Binary pattern deflectometry

Guillaume P. Butel,* Greg A. Smith, and James H. Burge

College of Optical Sciences, University of Arizona, 1630 E. Univ. Blvd., Tucson, Arizona 85721, USA

*Corresponding author: butel.guillaume@gmail.com

Received 12 November 2013; revised 3 January 2014; accepted 4 January 2014;
posted 10 January 2014 (Doc. ID 201095); published 6 February 2014

Deflectometry is widely used to accurately calculate the slopes of any specular reflective surface, ranging from car bodies to nanometer-level mirrors. This paper presents a new deflectometry technique using binary patterns of increasing frequency to retrieve the surface slopes. Binary Pattern Deflectometry allows almost instant, simple, and accurate slope retrieval, which is required for applications using mobile devices. The paper details the theory of this deflectometry method and the challenges of its implementation. Furthermore, the binary pattern method can also be combined with a classic phase-shifting method to eliminate the need of a complex unwrapping algorithm and retrieve the absolute phase, especially in cases like segmented optics, where spatial algorithms have difficulties. Finally, whether it is used as a stand-alone or combined with phase-shifting, the binary patterns can, within seconds, calculate the slopes of any specular reflective surface.

OCIS codes: (120.3940) Metrology; (120.5050) Phase measurement; (120.3930) Metrological instrumentation; (120.6650) Surface measurements, figure.

http://dx.doi.org/10.1364/AO.53.000923

1. Introduction

Deflectometry is used in a wide range of metrology applications, such as: plastic lens and car body inspection with sub-micrometer level accuracy [1], aspheric optic metrology to nanometer-level accuracy [2,3], and even measurements accurate to fractions of a nanometer for x-ray mirrors [4]. A deflectometry apparatus consists of digital display and camera, which are used to measure the slopes of a specular reflective surface by triangulation [1–9]. Phase-shifting the displayed pattern is commonly used to reach sub-pixel level accuracy and fast data acquisition [5]. To unwrap the phase, it is common practice to use spatial phase unwrapping techniques [10]. Spatial phase unwrapping uses neighboring pixels to reconstruct the true phase from the wrapped phase [11–15]; however, computational power (desktop or laptop) is needed because the algorithms are complex. Furthermore, noise sensitivity [10] and phase discontinuities make spatial phase unwrapping even more difficult. However, temporal phase unwrapping techniques can also be used in deflectometry. Temporal phase unwrapping algorithms use a series of pictures spaced in time to locate the $[-\pi, \pi]$ zones and apply the correct offset [16–25]. Various patterns are used, such as: binary patterns (or gray-code) [20–25] (common in 3D shape measurement [16–25]), color codes [18], or phase markers [17].

In this paper, we show that temporal phase unwrapping in deflectometry using binary patterns is advantageous for portable devices because it allows us to work around aspects out of our control such as: low computational power, automatic gain control, boundary errors, and matrix multiplication. As a stand-alone method, binary pattern deflectometry works very well on portable devices. However, we expand the binary technique and show that we can improve the measurement's accuracy by combining binary patterns with sinusoidal patterns and still retrieve the phase and surface slopes within seconds. An overview of this hybrid phase-shifting method and experimental results are presented in Section 3.

1559-128X/14/050923-08$15.00/0

2. Binary Patterns Theory and Implementation for Deflectometry

The binary pattern deflectometry technique displays on the screen a sequence of increasing frequency binary patterns, allowing us to find which pixel on the screen illuminates a given pixel on the mirror. A simple, yet powerful idea is to illuminate the screen pixel by pixel, then determine which mirror pixel is associated with the screen pixel being lit. This is, in essence, the original idea of a fast and low-cost deflectometry measurement proposed by Su *et al.* in [5]. However, this process is very slow because of the large number of pictures required. A much faster solution uses binary code.

The principle of this method is to create a unique binary code for each individual pixel on the mirror that uniquely registers it to a screen pixel location. The idea of uniquely encoding an object using phase is widely described in many domains and applications like security encryption [26]. If we were lighting several pixels on the display screen at the same time, several mirror pixels would have the same code or signature. We call this ambiguity. The goal of this method is to avoid any ambiguous pixels so that we do not need to unwrap the phase. This method achieves a registration between the location of a given mirror pixel and a location on the screen, by associating a sequence of binary patterns, that isolates a unique screen region. As the frequency increases, the size of the screen region decreases to reach, at the highest frequency, a unique pixel-to-pixel correspondence.

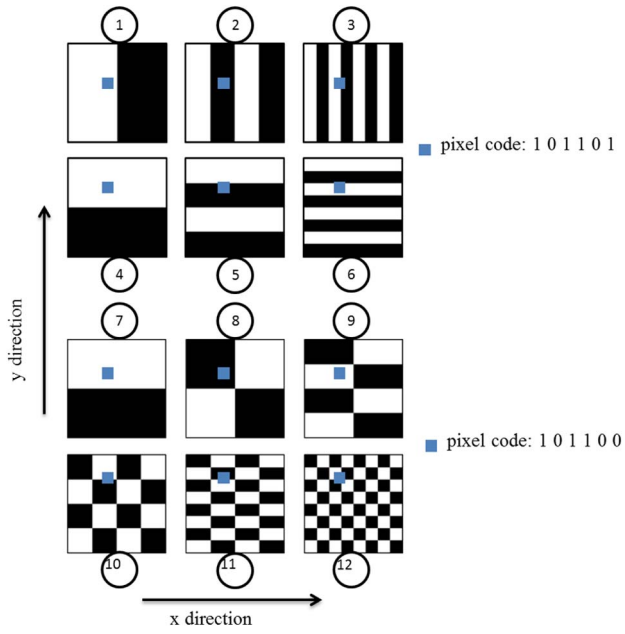

Fig. 1. Binary code pattern examples. The patterns Nos. 1–6 above represent the 1D binary code; the patterns Nos. 7–12 represent the 2D binary code. The 1D patterns are independent in x and y. The 2D patterns start with splitting the screen in two (No. 7) and then alternates between x and y frequency in increasing steps. The total number of patterns depends on the number of pixels used.

The patterns used can be affiliated to the Posdamer and Altschuler [27] nomenclature, as expressed in [28] and seen in Fig. 1. Two methods for creating the binary pattern are presented as one-dimensional and two-dimensional arrays, as seen in Fig. 1. The one-dimensional method is based on increasing frequency lines in both x and y directions. The two-dimensional method is based on increasing frequency squares; the frequency increase alternates between the x and y direction. Both methods require an equal number of screens to solve for the phase ambiguity. The main difference resides in how to combine the various patterns once reflected off the object surface.

This method is faster than the initial idea of illuminated pixels one by one. In the general case, the area used on the monitor has N^2 pixels. The initial idea requires the illumination of those pixels one by one and thus having to capture N^2 pictures. The binary code method only requires a total of $m = 2\,\text{ceil}[\log_2(N)]$ pictures (the ceil function rounds the number up, to the nearest integer). As an example, using $N = 100$, we need 10,000 pictures if taken one by one, but only 14 using the binary code.

A. One-Dimensional Binary Code

The one-dimensional binary code has the easiest reconstruction process of both cases, as the x and y directions are calculated separately. In the following, the nomenclature $()_2$ and $()_{10}$ designates an integer in base 2 (binary) and base 10 (decimal), respectively. If we only consider the x direction (Nos. 1–3 in Fig. 1), then we have pixels with binary codes from $(000)_2$ to $(111)_2$, which gives 8 possibilities for the screen pixel location in x and y. The screen location is obtained by converting the binary code into a decimal number, giving numbers from $(0)_{10}$ to $(7)_{10}$. The same process applies to the y direction. In the example of Fig. 2, we obtain that the point (x_1, y_1) on the mirror is associated with $x_{\text{screen}}(x_1, y_1) = P_1 =$ pixel No. 5. Similarly, (x_2, y_2) is associated to $x_{\text{screen}}(x_1, y_1) = P_2 =$ pixel No. 3. This is true for any given mirror pixel (x_j, y_j).

A given mirror pixel has a unique binary number, made with n digits, each digit associated with the

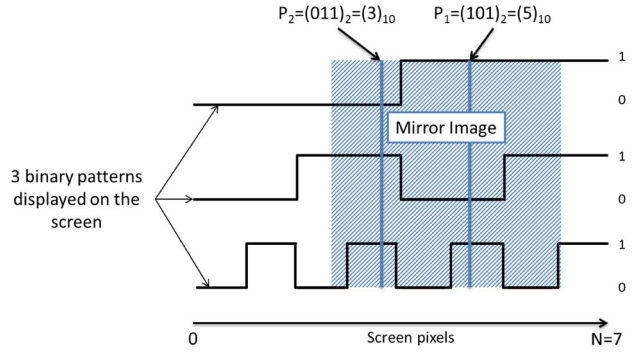

Fig. 2. Retrieving the pixel screen location using the 1D binary code patterns. Any point on the mirror has a binary code that gives its location in the screen space. The nomenclature $()_2$ and $()_{10}$ designates an integer in base 2 (binary) and base 10 (decimal), respectively.

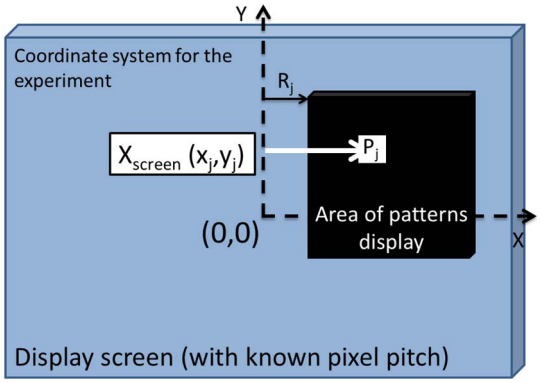

Fig. 3. Screen coordinates calculation in the 1D case. The points P_j are located on the area of patterns display thanks to their binary code and we calculate x_{screen} using R_j and the pixel pitch.

value obtained in the picture (0 or 1). The conversion of this binary number into a decimal number will give the screen location $x_{\text{screen}}(x_j, y_j)$ in unit of pixels. To get the real distance $x_{\text{screen}}(x_j, y_j)$ in meters, the screen has to be located in the coordinate system of the mirror and the camera, and the pixels converted to meters using the pixel pitch. Equation (1) shows the calculation:

$$x_{\text{screen}}(x_j, y_j) = [(P_j)_{10} + R_j]\text{pitch},$$
$$\forall\ (x_j, y_j) \in \text{Mirror}, \tag{1}$$

where R_j is the offset in pixels from the y axis to the edge of the area of patterns display (Fig. 3) and "pitch" the pixel pitch. A similar formula exists for $y_{\text{screen}}(x_j, y_j)$, using T_j for the offset from the x axis.

B. Two-Dimensional Binary Code

For the 2D pattern, we do not treat each direction independently. The sequence of patterns shown in Fig. 1 (Nos. 7–12) could be arranged differently, yet we would still follow the same path as the one we will present now. There are N^2 pixels used on our monitor and $m = 2n = 2\,\text{ceil}[\log_2(N)]$ pictures to be taken. To design the patterns, the number of lighted squares (or rectangles) has to be calculated. The numbers of squares in $x(Nx)$ and $y(Ny)$ vary whether the pattern index i is odd or even, as shown in Eq. (2):

$$\text{if } i \text{ odd } \begin{cases} Nx = 2^{\text{floor}(i/2)} \\ Ny = 2^{\text{floor}(i/2)+1} \end{cases}$$
$$\text{if } i \text{ even } \begin{cases} Nx = 2^{\text{floor}(i/2)} \\ Ny = Nx \end{cases}, \tag{2}$$

where the floor function rounds a number down to the nearest integer. We only have squares when i is even, because $Ny = Nx$. For an odd i, the x frequency is double the y frequency because $Ny = 2Nx$.

When we display those patterns onto the monitor, the camera will see their reflection from the mirror and, thus, each mirror pixel will have a sequence of 0 and 1 that registers it to a unique screen pixel. Using matrix multiplication, we can isolate the screen pixel for each mirror pixel and have a map of the screen locations, as shown in Eq. (3).

$$S_{x_0,y_0} = \prod_{i=1}^{m} [U - M^i]^{1-\alpha_i} [M^i]^{\alpha_i}, \tag{3}$$

where S_{x_0,y_0} is the final matrix we obtained after m iterations for the mirror pixel (x_0, y_0), M^i the N^2 matrix representing the pattern displayed on the screen at the ith step of the iteration, α_i the value of the mirror pixel in the ith picture (0 or 1), and U the unity N^2 matrix filled with 1. If α_i is 1, then the matrix M^i used is the same as the one displayed on the screen; however, if α_i is 0, we mirror the matrix using the unity matrix U to turn 0 into 1 and vice-versa. We multiply successively the screen patterns that were displayed when the pictures were taken to be able to associate a mirror pixel (x_0, y_0) to a screen location. The result we obtain for S_{x_0,y_0} is a matrix with a single white (value 1) pixel. This final matrix represents the unique screen pixel that illuminated the mirror pixel when the data was acquired. Therefore, we have a mirror pixel (x_0, y_0) associated with one screen pixel, whose coordinates can be retrieved by reading the coordinates of the white pixel in S_{x_0,y_0}.

The single white pixel in Figs. 4 and 5 has the x_{screen} and y_{screen} information in unit of display pixels

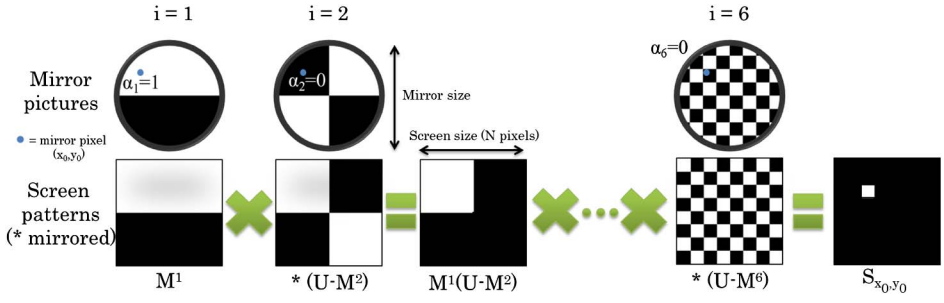

Fig. 4. Example of pixel-to-pixel multiplication used to sequentially isolate a screen pixel (a white pixel means 1 and a black pixel 0). The top row represents the pictures taken by the camera, the bottom row the matrix multiplication process. In this example, we consider a mirror pixel (x_0, y_0). Based on the first picture, the value at (x_0, y_0) is 1, so the matrix used in the product defined in Eq. (3) is M^1. Using the second picture ($i = 2$), $\alpha_2 = 0$ so the matrix used is $(U - M^2)$. The intermediate product is, thus, $M^1(U - M^2)$. The same process goes until we reach $i = m$ ($m = 6$ here).

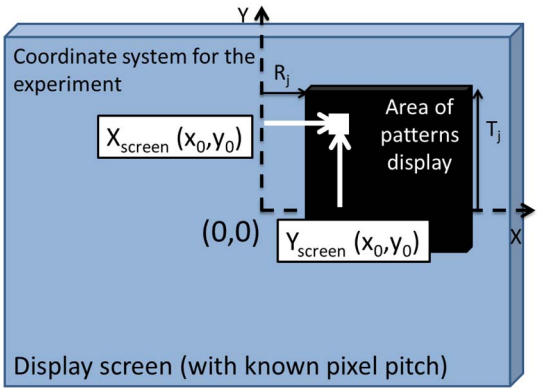

Fig. 5. Screen coordinates calculation in the 2D case. The matrix multiplication isolates a single pixel that gives the coordinates x and y for the mirror pixel (x_0, y_0). $x_{\text{screen}}(x_j, y_j)$ and $y_{\text{screen}}(x_j, y_j)$ are obtained using Eq. (1) that changes the coordinates of the pixel location from the original matrix S_{x_0,y_0} to a real space coordinate system.

for a single mirror pixel (x_0, y_0). This process has to be repeated for every mirror pixel in the pictures taken by the camera. Similar to the 1D case, a change of coordinate system and unit conversion have to be performed to be able to obtain the $x_{\text{screen}}(x_j, y_j)$ and $y_{\text{screen}}(x_j, y_j)$ in meters for any mirror pixel (x_j, y_j), using Eq. (1).

The calculation process is found to be a bit slower with the 2D algorithm (from 20 s to a minute), and the matrix multiplication is more computationally intensive. The 1D algorithm is almost instantaneous, usually less than a second.

C. Implementation of the Binary Patterns Method

When used in a real system, the binary patterns are strongly affected by the camera's modulation transfer function (MTF). The MTF alters the high frequency in a signal; however, turning the perfect binary shapes into round-edged shapes (Fig. 6) and setting up an efficient threshold is more difficult. With the wide variety of cameras and monitors available to acquire the data, other problems can occur, like nonlinearity and saturation. The threshold thus needs to be adjusted very carefully. This is a rather difficult process, especially if the algorithm has to be automated without the user's intervention. To prevent this pitfall, a better algorithm, called "shifted binary patterns," is introduced. It requires more pictures to be taken, but it is more robust against the errors at the binary boundaries.

The algorithm creates an undetermined, or intermediate region, separated by two threshold values, as shown in Fig. 7. The pictures then become separated into three regions, with the pixels in the intermediate region to be determined. The minimum "Min" is obtained by averaging a picture illuminated by a "black" display and the maximum "Max" by averaging a picture illuminated by a "white" display.

To achieve this goal, we use a different set of images that depict the same binary patterns shifted by half a period: the period being the size of a binary structure in the highest frequency binary pattern. Figure 8 shows the shifted binary pattern, as well as a profile along the y axis. A similar idea has been presented by Zhang *et al.* [24], but they used a higher frequency complementary binary pattern. Here, we are using the same frequency pattern, but shifted in space. Their technique is developed for profilometry and seems to work well in that case. However, when applying their algorithm for our deflectometry purposes, some phase errors were found at the boundaries of the binary patterns.

The pixels that belong to the intermediate region for the first set of images will not be undetermined for the shifted set of images and can, thus, be used as data in the slopes maps. Two slopes maps, one per binary patterns set, have to be computed. The second map, generated from the shifted patterns, will be used to fill the gaps of the main one. The shift offset has to be taken into account when calculating the second map to obtain the real values. As shown in Fig. 9, any white pixels in the left map will be using the value of the corresponding pixel in the middle map. The result is the map on the right of Fig. 4, where all the white pixels have disappeared but one, certainly due to noise. This map represents the $x_{\text{screen}}(x, y)$ values for any mirror pixel (x, y). The slopes of the mirror can then be calculated using the approximation

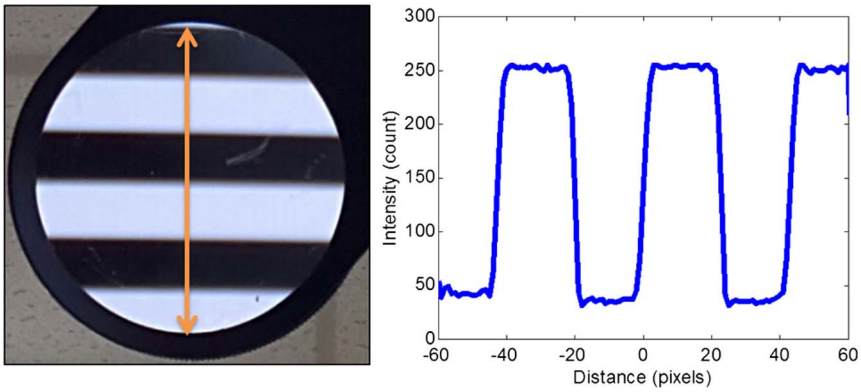

Fig. 6. Real data from a spherical mirror. The picture (left) clearly shows the different zones. The y-profile (right) displays a cross section of the picture.

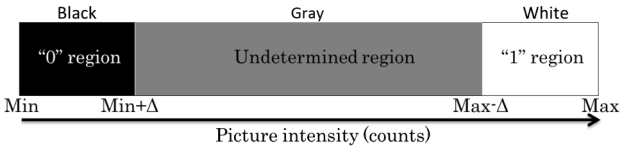

Fig. 7. Two thresholds now create an undetermined region between the 0 and 1 regions, with $\Delta = ((\text{Max} - \text{Min})/5)$.

$$X\text{Slopes}(x,y) = \frac{1}{2z_s}(x_{\text{mirror}}(x,y) - x_{\text{screen}}(x,y)) + \frac{1}{2z_c}(x_{\text{mirror}}(x,y) - x_{\text{camera}}(x,y)), \quad (4)$$

where z_s and z_c are the distances from the mirror to the display screen and to the camera, respectively (similar for $Y\text{Slopes}(x,y)$). Equation (4) assumes the sag is much smaller than the distance between the mirror and the screen, and that the viewing angles between camera, mirror, and screen are small.

If the intermediate region is bigger than the size of the shift, then the process breaks down. This happens when using patterns with frequencies not far from the cutoff frequency and for noisy environments. In most cases, the histogram of a picture will show that only a few numbers of pixels will belong to the intermediate region and, thus, this region, even though wide in intensity counts (Fig. 7), is narrow in actual pixels, and much smaller than the size of a white/black stripe.

This method is extremely fast (less than a second) because the slopes are directly obtained from the pictures. Their accuracy is built up bit-by-bit, by adding higher frequency binary patterns. The total time needed to obtain the slopes maps is the time needed to take the pictures; processing the data is instantaneous. Furthermore, there are no ambiguities in the maps, because of the unique mapping performed by the binary patterns.

D. Limitations of a Binary Code Method

As demonstrated above, it is possible to obtain slopes maps using purely binary patterns. However, two main limitations to this method remain. The first one is the mapping accuracy. The algorithm cannot map a mirror pixel to an area on the display smaller than a screen pixel, as the calculated location cannot be smaller than a pixel. At the theoretical limit, we have a one-to-one mapping, but we cannot do better than 1 pixel in the screen space.

The falling contrast that is experienced in the pictures taken by the camera represents the second limitation. As seen in Fig. 10, the theoretical limit based on screen resolution, is for pattern No. 14; however, the contrast has already fallen to 0 since pattern number 11. The first limitation thus becomes irrelevant due to the MTF of the system, and the method is therefore limited by the MTF of the camera.

To improve accuracy to about $1/100^{\text{th}}$ pixel, we can use the binary code together with phase-shifting, as phase-shifting can reach sub-pixel accuracy [29]. This process of combining binary code and phase-shifting is presented in the following section.

3. Direct Application: Phase Unwrapping Elimination

A. Implementation of the Hybrid Method

Combining binary patterns and phase-shifting is commonly used in profilometry [20–25], and was originally introduced by Bergmann [23]. In deflectometry, this combining technique has not been used as an unwrapping technique because the fringes and the binary patterns are not in focus. The critical aspect of such a technique is matching the boundaries of the binary patterns to end points of a sine wave period.

In Fig. 10, each pattern index (on the x axis) is related to a frequency in x and y directions that can be calculated at every step. The binary code allows us to locate small areas on the screen, rather than individual pixels. The phase-shifting can determine a wrapped (and incomplete) phase value for every pixel. In the following, we show that the unwrapping ambiguity can be overcome using the binary method. The binary code provides an offset map K at a given frequency. K is calculated by binary code conversion or matrix multiplication depending on the binary code patterns. K only contains integer values ranging between 0 and p-1, where p being the total number of sine wave periods. It can be summarized with the formula

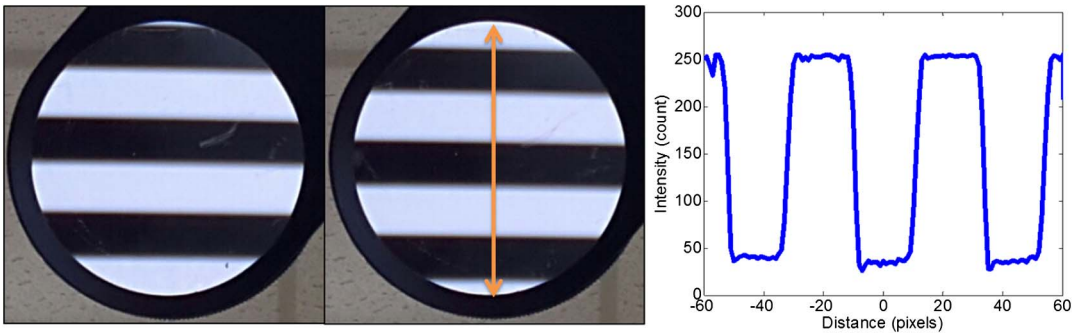

Fig. 8. Shifted binary patterns. The shifted mirror picture (middle) depicts a shift of half the size of a white stripe compared with mirror picture from Fig. 6 (left, for reference). The profile (right) shows a cross section of the shifted patterns picture.

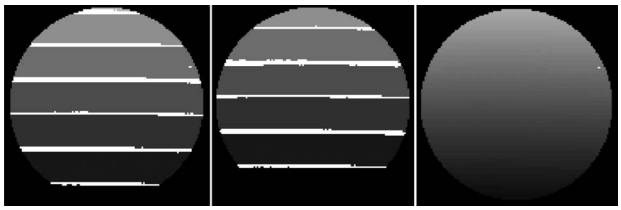

Fig. 9. Mirror maps representing the screen coordinates for standard (left), shifted (middle) patterns, and final map (right). The white lines show the intermediate regions.

$$\Phi_{\text{true}}(x,y) = \Phi_{\text{wrapped}}(x,y) + 2K(x,y)\pi \quad \forall\ (x,y) \in \text{Mirror}. \tag{5}$$

To correctly use the binary code information, the binary patterns are aligned in space with the sine waves, as shown below in Fig. 11. The frequency of the sine wave is determined by the desired contrast.

As stated in Section 2, two methods can be used to obtain K, one-dimensional or two-dimensional. Figure 4 shows the one-dimensional procedure. The only difference from Section 2 is that we stop the binary process retrieval at the frequency/contrast value chosen from Fig. 10 to give nonzero contrast. Since the sine wave locations match the offset zones in space (Fig. 11), the obtained phase offsets will be correct. Once we know K, we can recover the true phase Φ_{true}, for any point on the mirror using Eq. (5). The screen location can be calculated as a scaled version of the phase, as shown in Eq. (6):

$$x_{\text{screen}}(x,y) = \frac{\Phi^{x}_{\text{true}}(x,y)}{2p\pi}N, \tag{6}$$

where Φ^{x}_{true} is the phase for the x direction, N is the total number of screen pixels, and p the total number of sine periods used (similar equation for $y_{\text{screen}}(x,y)$). Equation (2) is then used to calculate the slopes of the mirror.

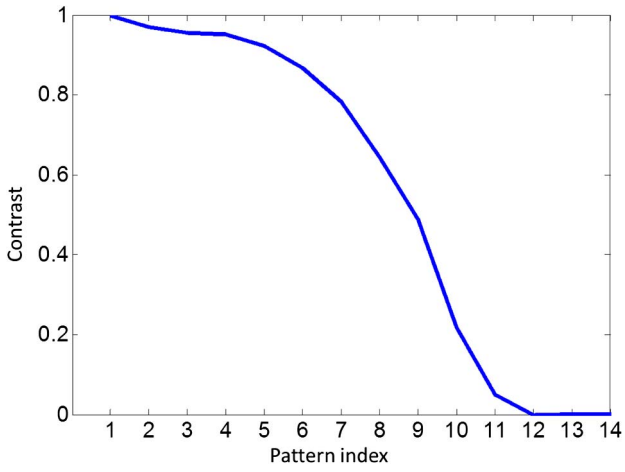

Fig. 10. Contrast obtained in the pictures taken by the camera.

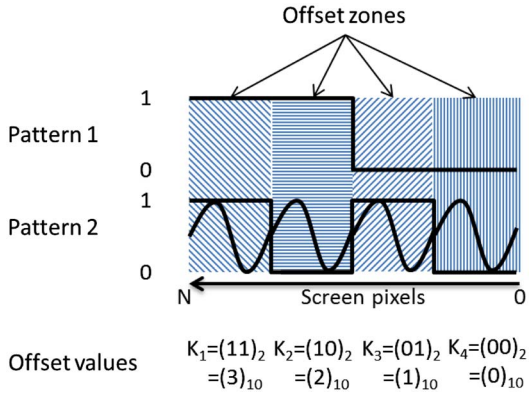

Fig. 11. Matching the square wave to the sine wave gives the correct offsets in decimal. Multiplying the offset values by 2π and adding them to the wrapped phase will give the true phase. In this example, K has 4 different values, $(0)_{10}$, $(1)_{10}$, $(2)_{10}$, and $(3)_{10}$.

B. Experimental Results

We tested a spherical mirror using binary code combined with phase-shifting and compared the results with using phase-shifting alone. The mirror was masked to show how algorithms can unwrap the phase in the case of segmented optics, where the mirror is only composed of "islands" of data. The results show the phase maps in x and y directions. The raw data used in both cases is the same, but the unwrapping algorithm is different: temporal (binary) or spatial [15]. The 2D binary algorithm has been used for the mirror under test.

The mirror is spherical, has a radius of curvature of 203.2 mm, and its f number is 2. The experiment was performed with a laptop camera and a laptop monitor (Fig. 12), placed at a distance of 220 mm. 8-period sine waves were used for the phase-shifting patterns. The displayed phase, therefore, goes from 0 to 16π rad. The true phase seen on the mirror can go from 0 to 16π rad but its range is usually smaller, as the mirror magnifies the object.

Figure 13 shows the phase maps obtained from the experiment. Map (a) shows the difference between the full and segmented mirrors, both spatially

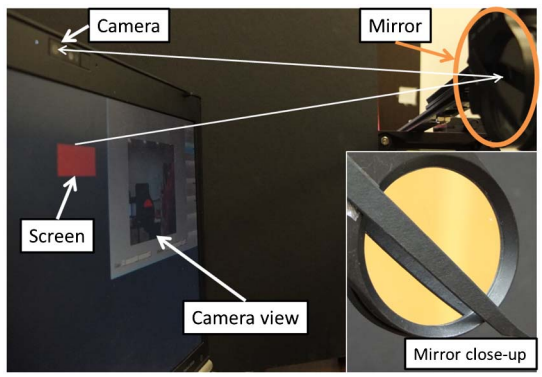

Fig. 12. Experimental setup. The patterns are displayed on the screen (left) and reflected off the mirror (right). Part of the mirror was masked to simulate a segmented mirror (close-up). The camera (left) takes pictures.

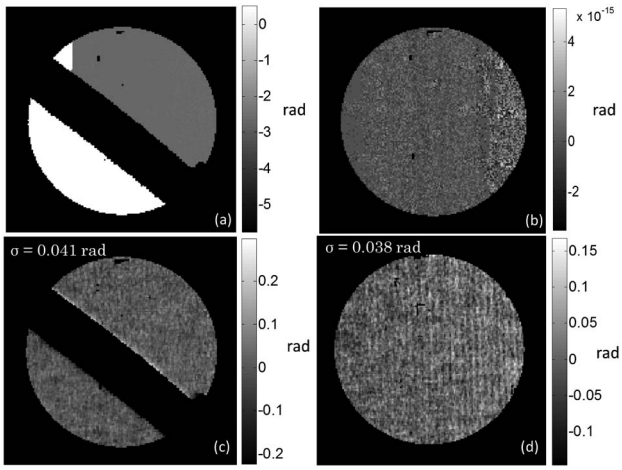

Fig. 13. Comparison of the phase maps differences. Map (a) depicts the difference between the full and segmented mirror (spatially unwrapped). Map (b) shows the phase difference between a spatial and temporal unwrapping (full mirror), which appears to be noise. Map (c) depicts the difference between the full and segmented mirror (temporally unwrapped). Map (d) shows the average of two maps temporally unwrapped. All numbers are in radians.

unwrapped. The difference is not small and we can clearly see 2 zones fairly constant, one at −2 rad (gray region) and the other at +0.2 rad (white region). Map (b) shows the difference between the two unwrapping algorithms for the full mirror. The result is on the order of MATLAB's noise, and we can, therefore, conclude that the two algorithms are equivalent for a full mirror. Using (a) and (b), we deduce that the particular spatial algorithm does not perform well in case of a segmented mirror, because this spatial algorithm cannot find any phase relationship between the two zones using neighboring pixels to correctly unwrap the phase. Map (c) shows the same difference as Map (a), but using temporal unwrapping this time. The difference standard deviation of 0.041 rad validates the binary pattern unwrapping. To further validate this result, Map (d) shows the difference of two measurements, using the temporal unwrapping on a full mirror. The obtained standard deviation is 0.038 rad. The standard deviations obtained in Maps (c) and (d) are on the same order of magnitude, therefore, the segmented mirror does not affect the result. In closing, the binary pattern algorithm works as well as the spatial unwrapping algorithm [15] for a full mirror, and can correctly unwrap the phase of a segmented mirror.

4. Conclusion

We present in this paper a new deflectometry method that can determine the slopes of an optic without any ambiguities, and much faster than the initial concept of pixels being illuminated one by one. This method uses a sequence of binary patterns, increasing in spatial frequency, to create as many unique binary codes as needed to solve for the screen pixel locations. Calculated screen locations are used in triangulation to calculate surface slopes. Two different cases can be derived from this method: a one-dimensional case and a two-dimensional case. This method is fast, almost instantaneous, but its accuracy is limited by the MTF of the system.

We showed it is possible use the binary method combined with the phase shifting algorithms with high accuracy. We performed an experiment proving the phase can be reconstructed without traditional spatial phase unwrapping algorithms. The binary method does not affect the resolution of the phase-shifting; Map (b) has an average of 10^{-15} rad. In the case of segmented optics, the temporal algorithm using binary patterns correctly unwraps the phase, which a spatial algorithm fails to accomplish. In our experiment, we found a difference standard deviation of 0.041 rad for the segmented optic, very close to our noise level, 0.038 rad. The spatial unwrapping could not unwrap the phase and left two distinct zones in the unwrapped phase map. The binary temporal unwrapping can be used instead of spatial unwrapping in all cases.

References

1. T. Bothe, W. Li, C. von Kopylow, and W. Jüptner, "High-resolution 3D shape measurement on specular surfaces by fringe reflection," Proc. SPIE **5457**, 411–422 (2004).
2. P. Su, S. Wang, M. Khreishi, Y. Wang, T. Su, P. Zhou, R. E. Parks, K. Law, M. Rascon, T. Zobrist, H. Martin, and J. H. Burge, "SCOTS: a reverse Hartmann test with high dynamic range for Giant Magellan Telescope primary mirror segments," Proc. SPIE **8450**, 84500W (2012).
3. P. Su, M. Khreishi, R. Huang, T. Su, and J. H. Burge, "Precision aspheric optics testing with SCOTS: a deflectometry approach," Proc. SPIE **8788**, 87881E (2013).
4. R. Huang, P. Su, T. Horne, G. Brusa Zappellini, and J. H. Burge, "Measurement of a large deformable aspherical mirror using SCOTS (Software Configurable Optical Test System)," Proc. SPIE **8838**, 883807 (2013).
5. P. Su, R. E. Parks, L. Wang, R. P. Angel, and J. H. Burge, "Software configurable optical test system: a computerized reverse Hartmann test," Appl. Opt. **49**, 4404–4412 (2010).
6. 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).
7. D. Pérard and J. Beyerer, "Three-dimensional measurement of specular free-form surfaces with a structured-lighting reflection technique," Proc. SPIE **3204**, 74–80 (1997).
8. Y. Tang, X. Su, Y. Liu, and H. Jing, "3D shape measurement of the aspheric mirror by advanced phase measuring deflectometry," Opt. Express **16**, 15090–15096 (2008).
9. W. Jüptner and T. Bothe, "Sub-nanometer resolution for the inspection of reflective surfaces using white light," Proc. SPIE **7405**, 740502 (2009).
10. J. C. Wyant, Phase-Shifting Interferometry [PDF document]. Retrieved from: http://fp.optics.arizona.edu/jcwyant/Optics513/optics513ChapterNotes.htm (2011).
11. T. R. Judge and P. J. Bryanston-Cross, "A review of phase unwrapping techniques in fringe analysis," Opt. Lasers Eng. **21**, 199–239 (1994).
12. X. Su and W. Chen, "Reliability-guided phase unwrapping algorithm: a review," Opt. Lasers Eng. **42**, 245–261 (2004).
13. R. Schöne and O. Schwarz, "Hybrid phase-unwrapping algorithm extended by a minimum-cost-matching strategy," Proc. SPIE **4933**, 305–310 (2003).
14. K. Zhou, M. Zaitsev, and S. Bao, "Reliable two-dimensional phase unwrapping method using region growing and local linear estimation," Magn. Reson. Med. **62**, 1085–1090 (2009).

15. G. Valadao and J. Bioucas-Dias, "PUMA: phase unwrapping via max flows," *Proceedings of Conference on Telecommunications—ConfTele*, Peniche, Portugal (2007), pp. 609–612.
16. H. H. Rapp, *Reconstruction of Specular Reflective Surfaces Using Auto-Calibrating Deflectometry* (KIT Scientific Publishing, 2012), retrieved from Universität Karlsruhe (Record ID: oai:EVASTAR-Karlsruhe.de:1000030538).
17. H. Cui, W. Liao, N. Dai, and X. Cheng, "A flexible phase-shifting method phase marker retrieval," Measurement **45**, 101–108 (2012).
18. X. Chen, C. Lu, M. Ma, X. Mao, and T. Mei, "Color-coding and phase-shift method for absolute phase measurement," Opt. Commun. **298**, 54–58 (2013).
19. H. O. Saldner and J. M. Huntley, "Temporal phase-unwrapping: application to surface profiling of discontinuous objects," Appl. Opt. **36**, 2770–2775 (1997).
20. S. Li, S. Liu, and H. Zhang, "3D shape measurement of optical free-form surface based on fringe projection," Proc. SPIE **8082**, 80822Z (2011).
21. G. Sansoni, M. Carocci, and R. Rodella, "Three-dimensional vision based on a combination of gray-code and phase-shift light projection: analysis and compensation of the systematic errors," Appl. Opt. **38**, 6565–6573 (1999).
22. G. Sansoni, S. Corini, S. Lazzari, R. Rodella, and F. Docchio, "Three-dimensional imaging based on gray-code light projection: characterization of the measuring algorithm and development of a measuring system for industrial applications," Appl. Opt. **36**, 4463–4472 (1997).
23. D. Bergmann, "New approach for automatic surface reconstruction with coded light," Proc. SPIE **2572**, 2–9 (1995).
24. Q. Zhang, X. Su, L. Xiang, and X. Sun, "3-D shape measurement based on complementary gray-code light," Opt. Lasers Eng. **50**, 574–579 (2012).
25. S. Zhang, "Composite phase-shifting algorithm for absolute phase measurement," Opt. Lasers Eng. **50**, 1538–1541 (2012).
26. J. Ohtsubo and A. Fujimoto, "Practical image encryption and decryption by phase-coding for optical security systems," Appl. Opt. **41**, 4848–4855 (2002).
27. J. L. Posdamer and M. D. Altschuler, "Surface measurement by space encoded projected beam systems," Comput. Graph. Image Process. **18**, 1–17 (1982).
28. J. Salvi, J. Pagès, and J. Batlle, "Pattern codification strategies in structured light systems," Pattern Recogn. **37**, 827–849 (2004).
29. G. P. Butel, G. A. Smith, and J. H. Burge, "Optimization of dynamic structured illumination for surface slope measurements," Proc. SPIE **8493**, 84930S (2012).