The variable aperture is used for rough alignment of the test sample. It is because as discussed in Chapter 3.7, this Talbot interferometer cannot measure the wavefront whose tilt is more than 4 mrad, and it is difficult to align the test sample roughly. A variable aperture is placed in front of the grating, and the aperture size is reduced in the case of alignment. It enables us to align the test sample by detecting the position of the transmitted light with CCD, which is similar to Alignment mode in Fizeau interferometer.

7. Conclusion and future work

In this thesis, we proposed a system for measuring freeform surface by combining the Talbot interferometer and the stitching technique and verified its feasibility by simulation. For the Talbot interferometer, theory was introduced, and simulation was performed to confirm that the wavefront can be retrieved from the Talbot image. We also clarified the specifications and accuracy with error estimation. For the stitching technique, simulation was performed, and it was confirmed that even if there was an alignment error of the test sample in sub-aperture measurement, it could be estimated and reduced. Assuming the Alvarez lens with the shape of 400 μmPV, we estimated the measurement error of both the Talbot interferometer and the stitching, which was 19.6 nmRMS except for the retrace error. The stitching error due to the system error is the dominant source of error. It may be improved by changing the lattice design. Otherwise, it is necessary to develop more robust stitching algorithm.

We investigated the retrace error by simulation using the optical design software Zemax and tried the Evans's method to reduce the retrace error, however, it turned out to be difficult to make the retrace error small. In particular, it was found that some method to reduce the coordinate error is necessary.

In the experiments, the cross grating and the windowless CCD were accurately placed using the Fizeau interferometer and a displacement sensor, and a Talbot interferometer was assembled. In addition, we suggested the way to assemble the optical system using Fizeau interferometer. It is a future work to measure the Alvarez lens surface which is one of freeform surfaces, and to verify measurement accuracy.

Since this system is high dynamic range, fast, compact, and ease to use, it will be beneficial to inspect various freeform optics.

References

1 F. Duerr, Y. Meuret, and H. Thienpont, '*Potential benefits of free-form optics in on-axis imaging applications with high aspect ratio*,' Optics Express **21** (25), 31072-31081 (2013).

2 K. Wang, F. Chen, Z. Liu, X. Luo, and S. Liu, '*Design of compact freeform lens for application specific lightemitting diode packaging*,' Optics Express **18** (2), 413-425 (2010).

3 T. Blalock, K. Medicus, and J. D. Nelson., '*Fabrication of freeform optics*,' Proc. SPIE **9575** (2015).

4 O. Cakmakcia, K. Thompsona, P. Valleeb, J. Coteb, and J. P. Rollandc, '*Design of a Freeform Single-Element Head-Worn Display*,' Proc. SPIE **7618**, 761803 (2010).

5 S. Minami, K. Minoura, and H. Yamamoto, '*Scanning Optical System Of The Canon Laser Beam Printer*,' Proc. SPIE **741**, 118-139 (1987).

6 Z. Challita, T. Agócs, E. Hugot, A. Jaskó, G. Kroes, W. Taylor, C. Miller, H. Schnetler, L. Venema, L. Mosoni, D. L. Mignant, M. Ferrari, and J. Cuby, '*Design and development of a freeform active mirror for an astronomy application*,' Proc. SPIE **53** (3), 031311 (2014).

7 J. P. Lormeau, C. Supranowitz, P. Dumas, T. Nitzsche, and R. Jenkins, '*Field proven technologies for fabrication of high-precision aspheric and freeform optical surfaces*,' Proc. SPIE **9442** (2015).

8 P. J. Smilie, B. S. Dutterer, J. L. Lineberger, M. A. Davies, and T. J. Suleski, '*Design and characterization of an infrared Alvarez lens*,' Proc. SPIE **51**, 13006 (2012).

9 James Babington, '*Alvarez Lenses: Theory and Applications*,' Proc. SPIE **9626**, 962615 (2015).

10 S. Barbero and J. Rubinstein, '*Adjustable-focus lenses based on the Alvarez principle*,' Journal of Optics **13** (12), 125705 (2011).

11 AMETEK Inc, '*Optics PGI*,' http://www.taylor-hobson.com/products/surface-profilers/optics-pgi.

12 H. Tsutsumi, K. Yoshizumi, and H. Takeuchi, '*Ultrahighly accurate 3D profilometer*,' Proc. SPIE **5638**, 387-394 (2005).

13 Mahr GmbH, '*MarSurf*,' https://www.mahr.com/en/Services/Production-metrology/Products/MarSurf---Automatic-Roughness-and-Contour-Measuring-Systems/.

14 H. Shen, R. Zhu, Z. Gao, E. Y. B. PUN, W. H. Wong, and X. Zhu, '*Design and fabrication of computer-generated holograms for testing optical freeform surfaces*,' Chinese Optics Letters **11** (3), 032201 (2013).

15 S. Scheiding, M. Beier, U. Zeitner, S. Risse, and A. Gebhardt, '*Freeform mirror fabrication and metrology using a high performance test CGH and advanced alignment features*,' Proc. SPIE **8613** (2013).

16 P. Su, Y. Wang, J. H. Burge, K. Kaznatcheev, and M. Idir, '*Non-null full field X-ray mirror metrology using SCOTS: a reflection deflectometry approach*,' Optics Express **20** (11), 12393-12406 (2012).

17 P. Su, M. A. H. Khreishi, T. Su, R. Huang, M. Z. Dominguez, A. Maldonado, G. Butel, Y. Wang, R. E. Parks, and J. H. Burge, '*Aspheric and freeform surfaces metrology with software configurable optical test system a computerized reverse Hartmann test*,' Proc. SPIE **53** (3), 031305 (2014).

18 M. C. Knauer, J. Kaminski, and G. Hausler, '*Phase measuring deflectometry: a new approach to measure specular free-form surfaces*,' Proc. SPIE **5457**, 366-376 (2004).

19 D. Malacara, '*Lateral Shear Interferometers*,' Optical Shop Testing Third Edition, 122-185 (1978).

20 G. W. R. Leibbrandt, G. Harbers, and P. J. Kunst, '*Wave-front analysis with high accuracy by use of a double-grating lateral shearing interferometer*,' Applied

Optics **35** (31), 6151-6161 (1996).

21 M. E. Riley and M. A. Gusinow, '*Laser beam divergence utilizing a lateral shearing interferometer*,' Applied Optics **16** (10), 2753-2756 (1977).

22 D. Nyyssonen and J. M. Jerke, '*Lens Testing with a Simple Wavefront Shearing Interferometer*,' Applied Optics **12** (9), 2061-2070 (1973).

23 M. E. Riley and M. A. Gusinow, '*Laser beam divergence utilizing a lateral shearing interferometer*,' Applied Optics **16** (10), 2753-2756 (1977).

24 R. S. Sirohi and M. P. Kothiyal, '*Double wedge plate shearing interferometer for collimation test*,' Applied Optics **26** (19), 4054-4056 (1987).

25 W. Yi, Z. Hongchen, S. Jutamulia, and M. Guoguang, '*Collimation test of a corrected laser diode beam using lateral shearing interferometer*,' Optics Communications **274** (2), 412-416 (2007).

26 W. Merzkirch, '*Generalized Analysis of Shearing Interferometers as Applied for Gas Dynamic Studies*,' Applied Optics **13** (2) (1974).

27 Y. Dong, Z. Hosseinimakarem, A. Davies, and C. J. Evans, '*Spurious mid-spatial frequency structure on optical surfaces reconstructed from surface slope measurements*,' Proc. SPIE **9203**, 7 (2014).

28 M. Bray, '*Stitching interferometer for large plano optics using a standard interferometer*,' **3134**, 39-50 (1997).

29 P. E. Murphy, J. Fleig, G. Forbes, D. Miladinovic, G. DeVries, and S. O'Donohue, '*Subaperture stitching interferometry for testing mild aspheres*,' Proc. SPIE **6293**, 62930J (2006).

30 M. Bray, '*Stitching Interferometry: The practical side of things*,' Proc. SPIE **7426**, 74260Q (2009).

31 K. K. Pant, D. R. Burada, M. Bichra, M. P. Singh, A. Ghosh, G. S. Khan, S. Sinzinger, and C. Shakher, '*Subaperture stitching for measurement of freeform wavefront*,' Applied Optics **54** (34), 7 (2015).

32 M. Otsubo, K. Okada, and J. Tsujiuchi, '*Measurement of large plane surface shapes by connecting small-aperture interferograms*,' Optical Engineering **33** (2), 608-613 (1994).

33 Paul Murphy, Greg Forbes, Jon Fleig, Paul Dumas, and Marc Tricard, '*Stitching Interferometry: A Flexible Solution for Surface Metrology*,' Optics and Photonics News **14** (5), 38-43 (2003).

34 C. Pruss, G. Baer, J. Schindler, W. Osten, and J. Siepmann, '*Flexibility and Rapid Measurement: Asphere and Freeform Metrology with Tilted Wave Interferometry*,' Optik & Photonik **4**, 61-64 (2013).

35 E. Garbusi and W. Osten, '*Perturbation methods in optics: application to the interferometric measurement of surfaces*,' Journal of the Optical Society of America A: Optics and Image Science, and Vision **26** (12), 2538-2549 (2009).

36 G. Baer E. Garbusi, W. Osten, '*Advanced studies on the measurement of aspheres and freeform surfaces with the Tilted-wave Interferometer*,' Proc. SPIE **8082**, 80821F (2011).

37 H. Kaijun, J. Jahns, and A. W. Lohmann, '*Talbot interferometry with a vibrating phase object*,' Optics Communications **45** (5), 295-300 (1983).

38 J. W. Goodman, '*Fresnel Diffraction by a Sinusoidal Amplitude Grating - Talbot Images*,' Introduction to Fourier Optics, second ed., 87-89 (1968).

39 J. W. Goodman, '*The Angular Spectrum of Plane Waves*,' Introduction to Fourier Optics, second ed., 55-61 (1968).

40 H. Ina M. Takeda, S. Kobayashi, '*Fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry*,' Journal of the Optical Society of America **72**, 156-160 (1982).

41 M. Takeda and K. Mutoh, '*Fourier transform profilometry for the automatic measurement of 3-D object shapes*,' Applied Optics **22**, 3977–3982 (1983).

42 S. Mirza and C. Shakher, '*Surface profiling using phase shifting Talbot*

interferometric technique,' Proc. SPIE **44**, 013601 (2005).

43 Y. Zhu, S. Odate, A. Sugaya, K. Otaki, K. Sugisaki, C. Koike, T. Koike, and K. Uchikawa, '*Method for designing phase-calculation algorithms for two-dimensional grating phase-shifting interferometry*,' Applied Optics **50** (18), 2815-2822 (2011).

44 C. Iaconis and I. A. Walmsley, '*Spectral phase interferometry for direct electric-field reconstruction of ultrashort optical pulses*,' Optics letters **23**, 792–794 (1998).

45 C. Karaalioglu and Y. Skarlatos, '*Fourier transform method for measurement of thin film thickness by speckle interferometer*,' Optical Engineering **42**, 1694–1698 (2003).

46 C. Ouchi, S. Kato, M. Hasegawa, T. Hasegawa, H. Yokota, K. Sugisaki, M. Okada, K. Murakami, J. Saito, M. Niibe, and M. Takeda, '*EUV wavefront metrology at EUVA*,' Proc. SPIE **6152** (2006).

47 D. C. Ghiglia and M. D. Pritt, '*Two-dimensional phase unwrapping: theory, algorithms, and software*,' (1998).

48 X. Su and W. Chen, '*Reliability-guided phase unwrapping algorithm: a review*,' Optics and Lasers in Engineering **42** (3), 245-261 (2004).

49 A. Asundi and Z. Wensen, '*Fast phase-unwrapping algorithm based on a gray-scale mask and flood fill*,' Applied Optics **37** (23), 5416-5416 (1998).

50 P. Liang, J. Ding, Z. Jin, C. Guo, and H. Wang, '*Two-dimensional wave-front reconstruction from lateral shearing interferograms*,' Optics Express **14** (2), 625-634 (2006).

51 D. L. Fried, '*Least-square fitting a wave-front distortion estimate to an array of phase-difference measurements*,' Journal of the Optical Society of America **67**, 370–375 (1977).

52 R. Cubalchini, '*Modal wave-front estimation from phase derivative measurements*,' Journal of the Optical Society of America **69**, 972–977 (1979).

53 G. Harbers, P. J. Kunst, and G. W. R. Leibbrandt, '*Analysis of lateral shearing interferograms by use of Zernike polynomials*,' Applied Optics **35** (31), 6162-6172 (1996).

54 W. H. Southwell, '*Wave-front estimation from wave-front slope measurements*,' Journal of the Optical Society of America **70**, 998–1006 (1980).

55 F. Dai, F. Tang, X. Wang, and O. Sasaki, '*Generalized zonal wavefront reconstruction for high spatial resolution in lateral shearing interferometry*,' Journal of the Optical Society of America A **29** (9), 2038-2047 (2012).

56 J. C. Wyant, '*Zernike polynomials and phase-shifting interferometry*,' http://wyant.optics.arizona.edu.

57 G. Gbur, '*Singular value decomposition*,' Mathematical Methods for Optical Physics and Engineering, 146-153 (2011).

58 P. J. Davis and P. Rabinowitz, '*APPROXIMATE INTEGRATION OVER A FINITE INTERVAL*,' Methods of Numerical Integration, Chapter 2 (2007).

59 K. lizuka, '*Engineering Optics Third Edition*,' 63-71 (2007).

60 K. Patorski, '*Talbot interferometry with increased shear*,' Applied Optics **24**, 4448–4453 (1985).

61 K. Patorski, '*Talbot interferometry with increased shear: further considerations*,' Applied Optics **25**, 1111–1116 (1986).

62 Q. Liu and R. Ohba, '*Effects of unparallel grating planes in Talbot interferometry*,' Applied Optics **38**, 4111–4116 (1999).

63 Q. Liu and R. Ohba, '*Effects of unparallel grating planes in Talbot interferometry II*,' Applied Optics **39**, 2084–2090 (2000).

64 T. Shi, D. Liu, Y. Zhou, T. Yan, Y. Yang, L. Zhang, J. Bai, Y. Shen, L. Miao, and W. Huang, '*Practical retrace error correction in non-null aspheric testing: A comparison*,' Optics Communications **383**, 378-385 (2017).

65 C. J. Evans and J. B. Bryan, '*Compensation for Errors Introduced by Nonzero*

Fringe Densities in Phase-Measuring Interferometers,' CIRP Annals - Manufacturing Technology **42** (1), 577-580 (1993).

66 P. E. Murphy, T. G. Brown, and D. T. Moore, '*Measurement and calibration of interferometric imaging aberrations*,' Applied Optics **39** (34), 6421-6429 (2000).

67 R. O. Gappinger and J. E. Greivenkamp, '*Iterative reverse optimization procedure for calibration of aspheric wave-front measurements on a nonnull interferometer*,' Applied Optics **43** (27), 5152-5161 (2004).

68 Newport, '*Visible Achromatic Doublet Lens, 25.4 mm, 125 mm EFL, 430-700 nm*,' https://www.newport.com/p/PAC055AR.14.

Appendix A. Calculation of five beams interference

When the electric fields of 0, ±1 order diffraction beams are respectively expressed by Eqs. (19) - (23), the intensity of their five beams interference ***I(x,y)*** is calculated by

$$I_{j_x,j_y}(x,y) = \left|E_0(x,y) + E_{1,0,j_x}(x,y) + E_{-1,0,j_x}(x,y) + E_{0,1,j_x}(x,y) + E_{0,-1,j_x}(x,y)\right|^2$$

$$= A_0^2 + 4A_1^2 + 2A_0A_1\cos\left[k\left\{W(x,y) - W(x-a,y) - \frac{\lambda}{p}\left(x - \frac{p}{N}j_x\right)\right\}\right]$$

$$+2A_0A_1\cos\left[k\left\{W(x,y) - W(x+a,y) + \frac{\lambda}{p}\left(x - \frac{p}{N}j_x\right)\right\}\right]$$

$$+2A_0A_1\cos\left[k\left\{W(x,y) - W(x,y-a) - \frac{\lambda}{p}\left(y - \frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_0A_1\cos\left[k\left\{W(x,y) - W(x,y+a) + \frac{\lambda}{p}\left(y - \frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{W(x-a,y) - W(x+a,y) + \frac{2\lambda}{p}\left(x - \frac{p}{N}j_x\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{W(x,y-a) - W(x,y-a) + \frac{2\lambda}{p}\left(y - \frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{W(x-a,y) - W(x,y-a) + \frac{\lambda}{p}\left(x - \frac{p}{N}j_x\right) - \frac{\lambda}{p}\left(y - \frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{W(x-a,y) - W(x,y+a) + \frac{\lambda}{p}\left(x - \frac{p}{N}j_x\right) + \frac{\lambda}{p}\left(y - \frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{W(x+a,y) - W(x,y-a) - \frac{\lambda}{p}\left(x - \frac{p}{N}j_x\right) - \frac{\lambda}{p}\left(y - \frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{W(x+a,y) - W(x,y+a) - \frac{\lambda}{p}\left(x - \frac{p}{N}j_x\right) + \frac{\lambda}{p}\left(y - \frac{p}{N}j_y\right)\right\}\right]$$

$$\sim A_0^2+4A_1^2+2A_0A_1\cos\left[k\left\{\frac{dW(x,y)}{dx}a-\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}-\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)\right\}\right]$$

$$+2A_0A_1\cos\left[k\left\{\frac{dW(x,y)}{dx}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}-\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)\right\}\right]$$

$$+2A_0A_1\cos\left[k\left\{\frac{dW(x,y)}{dy}a-\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}-\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_0A_1\cos\left[k\left\{\frac{dW(x,y)}{dy}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}-\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{-2a\left(\frac{dW(x,y)}{dx}+\frac{a^2}{6}\frac{d^3W(x,y)}{dx^3}\right)+\frac{2\lambda}{p}\left(x-\frac{p}{N}j_x\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{-2a\left(\frac{dW(x,y)}{dy}+\frac{a^2}{6}\frac{d^3W(x,y)}{dy^3}\right)+\frac{2\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{-\frac{dW(x,y)}{dx}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}-\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}+\frac{dW(x,y)}{dy}a-\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}+\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)-\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{-\frac{dW(x,y)}{dx}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}-\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}-\frac{dW(x,y)}{dy}a-\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}-\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}+\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)+\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2\cos\left[k\left\{\frac{dW(x,y)}{dx}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}+\frac{dW(x,y)}{dy}a-\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}-\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)-\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right]$$

$$+2A_1^2 cos\left[k\left\{\frac{dW(x,y)}{dx}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}-\frac{dW(x,y)}{dy}a-\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}-\right.\right.$$

$$\left.\left.\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}-\frac{\lambda}{p}\left(x-\frac{p}{N}j_x\right)+\frac{\lambda}{p}\left(y-\frac{p}{N}j_y\right)\right\}\right], \quad \textbf{(125)}$$

where $\boldsymbol{j_x}$, $\boldsymbol{j_y}$ indicate the number of the phase shift in the x-direction and y-direction, respectively. On the way to develop the above equations, the following approximations are used.

$$W(x+a,y)\sim W(x,y)+\frac{dW(x,y)}{dx}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}, \quad \textbf{(126)}$$

$$W(x-a,y)\sim W(x,y)-\frac{dW(x,y)}{dx}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dx^2}-\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}, \quad \textbf{(127)}$$

$$W(y+a,y)\sim W(x,y)+\frac{dW(x,y)}{dy}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}+\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}, \quad \textbf{(128)}$$

$$W(x-a,y)\sim W(x,y)-\frac{dW(x,y)}{dy}a+\frac{a^2}{2}\frac{d^2W(x,y)}{dy^2}-\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}. \quad \textbf{(129)}$$

We assume

$$B_x = A_0A_1 cos\left[\frac{ka^2}{2}\frac{d^2W(x,y)}{dx^2}\right], \quad \textbf{(130)}$$

$$B_y = A_0A_1 cos\left[\frac{ka^2}{2}\frac{d^2W(x,y)}{dy^2}\right], \quad \textbf{(131)}$$

$$\delta W_x = k\left\{\frac{dW(x,y)}{dx}a+\frac{a^3}{6}\frac{d^3W(x,y)}{dx^3}\right\}-\frac{2\pi}{p}x, \quad \textbf{(132)}$$

$$\delta W_y = k\left\{\frac{dW(x,y)}{dy}a+\frac{a^3}{6}\frac{d^3W(x,y)}{dy^3}\right\}-\frac{2\pi}{p}y. \quad \textbf{(133)}$$

The intensity of five beams interference ***I(x,y)*** is expressed by

$$I_{j_x,j_y}(x,y) = A_0^2 + 4A_1^2$$

$$+2A_0A_1 cos\left[\delta W_x - \frac{ka^2}{2}\frac{d^2W(x,y)}{dx^2}+\frac{2\pi}{N}j_x\right]$$

$$+2A_0A_1 cos\left[\delta W_x + \frac{ka^2}{2}\frac{d^2W(x,y)}{dx^2}+\frac{2\pi}{N}j_x\right]$$

$$+2A_0A_1cos\left[\delta W_y-\frac{ka^2}{2}\frac{d^2W(x,y)}{dy^2}+\frac{2\pi}{N}j_y\right]$$
$$+2A_0A_1cos\left[\delta W_y+\frac{ka^2}{2}\frac{d^2W(x,y)}{dy^2}+\frac{2\pi}{N}j_y\right]+2A_1^2cos\left[2\left\{\delta W_x+\frac{2\pi}{N}j_x\right\}\right]$$
$$+2A_1^2cos\left[2\left\{\delta W_y+\frac{2\pi}{N}j_y\right\}\right]$$
$$+2A_1^2cos\left[-\delta W_x+\delta W_y+\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2}-\frac{d^2W(x,y)}{dy^2}\right\}-\frac{2\pi}{N}j_x+\frac{2\pi}{N}j_y\right]$$
$$+2A_1^2cos\left[-\delta W_x-\delta W_y+\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2}-\frac{d^2W(x,y)}{dy^2}\right\}-\frac{2\pi}{N}j_x-\frac{2\pi}{N}j_y\right]$$
$$+2A_1^2cos\left[\delta W_x+\delta W_y+\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2}-\frac{d^2W(x,y)}{dy^2}\right\}+\frac{2\pi}{N}j_x+\frac{2\pi}{N}j_y\right]$$
$$+2A_1^2cos\left[\delta W_x-\delta W_y+\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2}-\frac{d^2W(x,y)}{dy^2}\right\}+\frac{2\pi}{N}j_x-\frac{2\pi}{N}j_y\right]$$
$$=A_0^2+4A_1^2+4B_xcos\left[\delta W_x+\frac{2\pi}{N}j_x\right]+4B_ycos\left[\delta W_y+\frac{2\pi}{N}j_y\right]$$
$$+2A_1^2cos\left[2\left\{\delta W_x+\frac{2\pi}{N}j_x\right\}\right]+2A_1^2cos\left[2\left\{\delta W_y+\frac{2\pi}{N}j_y\right\}\right]$$
$$+4A_1^2cos\left(\delta W_x+\frac{2\pi}{N}j_x\right)cos\left[+\delta W_y+\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2}-\frac{d^2W(x,y)}{dy^2}\right\}+\frac{2\pi}{N}j_y\right]$$
$$+4A_1^2cos\left(\delta W_x+\frac{2\pi}{N}j_x\right)cos\left[+\delta W_y-\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2}-\frac{d^2W(x,y)}{dy^2}\right\}+\frac{2\pi}{N}j_y\right]$$
$$=A_0^2+4A_1^2+4B_xcos\left[\delta W_x+\frac{2\pi}{N}j_x\right]+4B_ycos\left[\delta W_y+\frac{2\pi}{N}j_y\right]$$
$$+2A_1^2cos\left[2\left\{\delta W_x+\frac{2\pi}{N}j_x\right\}\right]+2A_1^2cos\left[2\left\{\delta W_y+\frac{2\pi}{N}j_y\right\}\right]$$
$$+8A_1^2cos\left(\delta W_x+\frac{2\pi}{N}j_x\right)cos\left(\delta W_y+\frac{2\pi}{N}j_y\right)$$
$$cos\left[\frac{ka^2}{2}\left\{\frac{d^2W(x,y)}{dx^2}-\frac{d^2W(x,y)}{dy^2}\right\}\right] \quad \textbf{(134)}$$

Appendix B. MATLAB Code of a Talbot interferometer

Talbot_sim.m

```
% ------------------------------------------------------------------------
%% Talbot simulation (one dimension)
% ------------------------------------------------------------------------
clear ;close all
path(path,'.¥module')
rad = pi/180;
%% condition ----------------------------------------------------------
% phase
A = 0.5;                          % coefficient of the input wavefront [um]
ZZ = 16;%-1;                          % Zernike number of the input wavefront
                                  % -1 : Alvarez surface
lamb = 0.6328;                    % lambda[um]
H = 64*1;
n = 4096;
N = n*H;                          % sampling number
Lp = 80;                          % lattice pitch of the grating [um]
p = Lp/8/H/2;                     % sampling pitch [um]
span = N*p;                       % width of calculation [um]
x = (-N/2:N/2-1)*p;               % coordinate[um]
gzi = (-N/2:N/2-1)/span;          % Fourier coordinate
Nz = 1;                           % Z sampling number
pz = 500;                         % Z pitch [um]
Ns = 4;                           % number of phase shift
Lflag = 0;                        % 0 : cosine transmittance of the grating
```

```
                                        % 1 : rectangle transmittance of the grating
noise_flag = 0;                         % 0 : No noise
                                        % 1 : add noise
Anoise = 0.0;                           % noise coefficient [%]
Zerror = 0;                             % Z setting error of the grating [um]
shft_error = zeros(1,Ns);
R0 = 2000;                              % analyzation radius [um]
R1 = 500;                               % extra radius [um]
k = 2*pi/lamb;                          % wavenumber
X = x/1e3;                              % x coordinate [mm]
RR = (R0+R1)/1e3;
Nt = 1;                                 % Talbot number
                                        % 1 : half Talbot distance
Zp = Nt*Lp^2/lamb+Zerror;               % Z position to calculate [um]
Zp0 = Nt*Lp^2/lamb;                     % Z position of the reference [um]
xshr = Nt*Lp;                           % amount of shear [um]
ins = abs(x) < R0;                      % analyzation region
%% grating ----------------------------------------------------
if Lflag == 0                           % cosine
    for j = 1:Ns
        tmp = 1/2+1/2*cos(2*pi*(x-j*Lp/Ns-shft_error(j))/Lp+eps);
        Amp(j,:) = tmp;%
    end
    clear tmp
elseif Lflag == 1                       % rectangle
    Amp = ones(Ns,N);
    nnp = Lp/p/2;
```

```
    for i = 1:Ns
        for j = 1:N/(2*nnp)-1
            Amp(i,1+((j-1)+i/Ns)*2*nnp:((j-1)+i/Ns)*2*nnp+nnp) = 0;
        end
    end
end
%% Input wavefront
phase = sperical_Zernike_function(x/R0,ZZ,A);
% calculate the wavefront outside R0
ff = sperical_Zernike_function((R0+R1)/R0,ZZ,A);
dff = (sperical_Zernike_function((R0+R1+0.1)/R0,ZZ,A)-ff)/0.1;
[ f ] = mild_curve_func( x,ff,dff,R0+R1,100 );
% figure;plot(x,f)
if ZZ ~= -1
    if ZZ ~= 0
        phase(abs(x) > R0+R1) = f(abs(x) > R0+R1) ;
    end
    if ZZ == 2
        phase(x < -(R0+R1)) = -phase(x < -(R0+R1));
    end
else
    [ W1 ] = Alvarez_phase_func( R0*1e-3,x*1e-3,-(R0+R1)*1e-3,(R0+R1)*1e-3 ,0 );
    [ W2 ] = Alvarez_phase_func( R0*1e-3,x*1e-3,-(R0+R1)*1e-3,(R0+R1)*1e-3 ,1 );
    phase = (W1+W2)/2;
    figure;plot(x*1e-3,phase-8.75);xlim([-R0*1e-3 R0*1e-3]);grid;title('phase[um]')
end
for j = 1:Ns
```

```
    Ea(j,:) = Amp(j,:).*exp(1i*k*phase);              % Ea: electric field on the grating
end
E0 = exp(1i*k*phase);                                 % E0: electric field of the reference
on the grating
clear Amp phase
%%
if Nz == 1
    z = Zp;                                           % z position [um]
else
    z = transpose(Zp+(-Nz/2:Nz/2-1)*pz);              % z position [um]
end
%% Angular spectrum propagation ---------------------------------------
for j = 1:Ns
    FT(j,:) = fftshift(fft(fftshift(Ea(j,:))))/N;
end
FT0 = fftshift(fft(fftshift(E0)))/N;
clear Ea E0 f
% figure,plot(gzi,abs(FT(j,:)),'.-')
if Nz == 1
    kernel = k*z*sqrt(1-(lamb*gzi).^2);               % exponential part
    for j = 1:Ns
        FTa(j,:) = FT(j,:).*exp(1i*kernel);           % Angular spectrum
    end
    kernel = k*Zp0*sqrt(1-(lamb*gzi).^2);
    FTa0 = FT0.*exp(1i*kernel);
else
    for j = 1:Nz
```

```
        kernel = k*z(j)*sqrt(1-(lamb*gzi).^2);          % exponential part
        FTa(j,:) = FT(1,:).*exp(1i*kernel);             % Angular spectrum
    end
    FTa0 = FT0.*exp(1i*kernel);
end
clear FT FT0 kernel
%% Selection of evanescent or propagating component
select = find((1-(lamb*gzi).^2) <= 0);                  % select whether propagating<= or
evanescent>
if Nz == 1
    for j = 1:Ns
        tmp = FTa(j,:);
        tmp(select) = 0;
        FTa(j,:) = tmp;
    end
    FTa0(select) = 0;
    % figure,imagesc(x,z(:,1),abs(FTa)),xlabel('x [¥mum]'),ylabel('z [¥mum]');axis tight
    Et = zeros(Ns,N);                                   % Et:
    for j = 1:Ns
        Et(j,:) = fftshift(ifft(fftshift(FTa(j,:))))*N; % FFT at the each z position
    end
    Et0 = fftshift(ifft(fftshift(FTa0)))*N; % FFT at the each z position
else
    Et = zeros(Nz,N);
    for j = 1:Nz
        tmp = FTa(j,:);
        tmp(select) = 0;
```

```
        Et(j,:) = fftshift(ifft(fftshift(tmp)))*N; % FFT at the each z position
    end
    clear FTa
    FTa0(select) = 0;
    Et0 = fftshift(ifft(fftshift(FTa0)))*N; % FFT at the each z position
end
phasd = atan2(imag(Et),real(Et));                    % phase
Iout = Et.*conj(Et);                                 % intensity
figure;plot(x*1e-3,1e3*Iout(1,:));xlim([-R0/1e3 R0/1e3]);grid
xlim([1.5 1.76])
%% reference wavefront
phas0 = atan2(imag(Et0),real(Et0)); % wrapped reference wavfront
[ phas0 ] = unwrap1d_func( phas0 )/k;              % unwrapping
phas0 = phas0-mean(phas0(ins));
figure;plot(x/1e3,phas0)
clear tmp tmp1
%% shear wavefront
dphas0 = zeros(1,N);
dphas0(1:N-1) = (phas0(2:N)-phas0(1:N-1))/p;
% figure;plot(x,dphas0);
%% noise ---------------------------
maxI = max(Iout(1,N*3/8:N*5/8));
Iout = Iout/maxI*1024;
if noise_flag == 1
    for i = 1:Ns
        noise = 0;
        for j = 1:10
```

```
                noise = noise+1024*Anoise.*rand(size(Iout(1,:)))/0.5773;
            end
            Iout(i,:) = Iout(i,:) + noise/10;
        end
        clear noise
end
%% wavefront retrieval ----------------------------------------------------
if Ns == 1
        fI = fftshift(fft(fftshift(Iout(1,:))));
        % figure;plot(real(fI));title('Fourier Spectrum'); xlim([N/2-1000 N/2+1000]);
        figure;plot(log10(abs(fI)));title('Fourier Spectrum'); xlim([N/2-1000 N/2+1000]);
        % cut the FT data ------------------------------------------------------
        ff = zeros(1,N);
        nr = 128;                                   % radius of cutting [pix]
        nc = N/2+1+round(n*H/(Lp/p));      % location of cutting [pix]
        ff(N/2+1-nr:N/2+1+nr) = fI(nc-nr:nc+nr);
        figure;plot(log10(abs(ff)));title('cut Fourier Spectrum');xlim([N/2-1000 N/2+1000]);
        tmp = fftshift(ifft(fftshift(ff)));
        dpha = -atan2(imag(tmp),real(tmp));
        dpha((abs(x)) > R0+R1) = 0;
        figure;plot(X,dpha);xlim([-RR RR]);title('wrapped phase[rad]')
        if dpha(N/2+1) < -pi+0.2
            dpha(N/2+1) = dpha(N/2+1)+2*pi;
        end
        [ dpha ] = unwrap1d_func( dpha );% unwrapping
        dpha((abs(x)) > R0+R1) = 0;
        if Lflag == 0
```

```
        pstn = -0.003955;
    else
        pstn = -0.0019735-1.26e-7;
    end
    dpha = dpha/k/Lp+pstn;

    [ ph ] = intg1D_func( x,dpha );                % integration
    tmp = ph-phas0;
    pst = mean(tmp(abs(x) < R0));
    ph = ph-pst;
else
    if Ns == 3
        rdph = atan2(sqrt(3)*(Iout(2,:)-Iout(3,:)),-2*Iout(1,:)+Iout(2,:)+Iout(3,:));
    elseif Ns == 4
        rdph = -atan2(Iout(4,:)-Iout(2,:),Iout(3,:)-Iout(1,:));
    elseif Ns == 6
        rdph = -atan2(sqrt(3)*(Iout(2,:)+Iout(3,:)-Iout(5,:)-Iout(6,:)),...
            2*Iout(1,:)+Iout(2,:)-Iout(3,:)-2*Iout(4,:)-Iout(5,:)+Iout(6,:));
    else
        phshft = 2*pi/Ns;
        tmp1 = 0;
        tmp2 = 0;
        for j = 1:Ns
            tmp1 = tmp1+Iout(j,:)*sin(phshft*j);
            tmp2 = tmp2+Iout(j,:)*cos(phshft*j);
        end
        rdph = atan2(-tmp1,tmp2);
```

```
        clear tmp1 tmp2
    end
    [ rdph ] = unwrap1d_func( rdph );
    if Lflag == 0
        pistn = -0.0019775;
    else
        pistn = 3.85*1e-6;
    end
    rdph = 1/xshr*(lamb/Lp*x+1/k*rdph)+pistn;
    rdph(abs(x) > R0+R1) = 0;
    rdph = sin(atan(rdph));
    %% integral
    [ ph ] = intg1D_func( x,rdph );
    ph = ph - mean(ph(ins));
end
%% diplay output
figure;plot(X,phas0,X,ph,'--r');
title('wavefront[um]');
xlim([-R0/1e3 R0/1e3]);grid;hold off
dif = 1e3*(ph-phas0);                                   % difference [nm]
dif = dif-mean(dif(ins));
[ PV,rms,maxh,minh ] = PV_rms_func( dif(ins));
figure;plot(X,dif);
xlim([-R0/1e3 R0/1e3]);grid;hold off
title([num2str(PV,3) ' nmPV, ' num2str(rms,3) ' nmRMS']);ylim([-10 10])

tmp = compensate_wavefront_func(ph,Lp,x);
```

```
figure;plot(x/1e3,tmp);xlim([-R0/1e3 R0/1e3]);
ph2 = ph-tmp;
figure;plot(x/1e3,phas0,x/1e3,ph2);
title('wavefront[um]')
xlim([-R0/1e3 R0/1e3]);grid;hold off

dif = 1e3*(ph2-phas0);
dif = dif-(max(dif(ins))+min(dif(ins)))/2;
[ PV,rms,maxh,minh ] = PV_rms_func( dif(ins));
figure;plot(x/1e3,dif);                                    % difference of output
xlim([-R0/1e3 R0/1e3]);grid;hold off;ylim([-15 015])
title([num2str(PV,3) ' nmPV, ' num2str(rms,3) ' nmRMS']);

%% end of this file
% programed by Yasunori Furukawa in 2017.06.07

function [ W ] = Alvarez_phase_func( R0,x,x0,x1,flag )
%% ----------------------------------------------------
% Output the wavefront reflected from the Alvarez surface in stitching

% W                : wavefront [um]

% R0               : analyzation radius [mm]
% x                : x coordinate [mm]
% x0               : minimum analyzation coordinate [mm]
% x1               : maximum analyzation coordinate [mm]
% flag           : 0 = wavefront on x axis
```

```
%                   1 = wavefront on diagonal direction
%% ------------------------------------------------------

lamb0 = 15;         % length of the edge [mm];
v = 1;              % magnification
shx = v*R0;         % amount of shear [mm]
A = 0.0012;         % coefficient [mm-1]
% wavefront
if flag == 0
    RR = 5;                     % radius [mm]
    B = 1e3*2/3*A;              % coefficient
    W = B*(v*x+RR-shx).^3;
elseif flag == 1
    RR = 5*sqrt(2);             % radius [mm]
    B = 1e3*2*sqrt(2)/3*A;    % coefficient
    W = B*(v*x+RR-shx).^3;
end

ins = abs(x) < R0;
% remove piston and tilt
n = 1;
p = polyfit(x(ins),W(ins),n);
for j = 1:n+1
    W = W-p(j)*x.^(n-j+1);
end
% calculate the wavefront outside the analyzation area with smooth
W0 = B*(v*x0+RR-shx).^3-p(1)*x0-p(2);
```

```
dW0 = 3*B*(v*x0+RR-shx).^2*v-p(1);
g0 = dW0*lamb0/2/pi*sin((x-x0)*2*pi/lamb0)+W0;

W1 = B*(v*x1+RR-shx).^3-p(1)*x1-p(2);
dW1 = 3*B*(v*x1+RR-shx).^2*v-p(1);
lamb1 = 1/dW1*(-lamb0*dW0+2*pi*(W0-W1));
g1 = dW1*lamb1/2/pi*sin(2*pi*(x-x1)/lamb1)+W1;
% substitute the wavefront outside the analyzation area
W(x < x0) = g0(x < x0);
W(x < x0-lamb0/4) = -dW0*lamb0/2/pi+W0;
W(x > x1) = g1(x > x1);
W(x > x1+lamb1/4) = -dW0*lamb0/2/pi+W0;
% figure;plot(x,W)
end
%% end of this function

function cW = compensate_wavefront_func(W,p,x,y)
%% -----------------------------------------
% compansate the wavefront of Talbot interferometer
% cW              : compensated wavefront [um]
% W               : wavefront [um]
% p               : pitch [um]
% x               : x coordinate [um]
%% -----------------------------------------
[n1,n2] = size(W);
if n1 == 1 || n2 == 1
    W1 = interp1(x,W,x-p,'PCHIP');
```

```
        W3 = interp1(x,W,x+p,'PCHIP');
        Wdd = W3+W1-2*W;                % second derivative wavefront
else
        W1 = interp2(x,y,W,x-p,y,'cubic');
        W3 = interp2(x,y,W,x+p,y,'cubic');
        Wdd = W3+W1-2*W;
        W1 = interp2(x,y,W,x,y-p,'cubic');
        W3 = interp2(x,y,W,x,y+p,'cubic');
        Wdd = Wdd+W3+W1-2*W;
end
cW = 1/6*Wdd;          % compansated wavefront [um]
%% end of this function
% programed by Yasunori Furukawa on Sep. 26, 2016

function [ intg_data ] = intg1D_func( x,dWx )
% ------------------------------------------------------------------------
% Integrate the shear wavefront in one-dimension

% intg_data : wavefront [um]

% x                : x coordinate [um]
% dWx              : shear wavefront

% ------------------------------------------------------------------------

[n1,n2] = size(dWx);
NaNflag = zeros(n1,n2);                    % NaN data
```

```
NaNflag(isnan(dWx)) = 1;
%%
warning off
Dx = CalcDelta(dWx, x, 2);                    % difference between two
data
%% integration
intg_data = Integral_func(Dx);
intg_data = intg_data-mean(intg_data(NaNflag==0));
intg_data(NaNflag==1) = NaN;
warning on
function [ data ] = Integral_func(Dx)
% ------------------------------------------------------------------------
% data    : integrated data
% Dx      : difference between two data
% ------------------------------------------------------------------------

[row, col] = size(Dx);
data = zeros(row, col);
cx = col/2+1;
% add Dx to right half data
for j = cx+1:col
    data(j) = data(j-1)+Dx(j-1);
end
% add Dx to left half data
for j = cx-1:-1:1
    data(j) = data(j+1)-Dx(j);
end
```

```
function [ delta ] = CalcDelta(diff, crd, dim)
% ------------------------------------------------------------------------
% calculate the integrated difference between two data
% diff    : local shear wavefront
% crd     : x coordinate [um]
% dim     : 2 when data size is [1,3]
% ------------------------------------------------------------------------

if dim == 2     % transpose
    diff = diff.';
    crd = crd.';
end
[row, col] = size(diff);
delta = zeros(row, col);
for j = 1:col
    for i = 1:row-2
        x = crd(i:i+2,j);
        y = diff(i:i+2,j);
        if sum(isnan(y)) == 0
            A = [x.^2 x ones(3,1)];
            p = A¥y;
            Q = p(1)/3*x.^3+p(2)/2*x.^2+p(3)*x;          % integrated data
            if i == 1
                delta(i,j) = Q(2)-Q(1);
            else
                if isnan(delta(i,j))
```

```
                delta(i,j) = Q(2)-Q(1);
            else
                delta(i,j) = (delta(i,j)+(Q(2)-Q(1)))/2;
            end
        end
        delta(i+1,j) = Q(3)-Q(2);
    else       % when there is NaN data
        if i == 1
            delta(i,j) = 0;
        else
            if isnan(delta(i,j))
                delta(i,j) = 0;
            end
        end
        if i == row-2
            delta(i+1,j) = 0;
        else
            delta(i+1,j) = NaN;
        end
    end
  end
end
if dim == 2     % transpose
  delta = delta.';
end
%% end of this function
```

```
function [ g ] = mild_curve_func( x,f,df,h,d )
% g         : wawvefront outside the analyzation area [um]

% x         : x coordinate [um]
% f         : wavefront at R0 [um]
% df        : slope wavefront at R0 [um]
% h         : extra coordinate [um]
% d         : width [um]

k = 2*pi/(4*d);
g = f+df/k*sin(k*(abs(x)-h));
g(abs(x) > h+d) = f+df/k;
g(abs(x) < h) = f;

end

function [ PV,rms,maxh,minh ] = PV_rms_func( data )
% ------------------------------------------------
% output PV and rms value

% PV                  : peak to valley
% rms                 : rms
% maxh, minh          : max and min value

% data                : data (only effective)
% ------------------------------------------------
maxh = max(max(data));
```

```
minh = min(min(data));
PV = maxh-minh;
rms = std(data);

end
%% end of this function

function W = sperical_Zernike_function(r,Zn,A)
% W                 : wavefront [um]

% r                 : normalized coordinate [um]
% Zn                : number of Fringe Zernike
% A                 : coefficient

switch Zn
    case 1
        W = ones(size(r));
    case 2
        W = r;
    case 4
        W = 2*r.^2-1;
    case 9
        W = 6*r.^4-6*r.^2+1;
    case 16
        W = 20*r.^6-30*r.^4+12*r.^2-1;
    case 25
        W = 70*r.^8-140*r.^6+90*r.^4-20*r.^2+1;
```

```
        otherwise
                W = 0;
end
W = A*W;
end

function [ unwrapped_data ] = unwrap1d_func( data )
% unwrap from center

[n1,n2] = size(data);
if n1 == 1
        N = n2;
else
        N = n1;
end
unwrapped_data = zeros(n1,n2);
tmp = unwrap(data(N/2+1:N));
tmp1 = flip(unwrap(flip(data(1:N/2+1))));
unwrapped_data(N/2+1:N) = tmp;
unwrapped_data(1:N/2) = tmp1(1:N/2);

end
```

Appendix C. MATLAB Code of a Talbot interferometer with five beams interference

five_beams_interferogram.m

```
%% Five beams interference
% one-shoot measurement
clear;close all
path(path,'.¥module')
%% condition
lamb = 0.6328;                          % wavelenth [um]
shr = 80;                               % amount of shear [um]
Mag = 1;                                % magnification
R = 2.5*1e3;                            % radius [um]
Rx = 2.5*1e3;                           % analyzation radius x [um]
Ry = Rx;                                % analyzation radius y [um]
xd = -0*2.4*1e3;yd = 2.4*1e3;           % center position of the test surface [mm]
flag = 0;                               % 0 : Alvarez surface
                                        % 1 : other surface
nx = 820;%640;                          % CCD pixel number (x)
ny = nx;%480;                           % CCD pixel number (y)
p = 7.4;                                % CCD pixel pitch [um]
X = (-nx/2:nx/2-1)*p+1e-10;
Y = (-ny/2:ny/2-1)*p+1e-10;
[x,y] = meshgrid(X,Y); y = -y;          % lateral coordinate [um]
fX = (-nx/2:nx/2-1)/(nx*p);             % Fourier coordinate
fY = (-ny/2:ny/2-1)/(ny*p);
```

```
k = 2*pi/lamb;                                  % wavenumber
xq = x+xd;yq = y+yd;
ins = find(x.^2+y.^2 < R^2);                    % analyzation area
ins2 = find(abs(x) < Rx+shr & abs(y) < Ry+shr);


A = 0.0012;                         % coefficient of the test surface [mm-2]
%% interference
W = zeros(ny,nx,4);
if flag == 0
    W0 = 1e3*2*A*(1/3*(xq*1e-3).^3+(xq*1e-3).*(yq*1e-3).^2);    % test surface[um]
    [ a ] = linear_Fit2D_func(W0(ins),x(ins),y(ins));
    W0 = W0 - a(1)-a(2)*x-a(3)*y;
    W(:,:,1) = interp2(x,y,W0,x-shr,y,'cubic');% -1st order x [um]
    W(:,:,2) = interp2(x,y,W0,x+shr,y,'cubic');% 1st order x [um]
    W(:,:,3) = interp2(x,y,W0,x,y-shr,'cubic');% -1st order y [um]
    W(:,:,4) = interp2(x,y,W0,x,y+shr,'cubic');% 1st order y [um]
else
    [ W0 ] = surf_func( x,y,A,R,flag );
    W(:,:,1) = surf_func( x-shr,y,A,R,flag );
    W(:,:,2) = surf_func( x+shr,y,A,R,flag );
    W(:,:,3) = surf_func( x,y-shr,A,R,flag );
    W(:,:,4) = surf_func( x,y+shr,A,R,flag );
end
tmp = zeros(ny,nx);
tmp(ins) = W0(ins);
figure;imagesc(X/1e3,-Y/1e3,tmp);colorbar;axis equal xy tight;
clear W0x W0y r2 tmp xx yy xq yq
```

```
%% Intensity
E0 = exp(1i*k*W0);                                    % 0 order
Ex1 = 0.3*exp(1i*k*(lamb/shr*x+W(:,:,1)));    % -1st order x
Ex2 = 0.3*exp(1i*k*(-lamb/shr*x+W(:,:,2))); % 1st order x
Ey1 = 0.3*exp(1i*k*(lamb/shr*y+W(:,:,3)));    % -1st order y
Ey2 = 0.3*exp(1i*k*(-lamb/shr*y+W(:,:,4))); % 1st order y
E = E0+Ex1+Ex2+Ey1+Ey2;
Iout = real(E.*conj(E));                              % irradiance

tmp2 = Iout(:,:,1);
tmp = zeros(ny,nx);
tmp(ins) = tmp2(ins);
figure;imagesc(tmp);colormap('gray');axis equal off
figure;imagesc(tmp(183:220,183:220));colormap('gray');axis equal tight off

%% wavefront retrieval (FT method)
Iout(isnan(Iout)) = 0;
FT = ifftshift(fft2(fftshift(Iout)));
% figure;imagesc(fX,-fY,log(abs(FT)));axis xy equal tight;colormap('gray');
fshftx = round(p*nx/shr);                             % shift length(x)
fshfty = round(p*ny/shr);                             % shift length(y)
mdftx = p*nx/shr-fshftx;                              % remainder (x)
mdfty = p*ny/shr-fshfty;                              % remainder (y)
if mod(fshftx,2) == 0
    fRx = fshftx/2;                                   % size of cutting (x)
else
    fRx = fshftx/2+0.5;
```

```
end
if mod(fshfty,2) == 0
    fRy = fshfty/2;                                    % size of cutting (y)
else
    fRy = fshfty/2+0.5;
end
tmpx = zeros(ny,nx);
tmpx(ny/2+1-fRy:ny/2+1+fRy,nx/2+1-fRx:nx/2+1+fRx) = ...
    FT(ny/2+1-fRy:ny/2+1+fRy,nx/2+1+fshftx-fRx:nx/2+1+fshftx+fRx);  % cut and paste to the center(x)
% figure;imagesc(fX,-fY,log(abs(tmpx)));axis xy equal tight;colorbar;
tmpy = zeros(ny,nx);
tmpy(ny/2+1-fRy:ny/2+1+fRy,nx/2+1-fRx:nx/2+1+fRx) = ...
    FT(ny/2+1-fRy-fshfty:ny/2+1+fRy-fshfty,nx/2+1-fRx:nx/2+1+fRx); % cut and paste to the center(y)
% figure;imagesc(fX,-fY,log(abs(tmpy)));axis xy equal tight;colorbar;
tmpx = ifftshift(ifft2(fftshift(tmpx)));
tmpy = ifftshift(ifft2(fftshift(tmpy)));
% figure;imagesc((abs(tmpx)));axis equal;colorbar;
tmpx = atan2(imag(tmpx),real(tmpx));
tmpy = atan2(imag(tmpy),real(tmpy));
%       figure;imagesc(X*1e-3,-Y*1e-3,tmpx);axis xy equal tight;colorbar;
%       figure;imagesc(X*1e-3,-Y*1e-3,tmpy);axis xy equal tight;colorbar;
tmp = zeros(ny,nx);
tmp(ins) = tmpy(ins);
figure;imagesc(X*1e-3,-Y*1e-3,tmp);axis xy equal tight;colorbar;
```

```
clear Iout
tmp = zeros(ny,nx);
tmp(ins2) = 1;
tmp(:,end) = 0;tmp(end,:) = 0;
%% unwrapping
[phx] = unwrap_func(tmpx,tmp,pi/3,0.001,ny/2,nx/2);
[phy] = unwrap_func(tmpy,tmp,pi/3,0.001,ny/2+10,nx/2);
phx = -(phx-2*pi/shr*mdftx/(fshftx+mdftx)*x)/shr/k;
tmpx = zeros(ny,nx);
tmpx(ins) = phx(ins);
phy = -(phy-2*pi/shr*mdfty/(fshfty+mdfty)*y)/shr/k;
tmpy = zeros(ny,nx);
tmpy(ins) = phy(ins);

[ ph ] = intg_func( x,y,phx,phy,400 );
ph = ph-mean(ph(ins));
[ K ] = linear_Fit2D_func(ph(ins),x(ins),y(ins));
ph = ph-K(1)-K(2)*x-K(3)*y;
tmp = zeros(ny,nx);
tmp(ins) = ph(ins);
figure;imagesc(X/1e3,-Y/1e3,tmp);axis equal tight xy;colorbar;title('recovered
wavefront[um]')
Ttmp = zeros(ny,nx);
Ttmp(ins) = W0(ins)-mean(W0(ins));
figure;imagesc(Ttmp);axis equal tight;colorbar;title('input wavefront[um]')

phc = compensate_wavefront_func(ph,shr,x,y);
```

```
tmp = zeros(size(phc));
tmp(ins) = phc(ins);
figure;imagesc(tmp);axis equal tight;colorbar;title('cwavefront[um]')

ph = ph-phc;
dif = (ph-Ttmp);
[ K ] = linear_Fit2D_func(dif(ins),x(ins),y(ins));
ph = ph-K(1)-K(2)*x-K(3)*y;
dif = 1e3*(ph-Ttmp);
dif = dif-mean(dif(ins));
tmp = zeros(ny,nx);tmp(ins) = dif(ins); dif = tmp;
figure;imagesc(dif,[-10 10]);axis equal tight;colorbar;title('difference[nm]')
[ PV,rms,maxh,minh ] = PV_rms_func(dif(ins));
figure;imagesc(X/1e3,-Y/1e3,dif);axis equal tight xy;colorbar;title('difference[nm]')
title(['error ',num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS'])
beep
%% end of this file
% programed by Yasunori Furukawa in 12/3/2016

function [ intg_data ] = intg_func( x,y,dWx,dWy,R )
%% -----------------------------------
% integrate the sheared or slope data

% x,y             : coordinate [mm]
% dWx,dWy      : slope
% R                : radius of the start position [pix]
%% -----------------------------------
```

```
[n1,n2] = size(dWx);
intg_data = 0;

NaNflag = zeros(n1,n2);
NaNflag(isnan(dWx) | isnan(dWy)) = 1;
%% decide the start position
N = 60;                                          % number of the start position
if N == 1
    tate = n1/2+1;yoko = n2/2+1;
else
tate = zeros(1,N);yoko = tate;
rad = pi/180;

nk = 360/30*rad;
for j = 1:30
    tate(j) = round(n1/2+R*sin(j*nk+10*rad));
    yoko(j) = round(n2/2+R*cos(j*nk+10*rad));
end
nk = 360/15*rad;
for j = 1:15
    tate(j+30) = round(n1/2+0.8*R*sin(j*nk+20*rad));
    yoko(j+30) = round(n2/2+0.8*R*cos(j*nk+20*rad));
end
nk = 360/7*rad;
for j = 1:7
    tate(j+45) = round(n1/2+0.6*R*sin(j*nk+30*rad));
    yoko(j+45) = round(n2/2+0.6*R*cos(j*nk+30*rad));
```

```
end
nk = 360/5*rad;
for j = 1:5
    tate(j+52) = round(n1/2+0.4*R*sin(j*nk+40*rad));
    yoko(j+52) = round(n2/2+0.4*R*cos(j*nk+40*rad));
end
nk = 360/3*rad;
for j = 1:3
    tate(j+57) = round(n1/2+0.2*R*sin(j*nk+50*rad));
    yoko(j+57) = round(n2/2+0.2*R*cos(j*nk+50*rad));
end
if sum(isnan(tate)) > 0 || sum(isnan(yoko)) > 0
    disp('Decrease R of start position')
else
    tmp = 0;
    for j = 1:N
        tmp = tmp+dWx(tate(j),yoko(j));
    end
    if isnan(tmp) == 1
        disp('There are NaN data at the start position')
    end
end
end
% figure,plot(tate,yoko,'.');axis square
%% Differenciation
Dx = CalcDelta(dWx, x, 2);
Dy = CalcDelta(dWy, y, 1);
```

```
%% Keep adding Dx and Dy from different start positions
for j = 1:N
    data = Integral_func(Dx,Dy,tate(j),yoko(j));
    intg_data = intg_data + data;
end
intg_data = intg_data/N;                                    % average
intg_data = intg_data-mean(intg_data(NaNflag==0));    % remove piston
intg_data(NaNflag==1) = NaN;

function [ data ] = Integral_func(Dx,Dy,cy,cx)
%% integration
% Dx,Dy : differenciation
% cy,cx : coordinate number
%%
[row, col] = size(Dx);
data = zeros(row, col);
flag = zeros(row, col);
if Dx(cy,cx) == 0
    disp('startˆÊ'u,ªNaN,Å,·?BR,Ì'l,ð?¬,³,-,µ,Ä‰º,³,¢');
end
%%
% +X
for j = cx+1:col
    data(cy,j) = data(cy,j-1)+Dx(cy,j-1);
end
% -X
for j = cx-1:-1:1
```

```
        data(cy,j) = data(cy,j+1)-Dx(cy,j);
end
% +Y
for j = cy-1:-1:1
        data(j,cx) = data(j+1,cx)-Dy(j,cx);
end
% -Y
for j = cy+1:row
        data(j,cx) = data(j-1,cx)+Dy(j-1,cx);
end
flag(:,cx) = 1;flag(cy,:) = 1;
%%
for ix = cx+1:col
        for iy = cy-1:-1:1
                if Dy(iy,ix) == 0               % if there is NaN
                        data(iy,ix) = data(iy,ix-1)+Dx(iy,ix-1);
                        flag(iy,ix) = flag(iy,ix-1);
                elseif Dx(iy,ix-1) == 0 % if there is NaN
                        data(iy,ix) = data(iy+1,ix)-Dy(iy,ix);
                        flag(iy,ix) = flag(iy+1,ix);
                else
                        data(iy,ix) = (flag(iy+1,ix)*(data(iy+1,ix)-Dy(iy,ix))+...
                                flag(iy,ix-1)*(data(iy,ix-1)+Dx(iy,ix-1)))/(flag(iy+1,ix)+flag(iy,ix-1));
                        flag(iy,ix) = flag(iy+1,ix)+flag(iy,ix-1);
                        if flag(iy,ix) > 1e15
                                flag(iy,ix) = 1e15;
                        end
```

```
            end
        end
    end
    %%
    for ix = cx+1:col
        for iy = cy+1:row
            if Dy(iy-1,ix) == 0          % if there is NaN
                data(iy,ix) = data(iy,ix-1)+Dx(iy,ix-1);
                flag(iy,ix) = flag(iy,ix-1);
            elseif Dx(iy,ix-1) == 0 % if there is NaN
                data(iy,ix) = data(iy-1,ix)+Dy(iy-1,ix);
                flag(iy,ix) = flag(iy-1,ix);
            else
                data(iy,ix) = (flag(iy-1,ix)*(data(iy-1,ix)+Dy(iy-1,ix))+...
                    flag(iy,ix-1)*(data(iy,ix-1)+Dx(iy,ix-1)))/(flag(iy-1,ix)+flag(iy,ix-1));
                flag(iy,ix) = flag(iy-1,ix)+flag(iy,ix-1);
                if flag(iy,ix) > 1e15
                    flag(iy,ix) = 1e15;
                end
            end
        end
    end
    %%
    for ix = cx-1:-1:1
        for iy = cy-1:-1:1
            if Dy(iy,ix) == 0             % if there is NaN
                data(iy,ix) =    data(iy,ix+1)-Dx(iy,ix);
```

```
            flag(iy,ix) = flag(iy,ix+1);
        elseif Dx(iy,ix) == 0      % if there is NaN
            data(iy,ix) = data(iy+1,ix)-Dy(iy,ix);
            flag(iy,ix) = flag(iy+1,ix);
        else
            data(iy,ix) = (flag(iy+1,ix)*(data(iy+1,ix)-Dy(iy,ix))+...
                flag(iy,ix+1)*(data(iy,ix+1)-Dx(iy,ix)))/(flag(iy+1,ix)+flag(iy,ix+1));
            flag(iy,ix) = flag(iy+1,ix)+flag(iy,ix+1);
            if flag(iy,ix) > 1e15
                flag(iy,ix) = 1e15;
            end
        end
    end
end
%%
for ix = cx-1:-1:1
    for iy = cy+1:row
        if Dy(iy-1,ix) == 0          % if there is NaN
            data(iy,ix) = data(iy,ix+1)-Dx(iy,ix);
            flag(iy,ix) = flag(iy,ix+1);
        elseif Dx(iy,ix) == 0      % if there is NaN
            data(iy,ix) = data(iy-1,ix)+Dy(iy-1,ix);
            flag(iy,ix) = flag(iy-1,ix);
        else
            data(iy,ix) = (flag(iy-1,ix)*(data(iy-1,ix)+Dy(iy-1,ix))+...
            flag(iy,ix+1)*(data(iy,ix+1)-Dx(iy,ix)))/(flag(iy-1,ix)+flag(iy,ix+1));
            flag(iy,ix) = flag(iy-1,ix)+flag(iy,ix+1);
```

```
                if flag(iy,ix) > 1e15
                    flag(iy,ix) = 1e15;
                end
            end
        end
end

function [ delta ] = CalcDelta(diff, crd, dim)
%%
% diff    : slope data
% crd     : coordinate
% dim     : 1: colum; 2: row

if dim == 2
    diff = diff.';
    crd = crd.';
end
[row, col] = size(diff);
delta = zeros(row, col);
%%
for j = 1:col
    for i = 1:row-2
        x = crd(i:i+2,j);
        y = diff(i:i+2,j);
        if sum(isnan(y)) == 0                    % if there is not NaN
            A = [x.^2 x ones(3,1)];
            G = inv_matrix_func(A);              % 3*3, inverse matrix
```

```
            p = G*y;
            Q = p(1)/3*x.^3+p(2)/2*x.^2+p(3)*x;          % integration
            if i == 1
                delta(i,j) = Q(2)-Q(1);
            else
                if isnan(delta(i,j))
                    delta(i,j) = Q(2)-Q(1);
                else
                    delta(i,j) = (delta(i,j)+(Q(2)-Q(1)))/2;
                end
            end
            delta(i+1,j) = Q(3)-Q(2);
        else       % if there is NaN
            if i == 1
                delta(i,j) = 0;
            else
                if isnan(delta(i,j))
                    delta(i,j) = 0;
                end
            end
            if i == row-2
                delta(i+1,j) = 0;
            else
                delta(i+1,j) = NaN;
            end
        end
    end
```

```
end
if dim == 2
    delta = delta.';
end
function G = inv_matrix_func(A)
%% calculate inverse matrix(3*3)
% A : 3*3 matrix
G = zeros(size(A));
detA = A(1,1)*A(2,2)*A(3,3)+A(2,1)*A(3,2)*A(1,3)+A(3,1)*A(1,2)*A(2,3)-
A(1,1)*A(3,2)*A(2,3)-...
    A(3,1)*A(2,2)*A(1,3)-A(2,1)*A(1,2)*A(3,3);
G(1,1) = A(2,2)*A(3,3)-A(3,2)*A(2,3);
G(2,1) = A(2,3)*A(3,1)-A(2,1)*A(3,3);
G(3,1) = A(2,1)*A(3,2)-A(2,2)*A(3,1);
G(1,2) = A(1,3)*A(3,2)-A(1,2)*A(3,3);
G(2,2) = A(1,1)*A(3,3)-A(1,3)*A(3,1);
G(3,2) = A(1,2)*A(3,1)-A(1,1)*A(3,2);
G(1,3) = A(1,2)*A(2,3)-A(1,3)*A(2,2);
G(2,3) = A(1,3)*A(2,1)-A(1,1)*A(2,3);
G(3,3) = A(1,1)*A(2,2)-A(1,2)*A(2,1);
G = G/detA;
%% end of this function
%   programed by Yasunori Furukawa in 2016.04.04
function [ a ] = linear_Fit2D_func(f,x,y)
%% ------------------------------
% Fit f with x, y and piston
% input one-dimensional f,x,y
```

```
% a(1) : constant
% a(2) : x tilt
% a(3) : y tilt
%% ----------------------------


[n1,n2] = size(f);
if n1 == 1
      N = n2;
elseif n2 ==1
      N = n1;
end
M12 = sum(x);
M13 = sum(y);
M22 = sum(x.^2);
M23 = sum(x.*y);
M33 = sum(y.^2);
M = [N M12 M13;
      M12 M22 M23;
      M13 M23 M33];
a = M¥[sum(f); sum(f.*x); sum(f.*y)];


end


function [Z,DZx,DZy] = make_Z_DZ_func(x,y,Zn,maxr,flag)
% ---------------------------------------------------
%% calculate Zernike or derivative Zernike function
```

```
% Z                : Zernike function
% DZx,DZy      : derivative Zernike function

% x,y             : coordinate [mm] (one-dimension)
% Zn               : maximum number of Zernike (16,25,36....)
% maxr           : normalization value [mm]
% flag           : 1 = Zernike, 2 = derivative Zernike
% ----------------------------------------------------

nnum = sqrt(Zn)-1;
[ang,R] = cart2pol(x,y);
nn = size(x,1);
Z = zeros(nn,Zn);DZx = Z;DZy = Z;
R = R/maxr;                              % normalization
%% Zernike
if flag == 1
      for n = 0:nnum
            for m = nnum :-1:0
                  for k = 0:1          % sin / cos
                        Znum = (n+1)^2-2*m + k;
                        if (n >= m) && (m>=k)
                              Z(1:nn,Znum) = 0;
                              for s = 0:n-m
                                    Z(:,Znum)      =      Z(:,Znum)      +(-1)^s*(prod(1:2*n-m-
s)/prod(1:s)/prod(1:n-s)/prod(1:n-m-s).*R.^2.^(n-m-s)) ;
                              end
```

```
                    if k == 0
                        Z(:,Znum) = Z(:,Znum).* R.^m.*cos(m*ang);
                    elseif k == 1
                        Z(:,Znum) = Z(:,Znum).*R.^m.*sin(m*ang);
                    end
                end
            end
        end
        Z(:,1) = ones(nn,1);
        DZx = 0;DZy = 0;
%% derivative Zernike
else
    for n = 0:nnum
        for m = nnum:-1:0    % for the right order
            for k = 0:1      % cos or sin                    x = R*cos(ang) , y = R*sin(ang)
                Zj = (n+1)^2-2*m+k;
                if n == 0 && m == 0
                elseif (n >= m) && (m >= k)
                    Z(:,Zj) = 0;DZx(:,Zj) = 0; DZy(:,Zj) = 0;
                    for s = 0:n-m
                        N = n-m-s;
                        Z(:,Zj) = Z(:,Zj)+(-1)^s * prod(1:2*n-m-
s)/prod(1:s)/prod(1:n-s)/prod(1:N)*R.^(2*N) ;
                        DZx(:,Zj) = DZx(:,Zj)+(-1)^s*prod(1:2*n-m-
s)/prod(1:s)/prod(1:n-s)/prod(1:N)*R.^(2*(N-1))*N*2.*x/maxr^2 ;
                        DZy(:,Zj) = DZy(:,Zj)+(-1)^s*prod(1:2*n-m-
```

```
s)/prod(1:s)/prod(1:n-s)/prod(1:N)*R.^(2*(N-1))*N*2.*y/maxr^2 ;
                                end
                                if k == 0 % cos
                                    DZx(:,Zj)      =      (DZx(:,Zj).*R.^m.*cos(m*ang)+Z(:,Zj)
*m.*R.^(m-2)/maxr^2.*(x.*cos(m*ang)+y.*sin(m*ang)));
                                    DZy(:,Zj)      =      (DZy(:,Zj).*R.^m.*cos(m*ang)+Z(:,Zj)
*m.*R.^(m-2)/maxr^2.*(y.*cos(m*ang)-x.*sin(m*ang)));
                                elseif k == 1 % sin
                                    DZx(:,Zj)      =      (DZx(:,Zj).*R.^m.*sin(m*ang)+Z(:,Zj)
*m.*R.^(m-2)/maxr^2.*(x.*sin(m*ang)-y.*cos(m*ang)));
                                    DZy(:,Zj)      =      (DZy(:,Zj).*R.^m.*sin(m*ang)+Z(:,Zj)
*m.*R.^(m-2)/maxr^2.*(y.*sin(m*ang)+x.*cos(m*ang)));
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
%% end of this function


function [ z ] = surf_func( x,y,A,R0,flag )
% z                : surface [um]


% x,y              : coordinate[um]
% A                : coefficient [um]
% R0               : radius [um]
% flag             : 0 = Z25
```

```
%                       1 = Z16
r2 = x.^2+y.^2;
if flag == 0


elseif flag == 1
      z = A*(20*r2.^3/R0^6-30*r2.^2/R0^4+12*r2/R0^2-1);
elseif flag == 2
      z = A*(1./(r2*1e-6/2+1));
end


function [ph2] = unwrap_func(ph,data,change,limit,x_start,y_start)
% ---------------------------------------------------
% unwrap data
% ph2             : unwrapped data


% ph              : wrapped phase data
% data            : irradiance
% change          : threshold of change per 1 pixel phase
% intense_limit : threshold of the irradiance
% x_start and y_start : start position of unwrap
% ---------------------------------------------------
%% parameter setting
[n1,n2] = size(ph);                                    % n1 * n2 pixel
flag = zeros(n1,n2);                                   % for judge
ph2 = ph ;
first_pos = (y_start-1)*n1+x_start;
flag(first_pos) = 1;
```

```
ph2(first_pos) = ph(first_pos);                    % insert the data of start position

% about flag
% 3 means Not unwrapping
% 1 means boundary , 2 means unwrapped already ,
% 0 means still not unwrapping
%% unwrapping
flag(data < limit) = 3;                % except for the region of low amplitude
nn = 1;
n_right = first_pos;n_left = []; n_up = []; n_down = [];
while isempty(nn) ~= 1                 % if nn is empty ,it finished
    nn = [n_right n_left n_up n_down];
    %right
    n_right = nn + n1 ;
    n_right = n_right(n_right < n1*n2 + 1);
    n_right = n_right(flag(n_right) == 0);
    n_right = n_right(((abs(ph(n_right) - ph(n_right - n1)) < change ) | ...          %
if change is too high,do not unwrap
                                    (abs(ph(n_right) - ph(n_right - n1) + 2*pi)<change ) |...
                                    (abs(ph(n_right) - ph(n_right - n1)-2*pi)<change)));
    ph2(n_right) = ph(n_right)-ph(n_right-n1)+ph2(n_right-n1)-2*pi*fix((ph(n_right)-
ph(n_right-n1))/pi);
    flag(n_right) = 1;
    %left
    n_left = nn - n1 ;
    n_left = n_left(n_left > 1);
    n_left = n_left(flag(n_left) == 0);
```

```
n_left = n_left(((abs(ph(n_left) - ph(n_left + n1)) < change ) | ...
                                (abs(ph(n_left) - ph(n_left + n1) + 2*pi ) < change ) |...
                                (abs(ph(n_left) - ph(n_left + n1) - 2*pi ) < change )));
ph2(n_left) = ph(n_left) - ph(n_left + n1) + ph2(n_left + n1 ) - 2*pi*fix((ph(n_left) - ph(n_left + n1))/pi);
flag(n_left) = 1;
%up
n_up = nn - 1 ;
n_up = n_up(n_up > 1);
n_up = n_up(mod(n_up,n1) ~= 0 & flag(n_up) == 0);
n_up = n_up(((abs(ph(n_up) - ph(n_up + 1)) < change ) | ...
                                (abs(ph(n_up) - ph(n_up + 1) + 2*pi ) < change ) |...
                                (abs(ph(n_up) - ph(n_up + 1) - 2*pi ) < change )));
ph2(n_up) = ph(n_up) - ph(n_up + 1) + ph2(n_up + 1) - 2*pi*fix((ph(n_up) - ph(n_up + 1))/pi);
flag(n_up) = 1;
%down
n_down = nn + 1 ;
n_down = n_down(n_down < n1*n2 + 1);
n_down = n_down(mod(n_down,n1) ~= 1 & flag(n_down) == 0);
n_down = n_down(((abs(ph(n_down) - ph(n_down - 1)) < change ) | ...
                                (abs(ph(n_down) - ph(n_down - 1) + 2*pi ) < change ) |...
                                (abs(ph(n_down) - ph(n_down - 1) - 2*pi ) < change )));
ph2(n_down) = ph(n_down) - ph(n_down - 1) + ph2(n_down - 1) - 2*pi*fix((ph(n_down) - ph(n_down - 1))/pi);
flag(n_down) = 1;
```

```
        flag(nn) = 2;
end
%% end of this function
```

Appendix D. MATLAB code of a Talbot interferometer with a tilted grating using three beams interference

three_interferogram_with_tilted_grating.m

```
%% Three beams interferogram with tilted grating
clear;close all
path(path,'.¥module')
%% condition
rad = pi/180;
lamb = 0.6328;                % wavelength [um]
k = 2*pi/lamb;                % wavenumber
R0 = 2.0*1e3;                 % analyzation radius [um]
Lp = 80;                      % grating pitch [um]
L = Lp^2/lamb;                                                    %
distance of propagation [um]
Ns = 1;                       % phase shift number
gamma = -0.000;               % angle of grating [rad]
Af = 1;                       % coefficient
xshr = Lp;                    % amount of shear [um]
N = 2^13*1;                   % sampling number
p = (R0+10*Lp)/(N/2);      % pixel pitch [um]
x = (-N/2:N/2-1)*p;         % x coordinate [um]
ref_pist = -2.60516586238;          % piston phase of the reference [rad]
if Ns == 1
    zero_pist = -0.02684914;       % piston phase when input wavefront is zero,
elseif Ns == 3
```

```
    zero_pist = 3.115;
elseif Ns == 4
    zero_pist = 3.112713048;
end
flag = -1;
dx = 1e-4;    % [um]
R1 = 10*Lp;
W1 = 1*Alvarez_phase_func( R0*1e-3,x*1e-3,-(R0+R1)*1e-3,(R0+R1)*1e-3,0 ); % wavefront on x-axis
W2 = 1*Alvarez_phase_func( R0*1e-3,x*1e-3,-(R0+R1)*1e-3,(R0+R1)*1e-3,1 ); % wavefront on diagonal
W = Af*(W1+W2)/2;clear W1 W2
Ws1 = 1*Alvarez_phase_func( R0*1e-3,(x-dx)*1e-3,-(R0+R1)*1e-3,(R0+R1)*1e-3,0 );% shifted wavefront on x-axis
Ws2 = 1*Alvarez_phase_func( R0*1e-3,(x-dx)*1e-3,-(R0+R1)*1e-3,(R0+R1)*1e-3,1 );% shifted wavefront on diagonal
Ws = Af*(Ws1+Ws2)/2;clear Ws1 Ws2
dW = (W-Ws)/dx;
% figure;plot(x,W);xlim([-R0 R0])
alfa = asin(dW);                                % incident ray angle [rad]
%% wavefront without grating (reference)
Wt = W+L./cos(alfa)-L;
xt = x+L*tan(alfa);
Wt = interp1(xt,Wt,x,'spline');
clear xt
%%
x1 = x./(1+tan(alfa)*tan(gamma));           % x coordinate on the grating
```

```
y1 = tan(gamma)*x1;                                % y coordinate on the grating
nega = y1 < 0;
ph1 = W-sqrt((x-x1).^2+y1.^2);
ph1(nega) = W(nega)+sqrt((x(nega)-x1(nega)).^2+y1(nega).^2);
clear nega
%% diffraction angle
m = 0;                                  % 0 order
ang = asin(sin(alfa-gamma)+m*lamb/Lp);
beta = gamma+ang;
ph2 = (L+y1)./cos(beta);
W20 = ph1+ph2-L;
x2 = x1+(L+y1).*tan(beta);
W0 = interp1(x2,W20,x,'spline');
m = 1;                                  % 1 order
ang = asin(sin(alfa-gamma)+m*lamb/Lp);
beta = gamma+ang;
ph2 = (L+y1)./cos(beta);
[ diffr_ph ] = intg1D_func( x1/cos(gamma),sin(beta)-sin(alfa));
W21 = ph1+ph2+1*m*diffr_ph-L;
x21 = x1+(L+y1).*tan(beta);
W1 = interp1(x21,W21,x,'spline')+lamb/2;
m = -1;                                  % -1 order
ang = asin(sin(alfa-gamma)+m*lamb/Lp);
beta = gamma+ang;
ph2 = (L+y1)./cos(beta);
[ diffr_ph ] = -intg1D_func( x1/cos(gamma),sin(beta)-sin(alfa));
W2n1 = ph1+ph2+1*m*diffr_ph-L;
```

```
x2n1 = x1+(L+y1).*tan(beta);
Wn1 = interp1(x2n1,W2n1,x,'spline')+lamb/2;
%% irradiance
E0 = exp(1i*k*W0);
Iout = zeros(Ns,N);
for j = 1:Ns
    E1 = sqrt(0.3)*exp(1i*(k*W1-2*pi/Ns*(j-1)));
    E2 = sqrt(0.3)*exp(1i*(k*Wn1+2*pi/Ns*(j-1)));
    E = E0+E1+E2;
    Iout(j,:) = E.*conj(E);
end
figure;plot(x,Iout(1,:))

if Ns == 3
    Wsh = atan2(sqrt(3)*(Iout(2,:)-Iout(3,:)),-2*Iout(1,:)+Iout(2,:)+Iout(3,:));
elseif Ns == 4
    Wsh = -atan2(Iout(4,:)-Iout(2,:),Iout(3,:)-Iout(1,:));
elseif Ns == 6
    Wsh = -atan2(sqrt(3)*(Iout(2,:)+Iout(3,:)-Iout(5,:)-Iout(6,:)),...
                                    2*Iout(1,:)+Iout(2,:)-Iout(3,:)-2*Iout(4,:)-
Iout(5,:)+Iout(6,:));
end
if Ns > 1
    Wsh = Wsh-zero_pist;
    if Wsh(N/2+1) < -pi
        Wsh = Wsh+ 2*pi;
    end
```

```
Wsh = unwrap1d_func(Wsh);      % unwrap
Wsh = 1/xshr*(lamb/Lp*x+1/k*Wsh);
Wt2 = interp1(x,Wt,x+Lp,'spline');
figure;plot(x,(Wt2-Wt)/Lp,x,Wsh,'--'),title('slope');grid
%% integral
[ ph ] = intg1D_func( x,Wsh );
ins = abs(x) < R0;
ph = ph-mean(ph(ins)-Wt(ins));
figure;plot(x,Wt,x,ph,'--'),title('wavefront [um]');xlim([-R0 R0])
figure;plot(x,1e3*(ph-Wt)),title('difference [nm]');xlim([-R0 R0])
Wc = compensate_wavefront_func(ph,Lp,x);
figure;plot(x,Wc);xlim([-R0 R0])
figure;plot(x,1e3*(ph-Wt-Wc+mean(Wc(ins)))),title('error [nm]');xlim([-R0 R0])
ph = ph-Wc;
if flag == -1
    dif = ph-Wt;
    n = 1;
    p = polyfit(x(ins),dif(ins),n);
    for j = 1:n+1
        ph = ph-p(j)*x.^(n-j+1);
    end
    dif = ph-Wt;
    figure;plot(x,Wt,x,ph,'--'),title('[um]')
    [ PV,rms,maxh,minh ] = PV_rms_func(1e3*dif(ins));
    figure;plot(x*1e-3,1e3*(dif)),          title(['error          ',num2str(PV,3),'nmPV,
',num2str(rms,3),'nmRMS']);%ylim([-10 10])
    xlim([-R0*1e-3 R0*1e-3])
```

```
    end
else      % FFT method
    %% phase recovery ----------------------------------------------------
    fI = fftshift(fft(fftshift(Iout(1,:))));
    figure;plot(log10(abs(fI)));title('Fourier Spectrum'); xlim([N/2-1000 N/2+1000]);
    % cut the FFT data ---------------------------------------------------------
    ff = zeros(1,N);
    fshftx = round(p*N/xshr);                          % shift length(x)
    mdftx = p*N/xshr-fshftx;                           % remainder (x)
    if mod(fshftx,2) == 0
        fRx = fshftx/2;                                % size of cutting (x)
    else
        fRx = fshftx/2+0.5;
    end
    fRx = fRx-7;
    ff(N/2+1-fRx:N/2+1+fRx) = fI(N/2+1+fshftx-fRx:N/2+1+fshftx+fRx); % cut(x)
    nc = N/2+1+round(N/(Lp/p));
    figure;plot(real(ff));title('cut Fourier Spectrum');xlim([N/2-1000 N/2+1000]);
    % figure;plot(log10(abs(ff)));title('Fourier Spectrum'); xlim([N/2-1000 N/2+1000]);
    tmp = fftshift(ifft(fftshift(ff)));
    Wsh = -atan2(imag(tmp),real(tmp))-zero_pist;
    RR = R0+R1;
    Wsh((abs(x)) > RR) = 0;
    figure;plot(x,Wsh);xlim([-RR RR]);title('wrapped phase[rad]')
    if Wsh(N/2+1) > pi
        Wsh = Wsh-2*pi;
    end
```

```
[ Wsh ] = unwrap1d_func( Wsh ); % unwrap
Wsh((abs(x)) > RR) = 0;
Wsh = Wsh/k/Lp;
figure;plot(x,Wsh);title('unwrapped subtraction phase');xlim([-RR RR])

[ W ] = intg1D_func( x,Wsh ); % integration
tmp = compensate_wavefront_func(W,Lp,x);
ph2 = W-tmp;
dif = ph2-Wt;
tmp = mean(dif(abs(x) < R0));
dif = 1e3*(dif-tmp);
ph2 = ph2-tmp;
figure;plot(x/1e3,Wt,x/1e3,ph2,'--');
title('wavefront[um]')
xlim([-R0/1e3 R0/1e3]);grid;hold off
[ PV,rms,maxh,minh ] = PV_rms_func(dif(abs(x) < R0));
figure;plot(x/1e3,dif);                % difference of output
xlim([-R0/1e3 R0/1e3]);grid;hold off
title(['error ',num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS']);ylim([-10 10])
%% remove piston and tilt
ins = abs(x) < R0;
n = 1;
p = polyfit(x(ins),dif(ins),n);
for j = 1:n+1
      dif = dif-p(j)*x.^(n-j+1);
end
  [ PV,rms,maxh,minh ] = PV_rms_func(dif(abs(x) < R0));
```

```
        figure;plot(x/1e3,dif);                % difference of output
        xlim([-R0/1e3 R0/1e3]);grid;hold off
        title(['error ',num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS']);ylim([-10 10])
end
% save('dif_noerror','dif')
```

Appendix E. MATLAB code of a stitching simulation

stitch_sim.m

```
%% ------------------------------------------------------
%% stitch simulation

% 1. Set parameters
% 2. Create error shape
% 3. Determine lattice design
% 4. Assume alignment error and system error
% 5. Calculate sub-aperture shape
% 6. Coordinate transform before stitching
% 7. Stitching
% 8. Evaluate coefficients
% 9. Calculate stitch shape
% 10. Evaluate the error shape
% 11. Estimate the stitching error from difference between overlapped data

%% ------------------------------------------------------
clear;close all
path(path,'.¥module')
path(path,'..¥module')
%% 1. Set parameter
Q = 4;
Ncx = 640/Q;                    % CCD pixel number (x)
Ncy = 640/Q;                    % CCD pixel number (y)
```

```
cp = 0.0074*Q;                          % CCD pixel pitch [mm]

flag = 0;                               % 0 = Alvarez surface
mtrx_flag =0;                           % 0 = load Zernike function
                                        % 1 = calculate Zernike function which expresses alignment error and system error stitching
shftflag = 0;                           % 0 = estimate shift error
                                        % 1 = Not estimate shift error
syserrflag = 1;                         % 0 = estimate sysyem error
                                        % 1 = Not estimate sysyem error
                                        % 2 = Not estimate, but input sysyem error
spike_noise_flag = 0;                   % 0 = Not add spike noise
                                        % 1 = add spike noise
Cnoise = 0;                             % noise [Cnoise*2 nmPV]
ptime    = 0.2;                         % pause time[s]
Lx = 10;                                % sample size (x) [mm]
Ly = 10;                                % sample size (y) [mm]
pp = 0.0074*7;                          % sample pixel pitch [mm]
Nx = 320;                               % sample pixel number (x)
Ny = 320;                               % sample pixel number (y)
Nset = 6;
Nsys = 16;
Zn = 16;
Znin = 16;

slx = 4.2;                              % length(x) of sub-aperture [mm]
sly = 3;                                % length(y) of sub-aperture [mm]
```

```
Nsx = 3;                                % number of stitching (x)
Nsy = 4;                                % number of stitching (y)
N = Nsx*Nsy;                            % total number of stitching
rr = sqrt((Lx/2).^2+(Ly/2).^2);
% CCD coordinate [mm]
Ncx = round(Ncx*cp/pp);
Ncy = round(Ncy*cp/pp);
if mod(Ncx,2) == 0
    cX = (-Ncx/2:Ncx/2-1)*pp;
else
    cX = (-(Ncx-1)/2:(Ncx+1)/2-1)*pp;
end
if mod(Ncy,2) == 0
    cY = (-Ncy/2:Ncy/2-1)*pp;
else
    cY = (-(Ncy-1)/2:(Ncy+1)/2-1)*pp;
end
% global coordinate [mm]
X = (-Nx/2:Nx/2-1)*pp ;
Y = (-Ny/2:Ny/2-1)*pp ;
[x,y] = meshgrid(X,Y) ;y = -y;
%% 2. Create error shape
f = surface_func( x,y,flag ); % design shape [mm]
% outa = abs(x) > Lx/2 | abs(y) > Ly/2;
% f(outa) = 0;
% figure,imagesc(X,-Y,f);axis equal;axis tight;colorbar;axis xy;axis([-Lx/2 Lx/2 -Ly/2
Ly/2])
```

```
ferr = 45*(6*(x.^2+y.^2).^2/(Lx/2)^4-6*(x.^2+y.^2)/(Lx/2)^2+1);    % error shape [nm]
% ferr(outa) = 0;
% figure,imagesc(X,-Y,ferr);axis equal;axis tight;colorbar;axis xy;axis([-Lx/2 Lx/2 -Ly/2 Ly/2])
Az = f+1e-6*ferr;                                                  % sample shape [mm]
%% 3. Determine lattice design
spx = lattice_design_func( Nsx,slx,Lx );
spy = -lattice_design_func( Nsy,sly,Ly );
% disp_lattice_design_func( spx,spy,slx,sly,Lx,Ly,0 );
[ mxyz,th ] = calc_stitch_move_func( spx,spy,slx,sly,Lx,Ly,0);    % amount of move
%% 4. System error coefficient
if syserrflag == 0 || syserrflag == 2
    Csys = 40*rand(1,49);Csys(1:3) = 0;
    Xa = -slx/2:0.1:slx/2;
    Ya = -sly/2:0.1:sly/2;
    [xa,ya] = meshgrid(Xa,Ya);ya = -ya;
    Zq = make_Z_DZ_func(xa(:),ya(:),Znin,rr,1);
    sys_err = Zq(:,Nset+1:Znin)*Csys(Nset+1:Znin)';
    tmp = reshape(sys_err,size(xa,1),size(ya,2));
    rms = std(tmp(:))
    PV = max(tmp(:))-min(tmp(:));
    figure,imagesc(Xa,-Ya,tmp,[-3*rms 3*rms]);axis equal;axis tight;colorbar;axis xy;axis([-slx/2 slx/2 -sly/2 sly/2])
    title(['error ',num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS'])
else
    Csys = zeros(1,49);
end
```

```
%% Alignment error coefficient
rand('twister',2');
Cset = (rand(1,N*Nset)-0.5)*2*1200*1/2 ;                    % z, thetax, thetay error [nm]
Cset(4:6:end) = 0;
if shftflag == 0
    Cset(5:6:end) = (rand(1,N)-0.5)*2*10*1e3;           % x shift error [nm]
    Cset(6:6:end) = (rand(1,N)-0.5)*2*10*1e3;           % y shift error [nm]
else
    Cset(5:6:end) = 0;          % x shift error [nm]
    Cset(6:6:end) = 0;          % y shift error [nm]
end
rand('twister',sum(100*clock)) ;
%% 5. Calculate sub-aperture shape
k = 1;
px = zeros(N,2);py = px;
Dflag    = false(Ncy,Ncx,N);
Qx = zeros(Ncy,Ncx,N) ;Qy = zeros(Ncy,Ncx,N) ;Qz = zeros(Ncy,Ncx,N);
if mtrx_flag == 1
    MtxZ = zeros(Ncy,Ncx,N,Zn) ;
end
for i = 1:Nsx
    for j = 1:Nsy
        perc = (k-1)/N*100 ;
        %% local sample coordinate?isame interval?j
        [ ~,ix ] = min(abs(X-mxyz(k,1))) ;                  % center coordinate
of sub-aperture (x)[pix]
        [ ~,iy ] = min(abs(-Y-mxyz(k,2))) ;                 % center coordinate
```

```
of sub-aperture (y)[pix]
        if mod(Ncx,2) == 0
            px(k,1) = ix-Ncx/2 ;        px(k,2) = ix+Ncx/2-1 ;
        else
            px(k,1) = ix-(Ncx-1)/2 ;        px(k,2) = ix+(Ncx+1)/2-1 ;
        end
        if mod(Ncy,2) == 0
            py(k,1) = iy-Ncy/2 ;        py(k,2) = iy+Ncy/2-1 ;
        else
            py(k,1) = iy-(Ncy-1)/2 ;        py(k,2) = iy+(Ncy+1)/2-1 ;
        end
        tx = x(py(k,1):py(k,2),px(k,1):px(k,2));
        ty = y(py(k,1):py(k,2),px(k,1):px(k,2));
        tz = f(py(k,1):py(k,2),px(k,1):px(k,2));                % designed shape in sub-
aperture [mm]
        Tz = Az(py(k,1):py(k,2),px(k,1):px(k,2));                % designed and error
shape in sub-aperture[mm]

        pos = find(abs(tx) > Lx/2+0.5 | abs(ty) > Ly/2+0.5);
        tx(pos) = NaN;ty(pos) = NaN;tz(pos) = NaN;
        Dflag(:,:,k) = abs(tx-mxyz(k,1)) <= slx/2 & abs(ty-mxyz(k,2)) <= sly/2 & abs(tx)
<= Lx/2 & abs(ty) <= Ly/2;
%% drive sample shift,tip,tilt), from local sample to CCD coordinate (non-equal pitch)
        [ qx,qy,qz ] = trans_func( mxyz(k,:),th(k,:),1,tx,ty,tz);
%           figure;imagesc(qz);colorbar% designed value
        [                Qx(:,:,k),Qy(:,:,k),Qz(:,:,k)                ]                =
trans_func( mxyz(k,:),th(k,:),1,tx,ty,Tz );        % error shape
```

```
%% create Zernike polynomials (on local sample coordinate, equal pitch)[um]
 if mtrx_flag == 1
      [ MtxZ(:,:,k,:) ] = make_stitchZ_func(mxyz(k,:),th(k,:),rr,Zn,qx,qy,qz,flag);
 end
%% add alignment error and system error
 xxq = Qx(:,:,k);yyq = Qy(:,:,k);
 Zq = make_Z_DZ_func(xxq(:),yyq(:),Znin,rr,1);
 if k == 1
      set_err = 0;
 else
      set_err = 1e-6*(Cset((k-1)*Nset+1)*Zq(:,1)+Cset((k-1)*Nset+2)*Zq(:,2)+
Cset((k-1)*Nset+3)*Zq(:,3)+ Cset((k-1)*Nset+4)*Zq(:,4)*0);
      set_err = reshape(set_err,Ncy,Ncx);      % alignment error [mm]
 end
 sys_err = Zq(:,Nset+1:Znin)*Csys(Nset+1:Znin)';
 sys_err = reshape(sys_err,Ncy,Ncx);
 sys_err = sys_err*1e-6;                          % system error [mm]
 noise = Cnoise*1e-6*(rand(Ncy,Ncx)-0.5)*2;
 spike_noise = zeros(Ncy,Ncx);
 if spike_noise_flag == 1
      qp = round((Ncx-40)*rand(1,2))+20;
      spike_noise(qp(1)-3:qp(1)+3,qp(2)-3:qp(2)+3) = 10*1e-6;
 end
 tmp = rot90(Qz(:,:,k),2);
 figure(4),imagesc(cX,-cY,-2*tmp,[-0.1 0.1]);
 axis equal;axis tight xy;colorbar;axis([-slx/2 slx/2 -sly/2 sly/2]);
 Qz(:,:,k) = Qz(:,:,k)+(set_err+sys_err)+noise+spike_noise;
```

```
            tmp = Qz(:,:,k).*Dflag(:,:,k);tmp(isnan(tmp)) = 0;
            figure(4),imagesc(cX,-cY,tmp,[-0.05 0.05]);axis equal;axis tight
xy;colorbar;title([num2str(k)]);ylim([-1.75 1.75])
            k = k+1;
            pause(ptime);
        end
end
if mtrx_flag == 0
        load('matrixZ.mat');
elseif mtrx_flag == 1
        save('matrixZ','MtxZ');
end
clear Zt xxq yyq M
%% 6. Coordinate transform before stitching
k = 1;
wz = zeros(Ncy,Ncx,N);
for i = 1:Nsx
        for j = 1:Nsy
            %% Coordinate transform (cx?¨x, CCD coor?¨local sample coor?iNon-equal
pitch)
            if k ==1
                [ qx2,qy2,qz2 ] = trans_func( mxyz(k,:),th(k,:),-
1,Qx(:,:,k),Qy(:,:,k),Qz(:,:,k));
            else
                [ qx2,qy2,qz2 ] = trans_func( mxyz(k,:),th(k,:),-1,Qx(:,:,k)+Cset((k-
1)*Nset+5)/1e6,Qy(:,:,k)+Cset((k-1)*Nset+6)/1e6,Qz(:,:,k));
            end
```

```
            wz(:,:,k) = 1e3*(qz2-surface_func( qx2,qy2,flag ));          % subtract designed
shape [um]
            tmp = wz(:,:,k);tmp(isnan(tmp)) = 0;
            qx2(isnan(qx2)) = 10;qy2(isnan(qy2)) = 10;
            wz(:,:,k)                                                                    =
griddata(qx2,qy2,tmp,x(py(k,1):py(k,2),px(k,1):px(k,2)),y(py(k,1):py(k,2),px(k,1):px(k,2
)),'cubic');
            wz(:,:,k) =    wz(:,:,k) + 0*noise;
            %% calculate sample shape on local sample coordinate (equal pitch)
            tmp = wz(:,:,k).*Dflag(:,:,k);
            tmp(isnan(tmp)) = 0;
            figure(5),imagesc(tmp,[-0.5 0.5]);axis equal;axis tight;colorbar;title(['stitch data
[um] ',num2str(k)]);% err+sys+align
            k = k+1;
            pause(ptime);
        end
end
if shftflag ~= 0
        MtxZ(:,:,:,5) = 0;
        MtxZ(:,:,:,6) = 0;
end
if syserrflag ~= 0;
            MtxZ(:,:,:,Nset+1:Nsys) = 0;
end
%% 7. Stitching ------------------------------------------
tic
[ W,Cset_out,Csys_out ] = stitching_func(wz,Dflag,N,Nset,Nsys,MtxZ,px,py,Nx,Ny,0) ;
```

```
toc
Cset_out = 1e3*Cset_out;        % from [um] to [nm]
Cset_out(5:Nset:end) = 10*Cset_out(5:Nset:end);
Cset_out(6:Nset:end) = 10*Cset_out(6:Nset:end);
Csys_out = 1e3*Csys_out;            % from [um] to [nm]
%% 8. Evaluate results
wn = Nset+1;
figure ; plot(wn:Zn,Csys(wn:Zn),'--o',(wn:Zn),Csys_out,'--rs') ;
grid;title('system error [nm]');xlim([wn Zn])
figure ; plot(Cset(wn:end),'--o');hold on
plot(Cset_out,'--rs') ;hold off;grid;title('alignment error [nm]')
figure ; plot(wn:Zn,Csys_out-Csys(wn:Zn),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',10) ;
grid;title('system error difference [nm]');xlim([wn Zn])
figure ; plot(Cset_out-Cset(wn:end),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',10);
grid;title('alignment error difference [nm]')
%% 9. Calculate stitch shape
W(isnan(W)) = 0;
st_fig = zeros(Ny,Nx) ;st_fig2 = zeros(Ny,Nx) ;
num_flag = zeros(Ny,Nx) ;                % overlapping number
for j = 1:N
    st_fig2(py(j,1):py(j,2),px(j,1):px(j,2)) = st_fig2(py(j,1):py(j,2),px(j,1):px(j,2)) +
W(:,:,j).*Dflag(:,:,j) ;
    temp_W = false(Ny,Nx) ;
    temp_W(py(j,1):py(j,2),px(j,1):px(j,2)) = Dflag(:,:,j) ;
    num_flag = num_flag + temp_W ;
```

```
end
ins = num_flag > 0;
st_fig(ins) = st_fig2(ins)./num_flag(ins);
figure,imagesc(X,-Y,1e3*st_fig);axis equal;axis tight;colorbar;axis xy;axis([-Lx/2 Lx/2 -
Ly/2 Ly/2])
clear st_fig2
% save('output.mat','Cset_out','Csys_out','st_fig');
%% 10. Evaluate the error shape
dif = zeros(Ny,Nx);                                    %
dif(ins) = 1e3*st_fig(ins)-ferr(ins);      % difference [nm]
tmp = dif(ins);
rms = std(tmp);
PV = max(tmp)-min(tmp);
figure,imagesc(X,-Y,dif,[-3*rms 3*rms]);axis equal;axis tight;colorbar;axis xy;axis([-Lx/2
Lx/2 -Ly/2 Ly/2])
title(['error ',num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS'])
%% 11. Estimate the stitching error from difference between overlapped data
W2 = NaN(Ny,Nx,N);
for j = 1:N
    W2(py(j,1):py(j,2),px(j,1):px(j,2),j) = W(:,:,j).*Dflag(:,:,j);
end
W2(W2==0) = NaN;
[ dev,sgma ] = stitch_err_func(W2,num_flag,st_fig);% um
dev = 1e3*dev;sgma = 1e3*sgma;
figure,imagesc(X,-Y,dev,[0 5*sgma]);axis equal;axis tight;colorbar;axis xy
axis([-Lx/2 Lx/2 -Ly/2 Ly/2])
title(['stitch error ' ,num2str(sgma,3),'nmRMS'])
```

```
[ st_fig4 ] = Alignment_remove_func( 1e3*st_fig,f,x,y,ins );
dif = zeros(Ny,Nx);                                    %
dif(ins) = st_fig4(ins)-ferr(ins);      % difference [nm]
dif(ins) = dif(ins) - mean(dif(ins));
tmp = dif(ins);
rms = std(tmp);
PV = max(tmp)-min(tmp);
figure,imagesc(X,-Y,dif,[-3*rms 3*rms]);axis equal;axis tight;colorbar;axis xy;axis([-Lx/2
Lx/2 -Ly/2 Ly/2])
title(['error ',num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS'])
```

loop_stitch.m

```
%% ----------------------------------------------------------
%% stitch again
%% ----------------------------------------------------------
close all
k = 1;
for i = 1:Nsx
      for j = 1:Nsy
            % display bar
            perc = (k-1)/N*100 ;
           %% local sample coordinate?isame interval?j
            tx = x(py(k,1):py(k,2),px(k,1):px(k,2));
            ty = y(py(k,1):py(k,2),px(k,1):px(k,2));
            tz = f(py(k,1):py(k,2),px(k,1):px(k,2));                % designed shape in sub-
```

```
aperture [mm]
        Tz = Az(py(k,1):py(k,2),px(k,1):px(k,2));          % designed and error
shape in sub-aperture[mm]

        pos = find(abs(tx) > Lx/2+0.5 | abs(ty) > Ly/2+0.5);
        tx(pos) = NaN;ty(pos) = NaN;tz(pos) = NaN;
        %% subtract alignment error and system error
        xxq = Qx(:,:,k);yyq = Qy(:,:,k);
        Zq = make_Z_DZ_func(xxq(:),yyq(:),Zn,rr,1);
        if k == 1
            set_err2 = 0;
        else
            set_err2 = 1e-6*(Cset_out((k-2)*Nset+1)*Zq(:,1)+Cset_out((k-
2)*Nset+2)*Zq(:,2)+ Cset_out((k-2)*Nset+3)*Zq(:,3)+ Cset_out((k-
2)*Nset+4)*Zq(:,4)*0);
            set_err2 = reshape(set_err2,Ncy,Ncx);   % alignment error [mm]
        end
        sys_err2 = Zq(:,Nset+1:Zn)*Csys_out(1:Zn-Nset)';
        sys_err2 = reshape(sys_err2,Ncy,Ncx);
        sys_err2 = 1*sys_err2*1e-6;                % system error [mm]
        Qz(:,:,k) = Qz(:,:,k)-(set_err2+sys_err2);
        if k ~= 1
            Qx(:,:,k) = Qx(:,:,k)+(Cset((k-1)*Nset+5)-Cset_out((k-2)*Nset+5))/1e6;
            Qy(:,:,k) = Qy(:,:,k)+(Cset((k-1)*Nset+6)-Cset_out((k-2)*Nset+6))/1e6;
        end
        figure(1),imagesc(Qz(:,:,k).*Dflag(:,:,k),[-0.3 0.3]);axis equal;axis
tight;colorbar;title(['measurement ',num2str(k)])
```

```
            pause(ptime);
            [ qx2,qy2,qz2 ] = trans_func( mxyz(k,:),th(k,:),-1,Qx(:,:,k),Qy(:,:,k),Qz(:,:,k));
            wz(:,:,k) = 1e3*(qz2-surface_func( qx2,qy2,flag ));          % subtract designed
shape [um]
            tmp = wz(:,:,k);tmp(isnan(tmp)) = 0;
            qx2(isnan(qx2)) = 10;qy2(isnan(qy2)) = 10;
            wz(:,:,k)                                                                    =
griddata(qx2,qy2,tmp,x(py(k,1):py(k,2),px(k,1):px(k,2)),y(py(k,1):py(k,2),px(k,1):px(k,2
)),'cubic');
            tmp = wz(:,:,k).*Dflag(:,:,k);
            tmp(isnan(tmp)) = 0;
            figure(2),imagesc(tmp,[-0.6 0.6]);axis equal;axis tight;colorbar;title(['stitch data
[um] ',num2str(k)]);% err+sys+align
            k = k+1;
            pause(ptime);
        end
end
tic
[                          W2,Cset_out2,Csys_out2                          ]                    =
stitching_func(wz,Dflag,N,Nset,Nsys,MtxZ,px,py,Nx,Ny,0) ;
toc
Cset_out2 = 1e3*Cset_out2;        % [um]?¨[nm]
Cset_out2(5:Nset:end) = 10*Cset_out2(5:Nset:end);
Cset_out2(6:Nset:end) = 10*Cset_out2(6:Nset:end);
Csys_out2 = 1e3*Csys_out2;                                                          %
[um]?¨[nm]
Csys_out = Csys_out+Csys_out2;
```

```
Cset_out = Cset_out+Cset_out2;


wn = Nset+1;
figure ; plot(wn:Zn,Csys(wn:Zn),'--o',(wn:Zn),Csys_out,'--rs') ;
grid;title('system error [nm]');xlim([wn Zn])
figure ; plot(Cset(wn:end),'--o');hold on
plot(Cset_out,'--rs') ;hold off;grid;title('alignment error [nm]')
figure;                                                    plot(wn:Zn,Csys_out-Csys(wn:Zn),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',10) ;
grid;title('system error difference [nm]');xlim([wn Zn])
figure;                                                    plot(Cset_out-Cset(wn:end),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',10);
grid;title('alignment error difference [nm]')
%% stitch shape
W2(isnan(W2)) = 0;
st_fig = zeros(Ny,Nx) ;st_fig2 = zeros(Ny,Nx) ;
num_flag = zeros(Ny,Nx) ;                  % overlapping number
for j = 1:N
    st_fig2(py(j,1):py(j,2),px(j,1):px(j,2))   =   st_fig2(py(j,1):py(j,2),px(j,1):px(j,2))   +
W2(:,:,j).*Dflag(:,:,j) ;
    temp_W = false(Ny,Nx) ;
    temp_W(py(j,1):py(j,2),px(j,1):px(j,2)) = Dflag(:,:,j) ;
    num_flag = num_flag + temp_W ;
end
ins = num_flag > 0;
st_fig(ins) = st_fig2(ins)./num_flag(ins);
clear st_fig2
```

```
%% Evaluate
dif = zeros(Ny,Nx);
dif(ins) = 1e3*st_fig(ins)-ferr(ins);      % difference [nm]
tmp = dif(ins);
rms = std(tmp);
PV = max(tmp)-min(tmp);
figure,imagesc(X,-Y,dif,[-3*rms 3*rms]);axis equal tight xy;colorbar; axis([-Lx/2 Lx/2 -
Ly/2 Ly/2])
title(['stitch error ',num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS'])
%% 11. Estimate the stitching error from difference between overlapped data
W3 = NaN(Ny,Nx,N);
for j = 1:N
      W3(py(j,1):py(j,2),px(j,1):px(j,2),j) = W2(:,:,j).*Dflag(:,:,j);
end
W3(W3==0) = NaN;
[ dev,sgma ] = stitch_err_func(W3,num_flag,st_fig);% um
dev = 1e3*dev;sgma = 1e3*sgma;
figure,imagesc(X,-Y,dev,[0 2*sgma]);axis equal tight xy;colorbar; axis([-Lx/2 Lx/2 -Ly/2
Ly/2])
title(['stitch error ' ,num2str(sgma,3),'nmRMS'])

[ st_fig4 ] = Alignment_remove_func( 1e3*st_fig,f,x,y,ins );
dif = zeros(Ny,Nx);                                    %
dif(ins) = st_fig4(ins)-ferr(ins);      % difference [nm]
dif(ins) = dif(ins) - mean(dif(ins));
tmp = dif(ins);
rms = std(tmp);
```

```
PV = max(tmp)-min(tmp);
figure,imagesc(X,-Y,dif,[-3*rms 3*rms]);axis equal;axis tight;colorbar;axis xy;axis([-Lx/2
Lx/2 -Ly/2 Ly/2])
title([num2str(PV,3),'nmPV, ',num2str(rms,3),'nmRMS'])

function [ W ] = Alignment_remove_func( W,f,x,y,ins )
%% remove the alignment component
% W          : wavefront or surface [nm]
% f          : nominal shape [mm]
% x,y        : lateral coordinate [mm]
% ins        : fitting area
%% -------------------------------------
N = size(find(ins),1);
A = zeros(N,5);
A(:,1) = 1;
A(:,2) = x(ins);
A(:,3) = y(ins);

tmpx = zeros(size(x));
tmpx(:,1:end-1) = f(:,2:end) - f(:,1:end-1);
tmpy = zeros(size(x));
tmpy(2:end,:) = f(1:end-1,:) - f(2:end,:);
A(:,4) = tmpx(ins);
A(:,5) = tmpy(ins);

[U,S,V] = svd(A,0);
```

```
for j = 1:size(S,1)
    if S(j,j) < 2
        S(j,j) = 0;
    else
        S(j,j) = 1/S(j,j);
    end
end
C = V*S*U'*W(ins);
tmp = W(ins)-A*C;
W = zeros(size(x));
W(ins) = tmp;
end

function [ move_xyz,th ] = calc_stitch_move_func( px,py,sx,sy,Lx,Ly,flag)
% ------------------------------------------------------------------
%% Calculate the amount of moving

% move_xyz          : amount of moving (x,y,z) [mm]
% th                : theta (x,y) [rad]

% px,py             : center position of the sub-aperture [mm]
% sx,sy             : size of the sub-aperture [mm]
% Lx,Ly             : size of the sample [mm]
% flag              : 0 = Alvarez lens
% ------------------------------------------------------------------

Nx = size(px,2);      Ny = size(py,2);
```

```
% create sub-aperture coordinate
N = 50*4;                                                    % sampling number
X = (-N/2:N/2-1)/(N/2)*sx/2;
Y = (-N/2:N/2-1)/(N/2)*sy/2;
[x,y] = meshgrid(X,Y); y = -y;                    % sub-aperture coordinate

move_xyz = zeros(Nx*Ny,3);
th = zeros(Nx*Ny,2);

num = 1;
for i = 1:Ny
    for j = 1:Nx
        xt = x+px(j);                                    % sample coordinate (x)
        yt = y+py(i);                                    % sample coordinate (y)
        [ f ] = surface_func( xt,yt,flag ); % sample shape [mm]
        ins = abs(xt) < Lx/2 & abs(yt) < Ly/2;          % area in the sample
        [ a ] = linear_Fit2D_func(f(ins),x(ins),y(ins));% coefficient
        f = f - a(1)-a(2)*x-a(3)*y;
        move_xyz(num,1) = px(j);
        move_xyz(num,2) = py(i);
        move_xyz(num,3) = a(1);
        th(num,1) = atan(a(2));
        th(num,2) = atan(a(3));
        num = num+1;
    end
end
end
```

```
%% end of this function
% programed by Yasunori Furukawa in Dec.5,2016

function disp_lattice_design_func( px,py,lx,ly,Lx,Ly,flag )
% -----------------------------------------------------------------
%% display the lattice design
% px,py              : center position of the sub-aperture
% lx,ly              : size of the sub-aperture
% Lx,Ly              : size of the sample
% flag               : select the type of the sample figure
%                      : 0 = rectangular
%                      : 1 = circle
% -----------------------------------------------------------------

Nx = size(px,2);       Ny = size(py,2);
%% display the overlapped sub-aperture
nn = 100;
X = (-nn/2:nn/2-1)/(nn/2)*1.2*Lx/2;
Y = (-nn/2:nn/2-1)/(nn/2)*1.2*Ly/2;
[x,y] = meshgrid(X,Y);y = -y;
over_flag = zeros(size(x));                                    % overlapp number
if flag == 0
    for i = 1:Nx
        for j = 1:Ny
            ins = find(abs(x-px(i)) < lx/2 & abs(y-py(j)) < ly/2);    % area of sub-
aperture
            over_flag(ins) = over_flag(ins)+1;                          % add 1
```

```
            end
        end
elseif flag == 1
end
over_flag(abs(x) > Lx/2 & abs(x) < Lx/2+0.1 & abs(y) < Ly/2+0.1) = -1;% draw the rim of the sample
over_flag(abs(y) > Ly/2 & abs(y) < Ly/2+0.1 & abs(x) < Lx/2+0.1) = -1;% draw the rim of the sample

figure,imagesc(X,-Y,over_flag);title('overlap number');
colorbar;axis equal xy tight;
%% display the center position of the sub-aperture
figure;
for i = 1:Nx
        for j = 1:Ny
            plot(px(i),py(j),'ko');hold on
        end
end
plot([-Lx/2 Lx/2 Lx/2 -Lx/2 -Lx/2],[-Ly/2 -Ly/2 Ly/2 Ly/2 -Ly/2],'g') % draw the rim of the sample
axis([-Lx/2*1.2 Lx/2*1.2 -Ly/2*1.2 Ly/2*1.2]);title('center position')
axis square;
hold off
end
%% end of this function
% programed by Yasunori Furukawa    Dec.5,2016
```

```
function [ pos ] = lattice_design_func( N,h,L )
% ------------------------------------------------------------------
%% determine the lattice design(sub-aperture location)
% pos             : center position of the sub-aperture

% N                : number of stitching
% h                : size of sub-aperture
% L                : size of sample
% ------------------------------------------------------------------

pos = zeros(1,N);
pos(1) = (-L+h)/2;    pos(N) = -pos(1);
if N == 3
      pos(2) = 0;
elseif N == 4
      pos(2) = (-L+h)/6;    pos(3) = -pos(2);
elseif N == 5
      pos(2) = (-L+h)/4;    pos(3) = 0;          pos(4) = -pos(2);
end
%% end of this function
% programed by Yasunori Furukawa    Dec.5,2016

function [ Z ] = make_stitchZ_func(mxyz,th,rr,Zn,qx,qy,qz,flag    )
% ------------------------------------------------
%% Calculate the Zernike polynomials on the local sample coordinate

% Z                      : Zernike function [um]
```

```
% mxyz                 : amount of move [mm]
% th                   : theta x,y [rad]
% rr                   : [mm]
% Zn                   : max number of Zernike (9,16,25,36 ...etc)
% qx,qy,qz            : CCD cordinate [mm]
% flag                 :

% ------------------------------------------------

[ n1,n2 ] = size(qx);
Z = zeros(n1,n2,Zn);
Zt = Z;
%% Zernike on the CCD coordinate
Ztmp = make_Z_DZ_func(qx(:),qy(:),Zn,rr,1);
for j = 1:Zn
    Zt(:,:,j) = reshape(Ztmp(:,j),n1,n2);
end
Zt(:,:,5) = 0;Zt(:,:,6) = 0;
%% Zernike on the local sample coordinate
for j = 1:Zn
    if j == 5
        [ xa,ya,~ ] = trans_func( mxyz,th,-1,qx,qy,qz );
        [ xx,yy,zz ] = trans_func( mxyz,th,-1,qx+1e-2,qy,qz ); % 10um x-shift
    elseif j == 6
        [ xa,ya,~ ] = trans_func( mxyz,th,-1,qx,qy,qz );
        [ xx,yy,zz ] = trans_func( mxyz,th,-1,qx,qy+1e-2,qz ); % 10um y-shift
```

```
        else
            [ xx,yy,zz ] = trans_func( mxyz,th,-1,qx,qy,qz+1e-3*Zt(:,:,j) );
        end
        if j == 5 || j == 6
            xx = xx(isfinite(xx));yy = yy(isfinite(yy));zz = zz(isfinite(zz));
            ina = isfinite(xa);
            Ztmp = 1e3*(zz-surface_func( xx,yy,flag )); %[um]
            tmp2 = griddata(xx,yy,Ztmp,xa(ina),ya(ina),'cubic');
            tmp = NaN(size(qx));
            tmp(ina) = tmp2;
            tmp(isnan(tmp)) = 0;
            Z(:,:,j) = tmp;
        else
            Z(:,:,j) = 1e3*(zz-surface_func( xx,yy,flag )); %[um]
        end
    end
    Z(:,:,4) = 0;
    end
    %% end of this function
    % programed by Yasunori Furukawa in Dec.9,2016

    function [ dev,sgma ] = stitch_err_func(W,flag,Q)
    % -------------------------------------------------------------------------
    %% compare the stitch data with the sub-aperture data

    % W         : wavefront at sub-aperture [n1,n2 n]
    % flag    : overlap number [n1,n2]
```

```
% Q          : stitch data [n1,n2]
% ------------------------------------------------------------------------

n = size(W,3);
dev = zeros(size(flag));
Ni = max(max(flag));                    % maximum overlap number

for i = 2:Ni
    ins = flag == i;                                                    %
choose the overlapped area
    for j = 1:n
        tmp = W(:,:,j);
        ing = isfinite(tmp);
        inb = ins & ing;
        dev(inb) = dev(inb)+(tmp(inb)-Q(inb)).^2/i;
    end
end
dev = sqrt(dev);
ins = flag > 1;
sgma = mean(dev(ins));

%% end of this function


function [W,Cset,Csys] = stitching_func(W,Dflag,Nf,Nset,Nsys,Zin,px,py,Nx,Ny,ip_flag)
% ------------------------------------------------------------------
%% stitching
```

```
% W                : wavefront that alignment error and system error are removed
% Cset             : coefficient of alignment error
% Csys             : coefficient of system error

% W                : sub-aperture shape
% Dflag            : 1 = effective,0 = non-effective
% Nf               : data number
% Nset             : alignment error Zernike number
% Nsys             : system error Zernike number
% Zin              : Zernike polynomial [Nky,Nkx,Nf,Zn] [mm]
% px,py            : center position of sub-aperture [Nf,2]
% Nx,Ny : pixel number of sample coordinate
% ----------------------------------------------------------------

DeltaPhi = zeros(Nset,Nf) ;                % DeltaPhi(4,i)
Zij = zeros(Nset,Nset,Nf,Nf) ;             % Zij(4,4,i,j)
DeltaE = zeros(Nsys-Nset,1) ;              % DeltaE(i,j,4)
S = zeros(Nsys-Nset,Nsys-Nset) ;                                        %
S(32,32,i)
Si = zeros(Nset,Nsys-Nset,Nf) ;            % Si(4,32,i)

handle1 = waitbar(0,'Stitching status','Name','Stitching calculation') ;
for i = 1:Nf
    flag1 = false(Ny,Nx) ;
    flag1(py(i,1):py(i,2),px(i,1):px(i,2)) = Dflag(:,:,i) ;

    W_i = zeros(Ny,Nx) ;
```

```
W_i(py(i,1):py(i,2),px(i,1):px(i,2)) = W(:,:,i) ; % substitute sub-aperture data
for j = 1:Nf
    perc = ((i-1)*Nf+j)/Nf^2*100 ;
    pec_str = strcat(num2str(perc,'%5.1f'),' % finished.') ;
    waitbar(perc/100,handle1,pec_str) ;
    if i == j
        for k = 1:Nf
            if i ~= k
                flag2 = false(Ny,Nx) ;
                flag2(py(k,1):py(k,2),px(k,1):px(k,2)) = Dflag(:,:,k) ;
                flag_tmp = and(flag1,flag2) ;
               % figure(1),imagesc(flag_tmp);axis equal
                W_flag = flag_tmp(py(i,1):py(i,2),px(i,1):px(i,2)) ;
                Zij(:,:,i,j) = Zij(:,:,i,j) +
calc_Zii_Dphi_func(Nset,0,W_flag,Zin(:,:,i,:),1) ;
            end
        end
    else
        flag2 = false(Ny,Nx) ;
        flag2(py(j,1):py(j,2),px(j,1):px(j,2)) = Dflag(:,:,j) ;

        W_j = zeros(Ny,Nx) ;
        W_j(py(j,1):py(j,2),px(j,1):px(j,2)) = W(:,:,j) ;

        W_flag = and(flag1,flag2) ;
        flagi = W_flag(py(i,1):py(i,2),px(i,1):px(i,2)) ;
        flagj = W_flag(py(j,1):py(j,2),px(j,1):px(j,2)) ;
```

```
                if max(max(flagi)) > 0
                    dif_tmp = W_j - W_i ;
                    difW = dif_tmp(py(i,1):py(i,2),px(i,1):px(i,2)) ;
                    DeltaPhi(:,i) = DeltaPhi(:,i) +
calc_Zii_Dphi_func(Nset,difW,flagi,Zin(:,:,i,:),2) ;
                    if j > i
                        Zij(:,:,i,j) =
calc_matrix_func(Nset,0,0,flagi,flagj,Zin(:,:,i,:),Zin(:,:,j,:),1) ;
                        Zij(:,:,j,i) = Zij(:,:,i,j).' ;
                    end
                    DeltaE = DeltaE +
calc_matrix_func(Nset,Nsys,difW,flagi,flagj,Zin(:,:,i,:),Zin(:,:,j,:),2) ;
                    S(:,:) = S(:,:) +
calc_matrix_func(Nset,Nsys,0,flagi,flagj,Zin(:,:,i,:),Zin(:,:,j,:),3) ;
                    Si(:,:,i) = Si(:,:,i) +
calc_matrix_func(Nset,Nsys,0,flagi,flagj,Zin(:,:,i,:),Zin(:,:,j,:),4) ;
                end
            end
        end
end
S = -0.5*S ;
DeltaE = 0.5*DeltaE ;
waitbar(1,handle1,'Making the stitching equation.') ;
%%    Stitching equation
baseN = 1 ;
Phi_Mtx = zeros(Nset*(Nf-1)+Nsys-Nset,1) ;
Z_Mtx = zeros(Nset*(Nf-1)+Nsys-Nset) ;
```

```
% Phi
Nt = 1:Nf ;
Nt(baseN) = [] ;
for i = 1:Nf-1
    Phi_Mtx((i-1)*Nset+1:i*Nset) = DeltaPhi(:,Nt(i)) ;
end
Phi_Mtx(Nset*(Nf-1)+1:Nset*(Nf-1)+Nsys-Nset) = DeltaE ;
for i = 1:Nf-1
    for j = 1:Nf-1
        if i == j
            Z_Mtx((i-1)*Nset+1:i*Nset,(j-1)*Nset+1:j*Nset) = Zij(:,:,Nt(i),Nt(j)) ;
        else
            Z_Mtx((i-1)*Nset+1:i*Nset,(j-1)*Nset+1:j*Nset) = -Zij(:,:,Nt(i),Nt(j)) ;
        end
    end
end
for i = 1:Nf-1
    Z_Mtx((i-1)*Nset+1:i*Nset,(Nf-1)*Nset+1:(Nf-1)*Nset+(Nsys-Nset)) = -
Si(:,:,Nt(i)) ;
    Z_Mtx((Nf-1)*Nset+1:(Nf-1)*Nset+(Nsys-Nset),(i-1)*Nset+1:i*Nset) =
Si(:,:,Nt(i)).' ;
end
Z_Mtx((Nf-1)*Nset+1:(Nf-1)*Nset+(Nsys-Nset),(Nf-1)*Nset+1:(Nf-1)*Nset+(Nsys-
Nset)) = S ;
%% solve the matrix problem, and obtain the alignment error and system error
waitbar(1,handle1,'SVD.') ;
% [Usvd,Ssvd,Vsvd] = svd(Z_Mtx) ;
```

```
% svd_eigv_th = max(max(Ssvd)) * 1.0e-10 ;
% Ssvd(Ssvd<svd_eigv_th) = 1.0e21 ;
% Sisvd = 1./diag(Ssvd) ;
% Sisvd(abs(Sisvd)<1.0e-20) = 0 ;
% Sisvd = diag(Sisvd) ;
% CZ_Mtx = Vsvd*Sisvd*Usvd'*Phi_Mtx ;
% difg = g(2:end)-g(1:end-1);
[U,S,V] = svd(Z_Mtx,0);
g = log10(diag(S));
figure,plot(g,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',10);grid
dif_g = g(1:end-1)-g(2:end);
dif_g(end-3:end) = 0;
%                                                                    figure,plot(dif_g,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',10);grid
[ ~,pos ] = max(dif_g);

% if ip_flag == 0
%         limN = 1e-10;       % •âŠÔ,È,µ,Ì,Æ,«
%         % num = rank(Z_Mtx);
% else
%     %    num = rank(Z_Mtx);
%         limN = 1e-4;      % 10^(-1.8);%•âŠÔ, ,è,Ì,Æ,«
% end
% pos = 48;%
for j = 1:size(S,1)
      if j > pos
```

```
        %if abs(S(j,j)) < limN
            S(j,j) = 0;
        else
            S(j,j) = 1/S(j,j);
        end
    end
    CZ_Mtx = V*S*U'*Phi_Mtx;
    %%
    Cset = -CZ_Mtx(1:(Nf-1)*Nset)' ;                    % coefficient of alignment
    error [um]
    Csys = -CZ_Mtx((Nf-1)*Nset+1:(Nf-1)*Nset+Nsys-Nset)' ; % coefficient of system
    error[um]
    waitbar(1,handle1,'Removing the system error and setting error.') ;
    %% alignment error removal
    for i = 1:Nf-1
        for j = 1:Nset
            W(:,:,Nt(i)) = W(:,:,Nt(i))-Cset((i-1)*Nset+j)*Zin(:,:,Nt(i),j);
        end
    end
    %% sysetm error removal
    for i = 1:Nf
        sys_err = zeros(size(Zin(:,:,1,1))) ;
        for j = 1:Nsys-Nset
            sys_err = sys_err + Csys(j)*Zin(:,:,i,j+Nset) ;
        end
        W(:,:,i) = W(:,:,i) - sys_err ;
        W(:,:,i) = W(:,:,i).*Dflag(:,:,i) ;
```

```
end
close(handle1) ;
%% end of this function
% programed by Yasunori Furukawa Dec. 10, 2016
function [ Q ] = calc_Zii_Dphi_func(Nset,dif_W,W_flag,Zin,flag)
% ----------------------------------------------------------------


% Q                 : flag:1 = Zii,2 = DeltaPhi


% Nset            : number of alignment error
% dif_W          :
% W_flag        : [Ny,Nx]
% Zin              : Zernike polynomial [Nky,Nkx,1,Nset]
% flag            : 1 = Zii,2 = DeltaPhi


% ----------------------------------------------------------------
cut = find(W_flag == true) ;% obtain overlapped data
%%
Zi =    zeros(size(cut,1),Nset) ;
for i = 1:Nset
      tmp = Zin(:,:,i);
      Zi(:,i) = tmp(cut);
end
if flag == 1               % Zii
      Q = Zi'*Zi;
elseif flag == 2        % DeltaPhi
      tmp = dif_W(cut);
```

```
    tmp(isnan(tmp)) = 0;
    Q = Zi'*tmp;
end
function [ Q ] = calc_matrix_func(Nset,Nsys,dif_W,flagi,flagj,Zini,Zinj,flag)
% ------------------------------------------------------------
% Q                    : flag : 1 = Zij,2 = E, 3 = S, 4 = Si

% Nset                 : alignment error number
% Nsys                 : system error number
% dif_W                : [Nky Nkx]
% flagi                : [Nky Nkx]
% flagj                : [Nky Nkx]
% Zini                 : [Nky Nkx 1 NN]
% Zinj                 : [Nky Nkx 1 NN]
% flag                 : 1 = Zij,2 = E, 3 = S, 4 = Si

% ------------------------------------------------------------

if flag == 1
    NN = Nset;
else
    NN = Nsys;
end
[n1,n2] = size(Zini(:,:,1));
tmp = reshape(Zini(:,:,1,1:NN),n1*n2,NN);
Zi = tmp(flagi,:) ;
tmp = reshape(Zinj(:,:,1,1:NN),n1*n2,NN);
```

```
Zj = tmp(flagj,:) ;
%% calculate matrix
if flag == 1 % Zij
    Q = Zi'*Zj;
elseif flag == 2 % E
    tmp = dif_W(flagi);
    tmp(isnan(tmp)) = 0;
    Q = (Zj(:,Nset+1:Nsys)-Zi(:,Nset+1:Nsys))'*tmp;
elseif flag == 3 % S
    Q = (Zj(:,Nset+1:Nsys) - Zi(:,Nset+1:Nsys))'*(Zj(:,Nset+1:Nsys) -
Zi(:,Nset+1:Nsys));
elseif flag == 4 % Si
    Q = Zi(:,1:Nset)'*(Zj(:,Nset+1:Nsys) - Zi(:,Nset+1:Nsys));
end

function [ f ] = surface_func( x,y,flag )
% -------------------------------------------------------------------
%% Output the test surface
% f                    : test surface [mm]

% x,y                  : coordinate [mm]
% flag                 : 0 = Alvarez lens
% -------------------------------------------------------------------
if flag == 0
    A = 0.0012;
    f = A*(1/3*x.^3+x.*y.^2);
end
```

```
end
%% end of this function
% programed by Yasunori Furukawa    Dec.5,2016


function [ qx,qy,qz ] = trans_func( mxyz,th,flag,x,y,z )
% -----------------------------------------------------
%% transform the coordinate
% qx,qy,qz           : coordinate after transform [mm]


% mxyz                 : amount of shift (x,y,z) [mm]
% th                   : amount of rotation (thx, thy) [rad]
% flag                 : 1 = transform from local sample coord to CCD coord
%                      : -1 = transform from CCD coord to local sample coord
% x,y,z                : coordinate before transform [mm]
% -----------------------------------------------------


if flag == 1
                        [ x,y,z ] = Shift_samp_func(-mxyz,x,y,z);              % XYZ shift
                        [ x,y,z ] = Rot_samp_func(-th(1),0,x,y,z);         % theta x rotation
    [ qx,qy,qz ] = Rot_samp_func(-th(2),1,x,y,z);         % theta y rotation
elseif flag == -1
    [ x,y,z ] = Rot_samp_func(th(2),1,x,y,z);             % theta y rotation
    [ x,y,z ] = Rot_samp_func(th(1),0,x,y,z);       % theta x rotation
    [ qx,qy,qz ] = Shift_samp_func(mxyz,x,y,z);         % XYZ shift
end
% if flag == 1
%                       [ x,y,z ] = Shift_samp_func(mxyz,x,y,z);              % XYZ shift
```

```
%                           [ x,y,z ] = Rot_samp_func(th(1),0,x,y,z);          % theta x rotation
%          [ qx,qy,qz ] = Rot_samp_func(th(2),1,x,y,z);          % theta y rotation
% elseif flag == -1
%          [ x,y,z ] = Rot_samp_func(-th(2),1,x,y,z);              % theta y rotation
%          [ x,y,z ] = Rot_samp_func(-th(1),0,x,y,z);        % theta x rotation
%          [ qx,qy,qz ] = Shift_samp_func(-mxyz,x,y,z);          % XYZ shift
% elseif flag == 0
%          [ x,y,z ] = Shift_samp_func(-mxyz,x,y,z);
%          [ x,y,z ] = Rot_samp_func(-th(1),0,x,y,z);
%          [ qx,qy,qz ] = Rot_samp_func(-th(2),1,x,y,z);
% end
end
function [xo,yo,zo] = Shift_samp_func(mxyz,x,y,z)
% ---------------------------------------------------------------------
%%    Shift the sample
% xo,yo,zo    : coordinate after shifting [mm]

% mxyz            : amount of shift (x,y,z) [mm]
% x,y,z          : coordinate before shifting [mm]
% ---------------------------------------------------------------------

xo = x + mxyz(1) ;
yo = y + mxyz(2) ;
zo = z + mxyz(3) ;
end
function [ xo,yo,zo ] = Rot_samp_func(th,flag,x,y,z)
% ---------------------------------------------------------------------
```

```
%%    Rotate the sample thetax or thetay
% xo,yo,zo    : coordinate after rotation [mm]

% th               : Rotation angle x,y [rad]
% flag           : 0 = theta x, 1 = theta y
% x,y,z          : coordinate before rotation [mm]
% ------------------------------------------------------------------------

if flag == 0
      xo = cos(th)*x-sin(th)*z;
      yo = y;
      zo = sin(th)*x+cos(th)*z;
elseif flag == 1
      xo = x;
      yo = cos(th)*y-sin(th)*z;
      zo = sin(th)*y+cos(th)*z;
end
end
%% end of this function
% programed by Yasunori Furukawa in Dec.8,2016
```