NIST Special Publication 260
NIST SP 260-228

Certification of Standard Reference Material® 2196

Axial Resolution Standard for Optical Medical Imaging

Jeeseong Hwang
Kimberly Briggman
Nikki Rentz
Hyun-Jin Kim*
David W. Allen
Lee Richter
Sowon Yoon*
John Lu

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.260-228

NATIONAL INSTITUTE OF
STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Special Publication 260
NIST SP 260-228

Certification of Standard Reference Material® 2196

Axial Resolution Standard for Optical Medical Imaging

Jeeseong Hwang
Kimberly Briggman
Nikki Rentz
Hyun-Jin Kim*
Applied Physics Division
Physical Measurement Laboratory

David W. Allen
Sensor Science Division
Physical Measurement Laboratory

Lee Richter
Material Science and Engineering Division
Material Measurement Laboratory

Sowon Yoon*
John Lu
Statistical Engineering Division
Information Technology Laboratory

**Former NIST employee; all work for this publication was done while at NIST*

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.260-228

February 2023

NIST SP 260-228
February 2023

NIST Technical Series Policies

Publication History

Approved by the NIST Editorial Review Board on 2022-07-25

How to Cite this NIST Technical Series Publication

Hwang J, Briggman K, Rentz N, Kim H-J, Allen DW, Richter L, Yoon S, Lu J (2023) Certification of Standard Reference Material® 2196, Axial Resolution Standard for Optical Medical Imaging. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 260-228. https://doi.org/10.6028/NIST.SP.260-228

NIST Author ORCID iDs

Jeeseong Hwang: 0000-0002-9801-8529
Kimberly Briggman: 0000-0001-8341-998X
Nikki Rentz: 0000-0002-7328-5639
David W. Allen: 0000-0001-8299-7956
Lee Richter: 0000-0002-9433-3724
John Lu: 0000-0003-2775-6644

Contact Information

Jeeseong.Hwang@nist.gov
Kimberly.Briggman@nist.gov

Abstract

Medical imaging devices and systems must be calibrated to ensure uniformity and reliability of test results. A standard reference material (SRM) or "phantom", as it is known in the medical imaging community, is used to replicate fundamental characteristics of tissue and/or the material for which the imaging device is intended for use. SRMs can be readily deployed to sites where devices are being used, negating the need to physically relocate instruments for calibration or performance testing. The SRM produced in this work is appropriate for calibrating depth-resolving 3D optical systems such as optical coherent tomography (OCT). The SRM consists of three layers of polydimethylsiloxane (PDMS) films on a gridded glass slide. A thin, clear layer of PDMS is sandwiched between two thicker scattering layers of PDMS. The scattering layers contain titanium dioxide (TiO_2) as scattering particles to allow easy identification of the thicknesses of the three layers as a dimensional calibration of the axial resolution of an optical coherence tomographic imaging device. The concentration of TiO_2 scattering particles in PDMS can be adjusted to control its scattering coefficient to mimic that of biological tissue which was independently measured by a broadband integrating sphere system at NIST for these concentrations. The thickness of each layer was measured by NIST's spectral domain optical coherence tomography (SD-OCT) instrument to produce data similar to that obtained using clinical devices. The axial dimensions of each region of interest of the SRM was determined from the pixelated, 3D tomographs acquired with an index of refraction (n) = 1, then converted from pixels to µm using a NIST-traceable height standard to calibrate the NIST SD-OCT and then corrected by the index of refraction of PDMS as measured by spectroscopic ellipsometry at the appropriate OCT wavelength. For the users of the SRMs, we have developed an algorithm which can be applied to a 3D tomograph obtained from any OCT to detect the interfacial planes between the three layers of the SRM. The algorithm reports the local layer thicknesses for each pixel across the entire lateral dimension of the tomographic data cube. The user can then compare the output of the algorithm generated by their OCT tomograph to the values from NIST to determine the calibration of their instrument. For each SRM, we report a mean thickness and standard deviation for each of the three-layer thicknesses across 9 regions of interest as defined by a target grid registered in the SRM.

Keywords

Imaging device calibration; Optical coherence tomography; Optical medical imaging phantoms; Tissue-mimicking phantoms; Polydimethylsiloxane.

Table of Contents

List of Tables

List of Figures

Acknowledgments

Authors thank Dr. Steven Choquette for helpful discussion on developing standard reference materials. This work was supported by NIST intramural Standards Development program on optical medical imaging phantoms and by Scientific and Technical Research and Services funding from the NIST Physical Measurement Laboratory.

1. Purpose and Description

Standard reference materials (SRMs) have been created to simulate structural (e.g., multilayer structures with thicknesses of a few ten to a few hundred micrometers) and optical (e.g., refractive index) characteristics of tissues as an axial (z-directional) resolution standard primarily for optical coherence tomography (OCT). The SRMs are composed of three layers of polydimethylsiloxane (PDMS), in which a thin, clear layer is sandwiched between two thicker scattering layers to allow clear identification of the thickness of the middle layer as a measure of the axial resolution of the instrument. A NIST certificate of analysis is provided for each SRM: the mean thickness and the standard deviation of each of the three layers tabulated for each 2 mm x 2 mm square region of interest (ROI) over the entire 6 mm x 6 mm sample area (i.e., for each of the 3 x 3 ROIs). A grid pattern and grid coordinates are engraved on the back side of the SRM slides to identify these 9 ROIs. A white coated fiduciary slide with five 5 mm clearance holes is provided along with the SRM to help align the target grids under an imaging device.

2. Storage and Use

For the best stability, SRMs based on PDMS should be stored at 25 °C or below and precautions should be taken to prevent moisture from contacting this material to avoid swelling of the layers. For long term storage, a container filled with dry air and dry nitrogen is recommended according to the manufacturer's data sheet (Dow Corning MSDS form 06-1009-01). Other cured PDMS devices with microstructures have been imaged by optical coherence tomography (OCT) over 8 years without any noticeable structural change (unpublished results), therefore the shelf life of the PDMS-based SRMs is expected to be at least 5 years from the delivery date under these recommended storage conditions for this application. In handling the SRMs, organic contamination such as fingerprinting should be avoided. If the surface of SRM needs to be cleaned, a few drops of 70% ethanol diluted in water (in which the PDMS is minimally soluble [1]) should be used to briefly rinse the surface followed immediately by gentle, non-contact drying with clean dry air or nitrogen.

To use the SRM, align the white and ground glass sides of both the sample and fiduciary slides (Fig. 1, left) with the SRM sample on top of the fiduciary slide. The number 3 hole in the fiduciary slide aligns with the target grids but slightly off center of the regions of interest (Fig. 1, right). After alignment, the fiduciary slide may be removed if desired.

The certificate of analysis includes a table for each SRM, listing the mean and standard deviation values for each layer thickness for each of the 9 ROIs [i.e., the ROI numbers (1, 2, 3, 4, 5, 6, 7, 8, 9) corresponding to each square region (3C, 3D, 3E, 4C, 4D, 4E, 5C, 5D, 5E) in Fig.1, respectively]. An example is shown in Table 2 in the Sample Characterization section of this document.

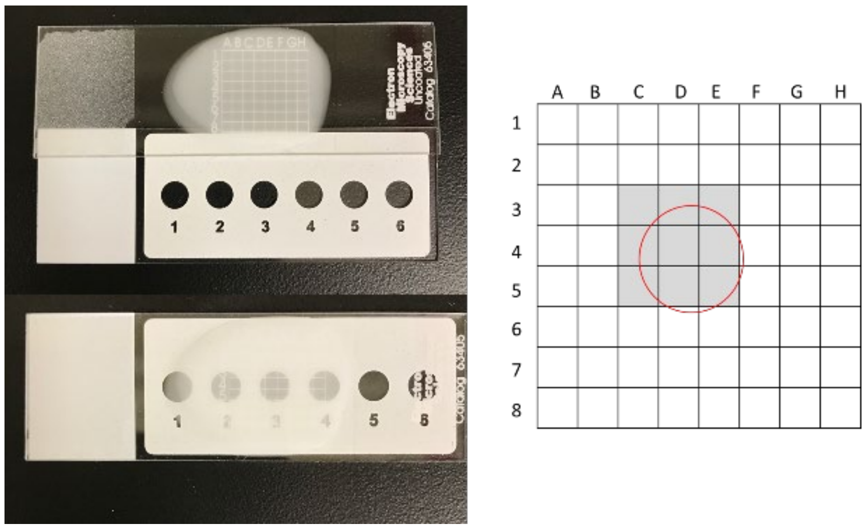


Fig. 1. How to use the fiduciary slide to find the ROIs of the target.

The certified thickness value from one or more of the ROIs can be used to check and correct the calibration of the end user's imaging device. Depending on the device, the discrepancy needs to be corrected in the image acquisition and/or data analysis software. For example, in OCT, the data acquisition needs to be set for the correct refractive index value of the PDMS material at the wavelength of the light source (see Index of Refraction Determination section for values). Then the dimension of each axial pixel can be calibrated according to the certified values. In most microscopes, the sample refractive index value is not adjustable during the acquisition, therefore post processing calibration can be implemented, following the procedure described later in this document. For best practice in calibration, it is strongly encouraged to use the NIST-developed image analysis algorithm provided in this document on the image data obtained from the user's imaging device to mitigate any algorithm-dependent uncertainty variations. Program codes of the NIST-developed image analysis algorithm are presented in Appendix A, and the format and structure of the user's raw image data to run the program are explained in the commented lines in the program codes.

3. History and Background

Medical imaging devices and systems must be calibrated to ensure uniformity and reliability of test results [2]. Calibration is necessary to ensure the highest possible degree of accuracy for data obtained through imaging. Calibration may require the use of a sample or object on which testing, or imaging can be performed. With respect to human tissue and fragile or volatile test materials, it is useful to have a material, a so called "phantom," which may emulate the physical and optical characteristics of the material on which imaging is to be performed for purposes of calibrating the imaging instrument [3]. Axial (z-direction) resolution calibration is particularly important for depth-resolving 3D optical systems such as OCT (for FDA-approved ophthalmic imaging and for intravenous diagnostics), confocal microscopy (mostly for superficial cancer imaging and pathology), and wide-field 3D microscopy (for optical biopsies in surgery) [4] [5] [6]. It is desirable to have standard reference materials for calibration and as a test for measurement consistency and proficiency of these imaging devices. For the 3D optical systems mentioned above, a phantom SRM in this report is a physical object that simulates the layered structure of tissue or other material for which the imaging device or system is intended to be tested. The SRMs can also be readily deployed to the sites where devices are being used, negating the need to physically relocate devices to a test site for calibration or performance testing.

The SRM produced in this work consists of 3 layers of PDMS, in which a thin, clear PDMS layer, created by spin coating, is sandwiched between two thick, scattering layers (PDMS with TiO_2) applied by blade coating. The wavelength-dependent scattering coefficient can be controlled by the concentration of titanium dioxide (TiO_2) scattering particles in the PDMS to make the scattering coefficient relevant to that of biological tissues. Measurement of the wavelength-dependent reduced scattering coefficient of these samples was achieved using an integrating sphere system and an inverse adding-doubling algorithm [7]. The thickness of each layer was measured by a spectral domain OCT (SD-OCT) at NIST. In SD-OCT, the axial length scale is measured from the time delayed back scattering signal reflected from the sub-surface layers, therefore the wavelength-dependent effect of the refractive index of the material influences the OCT measurement [8]. Accordingly, the axial dimensional calibration in the PDMS material needs to be corrected by the refractive index at the device wavelength, which was independently measured by spectroscopic ellipsometry (model M-2000 DI, Woollam). To make the measurement SI-traceable, the NIST SD-OCT conversion from pixels to µm was independently calibrated with a NIST-traceable step height reference material (step height standard, part #SHS – 50.0 Q, serial #13153-01-11, VLSI Standards Incorporated).

As multiple SRMs needed to be characterized, we developed an algorithm to automate the analyses of multiple samples. The algorithm detects the interfacial planes between layers from the 3D tomography data and measures the local layer thicknesses pixel by pixel across the lateral dimensions of 6 mm x 6 mm. The scattering layers in the SRMs have uniform thicknesses with the standard deviations less than 5% across the entire characterized sample area. Mean thicknesses and standard deviations of the layers for each of the 9 ROI areas within the lateral dimensions of 6 mm x 6 mm are reported for each SRM. This algorithm is available to consumers of the SRMs to use in their analysis as well.

4. Sample Preparation[1]

4.1. Processing of base materials

The layered phantoms are fabricated by a PDMS (Sylgard 184 Elastomer, Corning) polymer base material with titanium dioxide scattering particles (TiO_2), (TDP, particle size of 0.3 µm – 1.0 µm, TI-602 Atlantic Equipment Engineers, Inc.) as an additive to the first and third layers to introduce light scattering. A stock solution of TiO_2 (10 g/kg) in PDMS was prepared by the following procedure. Toluene (20 mL) was placed in a clean glass jar (125 mL) and 1 g of TiO_2 was added. The size distribution of the TiO_2 particles characterized by a backscatter confocal microscope is reported elsewhere [7]. The solution was sonicated in a bath sonicator for 6 h for homogenization and dispersion of the particles. PDMS (100 g) was added into the same jar and mixed with a blade mixer for 30 min, sonicated in a bath sonicator for 6 h, and placed in a

[1] Certain commercial instruments are identified to specify the experimental study adequately. This does not imply endorsement by NIST or that the instruments are the best available for the purpose.

rocking mixer for at least 24 h for homogeneous mixing. The jar was opened and placed in a vacuum desiccator for several days until the toluene was completely evaporated from the mixture. For the final preparation of the layers, this stock solution was diluted with clear PDMS for a final concentration of 0.15 % (mass fraction) of TiO_2 in PDMS. A curing reagent (10% mass fraction to the PDMS/TiO_2 mixture) was added to the mixture and the solution was mixed thoroughly with a blade mixer for 10 min. The sample was put in a vacuum desiccator for > 30 min or until all air bubbles were removed from the solution to prevent undesired scattering from the air bubbles once cured. For the clear layers, the PDMS was prepared as described above but without any TiO_2 additive.

4.2. Fabrication of layered phantoms

A schematic and a photograph of a three-layer sample is shown in Figs. 2(A), and 2(B), respectively. A bottom scattering layer was prepared on the reverse (smooth) side of a gridded glass slide (25.4 mm x 76.2 mm x 1.0 mm) (Electron Microscopy Sciences, part #63405) by a blade coating technique shown in Fig. 2(C). In the blade coating system, a glass slide (sample substrate) is placed on a horizontally movable stage and a second plain glass slide (blade slide) of the same size, held at a tilt angle of 5°, is dragged across the sample substrate after a small amount of PDMS/TiO_2 solution is placed between the two slides. Prior to mounting, the glass slides were cleaned by wiping with low-lint wipe (Kimberly-Clark, Kimtech) saturated with methanol followed by rinsing with a stream of methanol and blow-drying with clean nitrogen gas. The drag speed was kept at 1 mm/s. The height of the tilted blade slide maintained a constant gap between the blade edge and the top surface of the sample slide to provide a uniform layer thickness of the PDMS/TiO_2. Only one pass of the blade was used to produce the layer. To achieve uniformity in the thickness of the coated films, the gap between the sample and blade slides was adjusted and monitored by imaging a laser spot reflected off the top surface of the sample slide with a charge-coupled device camera. On the back of the gridded substrate slide, a pattern of 8 x 8 squares, each with dimension 2 mm x 2 mm is engraved. The blade coated layer covered the entire 8 x 8 gridded area.

After applying the first blade coated layer to the sample, the sample was cured in an oven at 75 °C for at least 2 h before applying the second layer. After the first scattering layer was coated and cured, a second non-scattering layer (clear PDMS only, no scattering particles) was prepared by spin coating as shown in Fig. 2(D). To produce a target thickness of the second layer of approximately 15 μm, we diluted PDMS solutions prior to spin coating. The PDMS film thickness depends on rotational speed, spin time and viscosity of the PDMS solution. To produce a target thickness of the second layer of approximately 15 μm, we used a 2:3 dilution (mass ratio PDMS : toluene, 67 % in weight fraction) of the PDMS solution. The gridded glass substrate with the first, cured scattering layer was completely covered by the diluted PDMS solution, then the slide was spun at a fixed speed of 209 rad/s (2000 revolutions per min) for 1 min, then the coated sample was placed back into a 75 °C oven to cure for at least 2 h. Upon curing this second layer at 75 °C for at least 2 h, a third scattering layer was prepared by the blade coater method described above and cured, following the same procedure as the first layer. The final triple layer sample was imaged with SD-OCT. A total of 200 samples were fabricated and characterized.

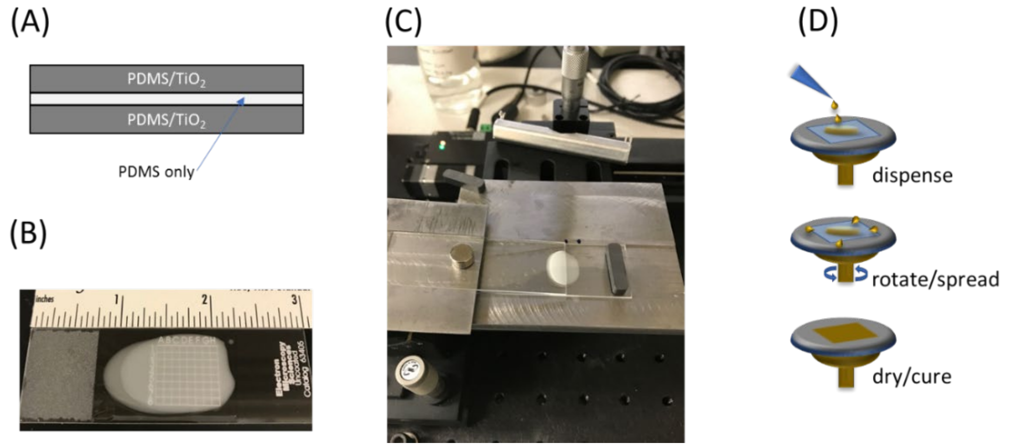


Fig. 2. Sample preparation techniques: (A) A schematic of the three-layer samples. (B) A SRM consisting of the three PDMS layers. (C) Blade coating system. Here, a clear microscope slide for the sample substrate is shown, but for the SRM, a slide with a grid on the bottom was used. (D) A schematic of the spin coating technique.

5. Sample Characterization

5.1. OCT Instrumentation

The SD-OCT system (Envisu R-Series, Bioptigen, Raleigh, NC) operates with a super luminescent diode (SLD) with a center wavelength of 840 nm and a nearly Gaussian spectrum with 93 nm full-width at the half-maximum (FWHM) bandwidth. The SLD light is focused through a long working distance objective lens with numerical aperture of 0.05 and laterally scanned by a pair of galvanometer mirrors. The spectrum collected from each one-dimensional A-scan was interferometrically combined with the reference arm signal and was projected onto a partial (1024 pixel) array of a 4096 pixel linear array camera, and the projected spectrum was processed by the fast Fourier transform algorithm to obtain a 10-bit spectrum (1024 points) for approximately 1.1 mm axial depth. The OCT images were obtained from a 6 mm x 6 mm lateral area with 600 linear two-dimensional B-scans at 600 A-scans per B-scan. A total of 1024 axial points for each lateral pixel provided sufficient imaging depth to image the three layers across the 6 mm x 6 mm lateral area. All measurements were performed at (21 ± 1) °C and at (30 ± 10) % relative humidity. OCT images were acquired using Bioptigen's InVivoVue software version 2.4.35.

For image acquisition, the sample was mounted on a platform with translation in both lateral and axial directions and adjustable in tip and tilt orientations. The sample was tilted in the y-direction to avoid back coupling of the specular reflection from the sample surface. The target image area consists of 3 x 3 grids, including grids numbered (3C, 3D, 3E), (4C, 4D, 4E), and (5C, 5D, 5E) as shown in Fig. 3(A). Each grid is 2 mm x 2 mm. From an OCT tomographic data cube, consisting of 600 B-scans across the y direction, 3 B-scan images (the 100th, 300th, and 500th B-scan) images are displayed in Fig. 3(B) as the upper, middle, and bottom panels, respectively. They correspond to the vertical cross-sectional images across the x-axis illustrated as dotted red lines in Fig. 3(A). These images resolve three layers: a clear PDMS layer in the middle with thicker scattering layers at the top and the bottom. The label 'top layer' refers to the layer at the air interface of the sample and the 'bottom layer' is the layer in contact with the glass substrate.

Each layer's thickness within each gridded section was extracted from the OCT data cube by a NIST-developed image analysis algorithm described in the Data Analysis section. Initially, the axial measurements in OCT were performed by setting the refractive index (n) = 1 in the OCT acquisition software. The algorithm was used to first analyze the optical thicknesses in pixels with n = 1, then the final thickness values were obtained by dividing the pixel values by the measured refractive index, 1.408, at λ = 840 nm and applying the length per pixel calibration of 1.074 µm/pixel for the NIST SD-OCT (see sections on Calibration of the OCT Axial Scale and Index of Refraction Determination). The errors in the length per pixel and the index of refraction were propagated in the final reported values. See Uncertainty Analysis for discussion of the final uncertainty calculations. The same measurements and analyses were repeated for all 200 samples, and an example of the certificate for SRMs including a table for the certified thickness values and uncertainties for SRM 002 is available in Appendix 1.

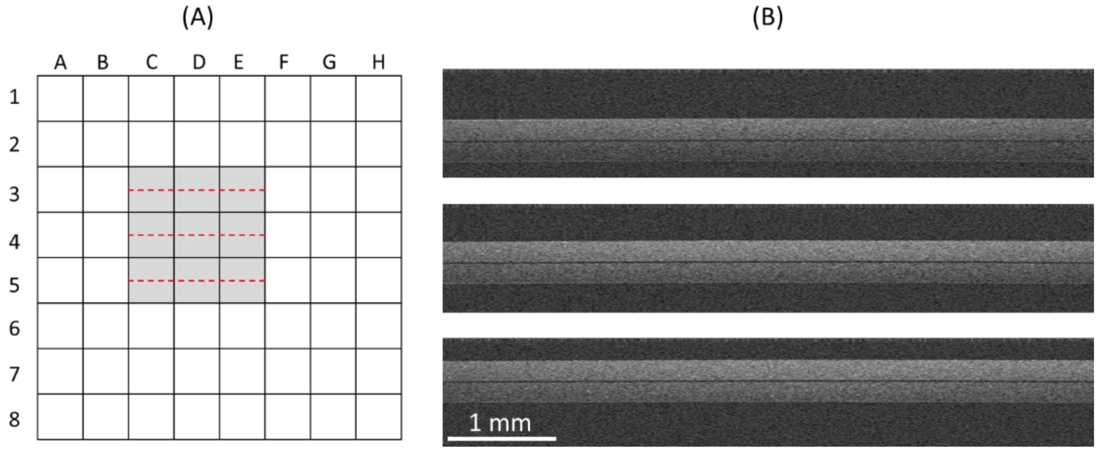


Fig. 3. Cross-section OCT images of the target regions of interest of a SRM. (A) Top view of a targeted imaging area (6 mm x 6 mm) of a SRM slide. (B) 3 B-scan (cross-section) images, 100th (upper), 300th (middle), and 500th (lower) of the 600 B-scans of the SRM 002, corresponding to the dotted red lines in (A). Although the images are acquired in pixels, a scale bar is shown which applies to the x-axis. The axial depth of the B-scan is approximately 1.1 mm.

5.2. Determination of axial and lateral resolutions of the NIST SD-OCT

In OCT, the theoretical axial resolution, R_{OCT}, is given by the following relation:

$$R_{OCT} = \frac{l_c}{2} \approx 0.44 \frac{\lambda_0^2}{\Delta\lambda}$$

where l_c represents the coherence length, λ_0 is the source center wavelength and $\Delta\lambda$ is the source bandwidth. For the NIST SD-OCT system, operating at a center wavelength of 840 nm with 93 nm FWHM spectral bandwidth, the theoretical axial resolution is $\approx$ 3 µm.

To determine the accuracy of our OCT thickness measurements, the axial and lateral resolution of the device were evaluated by measuring the point spread function (PSF) with a sample consisting of single 40 nm diameter gold nanoparticles dispersed in a clear PDMS matrix. To prepare this PSF sample, 40 µl of gold solution in aqueous buffer (40 nm Citrate NanoXact Gold, NanoComposix) was vortex mixed in 100 µl of ethanol for 10 min, then the mixture was added dropwise into 10 g of pure PDMS heated on a hot plate at 120 °C while blade mixing the PDMS. After further blade mixing for 30 min, the solution was cooled down to room temperature, then 0.88 gr of curing agent was added to the mixture and mixed with a blade mixer for 30 min. The mixture was then put in a vacuum desiccator for 2 h. A glass pipette was used to transfer a hanging drop of the final mixture to a clean microscope slide. The sample was cured at 75 °C for 2 h and then imaged by SD-OCT. After a tomograph of dispersed single gold nanoparticles under a 1 mm x 1 mm region was obtained, the scanning area was zoomed into a 0.1 mm x 0.1 mm region to obtain a higher resolution tomograph of one of the gold nanoparticles.

The size of the nanoparticle is sufficiently smaller than the device's spatial resolution. The device resolution is estimated using the nominal values of the wavelength, wavelength bandwidths, and numerical aperture (NA ≈ 0.05) of the objective lens of the SD-OCT, which are ≈ 3 µm and ≈ 14 µm (in PDMS with refractive index of 1.408 and nominal focal length of the telecentric lens in the SD-OCT) in the axial and lateral directions, respectively. The axial resolution value was calculated from the equation above, and the lateral resolution value from $\lambda/1.2 \cdot \mathrm{NA}$. Therefore, the single particle can be treated as a point scatterer for a PSF measurement. A 3D SD-OCT image of a single gold nanoparticle shown in Fig. 4(A) rendered from a tomograph (1000 x 1000 x 1024 pixels) of a 0.1 mm x 0.1 mm lateral area shows an oblong shape, elongated in the lateral direction indicative of an asymmetric PSF with lower spatial resolution in the lateral vs. the axial direction. From this raw SD-OCT tomographic data of the single particle, intensity profiles from 10 lines crossing the center of the particle, 5 lateral (gray in Fig. 4B) and 5 axial (black) lines, were obtained from the 5 adjacent B-scan (vertical cross sectioned at 0.1 µm spacing) images near the center of the Au particle.

These intensity profile curves were then fit to a Gaussian function, and the results are displayed in the same graph in Fig. 4B in red and blue. For the plots for axial intensity profiles and fitted curves, the axial length scale was corrected by the refractive index of the PDMS. But for the lateral dimension, the length scale was not corrected. From these fits, FWHM values for all lateral and axial line profiles were obtained and are displayed in Fig. 4(C). The profiles with the largest intensities exhibit the smallest FWHM values of 2.77 µm ± 0.26 µm and 7.55 µm ± 0.37 µm for the axial and lateral directions, respectively, indicating that these profiles are from the lines crossing closest to the center of the Au nanoparticle. These FWHM values describe the best OCT lateral and axial resolutions and agree with the estimations of the OCT resolution based on nominal values of the wavelength, wavelength bandwidth, and numerical aperture of the objective lens of the OCT. Here, the uncertainties are the standard errors calculated from the deviation of the intensity values from the fitted Gaussian curve based on the Levenberg-Marquardt algorithm [9].

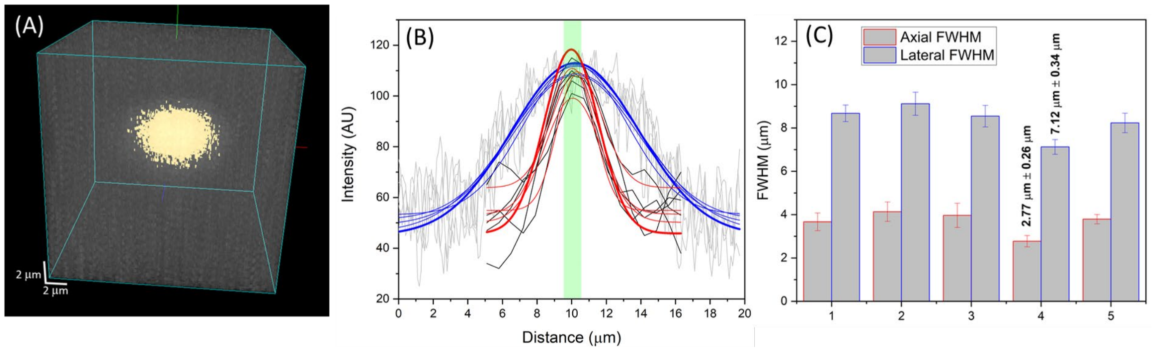


Fig. 4. (A) 3D rendered OCT image of a single 40 nm diameter gold nanoparticle. (B) Intensity profiles from 10 lines crossing the center of the particle, 5 lateral (gray) and 5 axial (black) lines obtained from the 5 adjacent B-scan (vertically cross sectioned) images near the center of the Au particle, and corresponding fitted Gaussian curves in blue and red. The thickest fitted curves represent the highest intensity and the smallest FWHM corresponding to #4 in C. Note that the peaks of all the Gaussian fit curves fall within the green vertical strip coinciding with the bound of one axial pixel. (C) FWHM results from the fitted curves are displayed in blue (lateral) and red (axial) with the smallest values displayed above the bar graphs.

Although single gold nanoparticles of 40 nm diameter have small scattering cross section sand the wavelength of the OCT's light source is not at the absorption peak, the high signal-to-noise ratio of the NIST SD-OCT allowed for resolving the PSF of a single particle. Furthermore, Fig. 4(B) demonstrates that the local intensity maxima of all the axial intensity profiles occur at the same axial position (see green vertical strip coinciding with the bound of one axial pixel), verifying that the OCT tomographic image of a single nanoparticle shows a well-localized maximum intensity at single axial pixel precision. Our image analysis algorithm discovers local maxima at the single pixel resolution in the axial direction, effectively deconvolving the local scattering intensity maximum from the signal from scattering particles. Therefore, our image analysis algorithm, based on local maxima finding, can localize the layer interface in the axial direction at 1 pixel resolution

5.3. Calibration of the SD-OCT axial scale

Since the 3D tomographs were acquired and analyzed in pixels, a conversion from pixel to length in µm is needed to determine the axial thicknesses. A calibration of the NIST SD-OCT was performed by imaging a NIST-traceable step height reference material (step height standard, part #SHS – 50.0 Q, serial #13153-01-11, VLSI Standards Incorporated). Step height standards have been used to establish SI traceability of the axial length scales in confocal and interference microscopy [10] in addition to OCT [11]. The step height standard used here consists of a 25 mm x 25 mm x 3 mm quartz block with a precisely etched trench (negative step). The negative step is 1 mm wide by 2.5 mm in length, as denoted with fiducial markers. The certificate of calibration provides a mean value of 49.594 µm ± 0.267 µm ($k = 2$) for the depth of the negative step in the calibrated area. OCT pixel images were obtained for the NIST-traceable step height standard over a 6 mm x 6 mm lateral area with 600 linear B-scans at 600 A-scans per B-scan. Shown in Fig. 5 are the (A) volume intensity projection and (B) the B-scan image corresponding to the slice along the green line drawn in (A).

Each vertical line in the B-scan in Fig. 5(B) was fit to a Gaussian profile to determine the position of maximum intensity. The maximum intensity was taken to be the surface of the quartz step height standard for either the upper quartz surface or the negative step surface. The centers of the Gaussian profiles were then fit across the upper and lower (trench) surfaces between the fiducial markers with a 5th-order polynomial to account for the distortion of the curvature of the collection lens across the field of view. The 2nd through 5th orders of the polynomial fit was constrained to be the same for each upper and lower curve pair. The difference in the vertical offset of each fit was used to determine the pixel spacing between the two curves. This procedure was applied to three of the B-scans between the fiducial markers closest to the center of the image. This pixel spacings were compared with the mean negative step value, 49.594 µm ± 0.267 µm ($k = 2$), to calculate an axial length per pixel of 1.074 µm ± 0.009 µm ($k = 2$). This value was used to convert the axial lengths in the SD-OCT images from pixels to microns.

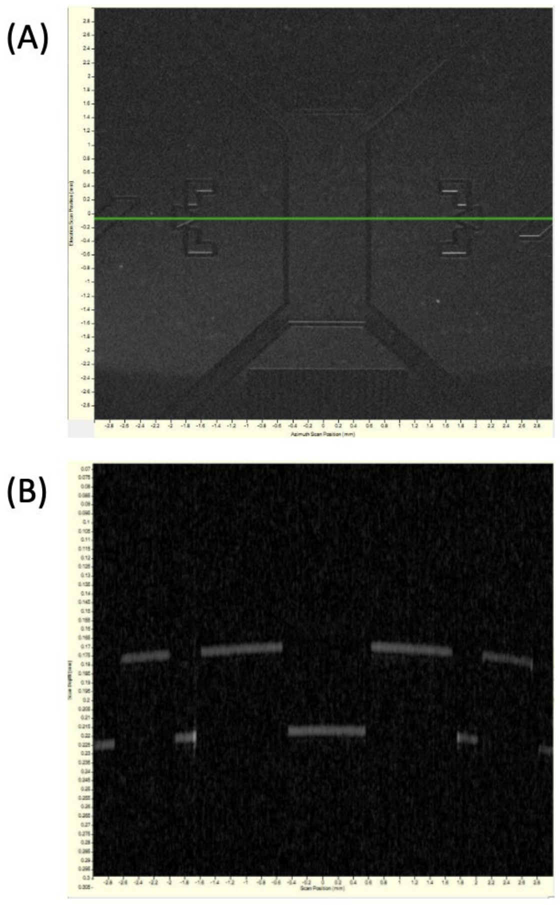


Fig. 5. (A) Volume intensity projection of the OCT image of the step height calibration standard. (B) The B-scan (depth) image corresponding to the green line slice drawn in (A).

5.4. Index of Refraction Determination

The length scale in OCT also needs to be corrected by the wavelength-dependent refractive index, n, of the material [12]. We initially acquired the OCT data by setting $n = 1$. We need to correct this value to that of the PDMS and TiO_2 containing PDMS materials used for these SRMs and at the OCT wavelength of 840 nm. In the literature, there are previous reports of the index of refraction for Sylgard 184 PDMS determined by Abbe refractometry [13], [14], but these measurements only span between wavelengths of 405 nm to 688 nm. Additionally, the Sylgard 184 PDMS product data sheet [15] lists values of refractive index at four wavelengths from 589 nm to 1554 nm, with an anomalous dispersion of PDMS reported at 632.8 nm. For these reasons we sought an in-house measurement of the index of refraction of the materials used in our SRMs.

For the spectroscopic ellipsometry measurement, we fabricated thick PDMS blocks from the same stock solutions that we used for the fabrication of the layered SRMs. In spectroscopic ellipsometry, a weakly focused beam (nominally 300 µm diameter) was used to suppress artifacts from reflection at the bottom of the sample and data was recorded at 9 spots on a 5 mm spacing grid (1 cm^2 area) across the sample. Additionally, three angles of incidence, nominally (50, 60 and 70) ° were used, calibrated to 0.006 ° precision (standard deviation over 9 spots) by measurement of a nominally 25 nm thick thermal oxide on a silicon wafer reference. The index of refraction was determined from a fit of the variable angle spectroscopic ellipsometry data to a Sellmeier dispersion[2],

$$n = \left(\epsilon_\infty + \frac{A\lambda^2}{\lambda^2 - B^2} - E\lambda^2\right)^{1/2}.$$

To correct the wavelength dependent refractive index in the OCT measurements, the value of $n = 1.408 \pm 0.002$ ($k = 1$, at 840 nm) was used for both the clear and the 0.15 % TiO_2 containing PDMS samples as no difference in the index of refraction at this scale was measured between these two samples. The uncertainties represent estimates of the type B errors due to instrument and angle of incidence calibration, based on measurements of fused silica. This value of the refractive index is consistent with that recently reported in [16], and if extrapolated to lower wavelengths, the index of refraction value is similar to [13], but lower than [14].

6. Data Analysis

Our thickness analysis algorithm is based on finding the top, bottom, and interfacial planes of different layers. The first step of the analysis involves the detection of the middle layer gap in all B-scan images. To this end, we first extracted horizontally stretched Haar-like features. Haar-like features are digital image features used in object recognition [17]. A Haar-like feature is defined by the difference in intensity values between rectangular regions, S1 and S2 in Fig. 6(A). The rectangular region is defined in pixels by its *w (width) x h (height)*. The values for *w* and *h* used in the experiment are *w = 31* and *h = [3,5,7,9,11,13,15]*. We chose *w* =31 because this is the safe axial length across which the curvature in the image is negligible with no distortion in the axial scale. For each *h*, we computed the feature response *R (x, y; z, h)* from each vertical slice (*z*) as follows: For each pixel location *(x, y)*, the algorithm computes two Haar-like features by computing the sum of pixel intensity values in each rectangular region, S1, S2, and S3 as depicted in Fig. 6. Then the value from the white region (S1 or S3) is added, and the value from

[2] The mean parameters of the Sellmeier model, from the 9 spots, are where $\epsilon_\infty = 1.4024$; A = 0.5656; B = 0.1349; and E = 0.0016. Note that there are strong correlations between the 4 parameters in a Sellmeier model.

the blue region (S2) is subtracted. More specifically, as shown in Fig. 6(A, a), the Haar-like feature for pixel *(x, y)*, *Ra,* is defined as *Ra (x, y; z, h) = S1(x, y) - S2(x, y)*. As illustrated in Fig. 6(A, b), the Haar-like feature for pixel *(x, y)*, *Rb*, is *Rb(x, y; z, h) = - S2(x, y)+S3(x, y)*. The key idea here is, if the local area of the image has the dark line segment with the thickness of *h*, the response will be positive and the magnitude of the response will be the intensity difference between the two regions: (1) Since we are interested in dark line segments surrounded by brighter areas, we disregard any pixels with $Ra(x, y; z, h) \leq 0$ or $Rb(x, y; z, h) \leq 0$; (2) Then, the response for *h* is *R(x, y; z, h) = Ra(x, y; z, h) + Rb(x, y; z, h)*; (3) Finally, the final response function is the summation of all responses across all *h* values: $R(x, y, z) = \sum_h R(x, y;\ z, h)$. To help visualizing these analysis steps, a vertical slice (B-scan) image before analysis, and responses of *Ra*, *Rb*, and the summed *R* for *h = 9* are shown in Fig. 7(A) from left to right, respectively, demonstrating intermediate results in the procedure to detect the middle layer gap.

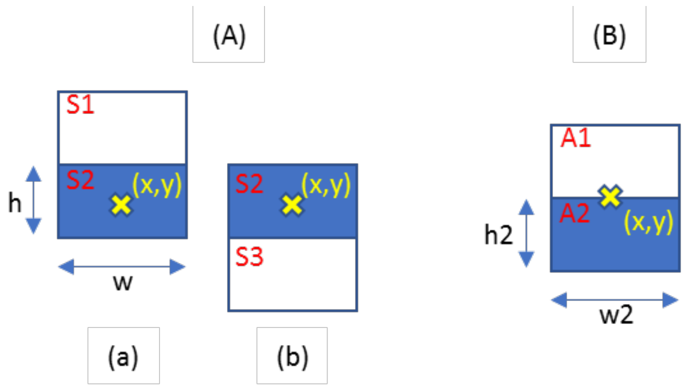


Fig. 6. Schematic of the interface detection algorithm. (A) Schematic of interfacial regions for the gap detection algorithm and (B) edge detection algorithm to detect the boundary between layers. S1 and S2 are regions from which the Haar-like features are calculated. A1 and A2 are regions where the upper and lower bound of the gap are determined. Details are described in the text.

To segment the gap in the 3D image, *R(x, y, z)* was binarized to find the largest connected component in 3D with the following steps: (1) binarize *R(x, y, z)* (assign 1 if above the threshold and 0 otherwise: *BW(x, y, z)= 1 or 0*) with the threshold determined by the Otsu's method [18], [19]; (2) find connected components in *BW(x, y, z)* using the 26-connectivity for 26 surrounding adjacent voxels around the center cube of a 3D volume defined by the adjacent 3 pixels by 3 pixels by 3 pixels; (3) the weight for each connected component is the sum of *R(x, y, z)*, where *(x, y, z)* belongs to that connected component; and (4) find the connected component with the maximum weight.

To find the upper bound and lower bound of the gap, we first fit a second-order polynomial, $z = a_0 + a_1x + a_2y + a_3x^2 + a_4y^2 + a_5xy$, to the points in the gap where *BW(x, y, z) = 1* and estimate the parameters using the least squares. This fit allows for *median estimate* of the gap locations, *Z*(x, y)*. As the final step to search for the exact upper bound and lower bound of the gap, for every *(x, y)* in the gap, the upper bound was found by moving the window pixel-by-pixel upward to 10 pixels (10 µm along the axial direction to make sure that this distance is sufficiently larger than the nominal thickness of the middle gap which < 20 µm) from *Z*(x, y)* as shown in Fig. 7(B) then: (1) If the window is located right at the edge, the intensity difference between the upper area (A1) and the lower area (A2) would be the maximum; (2) the difference between A1 and A2 is measured by the Euclidean distance between the histograms from A1 and A2; and (3) the average image intensity is checked to determine the upper bound of the gap. Similarly, the lower bound of the gap was searched by moving the window pixel-by-pixel downward to 10 pixels from *Z*(x, y)*, and the above procedure was repeated. The final upper and lower bounds are *Z_upper~(x, y)* and *Z_lower~(x, y)*, respectively.

In the OCT image, the sample is tilted in the y-direction to avoid back coupling of specular reflection from the sample surface. This tilt angle was calculated from the OCT data cubes. Although the tilt angle was maintained for the measurements of all the samples, we calculated the angle for each sample for the best precision following the steps below. This tilt angle was used to correct the gap thicknesses from the OCT measurements.

A 3D plane equation, $ax + by + cz + d = 0,$ was fit to each bound plane with the least squares estimation. The points in the bound plane are represented in metric coordinates such that the following matrix product equation is satisfied,

$$Ap = 0, \text{ where } A = \begin{bmatrix} x1 & y1 & z1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ xN & yN & zN & 1 \end{bmatrix} \text{ and } p = [abcd]^T,$$

where p is the eigenvector of $A^T A$ corresponding to the smallest eigenvalue (Lagrange multipliers), and the tilt angle of the plane is

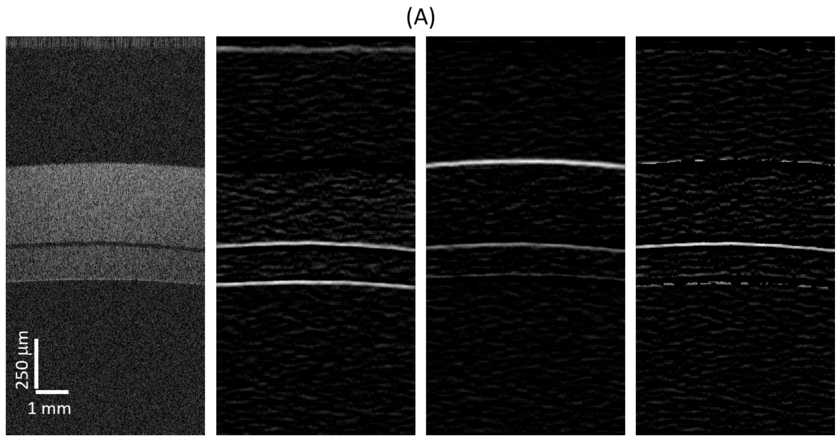
$$\theta = \phi - \frac{\pi}{2}, \ \theta \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right],$$

where $\phi = \cos^{-1}\left(\frac{b}{\sqrt{a^2+b^2+c^2}}\right), \ \phi \in [0, \pi]$ is the angle of the normal vector of the plane.

The layer thickness is defined as the vertical distance between the upper and lower bound at every pixel, converted from pixels to metric units, and divided by the refractive index,

$$\frac{1}{n} l_{pp_z}\left[Z_{upper}(x, y) - Z_{lower}(x, y)\right] \times cos\theta$$

where l_{pp_z} is the calibrated length per pixel (µm/pixel) in the axial direction, and n is the refractive index of PDMS at λ = 840 nm.



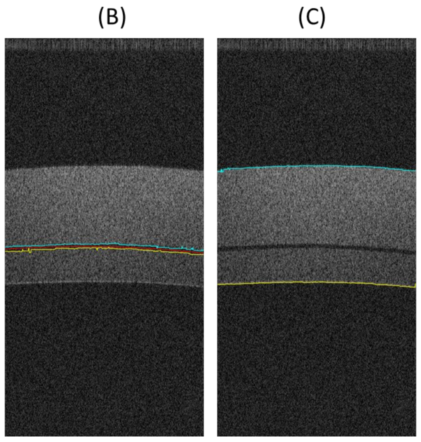


Fig. 7. Depiction of intermediate results to determine thicknesses. (A) Intermediate results following the procedure to detect the middle layer gap. From left to right: A raw vertical slice (B-scan) image before analysis, responses of *Ra*, *Rb*, and the summed *R* for *h = 9*, respectively. See text for details. (B) Pixel intensity detection of the final middle gap location (red) and upper (cyan) and lower bounds (yellow). (C) Pixel intensity detection of the upper bound of the top layer (cyan) and lower bound of the bottom layer (yellow).

7. Results and Discussion

7.1. Measurement of layer thicknesses determined by OCT

Layer thickness values analyzed from the experimental OCT data of SRM 002 by the analysis code are shown in Table 2. Table 1 lists raw layer thickness values and raw uncertainties (as acquired with $n = 1$). The final uncertainties were propagated using these values and those described above for the final certified thickness values and uncertainties for the SRMs. See Uncertainty Analysis section for discussion of the final uncertainty calculations. The same measurements and analyses were repeated for all 200 samples, and an example of the certificate for SRMs including a table for the certified thickness values and uncertainties for SRM 2 is available in Appendix 1.

7.2. Uncertainty Analysis

For each SRM, the mean thickness values of the layers are provided. To estimate the uncertainty values for each layer thickness, the following procedure was used.

All thickness measurements were first determined in pixels with $n = 1$, then the pixels were converted to metric units as follows:

$$T_m = \frac{T_p * lpp}{n}$$

where T_m is the measured thickness in metric units (microns), $T_p$is the measured thickness in pixels from the data analysis, lpp is the conversion factor (as calibrated by a NIST traceable step height standard as described in the 'Calibration of the SD-OCT Axial Scale' section) from a single axial pixel to the length in micrometer, and n is the measured refractive index (from the 'Refractive Index Determination' section). Then, δ_m, the uncertainty of T_m is :

$$\delta_m = |T_m| * \sqrt{\left(\frac{\delta_{Tp}}{T_p}\right)^2 + \left(\frac{\delta_{lpp}}{lpp}\right)^2 + \left(\frac{\delta_n}{n}\right)^2}$$

where δ_{Tp}, δ_{lpp}, and δ_n are uncertaintities associated with T_p, *lpp* and n, respectively. The budgets of these uncertainties contributing to the final thickness uncertainty, their origin, and determination methods are summarized in Table 3. The final uncertainties of thickness values are calculated and included in the certified values summarized in each SRM's certificate.

Sample ROI		Pixels Intensity Based mean (μ) and standard deviation (σ) output from the analysis code	
		Raw μ (pixel)	Raw σ (pixel)
1 (3C)	Top	159	2
	Middle	20	2
	Bottom	193	2
2 (3D)	Top	160	2
	Middle	20	2
	Bottom	192	2
3 (3E)	Top	161	2
	Middle	20	3
	Bottom	192	3
4 (4C)	Top	157	2
	Middle	19	2
	Bottom	194	1
5 (4D)	Top	159	1
	Middle	19	1
	Bottom	193	1
6 (4E)	Top	160	1
	Middle	19	1
	Bottom	193	1
7 (5C)	Top	157	3
	Middle	19	2
	Bottom	196	2
8 (5D)	Top	160	2
	Middle	19	1
	Bottom	196	1
9 (5E)	Top	162	2
	Middle	19	1
	Bottom	195	2

Table 1. Raw layer thickness values in pixels analyzed from the OCT data of SRM 002 by the analysis code**.** Listed are the mean and *k* =1 standard deviation values in pixels obtained for each layer thickness for each of the 9 ROIs (i.e., the ROI numbers (1, 2, 3, 4, 5, 6, 7, 8, 9) corresponding to each square region (3C, 3D, 3E, 4C, 4D, 4E, 5C, 5D, 5E) in Fig.1, respectively). The analysis method is described in the Data Analysis section of this document.

Sample ROI		Mean thickness(μ) and the final thickness uncertainty (ε)	
		μ (μm)	ε (μm)
1 (3C)	Top	121	2
	Middle	15	2
	Bottom	147	2
2 (3D)	Top	122	2
	Middle	15	2
	Bottom	147	2
3 (3E)	Top	123	2
	Middle	15	2
	Bottom	146	3
4 (4C)	Top	120	2
	Middle	15	1
	Bottom	148	2
5 (4D)	Top	121	1
	Middle	15	1
	Bottom	148	2
6 (4E)	Top	122	1
	Middle	15	1
	Bottom	147	2
7 (5C)	Top	119	2
	Middle	15	1
	Bottom	150	2
8 (5D)	Top	122	2
	Middle	15	1
	Bottom	150	2
9 (5E)	Top	124	2
	Middle	15	1
	Bottom	149	2

Table 2. Mean layer thickness values analyzed from the OCT data of SRM 002 by the analysis code and the final uncertainty values as calculated by the procedure described in the Uncertainty Analysis section for each of the 9 ROIs (i.e., the ROI numbers (1, 2, 3, 4, 5, 6, 7, 8, 9) corresponding to each square region (3C, 3D, 3E, 4C, 4D, 4E, 5C, 5D, 5E) in Fig.1, respectively).

Uncertainty	Origin	Determination method	Value
δ_{Tp}	Uncertainty of thickness values in pixels over a region of interest	Standard deviation of the layer thicknesses in pixels calculated by the analysis algorithm	Determined for each ROI from algorithm (k = 1)
δ_{lpp}	Uncertainty of conversion factor of micrometer per pixel	Standard deviation of the conversion factor from pixels to microns from NIST traceable standard	0.009 (k = 2)
δ_n	Measurement accuracy of the refractive index	Standard deviation of refractive index values from multiple measurements	0.002 (k =1)

Table 3. Budgets of the uncertainties contributing to the final uncertainty of the layer thicknesses.

8. Summary

Calibration of medical imaging devices ensures accuracy and precision of the measurements. NIST has developed a SRM to enable on-site calibration of OCT. The SRM consists of three PDMS layers, where a thin clear layer created by spin casting is sandwiched between two thick scattering layers made by a blade coating technique. The thickness of each layer was measured by a calibrated SD-OCT where the axial dimensional in the PDMS material was corrected by the refractive index as determined by spectroscopic ellipsometry. For automated analysis of multiple SRMs, we have also developed an algorithm which detects the interfacial planes between layers from the 3D tomography data and measures the local layer thicknesses at each pixel across the entire lateral dimension of the tomograph. The scattering layers in the SRMs have uniform thicknesses with the standard deviations less than 5% across the entire characterized sample area. The certificate of analysis for an individual SRM is provided, listing the mean and standard deviation values for each layer thickness in a table for the 9 ROIs each corresponding to a 2 mm x 2 mm region. The certified thickness values from one or more of the ROIs can be used to check and correct the calibration of the end user's imaging device. Depending on the device, the discrepancy needs to be corrected in the image acquisition and/or data analysis software. The certificate provides layer thicknesses values analyzed by a pixel intensity-based model described in the previous section. Although the target application of our SRMs is OCT, the SRMs may also be used to demonstrate the measurement capability and to calibrate measured optical properties in other depth-resolving 3D optical systems, such as confocal microscopy and wide-field 3D microscopy.

References

[1] J. N. Lee, C. Park, and G. M. Whitesides, "Solvent compatibility of poly(dimethylsiloxane)-based microfluidic devices," *Analytical Chemistry,* vol. 75, pp. 6544-6554, Dec 1 2003.

[2] J. Hwang, J. C. Ramella-Roman, and R. Nordstrom, "Introduction: Feature Issue on Phantoms for the Performance Evaluation and Validation of Optical Medical Imaging Devices," *Biomedical Optics Express,* vol. 3, pp. 1399-1403, Jun 1 2012.

[3] B. W. Pogue and M. S. Patterson, "Review of tissue simulating phantoms for optical spectroscopy, imaging and dosimetry," *Journal of Biomedical Optics,* vol. 11, Jul-Aug 2006.

[4] P. H. Tomlins, P. Woolliams, M. Tedaldi, A. Beaumont, and C. Hart, "Measurement of the three-dimensional point-spread function in an optical coherence tomography imaging system - art. no. 68472Q," *Coherence Domain Optical Methods and Optical Coherence Tomography in Biomedicine XII,* vol. 6847, pp. Q8472-Q8472, 2008.

[5] J. J. J. Dirckx, L. C. Kuypers, and W. F. Decraemer, "Refractive index of tissue measured with confocal microscopy," *Journal of Biomedical Optics,* vol. 10, Jul-Aug 2005.

[6] J. C. Erie, J. W. McLaren, and S. V. Patel, "Confocal Microscopy in Ophthalmology," *American Journal of Ophthalmology,* vol. 148, pp. 639-646, Nov 2009.

[7] P. Lemaillet, J. Hwang, H. Wabnitz, D. Grosenick, L. Yang, and D. W. Allen, "Broadband spectral measurements of diffuse optical properties by an integrating sphere instrument at the National Institute of Standards and Technology," *Design and Quality for Biomedical Technologies X,* vol. 10056, 2017.

[8] P. H. Tomlins and R. K. Wang, "Simultaneous analysis of refractive index and physical thickness by Fourier domain optical coherence tomography," *IEEE Proceedings-Optoelectronics,* vol. 153, pp. 222-228, Oct 2006.

[9] Theory of Nonlinear Curve Fitting, accessed 11/07/2021 from https://www.originlab.com/doc/Origin-Help/NLFit-Theory#How_Origin_Fits_the_Curve

[10] P. de Groot and D. Fitzgerald, “Measurement, certification, and use of step-height calibration specimens in optical metrology,” *Proc SPIE 10329, Optical Measurement Systems for Industrial Inspection X*, 1032919, June 2017.

[11] I. B. Couceiro, T. F. da Silva, L. v. G. Tarelho, C. L. S. Azeredo, I. Malinovski, H. P. H. Grieneisein, W. S. Barros, G. V. Faria, J. P. von der Weid, M. M. Amaral, M. P. Raele, and A. Z. de Freitas, "Development of traceability methodology for optical coherence tomography (OCT) using step height standard as calibration reference", *Proc. SPIE 8082, Optical Measurement Systems for Industrial Inspection VII*, 80822P, May 2011.

[12] G. Min, W. J. Choi, J. W. Kim, and B. H. Lee, "Refractive index measurements of multiple layers using numerical refocusing in FF-OCT," *Optics Express,* vol. 21, pp. 29955-29967, Dec 2, 2013.

[13] C. Meichner, A. E. Schedl, C. Neuber, K. Kreger, H. W. Schmidt, and. L. Kador, “Refractive-index determination of solids from first- and second-order critical diffraction angles of periodic surface patterns,” *AIP Advances*, vol. 5, 087135, 2015.

[14] F. Schneider, J. Draheim, R. Kamberger, U. Wallrabe, “Process and material properties of polydimethylsiloxane (PDMS) for Optical MEMS,” *Sensors and Actuators A: Physical*, vol. 151, pp. 95-99, 2009.

[15] Dow Corning Sylgard 184 Silicone Elastomer Technical Data Sheet, from https://www.dow.com/en-us/document-

viewer.html?ramdomVar=9006169780984819072&docPath=/content/dam/dcc/documents/en-us/productdatasheet/11/11-31/11-3184-sylgard-184-elastomer.pdf

[16] A. Cai, W. Qiu, G. Shao, and W. Wang, "A new fabrication method for all-PDMS waveguides," *Sensors and Actuators A: Physical*, vol. 204, pp. 44-47, 2013.

[17] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proc IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. TR-2004, pp. 1-9, May 2004.

[18] N. Otsu, "Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems Man and Cybernetics,* vol. 9, pp. 62-66, 1979.

[19] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging,* vol. 13, pp. 146-168, Jan 2004.

APPENDIX A

Appendix A is a collection of MATLAB programs for batch job of the thicknesses analyses on multiple OCT data cubes. Details of the algorithm are described in Data Analysis section of the SP 260 document.

<Main Program – main_batch.m>

```
clearvars; close all; clc;
addpath('./functions');
iptsetpref('ImshowBorder','tight')

%% Measure the thickness of the gap in OCT images
% [Note] Axes are defined as follows:
%   |- Each frame is X-Z view (origin at top-left corner of the image;
%   x direction to the right; z direction to down)
%   |- Y direction is the order of the frames
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--- Inputs ---%
% input file list: "list.txt" is a list of folders, each folder
% contains a series of B scan images of a sample in a 'tiff' format

list_file = '../list.txt';
img_ext = 'TIFF';

% debug mode
debug_mode = 1; % '1' to see interim results; '0' to skip

% save path for results
bSaveResults = 1; % Set '1' to save interim results; '0' not to save them
save_root_path = '../data/SRM002/';
save_fig_type = 'png'; % Other options include 'bmp', 'jpeg', 'tiff', 'epsc', etc.;
% if empty (i.e., []), only default 'fig' files will be saved

%--- Parameters ---%
% Haar-like features for gap detection (function 'DetectGap')
haar_filter_width = 31;
haar_filter_gap_height = 3:2:15; % possible thickness range of the gap; odd number
haar_filter_layer_height = 7; % thickness to estimate layer surfaces

% Upper and lower bounds of the gap (function 'FindGapBounds')
hist_window = [3 31]; % window size for local histogram computation
hist_st = 0.05; % bin size for local histogram
Lz = 15;

% Thickness (function 'MeasureThickness')
len_per_pix_x = 10; %scan length (um) / number of pixels in x (6mm/600)
len_per_pix_y = 10; %scan length (um) / number of pixels in y (6mm/600)
len_per_pix_z = 1.074; % from calibration by step-height standard
len_per_pix_x_std = 1; % 1 for convenience, does not affect the axial
len_per_pix_y_std = 1; % 1 for convenience, does not affect the axial
```

```
len_per_pix_z_std = 0.009; % from calibration by step-height standard

ref_ind = 1.408;
ref_ind_std = 0.002;
thickness_unit = 'um';

% Contrast per slice
z_top_crop = 40; % remove the top pixels in z from contrast computation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid = fopen(list_file, 'rt');
list = textscan(fid, '%s');
fclose(fid);
list = list{1};
N = length(list);

if bSaveResults
    if ~isdir(save_root_path)
        mkdir(save_root_path);
    end
end

for l = 1:N
    [data_path,file_name,~] = fileparts(list{l});
    fprintf('--- %s (%d/%d) ---\n', file_name, l, N);

    % Save path
    if bSaveResults
        save_path = fullfile(save_root_path,file_name);
        if ~isdir(save_path)
            mkdir(save_path);
        end
    end

    proc_time = struct; % processing time log

    %--- Load images ---%
    tic;
    Z = LoadImages(fullfile(data_path,file_name),img_ext);
    proc_time.LoadData = toc;

    %--- Detect the gap in OCT images ---%
    tic;
    BW_Line = DetectGap(Z, haar_filter_width, haar_filter_gap_height, debug_mode,
save_path);
    proc_time.DetectGap = toc;

    %--- Detect the upper bound of top layer and the lower bound of bottom layer --
-%
    tic;
    [BW_UpBound, BW_LowBound] = DetectLayers(Z, BW_Line, haar_filter_width,
haar_filter_layer_height, debug_mode, save_path);
    proc_time.DetectLayer = toc;
    %--- Find the upper and lower bounds of the gap ---%
    tic;
    GapBoundInfo = FindGapBounds(Z, BW_Line, hist_window, hist_st, Lz, debug_mode, 
save_path);
    proc_time.FindGapBounds = toc;

    %--- Find the upper bound of the top layer and the lower bound of the bottom 
layer ---%
    tic;
    [TopBoundInfo, BottomBoundInfo] = FindBounds_TopBottomLayers(Z, BW_UpBound, 
BW_LowBound, hist_window, hist_st, Lz, debug_mode, save_path);
    proc_time.FindLayerBounds = toc;

    %--- Estimate the tilt angle ---%
    tic;
    TiltAngle = EstimateTilt(Z, GapBoundInfo, TopBoundInfo, BottomBoundInfo, 
len_per_pix_x, len_per_pix_y, len_per_pix_z);
    proc_time.EstimateTilt = toc;

    %--- Measure the thickness of the gap ---%
    tic;
    GapThickness = MeasureGapThickness(Z, GapBoundInfo, len_per_pix_x, 
len_per_pix_z, ref_ind, TiltAngle);
    proc_time.MeasureGapThickness = toc;

    %--- Measure the thickness of top and bottom layers ---%
    tic;
    [TopThickness, BottomThickness] = MeasureLayerThickness(Z, GapBoundInfo, 
TopBoundInfo, BottomBoundInfo, len_per_pix_x, len_per_pix_z, ref_ind, TiltAngle);
    proc_time.MeasureLayerThicknesses = toc;

    fprintf('Processing time [s]:\n');
    disp(struct2table(proc_time));


    %--- Analysis results ---%
    % mean thickness in 3 by 3 regions
    GapThickness3by3 = GetStatistics(GapThickness, ref_ind, ref_ind_std, 
debug_mode, save_path, 'GapThickness');
    TopThickness3by3 = GetStatistics(TopThickness, ref_ind, ref_ind_std, 
debug_mode, save_path, 'TopLayerThickness');
    BottomThickness3by3 = GetStatistics(BottomThickness, ref_ind, ref_ind_std, 
debug_mode, save_path, 'BottomLayerThickness');

end
```

<Function - ConvertImage2Metric>

```
%%% Convert a point in image coordinates to metric coordinates %%%
% Inputs
%   |- p: location of a point in image coordinates (one dimension)
%   |- lpp: length per pixel in the corresponding axis
% Output
%   |- m: the point in metric coordinates

function m = ConvertImage2Metric(p, lpp)

m = (p-1)*lpp;
```

<Function - DetectGap>

```
%%% Detect the gap in OCT images %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- w: width of Haar-like features
%   |- h: height of Haar-like features
%   |- debug_mode: '1' to show interim results; '0' to skip
%   |- save_path: assign a path to save interim results; empty field not to save
them
% Output
%   |- BW_Line: binary 3D matrix; '1' for detected gap location, '0',
%                 otherwise

function BW_Line = DetectGap(Z, w, h, debug_mode, save_path)

% Defaults
if nargin < 3
    error('Inputs of Z, w, and h are required.');
elseif nargin < 4
    debug_mode = 0;
    save_path = [];
elseif nargin < 5
    save_path = [];
end

if ~isempty(save_path)
    if ~isdir(fullfile(save_path,'GapDetection'))
        mkdir(fullfile(save_path,'GapDetection'));
    end
end

% Normalize the image to [0,1]
Z = double(Z)/double(intmax(class(Z)));
[D,W,H] = size(Z);

% Haar-like features for horizontal lines
w2 = floor(w*0.5);

Ax = ones(1,W);
hx = ones(1,w);
Ax = imfilter(Ax, hx);

% Display progress
na = 30;
ystr = round(linspace(1,H,na));
astr = repmat(' ',1,na);
nastr = na;
```

```
fprintf('Computing Haar-like features... [*%s]',astr);

R = zeros(D,W,H);
for y = 1:H
    if ~isempty(find(ystr==y))
        fprintf(repmat('\b',1,nastr+1));
        nastr = nastr-1;
        astr = repmat(' ',1,nastr);
        fprintf('*%s]',astr);
    end

    % integral image
    I = Z(:,:,y);

    IS = zeros(D+1,W+w);
    I_sum = cumsum(I);
    I_sum = cumsum(I_sum,2);
    IS(2:end,w2+1+1:w2+1+W) = I_sum;
    IS(2:end,end-w2+1:end) = I_sum(:,end)*ones(1,w2);
    clear I_sum

    % haar-like features
    R_slice = zeros(D,W);
    for kt = 1:length(h)
        t = h(kt);
        t2 = floor(t*0.5);

        % sub-regions
        Ia = zeros(D,W);
        Ia(t+t2+1:end,:) = IS(1:end-(t+t2+1),1:W);
        Ib = zeros(D,W);
        Ib(t+t2+1:end,:) = IS(1:end-(t+t2+1),w+1:w+W);
        Ic = zeros(D,W);
        Ic(t+t2+1:end,:) = IS(t+1:end-(t2+1),1:W);
        Id = zeros(D,W);
        Id(t+t2+1:end,:) = IS(t+1:end-(t2+1),w+1:w+W);
        Ie = zeros(D,W);
        Ie(t+t2+1:end-t2,:) = IS(2*t+1:end,1:W);
        Ie(end-t2+1:end,:) = ones(t2,1)*Ie(end-t2,:);
        If = zeros(D,W);
        If(t+t2+1:end-t2,:) = IS(2*t+1:end,w+1:w+W);
        If(end-t2+1:end,:) = ones(t2,1)*If(end-t2,:);
        Ig = zeros(D,W);
        Ig(t+t2+1:end-(t+t2),:) = IS(3*t+1:end,1:W);
        Ig(end-(t+t2)+1:end,:) = ones(t+t2,1)*Ig(end-(t+t2),:);
        Ih = zeros(D,W);
        Ih(t+t2+1:end-(t+t2),:) = IS(3*t+1:end,w+1:w+W);
        Ih(end-(t+t2)+1:end,:) = ones(t+t2,1)*Ih(end-(t+t2),:);

        % areas
        hy = ones(t,1);
        Ay = [zeros(t,1); ones(D,1)];
        Ay = imfilter(Ay, hy);
        Ay = Ay(1:D);
        A1 = Ay*Ax;

        Ay = ones(D,1);
        Ay = imfilter(Ay, hy);
        A2 = Ay*Ax;

        Ay = [ones(D,1); zeros(t,1)];
        Ay = imfilter(Ay, hy);
        Ay = Ay(t+1:t+D);
        A3 = Ay*Ax;

        % region 1 (+); region 2 (-); region 3 (+)
        R1 = Ia + Id - Ib - Ic;
        R2 = Ic + If - Id - Ie;
        R3 = Ie + Ih - If - Ig;

        R1 = R1./A1;
        R2 = R2./A2;
        R3 = R3./A3;

        R1(A1==0) = NaN;
        R2(A2==0) = NaN;
        R3(A3==0) = NaN;

        % haar responses
        f_haar1 = R1-R2;
        f_haar2 = R3-R2;

        f_haar1(f_haar1 <= 0) = NaN;
        f_haar2(f_haar2 <= 0) = NaN;

        % dark line (gap) response
        f_line = f_haar1+f_haar2;

        R_slice = R_slice + f_line;
    end
    R(:,:,y) = R_slice;
end
fprintf(' Done.\n');


% Binarize the line response and find the largest connected component
fprintf('Binarizing the line response...');
BW_Line = FindLargestCC(R);
fprintf(' Done.\n');


% Display & Save
if debug_mode || ~isempty(save_path)
    fprintf('Displaying the detected gap...');

    fig1 = figure(1);
    fig1.PaperPositionMode = 'auto';
    fig1.PaperSize = [W D];
    for y = 1:H
        ix = find(BW_Line(:,:,y));
        I = Z(:,:,y);
        Ir = I; Ir(ix) = 0;
        Ig = I; Ig(ix) = 1;
        Ib = I; Ib(ix) = 0;
        Ic = cat(3,Ir,Ig,Ib);

        figure(fig1);
        set(fig1, 'Name', [num2str(y) '/' num2str(H)]);
        imshow(Ic);
        drawnow

        if ~isempty(save_path)
            print(fig1, fullfile(save_path,'GapDetection',num2str(y,'%04d')), '-
dpng');
        end

        pause(0.05);
        clf(fig1);
    end
    fprintf(' Done.\n');
end
```

<Function - DetectLayers>

```
%%% Detect the upper bound of top layer and lower bound of bottom layer in OCT
images %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- BW_Line: 3D binary mask for gap
%   |- w: width of Haar-like features
%   |- h: height of Haar-like features
%   |- debug_mode: '1' to show interim results; '0' to skip
%   |- save_path: assign a path to save interim results; empty field not to save
them
% Output
%   |- BW_UpBound: binary 3D matrix for upper bound of the top layer
%   |- BW_LowBound: binary 3D matrix for lower bound of the bottom layer

function [BW_UpBound, BW_LowBound] = DetectLayers(Z, BW_Line, w, h, debug_mode,
save_path)

% Defaults
if nargin < 4
    error('Inputs of Z, BW_Line, w, and h are required.');
elseif nargin < 5
    debug_mode = 0;
    save_path = [];
elseif nargin < 6
    save_path = [];
end

if ~isempty(save_path)
```

```
    if ~isdir(fullfile(save_path,'LayerDetection'))
        mkdir(fullfile(save_path,'LayerDetection'));
    end
end

% Normalize the image to [0,1]
Z = double(Z)/double(intmax(class(Z)));
[D,W,H] = size(Z);

% Haar-like features for horizontal lines
w2 = floor(w*0.5);

Ax = ones(1,W);
hx = ones(1,w);
Ax = imfilter(Ax, hx);

% Display progress
na = 30;
ystr = round(linspace(1,H,na));
astr = repmat(' ',1,na);
nastr = na;

fprintf('Computing Haar-like features... [*%s]',astr);

R = zeros(D,W,H);
R_up = zeros(D,W,H);
R_low = zeros(D,W,H);
for y = 1:H
    if ~isempty(find(ystr==y))
        fprintf(repmat('\b',1,nastr+1));
        nastr = nastr-1;
        astr = repmat(' ',1,nastr);
        fprintf('*%s]',astr);
    end

    % integral image
    I = Z(:,:,y);

    IS = zeros(D+1,W+w);
    I_sum = cumsum(I);
    I_sum = cumsum(I_sum,2);
    IS(2:end,w2+1+1:w2+1+W) = I_sum;
    IS(2:end,end-w2+1:end) = I_sum(:,end)*ones(1,w2);
    clear I_sum

    % haar-like features
    h2 = floor(h*0.5);

    % sub-regions
    Ia = zeros(D,W);
    Ia(h+h2+1:end,:) = IS(1:end-(h+h2+1),1:W);
    Ib = zeros(D,W);
    Ib(h+h2+1:end,:) = IS(1:end-(h+h2+1),w+1:w+W);
    Ic = zeros(D,W);
    Ic(h+h2+1:end,:) = IS(h+1:end-(h2+1),1:W);
    Id = zeros(D,W);
    Id(h+h2+1:end,:) = IS(h+1:end-(h2+1),w+1:w+W);
    Ie = zeros(D,W);
    Ie(h+h2+1:end-h2,:) = IS(2*h+1:end,1:W);
    Ie(end-h2+1:end,:) = ones(h2,1)*Ie(end-h2,:);
    If = zeros(D,W);
    If(h+h2+1:end-h2,:) = IS(2*h+1:end,w+1:w+W);
    If(end-h2+1:end,:) = ones(h2,1)*If(end-h2,:);
    Ig = zeros(D,W);
    Ig(h+h2+1:end-(h+h2),:) = IS(3*h+1:end,1:W);
    Ig(end-(h+h2)+1:end,:) = ones(h+h2,1)*Ig(end-(h+h2),:);
    Ih = zeros(D,W);
    Ih(h+h2+1:end-(h+h2),:) = IS(3*h+1:end,w+1:w+W);
    Ih(end-(h+h2)+1:end,:) = ones(h+h2,1)*Ih(end-(h+h2),:);

    % areas
    hy = ones(h,1);
    Ay = [zeros(h,1); ones(D,1)];
    Ay = imfilter(Ay, hy);
    Ay = Ay(1:D);
    A1 = Ay*Ax;

    Ay = ones(D,1);
    Ay = imfilter(Ay, hy);
    A2 = Ay*Ax;

    Ay = [ones(D,1); zeros(h,1)];
    Ay = imfilter(Ay, hy);
    Ay = Ay(h+1:h+D);
    A3 = Ay*Ax;

    % region 1 (+); region 2 (-); region 3 (+)
    R1 = Ia + Id - Ib - Ic;
    R2 = Ic + If - Id - Ie;
    R3 = Ie + Ih - If - Ig;

    R1 = R1./A1;
    R2 = R2./A2;
    R3 = R3./A3;

    R1(A1==0) = NaN;
    R2(A2==0) = NaN;
    R3(A3==0) = NaN;

    r_up = -R1-R2+2*R3;
    r_low = 2*R1-R2-R3;

    r_up(r_up<=0) = NaN;
    r_low(r_low<=0) = NaN;

    R_up(:,:,y) = r_up;
    R_low(:,:,y) = r_low;
end
fprintf(' Done.\n');
```

```
% Binarize the line response and find the largest connected component
fprintf('Binarizing the line response...');
BW_UpBound = FindLargestCC(R_up, BW_Line);
BW_LowBound = FindLargestCC(R_low, BW_Line);
fprintf(' Done.\n');


% Display & Save
if debug_mode || ~isempty(save_path)
    fprintf('Displaying the detected surface layers...');

    fig1 = figure(1);
    fig1.PaperPositionMode = 'auto';
    fig1.PaperSize = [W D];

    for y = 1:H
        ix1 = find(BW_UpBound(:,:,y));
        ix2 = find(BW_LowBound(:,:,y));

        I = Z(:,:,y);
        Ir = I; Ir(ix1) = 0; Ir(ix2) = 1;
        Ig = I; Ig(ix1) = 1; Ig(ix2) = 1;
        Ib = I; Ib(ix1) = 1; Ib(ix2) = 0;
        Ic = cat(3,Ir,Ig,Ib);

        figure(fig1);
        set(fig1, 'Name', [num2str(y) '/' num2str(H)]);
        imshow(Ic);
        drawnow

        if ~isempty(save_path)
            print(fig1, fullfile(save_path,'LayerDetection',num2str(y,'%04d')), '-
dpng');
        end

        pause(0.05);
        clf(fig1);
    end
    fprintf(' Done.\n');
end
```

<Function - EstimateTilt>

```
%%% Estimate tilt angle of the layers %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- Bnd_gap: a struct file with gap information
%   |- Bnd_up: a struct file with top layer information
%   |- Bnd_low: a struct file with bottom layer information
%   |- lpp_x: length per pixel in x-axis
%   |- lpp_y: length per pixel in y-axis
%   |- lpp_z: length per pixel in z-axis
% Output
%   |- tilt: estimated tilt angle

function tilt = EstimateTilt(Z, Bnd_gap, Bnd_up, Bnd_low, lpp_x, lpp_y, lpp_z)

[D,W,H] = size(Z);
[x,y] = meshgrid(0:lpp_x:lpp_x*(W-1),0:lpp_y:lpp_y*(H-1));
th = zeros(1,4);

% Fit a 3D plane to the gap upper bound
A = [x(:) y(:) ConvertImage2Metric(Bnd_gap.UpperPixelLocation(:), lpp_z)
ones(W*H,1)];
[V,D] = eig(A'*A);
[~,ind] = min(diag(D));
n_gu = V(1:3,ind); % normal vector of the plane
th(1) = acos(abs(n_gu(2))/norm(n_gu)) - pi/2; % angle of the plane

% Fit a 3D plane to the gap upper bound
A = [x(:) y(:) ConvertImage2Metric(Bnd_gap.LowerPixelLocation(:), lpp_z)
ones(W*H,1)];
[V,D] = eig(A'*A);
[~,ind] = min(diag(D));
n_gl = V(1:3,ind);
th(2) = acos(abs(n_gl(2))/norm(n_gl)) - pi/2;

% Fit a 3D plane to upper surface of top layer
A = [x(:) y(:) ConvertImage2Metric(Bnd_up.UpperPixelLocation(:), lpp_z)
ones(W*H,1)];
[V,D] = eig(A'*A);
[~,ind] = min(diag(D));
n_tu = V(1:3,ind);
th(3) = acos(abs(n_tu(2))/norm(n_tu)) - pi/2;

% Fit a 3D plane to lower surface of bottom layer
A = [x(:) y(:) ConvertImage2Metric(Bnd_low.LowerPixelLocation(:), lpp_z)
ones(W*H,1)];
[V,D] = eig(A'*A);
[~,ind] = min(diag(D));
n_bl = V(1:3,ind);
th(4) = acos(abs(n_bl(2))/norm(n_bl)) - pi/2;

% Find the average angle
tilt = mean(th);
```

<Function - FindBounds_TopBottomLayers>

```
%%% Find the upper bound of the top layer and lower bound of the bottom layer %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- BW_up: binary 3D matrix indicating upper bound location
%   |- BW_low: binary 3D matrix indicating lower bound location
%   |- wnd: window size for local histogram computation
%   |- d: bin size for the histogram
%   |- Lz: search range from the gap location; +/- Lz pixels in z-axis
%   |- debug_mode: '1' to show interim results; '0' to skip
%   |- save_path: assign a path to save interim results; empty field not to save
them
% Output
%   |- Bnd_up: a struct file with the bound information
%       |- CenterPolyFit: estimated upper bound location after polynomial fitting
%       |- CenterPolyParam: parameters of polynomial fit of the upper bound
%       |- UpperPixelLocation: pixel location of the upper bound of the top layer
%   |- Bnd_low: a struct file with the bound information
%       |- CenterPolyFit: estimated lower bound location after polynomial fitting
%       |- CenterPolyParam: parameters of polynomial fit of the lower bound
%       |- LowerPixelLocation: pixel location of the lower bound of the bottom
layer
%   |- Estimated bound information saved at
%       |- <save_path>/LayerBound/PixelwiseDetection

function [Bnd_up, Bnd_low] = FindBounds_TopBottomLayers(Z, BW_up, BW_low, wnd, d,
Lz, debug_mode, save_path)

% Defaults
if nargin < 6
    error('Inputs of Z, BW_up, BW_low, wnd, d, and Lz are required.');
elseif nargin < 7
    debug_mode = 0;
    save_path = [];
elseif nargin < 8
    save_path = [];
end

if ~isempty(save_path)
    if ~isdir(fullfile(save_path,'LayerBound','PixelwiseDetection'))
        mkdir(fullfile(save_path,'LayerBound','PixelwiseDetection'));
    end
end

[D,W,H] = size(Z);

% Find the bounds of top and bottom layers
Bnd_up = FindUpperBoundTopLayer(Z, BW_up, wnd, d, Lz);
Bnd_low = FindLowerBoundBottomLayer(Z, BW_low, wnd, d, Lz);

% Display and Save
if debug_mode || ~isempty(save_path)
    fig1 = figure(1);
    fig1.PaperPositionMode = 'auto';
    fig1.PaperSize = [W D];

    for y = 1:H
        I = Z(:,:,y);

        % detected upper bound of top layer and lower bound of bottom layer
        figure(fig1);
        set(fig1, 'Name', [num2str(y) '/' num2str(H)]);
        imshow(I);
        hold on
        plot(1:W, Bnd_up.UpperPixelLocation(y,:), 'c-');
        plot(1:W, Bnd_low.LowerPixelLocation(y,:), 'y-');
        hold off
        drawnow

        if ~isempty(save_path)
            print(fig1,
fullfile(save_path,'LayerBound','PixelwiseDetection',num2str(y,'%04d')), '-dpng');
        end

        pause(0.05);
        clf(fig1);
    end
end
```

<Function - FindGapBounds>

```
%%% Find the upper and lower bounds of the gap %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- BW_L: binary 3D matrix indicating gap location
%   |- wnd: window size for local histogram computation
%   |- d: bin size for the histogram
%   |- Lz: search range from the gap location; +/- Lz pixels in z-axis
%   |- debug_mode: '1' to show interim results; '0' to skip
%   |- save_path: assign a path to save interim results; empty field not to save
them
% Output
%   |- Bnd: a struct file with the gap information
%       |- CenterPolyFit: estimated gap location after polynomial fitting
%       |- CenterPolyParam: parameters of polynomial fit of the gap
%       For 'Pixel':
%       |- UpperPixelLocation: pixel location of the upper bound of the gap
%       |- LowerPixelLocation: pixel location of the lower bound of the gap

function Bnd = FindGapBounds(Z, BW_L, wnd, d, Lz, debug_mode, save_path)

% Defaults
if nargin < 5
    error('Inputs of Z, BW_L, wnd, d, and Lz are required.');
elseif nargin < 6
    debug_mode = 0;
    save_path = [];
elseif nargin < 7
    save_path = [];
end

if ~isempty(save_path)
    if ~isdir(fullfile(save_path,'GapBound','PixelwiseDetection'))
        mkdir(fullfile(save_path,'GapBound','PixelwiseDetection'));
    end
end

% Normalize the image to [0,1]
Z = double(Z)/double(intmax(class(Z)));
[D,W,H] = size(Z);
```

```
% Fit a second-order polynomial to the detected gap
fprintf('Fitting a second-order polynomial...');

% mesh points
[mx,my] = meshgrid(1:W,1:H);
mA = [ones(numel(mx),1) mx(:) my(:) mx(:).^2 my(:).^2 mx(:).*my(:)];

% least-squares estimation
ind = find(BW_L);
[pz,px,py] = ind2sub([D,W,H], ind);

A = [ones(length(px),1) px py px.^2 py.^2 px.*py];
a_ctr = pinv(A)*pz;

bA = [ones(numel(mx),1) mx(:) my(:) mx(:).^2 my(:).^2 mx(:).*my(:)];
fz_ctr = bA*a_ctr;
fz_ctr = reshape(fz_ctr, [H W]);

fprintf(' Done.\n');


% Find the upper and lower bounds
fprintf('Find the upper and lower bounds...');

% fitted line location
mz = mA*a_ctr;
mz = reshape(mz, [H W]);

% local histogram
wnd_h2 = floor(wnd(1)/2);
b = 0:d:1+d; % bins
b3 = reshape(b(1:end-1), 1,1,[]);

z_b1 = zeros(H,W);
z_b2 = zeros(H,W);
for y = 1:H
    I = Z(:,:,y);

    z1 = max(min(floor(mz(y,:)))-2*Lz, 1);
    z2 = min(max(ceil(mz(y,:)))+2*Lz, D);

    I_seg = I(z1:z2,:);
    I_hist = ImageHistogramFeature2D(I_seg, wnd, b);
    [h_hist,w_hist,~] = size(I_hist);

    % mean intensity value
    I_mean = sum(I_hist .* repmat(b3,h_hist,w_hist), 3);

    for x = 1:W
        kz0 = round(mz(y,x))-z1+1;

        % upper bound
        kz = kz0-Lz:kz0-wnd_h2;
        dist = zeros(length(kz),1);
        ind = find(squeeze(I_mean(kz,x,:)) > squeeze(I_mean(kz+wnd(1),x,:)));
        % top: bright, bottom: dark
        for cnt = 1:length(ind)
            dist(ind(cnt)) = sum((I_hist(kz(ind(cnt)),x,:)-
I_hist(kz(ind(cnt))+wnd(1),x,:)).^2);
        end
        % dist(1:wnd_h2,:) = 0;
        [~,idx] = max(dist);
        z_b1(y,x) = kz(idx)+wnd_h2+0.5 + z1-1;

        % lower bound
        kz = kz0+wnd_h2+1:kz0+wnd_h2+Lz;
        dist = zeros(length(kz),1);
        ind = find(squeeze(I_mean(kz-wnd(1),x,:)) < squeeze(I_mean(kz,x,:)));
        % top: dark, bottom: bright
        for cnt = 1:length(ind)
            dist(ind(cnt)) = sum((I_hist(kz(ind(cnt))-wnd(1),x,:)-
I_hist(kz(ind(cnt)),x,:)).^2);
        end
        % dist(end-wnd_h2+1:end,:) = 0;
        [~,idx] = max(dist);
        z_b2(y,x) = kz(idx)-wnd_h2-0.5 + z1-1;
    end
end

fprintf(' Done.\n');

Bnd = struct;
Bnd.CenterPolyFit = fz_ctr;
Bnd.CenterPolyParam = a_ctr;

Bnd.UpperPixelLocation = z_b1;
Bnd.LowerPixelLocation = z_b2;


% Display and Save
if debug_mode || ~isempty(save_path)
    fig1 = figure(1);
    fig1.PaperPositionMode = 'auto';
    fig1.PaperSize = [W D];

    for y = 1:H
        I = Z(:,:,y);

        % detected upper and lower bounds
        figure(fig1);
        imshow(I);
        hold on
        plot(1:W, z_b1(y,:), 'c-');
        plot(1:W, z_b2(y,:), 'y-');
        plot(1:W, mz(y,:), 'r-');
        hold off
        drawnow
        set(fig1, 'Name', [num2str(y) '/' num2str(H)]);
        if ~isempty(save_path)
            print(fig1,
fullfile(save_path,'GapBound','PixelwiseDetection',num2str(y,'%04d')), '-dpng');
        end

        pause(0.05);
        clf(fig1);
    end
end
```

<Function - FindLargestCC>

```
%%% Binarize the response and find the largest connected component %%%
% Inputs
%   |- R: 3D matrix of the haar-like filter response
% Output
%   |- BW: binary 3D matrix indicating the largest connected component

function BW = FindLargestCC(R, BW_Line)

if nargin < 2
    BW_Line = [];
end

[D,W,H] = size(R);

level = graythresh(R);
BW_R = false(D,W,H);
for y = 1:H
    BW_R(:,:,y) = im2bw(R(:,:,y), level);
end

CC_R = bwconncomp(BW_R);
wS_R = zeros(1,CC_R.NumObjects);
if isempty(BW_Line)
    for k = 1:CC_R.NumObjects
        wS_R(k) = sum(R(CC_R.PixelIdxList{k}));
    end
else
    for k = 1:CC_R.NumObjects
        if sum(BW_Line(CC_R.PixelIdxList{k}))
            continue
        end
        wS_R(k) = sum(R(CC_R.PixelIdxList{k}));
    end
end

[~,ind] = sort(wS_R, 'descend');
BW = false(D,W,H);
BW(CC_R.PixelIdxList{ind(1)}) = true;
```

<Function - FindLowerBoundBottomLayer>

```
%%% Find the lower bound of the bottom layer %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- BW_low: binary 3D matrix indicating bottom layer location
%   |- wnd: window size for local histogram computation
%   |- d: bin size for the histogram
%   |- Lz: search range from the gap location; +/- Lz pixels in z-axis
% Output
%   |- Bnd: a struct file with the bound information
%       |- CenterPolyFit: estimated lower bound location after polynomial fitting
%       |- CenterPolyParam: parameters of polynomial fit of the lower bound
%       |- LowerPixelLocation: pixel location of the lower bound of the bottom
layer

function Bnd = FindLowerBoundBottomLayer(Z, BW_low, wnd, d, Lz)

% Defaults
if nargin < 5
    error('Inputs of Z, BW_low, wnd, d, and Lz are required.');
end

% Normalize the image to [0,1]
Z = double(Z)/double(intmax(class(Z)));
[D,W,H] = size(Z);

% Fit a second-order polynomial to the detected gap
fprintf('Fitting a second-order polynomial...');

% mesh points
[mx,my] = meshgrid(1:W,1:H);
mA = [ones(numel(mx),1) mx(:) my(:) mx(:).^2 my(:).^2 mx(:).*my(:)];

% least-squares estimation
ind = find(BW_low);
[pz,px,py] = ind2sub([D,W,H], ind);

A = [ones(length(px),1) px py px.^2 py.^2 px.*py];
a_ctr = pinv(A)*pz;

bA = [ones(numel(mx),1) mx(:) my(:) mx(:).^2 my(:).^2 mx(:).*my(:)];
fz_ctr = bA*a_ctr;
fz_ctr = reshape(fz_ctr, [H W]);

fprintf(' Done.\n');


% Find the upper and lower bounds
fprintf('Find the upper and lower bounds...');

% fitted line location
mz = mA*a_ctr;
mz = reshape(mz, [H,W]);
```

```
% local histogram
wnd_h2 = floor(wnd(1)/2);
b = 0:d:1+d; % bins
b3 = reshape(b(1:end-1), 1,1,[]);

z_b = zeros(H,W);
for y = 1:H
    I = Z(:,:,y);

    z1 = max(min(floor(mz(y,:)))-2*Lz, 1);
    z2 = min(max(ceil(mz(y,:)))+2*Lz, D);

    I_seg = I(z1:z2,:);
    I_hist = ImageHistogramFeature2D(I_seg, wnd, b);
    [h_hist,w_hist,~] = size(I_hist);

    % mean intensity value
    I_mean = sum(I_hist .* repmat(b3,h_hist,w_hist), 3);

    for x = 1:W
        kz0 = round(mz(y,x))-z1+1;

        % lower bound
        kz1 = kz0-Lz;
        kz2 = kz0+Lz-wnd_h2;
        kz = max(kz1,1):min(kz2,h_hist-wnd(1));
        dist = zeros(length(kz),1);

        ind = find(squeeze(I_mean(kz,x,:)) > squeeze(I_mean(kz+wnd(1),x,:))); %
top: bright, bottom: dark
        for cnt = 1:length(ind)
            dist(ind(cnt)) = sum((I_hist(kz(ind(cnt)),x,:)-
I_hist(kz(ind(cnt))+wnd(1),x,:)).^2);
        end
        [~,idx] = max(dist);
        z_b(y,x) = kz(idx)+wnd_h2+0.5 + z1-1;
    end
end

fprintf(' Done.\n');

Bnd = struct;
Bnd.CenterPolyFit = fz_ctr;
Bnd.CenterPolyParam = a_ctr;

Bnd.LowerPixelLocation = z_b;
```

<Function - FindUpperBoundTopLayer>

```
%%% Find the upper bound of the top layer %%%
% Inputs
%    |- Z: 3D matrix of the z-stack
%    |- BW_up: binary 3D matrix indicating upper surface of the top layer location
%    |- wnd: window size for local histogram computation
%   |- d: bin size for the histogram
%   |- Lz: search range from the gap location; +/- Lz pixels in z-axis
% Output
%   |- Bnd: a struct file with the bound information
%       |- CenterPolyFit: estimated upper bound location after polynomial fitting
%       |- CenterPolyParam: parameters of polynomial fit of the upper bound
%       |- UpperPixelLocation: pixel location of the upper bound of the top layer

function Bnd = FindUpperBoundTopLayer(Z, BW_up, wnd, d, Lz)

% Defaults
if nargin < 5
    error('Inputs of Z, BW_up, wnd, d, and Lz are required.');
end

% Normalize the image to [0,1]
Z = double(Z)/double(intmax(class(Z)));
[D,W,H] = size(Z);

% Fit a second-order polynomial to the detected gap
fprintf('Fitting a second-order polynomial...');

% mesh points
[mx,my] = meshgrid(1:W,1:H);
mA = [ones(numel(mx),1) mx(:) my(:) mx(:).^2 my(:).^2 mx(:).*my(:)];

% least-squares estimation
ind = find(BW_up);
[pz,px,py] = ind2sub([D,W,H], ind);

A = [ones(length(px),1) px py px.^2 py.^2 px.*py];
a_ctr = pinv(A)*pz;

bA = [ones(numel(mx),1) mx(:) my(:) mx(:).^2 my(:).^2 mx(:).*my(:)];
fz_ctr = bA*a_ctr;
fz_ctr = reshape(fz_ctr, [H W]);

fprintf(' Done.\n');



% Find the upper lower bound
fprintf('Find the upper bound of top layer...');

% fitted line location
mz = mA*a_ctr;
mz = reshape(mz, [H W]);

% local histogram
wnd_h2 = floor(wnd(1)/2);
b = 0:d:1+d; % bins
b3 = reshape(b(1:end-1), 1,1,[]);

z_b = zeros(H,W);
for y = 1:H
```

```
    I = Z(:,:,y);

    z1 = max(min(floor(mz(y,:)))-2*Lz, 1);
    z2 = min(max(ceil(mz(y,:)))+2*Lz, D);

    I_seg = I(z1:z2,:);
    I_hist = ImageHistogramFeature2D(I_seg, wnd, b);
    [h_hist,w_hist,~] = size(I_hist);

    % mean intensity value
    I_mean = sum(I_hist .* repmat(b3,h_hist,w_hist), 3);

    for x = 1:W
        kz0 = round(mz(y,x))-z1+1;

        % upper bound
        kz1 = kz0-Lz;
        kz2 = kz0+Lz-wnd_h2;
        kz = max(kz1,1):min(kz2,h_hist-wnd(1));
        dist = zeros(length(kz),1);

        ind = find(squeeze(I_mean(kz,x,:)) < squeeze(I_mean(kz+wnd(1),x,:))); %
top: dark, bottom: bright
        for cnt = 1:length(ind)
            dist(ind(cnt)) = sum((I_hist(kz(ind(cnt)),x,:)-
I_hist(kz(ind(cnt))+wnd(1),x,:)).^2);
        end
        [~,idx] = max(dist);
        z_b(y,x) = kz(idx)+wnd_h2+0.5 + z1-1;
    end
end

fprintf(' Done.\n');

Bnd = struct;
Bnd.CenterPolyFit = fz_ctr;
Bnd.CenterPolyParam = a_ctr;

Bnd.UpperPixelLocation = z_b;
```

<Function - GetStatistics>

```
%%% Get statistics of the gap thickness by region %%%
% Inputs
%   |- Thk: estimated thickness of the gap
%   |- ri: refractive index
%   |- std_ri: standard deviation of refractive index
%   |- debug_mode: '1' to show interim results; '0' to skip
%   |- save_path: assign a path to save interim results; empty field not to save
them
%   |- save_name: file name to be saved
% Output
%   |- stat: mean and uncertainty of thickness values in 3 by 3 regions
```

```
function stat = GetStatistics(Thk, ri, std_ri, debug_mode, save_path, save_name)

nx = 3;
ny = 3;

method = fieldnames(Thk);
nMethod = length(method);

if ~isempty(save_path)
    if ~isdir(fullfile(save_path,'Thickness'))
        mkdir(fullfile(save_path,'Thickness'));
    end
end

%%% save for possible update %%%
if ~isempty(save_path)
    save(fullfile(save_path,'Thickness',['Thickness_' save_name '.mat']), 'Thk');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Mean and standard deviation of thickness in ny by nx regions
mean_thk = zeros(length(method),nx*ny);
std_thk = zeros(length(method),nx*ny);
for k = 1:nMethod
    [H,W] = size(Thk.(method{k}));

    xt = round(linspace(0,W,nx+1));
    yt = round(linspace(0,H,ny+1));

    x1 = xt(1:nx)+1;
    x2 = xt(2:end);
    y1 = yt(1:ny)+1;
    y2 = yt(2:end);

    for ky = 1:ny
        for kx = 1:nx
            ki = (ky-1)*nx + kx;
            mean_thk(k,ki) =
mean(reshape(Thk.(method{k})(y1(ky):y2(ky),x1(kx):x2(kx)), [],1), 'omitnan');
            std_thk(k,ki) =
std(reshape(Thk.(method{k})(y1(ky):y2(ky),x1(kx):x2(kx)), [],1), 'omitnan');
        end
    end

    % mean
    stat.(method{k}).mean = mean_thk(k,:);

    % uncertainty
    perc_std = sqrt( (std_thk(k,:)./mean_thk(k,:)).^2 + (std_ri/ri)^2
+(0.009/1.074)^2);
    stat.(method{k}).unc = mean_thk(k,:) .* perc_std;

    % Display and Save
    if debug_mode
        fig1 = figure(1);
        imshow(Thk.(method{k}),[]);
        for ky = 1:ny
            for kx = 1:nx
                ki = (ky-1)*nx + kx;
                text(x1(kx)+(x2(kx)-x1(kx))/2, y1(ky)+(y2(ky)-y1(ky))/2, ...
                      {num2str(round(stat.(method{k}).mean(ki))),['('
num2str(round(stat.(method{k}).unc(ki))) ')']}, ...
                      'FontWeight','bold', 'FontSize',20, 'Color','r',
'HorizontalAlignment','center');
            end
        end
        drawnow

        if ~isempty(save_path)
            print(fig1, fullfile(save_path,'Thickness',[save_name '_' method{k}]),
'-dpng');
        end

        pause(0.05);
        clf(fig1);
    end

    if ~isempty(save_path)
        [kx,ky] = meshgrid(1:nx,1:ny);
        T = array2table([ky(:), kx(:), round(stat.(method{k}).mean'),
round(stat.(method{k}).unc')], ...
            'VariableNames', {'Row','Column','Mean','Uncertainty'});
        writetable(T, fullfile(save_path,'Thickness',[save_name '_3by3_' method{k}
'.csv']));
    end
end


% Mean and standard deviation of thickness per slice
mean_thk_slice = zeros(nMethod,H);
std_thk_slice = zeros(nMethod,H);
unc_thk_slice = zeros(nMethod,H);
for k = 1:nMethod
    mean_thk_slice(k,:) = mean(Thk.(method{k}), 2, 'omitnan');
    std_thk_slice(k,:) = std(Thk.(method{k}), 0, 2, 'omitnan');

    % uncertainty
    perc_std = sqrt( (std_thk_slice(k,:)./mean_thk_slice(k,:)).^2 + (std_ri/ri)^2
);
    unc_thk_slice(k,:) = mean_thk_slice(k,:) .* perc_std;

    if ~isempty(save_path)
        T = array2table([ [1:H]', round(mean_thk_slice(k,:)'),
round(unc_thk_slice(k,:)') ], ...
            'VariableNames', {'SliceNumber','Mean','Uncertainty'});
        writetable(T, fullfile(save_path,'Thickness',[save_name '_Slice_' method{k}
'.csv']));
    end
end
```

<Function - ImageHistogramFeature2D>

```
%%% Compute local histogram of the image %%%
% Inputs
%   |- img: a vertical slice segmented around the gap location
%   |- window_size: window size to compute local histogram
%   |- bins: bins to compute histogram
% Output
%   |- hst: local histogram at every pixel in img

function hst = ImageHistogramFeature2D(img, window_size, bins)

half_window_size = (window_size-1)*0.5;
if sum(mod(half_window_size,1)) > 0
    error('Use odd numbers for ''window_size''.');
end

[h,w] = size(img);
nb = length(bins)-1;

% binning
bin_label = zeros(h,w);
for kb = 1:nb
    ind = find(img>=bins(kb) & img<bins(kb+1));
    bin_label(ind) = kb;
end

% counting
ft = fspecial('average', window_size);
hst = zeros(h,w,nb);
for kb = 1:nb
    bw = (bin_label==kb);
    bw_ft = imfilter(double(bw),ft);
    hst(:,:,kb) = bw_ft;
end
```

<Function - LoadImages>

```
%%% Load the images of a z-stack %%%
% Inputs
%   |- file_path: directory path where the images are
%   |- file_type: image file extension
% Output
%   |- Z: 3D matrix of the z-stack; D by W (each image) by H (# of vertical
%         slices); intensity values normalized in the range of [0,1]

function Z = LoadImages(file_path, file_type)

fprintf('Loading data...');

flist = dir(fullfile(file_path,['*.' file_type]));
H = length(flist); % # of vertical slices
```

```
% Check the image size
y = 1;
I = imread(fullfile(file_path,flist(y).name));
[D,W,~] = size(I);

% Read images (only the first channel)
Z = zeros(D,W,H, class(I));
for y = 1:H
    I = imread(fullfile(file_path,flist(y).name));
    Z(:,:,y) = I(:,:,1);
end

fprintf('Done.\n');
```

<Function - MeasureGapThickness>

```
%%% Measure the thickness of the gap %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- Bnd: a struct file with gap information
%   |- lpp: length per pixel
%   |- ri: refractive index
%   |- tilt: estimated tilt angle of the layers
% Output
%   |- Thk: estimated thickness per method

function Thk = MeasureGapThickness(Z, Bnd, lpp_x, lpp_z, ri, tilt)

% Normalize the image to [0,1]
Z = double(Z)/double(intmax(class(Z)));
[D,W,H] = size(Z);

Thk = struct;

% Pixelwise thickness
Thk.Pixel = Bnd.LowerPixelLocation - Bnd.UpperPixelLocation;
Thk.Pixel = (Thk.Pixel*lpp_z)*cos(tilt) / ri;
```

<Function - MeasureLayerThickness>

```
%%% Measure the thickness of the top and bottom layers %%%
% Inputs
%   |- Z: 3D matrix of the z-stack
%   |- Bnd_gap: a struct file with gap information
%   |- Bnd_up: a struct file with top layer information
%   |- Bnd_low: a struct file with bottom layer information
%   |- lpp_x: length per pixel for x-axis
%   |- lpp_z: length per pixel for z-axis
%   |- ri: refractive index
%   |- tilt: estimated tilt angle of the layers
% Output
%   |- Thk_up: estimated thickness per method for top layer
%   |- Thk_low: estimated thickness per method for bottom layer
```

```
function [Thk_up, Thk_low] = MeasureLayerThickness(Z, Bnd_gap, Bnd_up, Bnd_low,
lpp_x, lpp_z, ri, tilt)

% Normalize the image to [0,1]
Z = double(Z)/double(intmax(class(Z)));
[D,W,H] = size(Z);

Thk_up = struct;
Thk_low = struct;

% Pixelwise thickness
Thk_up.Pixel = Bnd_gap.UpperPixelLocation - Bnd_up.UpperPixelLocation;
Thk_up.Pixel = (Thk_up.Pixel*lpp_z)*cos(tilt) / ri;

Thk_low.Pixel = Bnd_low.LowerPixelLocation - Bnd_gap.LowerPixelLocation;
Thk_low.Pixel = (Thk_low.Pixel*lpp_z)*cos(tilt) / ri;
```

* * * * * * * * * * * * End of Appendix * * * * * * * * * * * *