meadowlark optics
spatial light modulators

User Manual

1920 x 1152 XY Phase Series and

1920 x 1200 XY Phase Series

Spatial Light Modulator

With HDMI Controller

Meadowlark Optics, Inc.

5964 Iris Parkway

Frederick, CO 80530

USA

Telephone: (303) 833-4333

Fax: (303) 833-4335

sales@Meadowlark.com

www.Meadowlark.com

Table of Contents

Contents

1 Introduction

This User Manual covers the 1920 x 1152 and 1920 x 1200 (8-bit and 10-bit) XY Phase Series of liquid crystal on silicon spatial light modulators (SLMs) now available from Meadowlark Optics. The manual instructs users on how to set up the PCIe electronic hardware, optics, and operate the system software. SLM system setup is complex, incorporating optical components, mounting and positioning hardware, custom drive electronics hardware, and a proprietary software interface. We strongly recommend that all users first read and familiarize themselves with the entire User Manual before initiating the setup and start up procedures.

1.1 Spatial Light Modulator Principles

A SLM is a device that modulates light according to a fixed spatial (pixel) pattern. The XY Phase Series SLMs convert digitized data into coherent optical information appropriate for a wide variety of applications, including beam steering, optical tweezers, diffractive optics, pulse shaping, and more. These applications require modulators that can easily and rapidly change the wavefront of a coherent beam. By combining the electro-optical performance characteristics of liquid crystal materials with silicon-based digital circuitry, Meadowlark Optics now offers high resolution SLMs that are also physically compact and optically efficient.

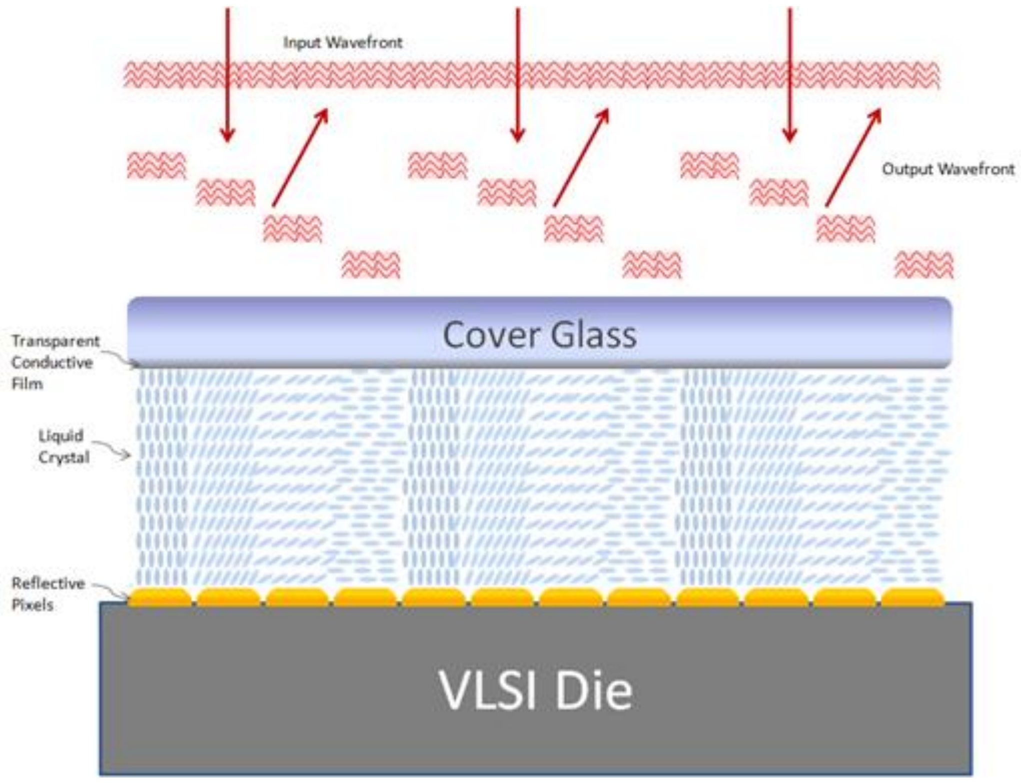


Figure 1 Cross sectional illustration of a liquid crystal spatial light modulator.

Figure 1 shows the cross section of an LC SLM. Polarized light enters the device from the top, passes through the cover glass, transparent conductive film, and liquid crystal layer. Light is reflected off the pixel electrodes and returns on the same path. Drive signals from the hardware interface independently drive each pixel on the SLM. The analog voltage applied to each pixel produces an electric field between that pixel and the cover glass, resulting in a change in the optical properties of the LC layer. Because each pixel is independently controlled, a phase pattern may be generated by loading different voltages onto each pixel.

1.2 Phase Modulation

The XY Phase Series Spatial Light Modulators are fabricated with nematic liquid crystal, aligned in a homogenous configuration. Nematic liquid crystal has a variable electro-optic response to voltage. Fig. 2 shows a simplified side view of a spatial light modulator's liquid crystal layer. When no voltage is applied to the LC, the molecules are parallel to the SLM coverglass and VLSI backplane. In this case, incident light will experience the largest difference between the extraordinary (n_e) and ordinary (n_o) index of refraction. As the voltage applied to the liquid crystal is increased, the LC will tilt until the extreme is reached and the LC is nearly normal to the SLM coverglass and VLSI backplane. At this point the difference between the extraordinary and ordinary index of refraction is nearly zero.

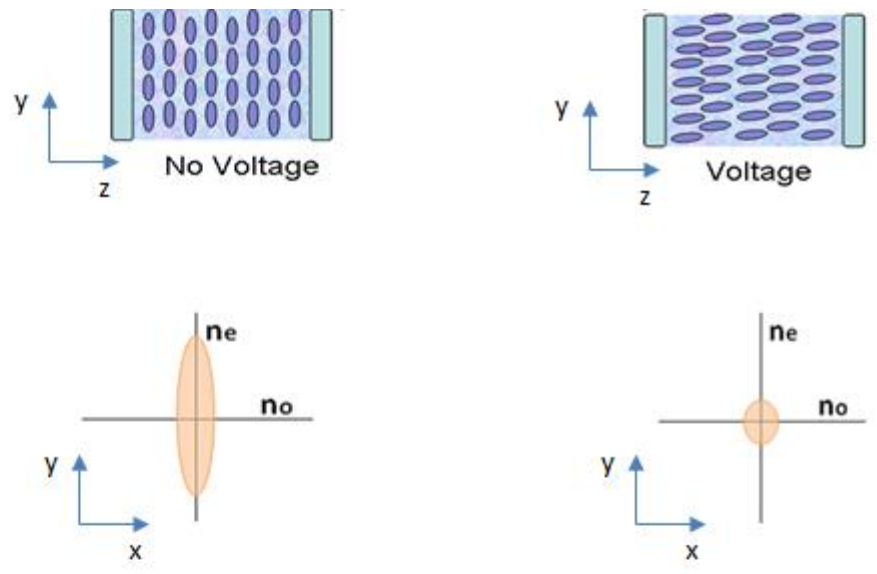


Figure 2 This diagram illustrates the liquid crystal orientation with respect to the coverglass and VLSI backplane as a function of applied voltage. The molecules are parallel to the coverglass and backplane when no field is applied, and are nearly perpendicular to the coverglass and backplane when full field is applied.

If light incident upon the SLM is linearly polarized parallel to the extraordinary axis, then a pure, voltage dependent phase shift will be observed. For example, if no voltage is applied to the pixel, the maximum phase retardation (typically a full wave at the design wavelength) will be applied. Likewise, if the pixel is programmed for maximum voltage, a minimum phase delay is applied. The result is a programmable phase modulation on a per pixel basis.

Each of the SLM pixels is independently programmable to 256 discrete voltage states, all providing phase modulation. The response of the liquid crystal within the SLM to the applied pixel value (voltage) is nonlinear. To account for this, each SLM is shipped with a custom look-up-table (LUT). When this LUT is applied to an input image, the result is a linear output phase response ranging from 0 to 2π.

1.3 Optical Test Setup

Depending on the application of the XY Phase Series SLM, many different optical setups can be used. Two examples of phase-only optical test setups are shown below.

The first optical setup, illustrated in Figure 3, is a modification of a Twyman-Green interferometer (Handbook of Optics. Vol. 1, pp. 2.28-2.29). Here a monochromatic, collimated light source (i.e. laser beam expanded so it is larger than the diagonal of the SLM) passes through a non-polarizing beamsplitter such that the beam is divided into two beams, with nearly equal intensity. One of these beams illuminates the XY Phase Series SLM, while the other illuminates a reference mirror. Each of these reflected beams is then recombined at the image plane of a lens. If the reference mirror and the SLM are carefully aligned such that they are nearly coplanar, interference fringes will be visible at the image plane. A camera is placed at the image plane with proper magnification for easier viewing of the active area of the SLM and the interference fringes. When the XY Phase Series SLM is driven with different phase patterns, the corresponding interference fringes can be viewed. Analyzing the interference fringes will then provide insight into the phase modulation provided by the XY Phase SLM.

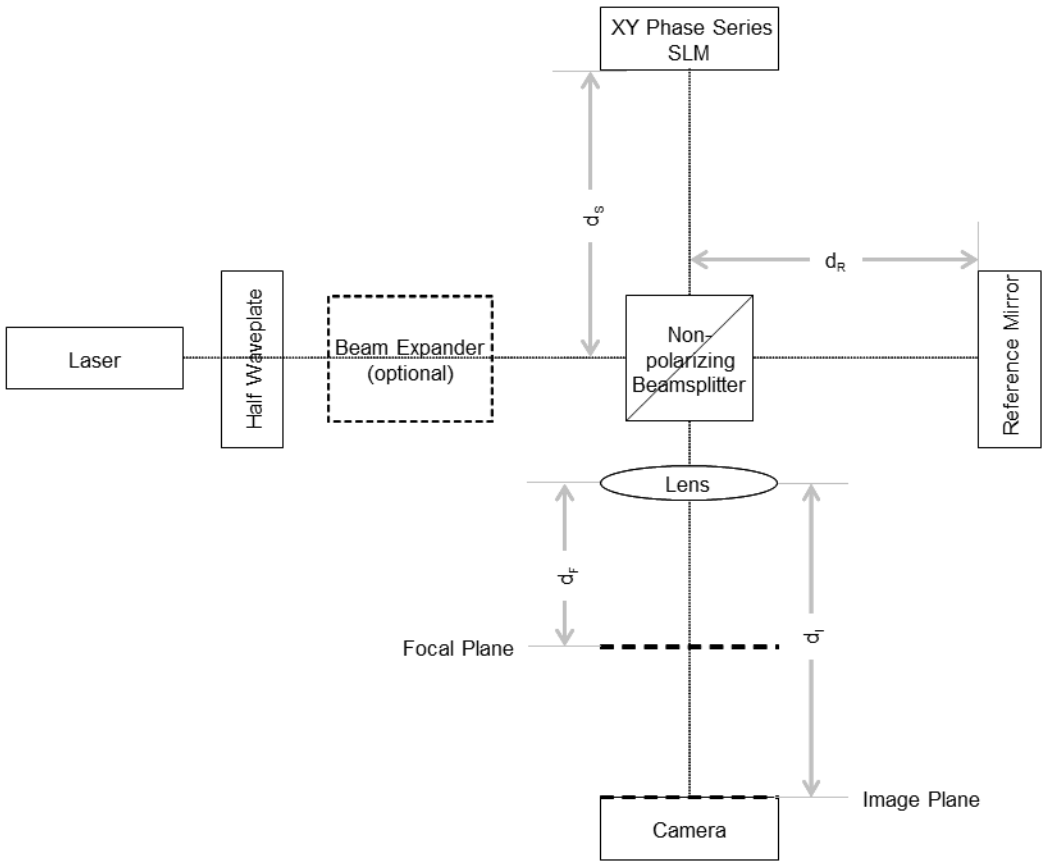


Figure 3 A Twyman-Green interferometer for testing an XY Phase SLM.

To ease the alignment of the reference mirror and the XY Phase Series SLM in the Twyman-Green interferometer, place the camera or a card into the beam path at the focal plane of the lens. There should be two spots visible, one from the reference mirror and one from the SLM (there may also be faint high-order diffraction spots from the SLM). If utilizing a beamsplitter cube, there are likely to be additional spots that are ghost images from the beamsplitter. When adjusting the tip/tilt of either the reference mirror or the SLM, the ghost spots will not move, but the true spots from the reference mirror and the SLM will move. Once the correct two spots are located, adjust the tip and tilt of the reference mirror and/or the SLM until these two spots are aligned on top of one another. Place the camera back in the image plane, or remove the card, and interference fringes should be visible in the image plane. By slowly adjusting the tip and/or tilt of just the reference mirror or SLM, the desired interference fringes should be readily obtained.

If working with a laser diode it is important to note that the coherence length of laser diodes are typically much shorter than the coherence length of a laser. With a Twyman-Green interferometer it is critical that both the SLM leg and the reference mirror leg have the same optical path length, ensuring that the short coherence length of a laser diode does not significantly reduce the visibility of the interference fringes. To ensure equal path lengths, set the camera such that it is focused onto the SLM – distinct features of the silicon backplane, such as individual or the edge of the active area, should become sharp. Blocking the reference leg beam will likely be necessary to prevent crosstalk in the camera. Then move the block from the reference leg to the SLM leg and move the reference mirror closer to, or further from, the beam splitter until it is also in sharp focus on the camera. Since the reference mirror will likely not have any features with which to see the best focal point, look for focus of dust particles, or perhaps very carefully place a card onto the reference mirror such that it covers about half of the beam. The card edge can then be used as a guide to find the best focal position of the reference mirror.

There are a few important issues to remember whenever working with an SLM.

1. **Polarization** – The SLM is basically a variable single-order retardation plate, or wave plate. Like all wave plates, there is a fast axis and a slow axis. However, with the XY Phase Series SLMs the index of refraction along the slow axis can be decreased electronically. **While the liquid crystal alignment is occasionally customized, in general on the 512x512 the slow axis is vertical, on the 1920x1152 the slow axis is vertical (parallel to the short axis of the SLM), and on the 1024x1024 the slow axis is horizontal.**

   When the light source is linearly polarized and parallel to the slow axis of the SLM, the result is phase-only modulation of the light source. If the orientation of the incident polarization is incorrect, there would be no modulation observed with variable voltage. Placing a half wave plate between the laser and the beamsplitter will greatly facilitate achieving the desired polarization alignment.

   If there is any question about the laser producing linearly polarized light, place a polarizer between the laser and the half wave plate. This will ensure that the output of the laser is linearly polarized, and the half wave plate can then be used to rotate the angle of the polarization.

2. **Diffraction** – The SLM has discrete, reflective pixel pads that diffract light into higher orders in addition to the 0th order. This diffraction can easily be seen in the focal plane of a lens. There will be a very bright center spot (0th-order), surrounded by a grid of spots becoming progressively dimmer as they get further from the 0th-order.

3. **Dispersion** – Liquid crystal devices are not achromatic due to the nonlinear change in index of refraction as a function of wavelength. This dispersion means that a device designed to provide a 2π phase stroke at one wavelength, will provide less than 2π phase stroke at a longer wavelength, and more than 2π phase stroke at a shorter wavelength. A wavelength dependent calibration is provided with the SLM such that input graylevels of 0 to 255 linearly map to phase delays of 0 to 2π. If the wavelength is changed it will be necessary to re-calibrate the SLM.

4. **Optical Quality** – Due to the very small pixel pitch of the SLM, it is important to use high quality optics. A single element lens will generally have too much spherical aberration to provide a good, sharp focus across the entire clear aperture of the SLM. As a result, it is recommended to always use at least a doublet lens. For the same reason, the longer the focal length of the lens, the better the resulting image at the image plane. In addition, the transmitted wavefront distortion of most beamsplitter cubes is typically ~$\lambda/4$, contributing to aberrations at the image plane. The use of a

beamsplitter plate in place of a beamsplitter cube could reduce aberrations but using a beamsplitter plate requires the use of a compensating plate in the reference leg of the Twyman-Green interferometer

An off-axis setup, shown in Figure 4, is designed to maximize throughput by eliminating the non-polarizing beam splitter. A laser beam is incident on the SLM at a slight angle reflects off the reflective pixel pads, and then is imaged with a lens onto a camera. In this setup it is most common to put the camera at the focal plane of the lens. The SLM can be used to modulate the 0th order or can be used as a diffractive optic manipulating the 1st order.

The off-axis angle should be kept to a minimum to reduce crosstalk effects due to the beam traveling through more than one pixel region. Minimizing the off-axis angle also keeps the phase stroke closer to the designed value. We typically recommend an angle of incidence of 15 degrees or less. If a wider angle is used the SLM should be recalibrated.

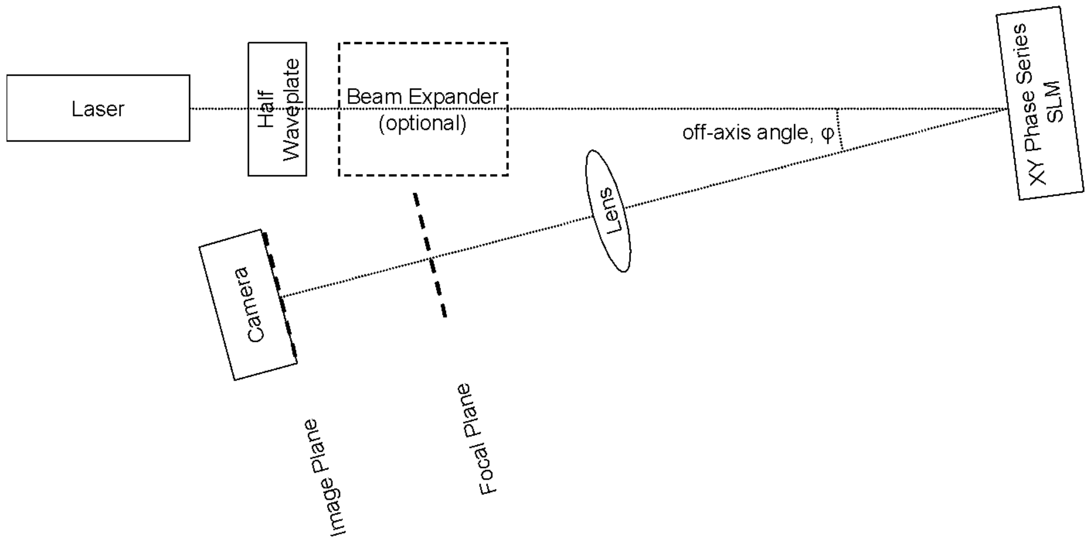
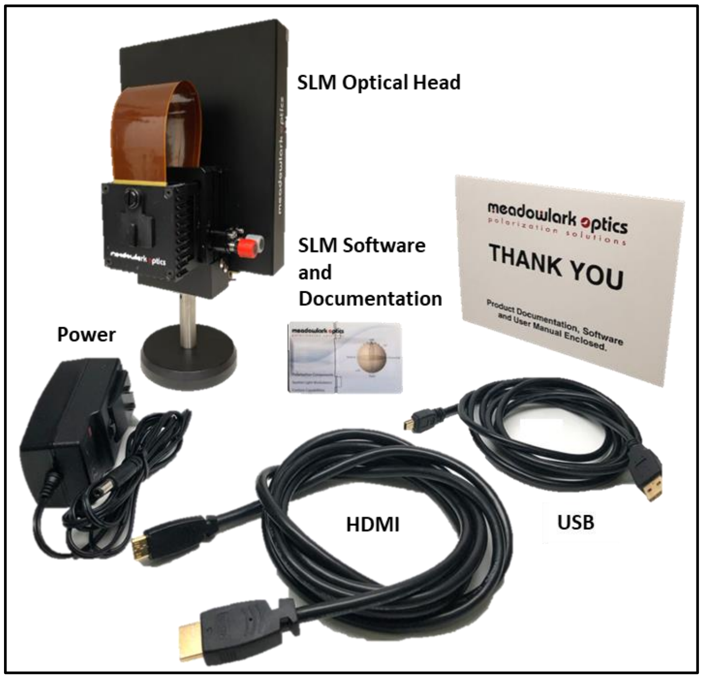


Figure 4 Off-axis optical setup for the XY Phase Series SLMs.

2 System Setup

The complete SLM system, which is driven through a HDMI interface, is shown in Figure 5. This system consists of the optical head containing the SLM and attached to the HDMI Driver Box, an external power cable, a USB cable, an HDMI cable, and a USB card that contains software, documentation, and user manual. The power cable must be connected from the controller to a power strip, the HDMI cable and USB cable must be connected from the controller to a HDMI port and USB port on the computer.


SLM Optical Head
SLM Software and Documentation
Power
HDMI
USB
meadowlark optics
polarization solutions
THANK YOU
Product Documentation, Software and User Manual Enclosed.

1

2 ***Figure 5 Complete SLM system with all associated cabling.***

Connect the power, USB, and HDMI cables to the HDMI controller. Then connect the USB and HDMI cables to your computer. After installation, the SLM will be recognized on your computer as a secondary monitor. The following sections will walk you through the graphics card settings and use of the software.

2.1 Minimum System Requirements

To effectively utilize your SLM system some basic computing hardware is required. The following components are essential to properly achieve the full performance of your SLM system.

- 1 GHz dual-core processor or better
- 2 GB RAM or better
- 500 MB available disk space
- Windows®-based computer operating system (Windows 10® or Windows 7® (64-bit))
- Available HDMI 1.2 port or newer (adapters are not recommended and may introduce performance issues)
- Available USB 2.0 port
- **A GPU supporting OpenCL 1.2 or higher is required, NIVIDA is preferred.**

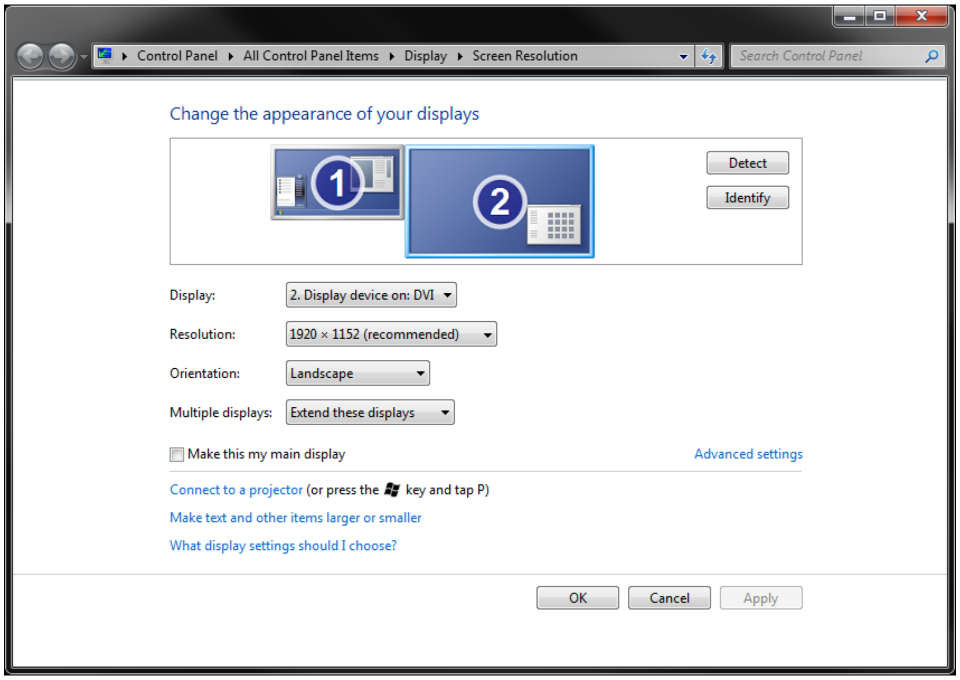
2.2 Software Installation Instructions

The Meadowlark software on the USB flash drive contains the executable code necessary to operate the SLM, as well as sample pattern files, calibrations, software development kits, documentation, and various other support files. Please follow these steps to install the software.

1. Insert the USB flash drive that came with your SLM system into the USB drive.
2. Copy the *.msi, Preferences.ini, your custom LUT file, and your custom calibration file to your computer. Double click on the *.msi file to launch the installer.
3. Follow the onscreen instructions to complete the installation.
4. When the installation is complete, eject the USB drive from the drive and store it away carefully.
5. After installation is complete, the software should be ready to use. A shortcut to the executable will be installed on the desktop, and in the Start Menu.
6. The software will be installed under C:\Program Files\Meadowlark Optics\Blink 1920 HDMI. The software development kit documentation and example programs can be found at C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\SDK.

2.3 Graphics Card Setup

The computer views the SLM as a secondary monitor. As such, when the system is initially connected it may be necessary to enable the SLM via the Windows Properties Dialog prior to being able to drive to the SLM. To do this, open the display properties window, right click on the secondary monitor, and select "Attached".


Control Panel ▸ All Control Panel Items ▸ Display ▸ Screen Resolution

Change the appearance of your displays

Display: 2. Display device on: DVI

Resolution: 1920 × 1152 (recommended)

Orientation: Landscape

Multiple displays: Extend these displays

Make this my main display

Search Control Panel
1
2
Detect
Identify
Advanced settings
Connect to a projector (or press the key and tap P)
Make text and other items larger or smaller
What display settings should I choose?
OK
Cancel
Apply

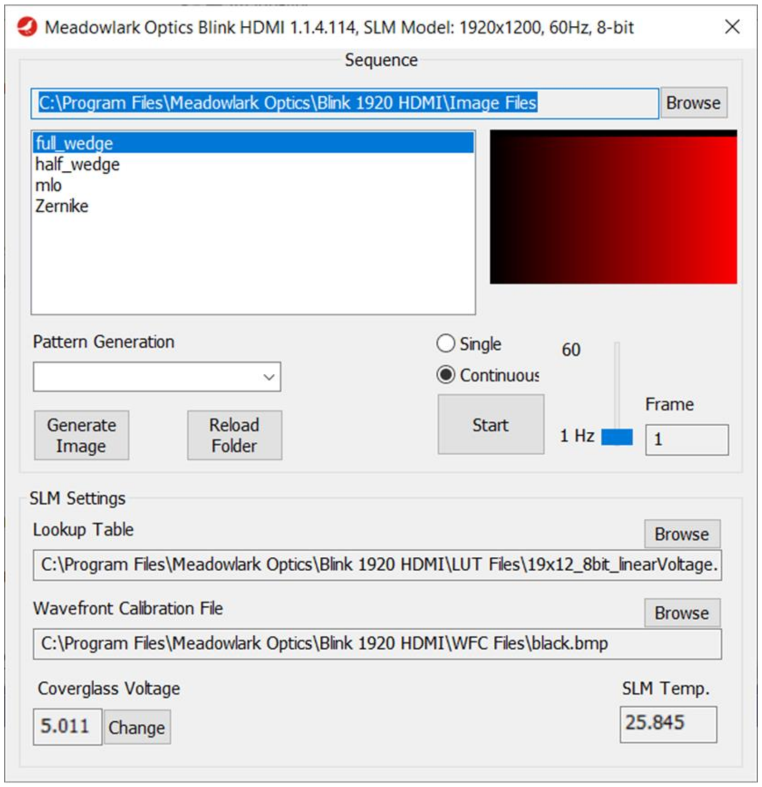
Figure 6 Display Properties with the HDMI hardware connected and the secondary "monitor" enabled.

Note the monitor arrangement with the primary monitor on the left, and the secondary monitor to the right. If this arrangement is incorrect simply click and drag the secondary monitor to the desired location. Right click on the secondary monitor to edit the properties. Color quality must be set to True Color (32 bit) for the device to work properly, and the Screen Refresh Rate should be set to 31 Hz for the 1920 x 1152, or 60 Hz for the 1920 x 1200. The resolution should be set to 1920 x 1152 or 1920 x 1200 depending on the SLM model in use, and "Extend these displays" must be selected.

3 Software Operation

To start the software, double click the "Blink HDMI" shortcut located on your desktop, or click on the Windows "Start" button, select, "Run…", type "C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\Blink HDMI.exe", then click "OK". When the software is started, the screen shown in Fig. 7 should be visible.

Along the title bar of the GUI is important information about your SLM software version, the resolution of the SLM found, the frame rate available, and the bit depth the SLM HDMI controller supports. The software will open if a SLM is not attached, but you will receive a warning that the software is running in simulation mode, the temperature will read 0, and the coverglass voltage will read 0. Running the software in simulation mode enables the software output to be loaded to a secondary monitor, which can be useful for troubleshooting system level questions.



Meadowlark Optics Blink HDMI 1.1.4.114, SLM Model: 1920x1200, 60Hz, 8-bit

Sequence

C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\Image Files

full_wedge
half_wedge
mlo
Zernike

Frame: 1

Lookup Table: C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\LUT Files\19x12_8bit_linearVoltage.

Wavefront Calibration File: C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\WFC Files\black.bmp

Coverglass Voltage: 5.011

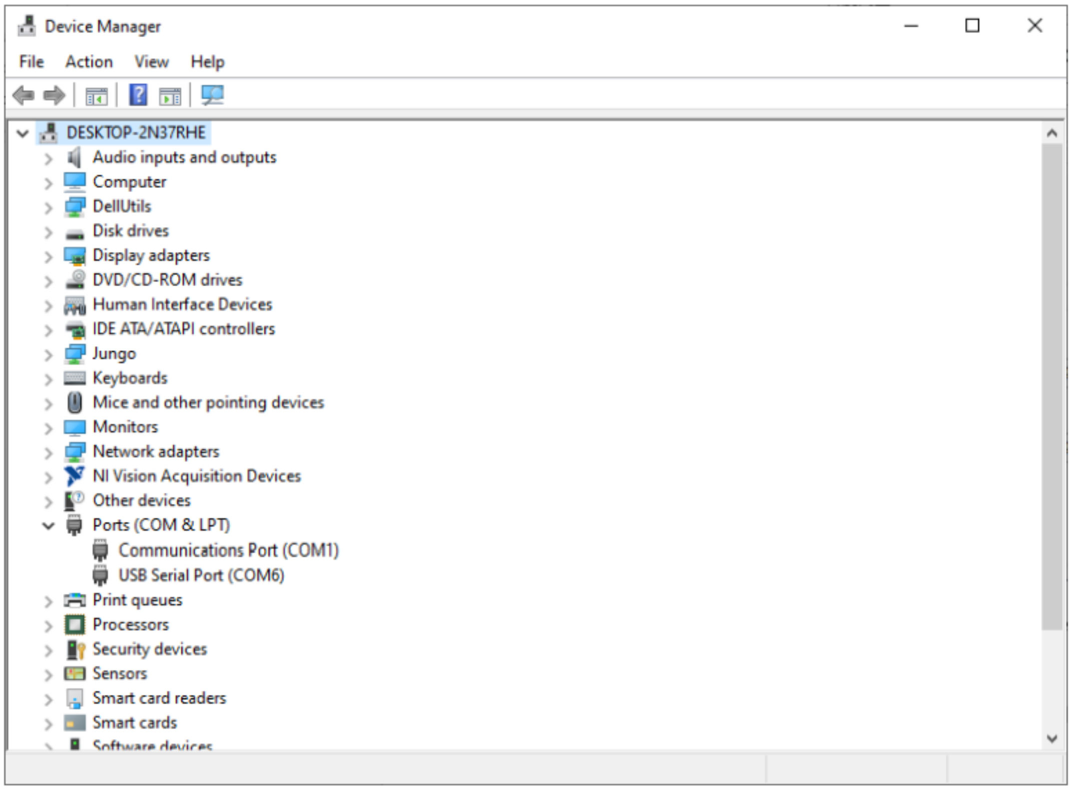
SLM Temp.: 25.845

Figure 7 Main window for the Meadowlark Blink HDMI software

The SLM will be powered automatically when the power is applied to the HDMI hardware. After the software has opened, the Pattern File selected in the Sequence List should appear both on the GUI and on the SLM.

3.1 SLM Settings

The software allows the user to load information about the SLM settings to the hardware. These settings are set and/or read through the USB cable using a COM port. The software will not know which COM port was assigned to the hardware by the computer. To set the COM Port, open the device manager and expand Ports. If the USB cable to the SLM hardware is attached/detached, then one of the ports will be added/removed from the device manager. In the screen capture below the SLM hardware is COM6. To set the COM port the software references, open C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\Preferences.ini and set COM Port=COM6.



The settings that are controlled by the USB interface include loading the LUT calibration, reading the SLM temperature, and adjusting the coverglass voltage. If the hardware is power cycled, and the USB cable is not attached then default settings will be restored. Blink automatically loads the last used settings on initialization, which are defined in C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\Preferences.ini. If using the software development Kit's (SDK's) the user has more control, so it is important to ensure that the SLM is properly initialized.

Look Up Table **(LUT) Information**

Of primary interest to the user is the ability to change the LUT, or "look up table". A LUT is used to map input image data that ranges from 0 to 255 for 8-bit systems or 0 to 1023 for 10-bit systems to a set of new output values that result in a linear from 0 to 2π phase shift. The software supports several LUT file formats: *.blt, *.LUT, and *.txt. In each case the LUT file is a text file.

- A *.blt file contains one column of 256 or 1024 numbers depending on the input bit depth supported that can range from 0 to 2048 for the 1920 x 1152 SLM, and 0 to 4096 for the 1920 x 1200 SLM. The position of the entry in the look up table corresponds to the input graylevel, and the value of the entry is the output pixel value that is loaded to the SLM hardware when that input pixel value is encountered in the Pattern File.
- A *.LUT file contains two columns: the first column in the input graylevel value of 0 to 255 or 0 to 1023, and the second column in the output values that can range from 0 to 2048 for the 1920 x 1152 SLM, and 0 to 4096 for the 1920 x 1200 SLM. For this file format the input graylevel that corresponds to pixel values in pattern files is defined by the first column, and the output graylevel is defined in the second column.

- A *.txt calibration file is a regional calibration. To ensure proper formatting, this type of calibration file should only be generated using Meadowlark Optics calibration tools, which are described in greater detail in Section 5.

Each SLM system is shipped with a custom LUT designed to enable a linear phase stroke versus pixel value. This LUT will be named “slmXXXX_atYYY.lut”, where XXXX corresponds to the serial number of your SLM, and YYY corresponds to the wavelength the calibration was generated at. When using this LUT, a linear progression of input 0 – 255 data should result in a linear output phase of 0 to 2π. If the customer generates their own LUT, then the modulation can be calibrated for 0 to ‘n’π.

Wavefront Correction File

This is an aberration correction image that is superimposed with each user defined phase pattern by adding the two images together modulo the input bit depth, and then processing the final image through the LUT calibration. This is used to remove aberrations from the SLM, or from the SLM and optical train if calibrated in the customer’s optical system.

Coverglass Voltage

It is generally not recommended that the user edit the coverglass voltage because improper settings could cause permanent damage to the SLM. The proper default will be set by Meadowlark Optics. However, if instructed by a Meadowlark Optics employee this value can be edited.

SLM Temperature

The SLM software will periodically poll the current SLM temperature. The liquid crystal will lose modulation depth as the SLM heats. If the SLM is used with a high-power laser it is recommended that the SLM be calibrated at the desired operating temperature.

3.2 Sequences

The sequence list is defined by reading the contents of a folder and searching for bitmaps. The software stores information about the last folder used and loads the images from this folder upon initialization. Note that the images are not sorted. To force a series of images to occur in a specific order it is recommended file names within a folder follow a numbering sequence such as: 000Img.bmp, 001Img.bmp, 002Img.bmp…etc.

The software will adapt to input images to the correct format for the SLM in use. If a 1920 x 1152 image is loaded to the 1920 x 1200 SLM then the remaining rows will be set to 0. If the hardware can support 10-bit operation, and an 8-bit image is provided the 8-bits will be put in the 8 most significant bits (red) and the remaining bits will be set to 0. If a 10-bit image is provided to an 8-bit system, then the 8 most significant bits will be used. When generating images with the image generation functions images will be forced to match the capabilities of the SLM model (i.e., the bit depth, and SLM dimensions).

To change the Pattern File currently loaded into the SLM, simply point the mouse at a different Pattern File in the Sequence List and click. This new Pattern File should now be highlighted and be immediately loaded into the SLM. The cursor key (↑ and ↓) can also be used to select different Pattern Files in the sequence list.

3.3 Running a sequence

The Pattern Files can be changed repetitively by clicking the "START" button. Once the "START" button has been pressed, the system will sequence through all the Pattern Files in the current Sequence List at a user defined frame rate (Hz). Using the single or continuous buttons the user can select to loop through the images once, or indefinitely. The frame rate can be set using the frame rate slider. The cursor key (↑ and ↓) can also be used to set the frame rate.

3.4 Opening new sequence list

To open a new sequence file, move the cursor over the "Browse…" button and click. This opens the standard Windows wizard for opening a file or folder. The user is asked to locate a folder on the hard disk. Once located, the new sequence is automatically loaded into the system with the images currently contained within the selected folder. If the images within the folder are moved or deleted after loading into the system, it is necessary to click the "Browse…" button again and reselect the folder.

3.5 Pattern Generation

The software allows users to dynamically generate images to load to the SLM. The supported modes include solids, stripes, checkerboards, random phases, blazed gratings, sinusoidal gratings, Bessel beams (spiral phase, forked phase, axicons, concentric rings), holograms, Zernike polynomials, Fresnel lenses (cylindrical and circular), and superimposed images. Image generation capabilities are supported both through the main GUI, and through the software development kits by loading the Meadowlark Optics Image Generation DLL (see Section 5 for further information).

To generate an image, select the image type from the Pattern Generation drop box, and then click "Generate Image". This will bring up one of the following dialogs. As the image parameters are edited, the SLM automatically updates with the current image data. The image can be saved to a user defined folder (the current folder is the default). By clicking the "Reload Folder" button the images in the current folder will be automatically updated to include the newly generated image.

Bessel Beam

The Bessel Beam GUI (Fig. 8) allows for generation of vortex beams, axicons, and concentric ring patterns. As the image parameters are edited the image written to the SLM will be updated, and the image preview will be refreshed. By default, the center of the pattern will be placed at the center of the SLM. For some patterns it is easier to shift the "center" of the image applied to the SLM with pixel-by-pixel control rather than to optically shift the center of the illumination. Thus, the user can define the center of the image along the x and y axis.

Selecting the spiral checkbox will generate a spiral phase where "Vortex Charge" determines the number of waves of rotation in the image. Selecting a fork will superimpose a repeating phase ramp with the spiral phase, which simply shifts the vortex off axis.

Selecting Axicon will generate a linear phase ramp that is either increasing or decreasing from the center of the SLM image to the edge of the SLM. Setting the number of waves will determine the slope of the phase ramp.

Selecting Rings will generate a repeating concentric ring pattern. The user can specify the diameter in pixels of the inner ring, and the diameter in pixels of the outer ring. The pattern will consist of two graylevels defined by Value One and Value Two. In general, if using a 0 - 2π LUT, the two graylevels used will be 0 and 128 which will generate a 0 and π ring pattern. The ring pattern can be superimposed with defocus by setting the number of waves of defocus to apply.

After editing the image parameters, the bitmap can be saved. The default location for saving is the current path to the current sequence list.

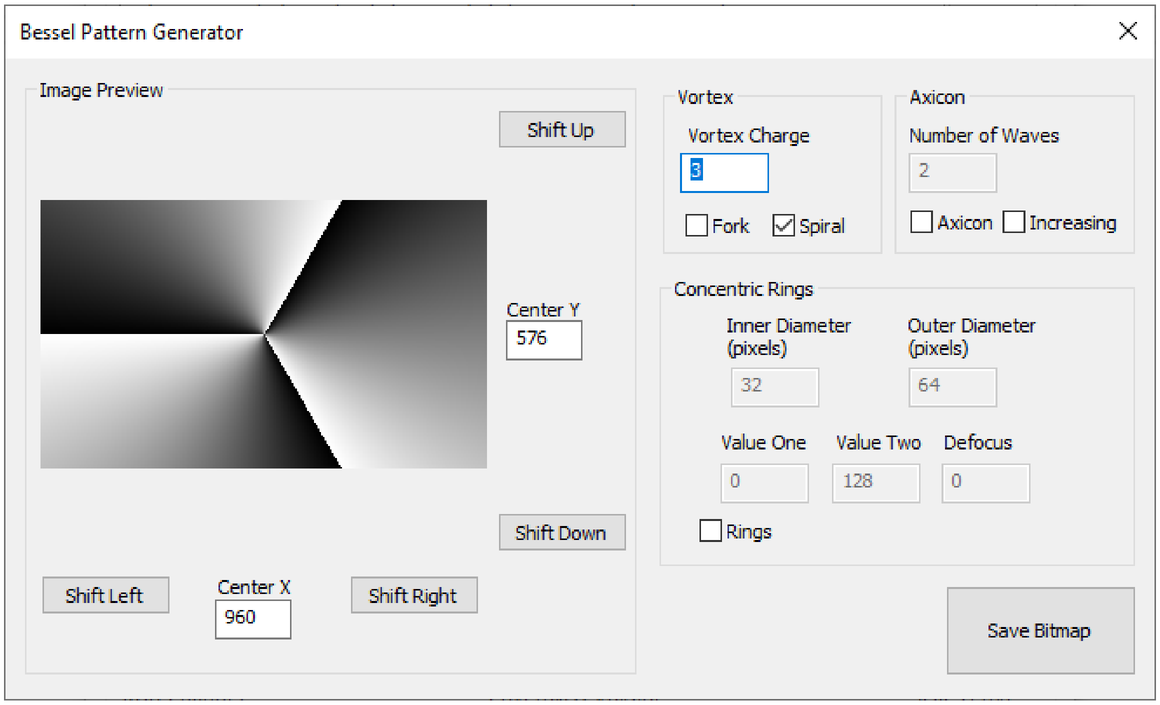

Bessel Pattern Generator

Image Preview

Shift Up

Center Y: 576

Shift Down

Shift Left | Center X: 960 | Shift Right

Vortex
Vortex Charge: 3
[ ] Fork [✓] Spiral

Axicon
Number of Waves: 2
[ ] Axicon [ ] Increasing

Concentric Rings
Inner Diameter (pixels): 32
Outer Diameter (pixels): 64
Value One: 0
Value Two: 128
Defocus: 0
[ ] Rings

Save Bitmap

Figure 8 Bessel GUI

BEST Bessel Beam

The BEST Bessel Beam dialog is based on Na Ji's Bessel Scanning Technology which enables video rate volumetric imaging in two-photon microscopes. For more information on this work it is highly recommended that the following paper and corresponding supplemental information be read:

> Lu, Rongwen, Wenzhi Sun, Yajie Liang, Aaron Kerlin, Jens Bierfeld, Johannes D. Seelig, Daniel E. Wilson et al. "Video-rate volumetric functional imaging of the brain at synaptic resolution." *Nature neuroscience* 20, no. 4 (2017): 620.

To reduce the hurdles of software development when integrating volumetric imaging into a custom system or two photon microscope, Meadowlark Optics included a GUI (Fig 10) used to compute the theoretical axial point spread function (PSF) of a Bessel Beam based on the user's optical setup and the desired numerical aperture of excitation. To begin, the software requires information about the optical system, which is sketched in Fig. 9. The user must define the wavelength of excitation, the beam diameter incident on the SLM, and the focal length of lens 1 which is placed between the SLM and the amplitude mask. Given the users desired effective numerical aperture for excitation (see Supplemental Information, Rongwen et. al.) the GUI will automatically calculate the intensity profile of the excitation beam at the amplitude mask. As outlined by Rongwen et. al., the amplitude mask can be designed to block all light except the 1st order peak and neighboring side lobes (Case I), or the amplitude mask can be designed to block all light except the 1st order peak (Case II). The specifications of the amplitude mask inner and outer radius requirements to support Case I or Case II are shown on the GUI, and are dynamically updated as the parameters in the GUI are adjusted.

To calculate the axial PSF of the excitation, it is necessary to know the manufacturer, numerical aperture (NA), magnification, and immersion type (air, water, or oil) of the objective in use. Additionally, it is necessary to know the magnification of the optical relays from the amplitude mask to the galvo scanning mirror, and from the galvo scanning mirror to the objective. The optical relays should be 4F imaging systems. The magnification of the relay from the mask to the galvo is: the focal length of lens 3 divided by the focal lens 2, and the magnification of the relay from the galvo to the objective is: the focal length of lens 5 divided by the focal lens 4. If an optical relay is not found in your particular optical design, set the magnification in the software to 1. The axial PSF is automatically plotted as GUI parameters are adjusted, and the full width half max of the PSF is shown.

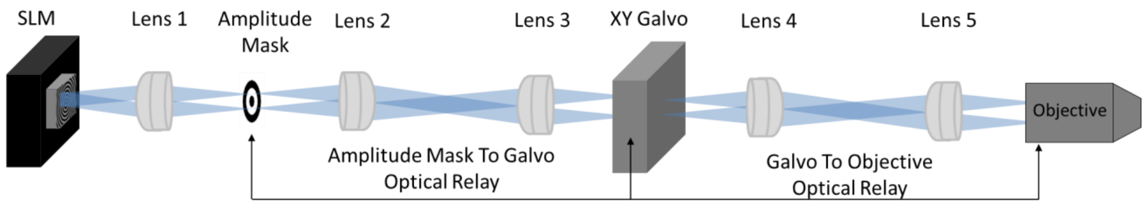
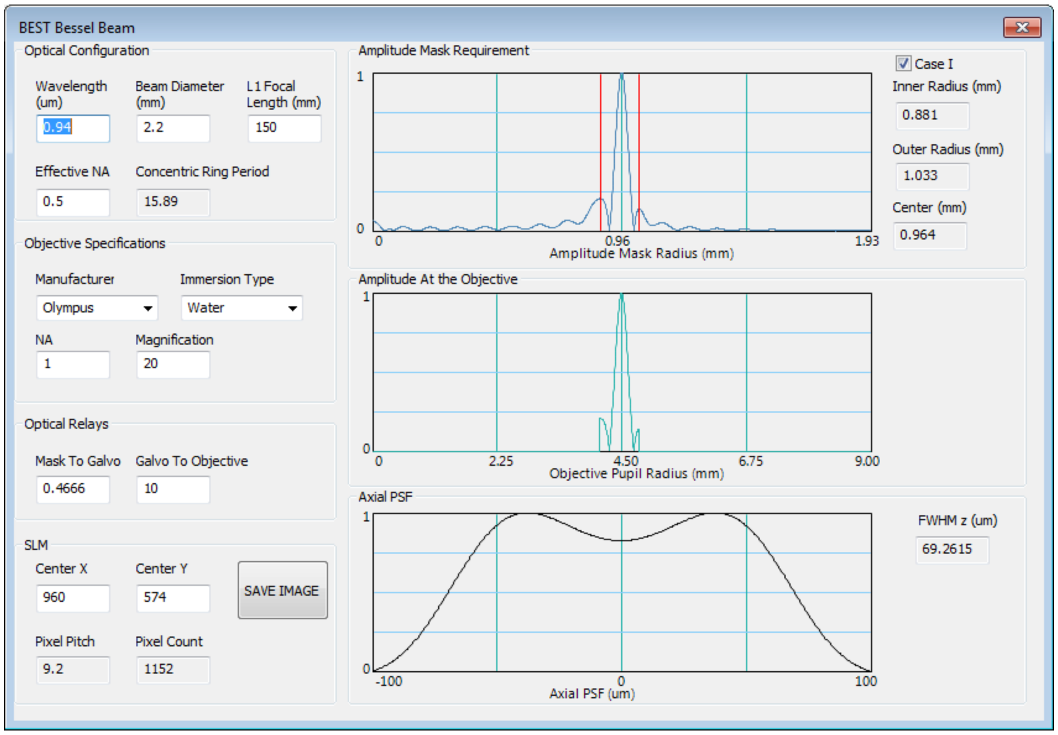

SLM
Lens 1
Amplitude Mask
Lens 2
Lens 3
XY Galvo
Lens 4
Lens 5
Objective
Amplitude Mask To Galvo Optical Relay
Galvo To Objective Optical Relay

Figure 9 Optical Layout for BEST volumetric imaging


BEST Bessel Beam

Optical Configuration

Wavelength (um): 0.94
Beam Diameter (mm): 2.2
L1 Focal Length (mm): 150
Effective NA: 0.5
Concentric Ring Period: 15.89

Objective Specifications

Manufacturer: Olympus
Immersion Type: Water
NA: 1
Magnification: 20

Optical Relays

Mask To Galvo: 0.4666
Galvo To Objective: 10

SLM

Center X: 960
Center Y: 574
Pixel Pitch: 9.2
Pixel Count: 1152

Case I: ✓
Inner Radius (mm): 0.881
Outer Radius (mm): 1.033
Center (mm): 0.964

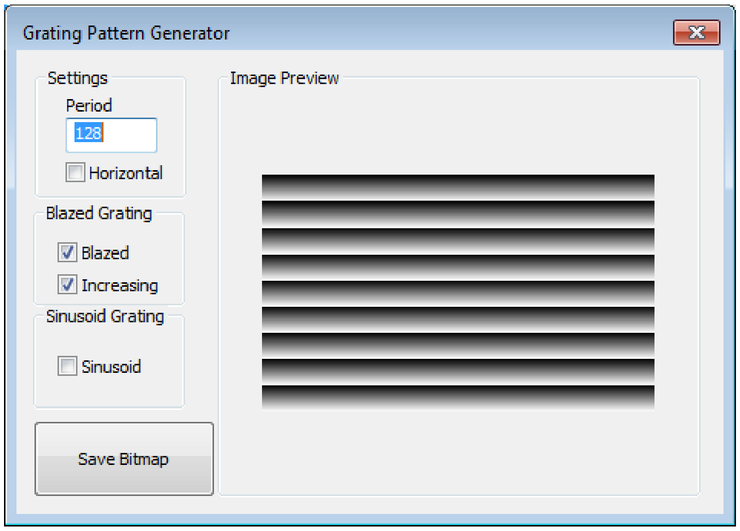
FWHM z (um): 69.2615

SAVE IMAGE
Amplitude Mask Requirement
1
0
0
0.96
1.93
Amplitude Mask Radius (mm)
Amplitude At the Objective
1
0
0
2.25
4.50
6.75
9.00
Objective Pupil Radius (mm)
Axial PSF
1
0
-100
0
100
Axial PSF (um)

Figure 10 BEST Bessel GUI

Blazed or Sinusoid Grating

The blazed or sinusoidal grating GUI enables the user to create repeating ramp patterns, and sinusoidal gratings. The user must define the period in pixels from 0 to 2π, set either a horizontal or vertical grating direction, and set the slope of the grating to either increase from 0 to 2π or decrease from 2π to 0.


Grating Pattern Generator

Settings

Period: 128
Horizontal: ☐

Blazed Grating

Blazed: ✓
Increasing: ✓

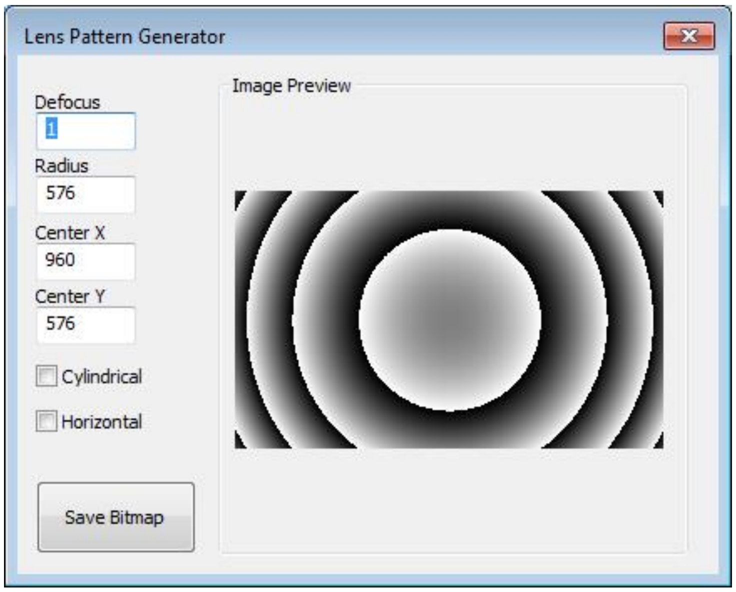
Sinusoid Grating

Sinusoid: ☐

Figure 11 Grating GUI

Fresnel lens

The Fresnel lens GUI allows users to define either circular or cylindrical lenses. The user must define the number of pixels in the radius of the lens, and the number of waves of defocus to apply. The default center of the lens is half of the SLM height and width but can be shifted to accommodate slight offsets in alignment.

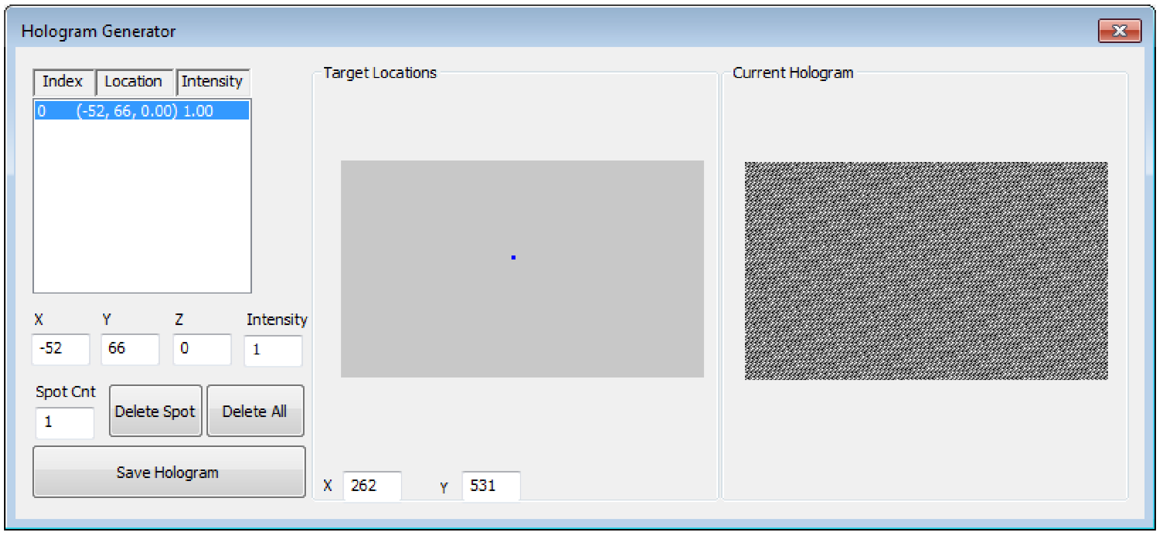


Defocus: 1
Radius: 576
Center X: 960
Center Y: 576
Cylindrical [ ]
Horizontal [ ]

Figure 12 Fresnel Lens GUI

Hologram

The hologram GUI allows the user to create a phase profile that will generate a volume of focal points in the Fourier Plane. The software utilizes a Weighted Gerchberg Saxton implemented on the GPU. This is an FFT based iterative algorithm that by default allows 10 iterations to compute a desired hologram. The user can point and click on the Target Locations to create a focal point, then manually edit the x, y, z location and intensity of the focal point. Location 0, 0, 0 corresponds to the location of the 0th order. Spot locations are shown in the list box in the upper left corner. Using the arrow keys on the keyboard different spot locations can be selected for editing or removal. As the hologram is computed, a preview of the hologram is shown on the GUI, and the hologram is loaded to the SLM.



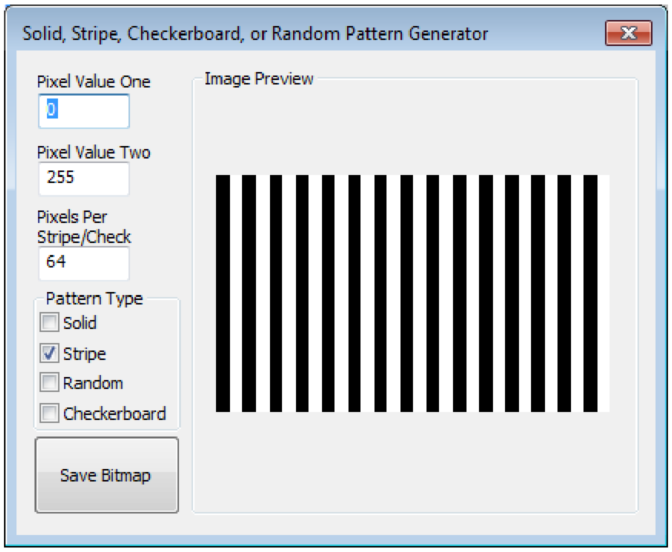
Index	Location	Intensity
0	(-52, 66, 0.00)	1.00

X: -52
Y: 66
Z: 0
Intensity: 1
Spot Cnt: 1
X: 262
Y: 531

Figure 13 Hologram Generation GUI

Solid, Stripe, Check, or Random Phase

This GUI allows the user to create simple stripe, checkerboard, solid, or random phase patterns.



Pixel Value One: 0
Pixel Value Two: 255
Pixels Per Stripe/Check: 64
Pattern Type: [✓ Stripe]

Figure 14 Solid, Stripe, Random Phase, Checkerboard GUI

Superimpose Images

This GUI allows the user to browse for any two images and superimpose them by adding the two images modulo 256. The output image is then processed through the LUT calibration.

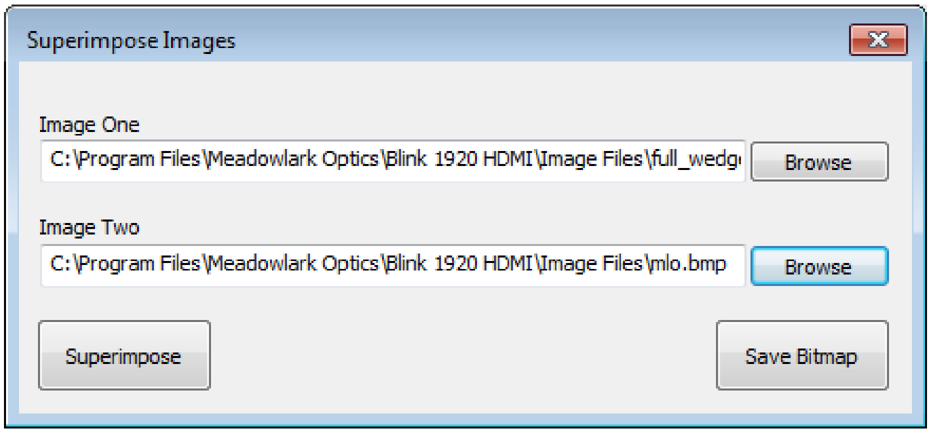
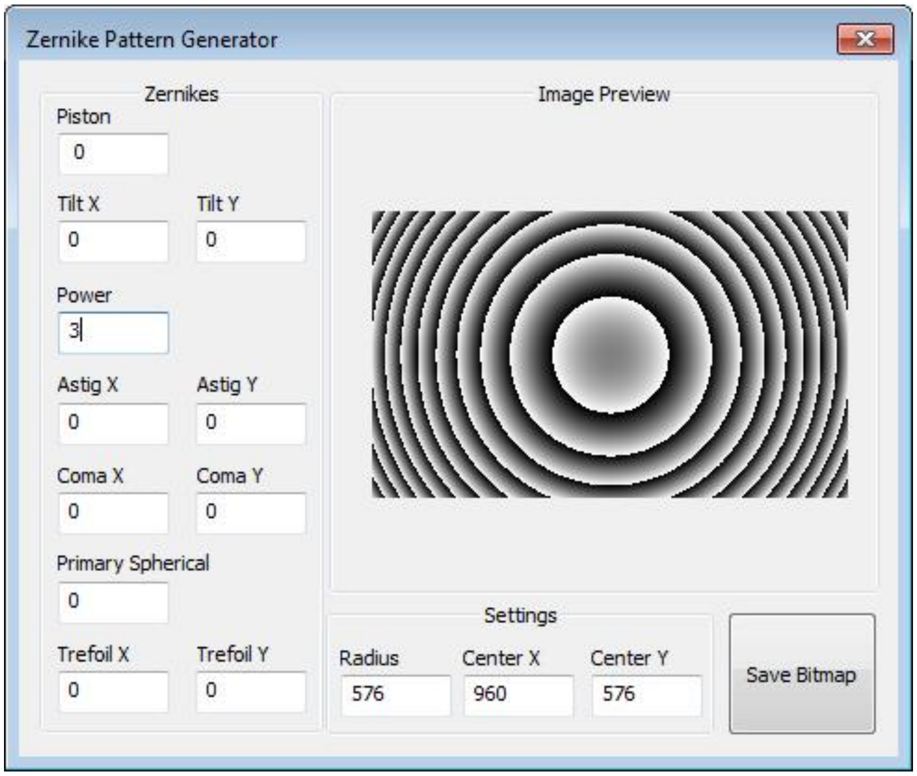


Image One: C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\Image Files\full_wedg
Image Two: C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\Image Files\mlo.bmp

Figure 15 Superimpose Images GUI

Zernike Polynomials

This GUI allows the user to input Zernike polynomial coefficients and output a bitmap. The user must define a radius for the equations, the x and y location of the center of the pattern. If Zernike polynomials are being used to compute a custom aberration correction, it is recommended that the radius be a mid-point between the distance from the center to short axis edge which tends to under-correct the corners, and the center to the diagonal which tends to over-correct the edges.



Piston: 0
Tilt X: 0
Tilt Y: 0
Power: 3
Astig X: 0
Astig Y: 0
Coma X: 0
Coma Y: 0
Primary Spherical: 0
Trefoil X: 0
Trefoil Y: 0
Radius: 576
Center X: 960
Center Y: 576

Figure 16 Zernike Polynomial GUI

4 Software Development Kit (SDK)

Meadowlark Optics offers several software development kit example programs demonstrating how to interface to our SLM from Matlab, LabVIEW, Python, and C++. The example programs can be found at C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\SDK. Each example program demonstrates the order of operations that functions should be called in, the core functions that should be used, and how to link to our Dynamic Linked Library (DLL) to drive the SLM. The following sections outline the functions available and explain the purpose of each parameter passed. The Blink_C_wrapper DLL, and corresponding Blink_C_wrapper.h file must be used for interfacing to the SLM. This section assumes familiarity with LabVIEW, Matlab, Python, and C++ and the Meadowlark SLM hardware.

4.1 Modes of Operation

Figure 17 shows the order of function calls with green indicating initialization, blue indicating the customers loop of loading images to the SLM, and red indicating the proper shut down procedure. The function calls are further detailed in Function Summary and Usage.

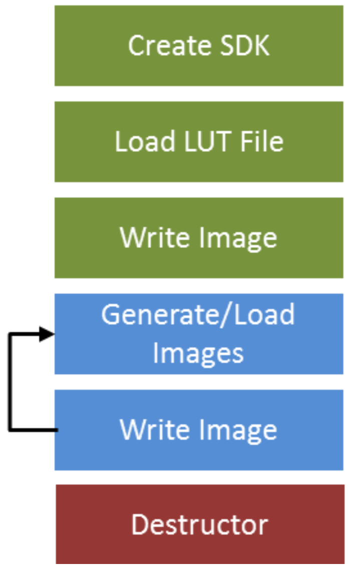
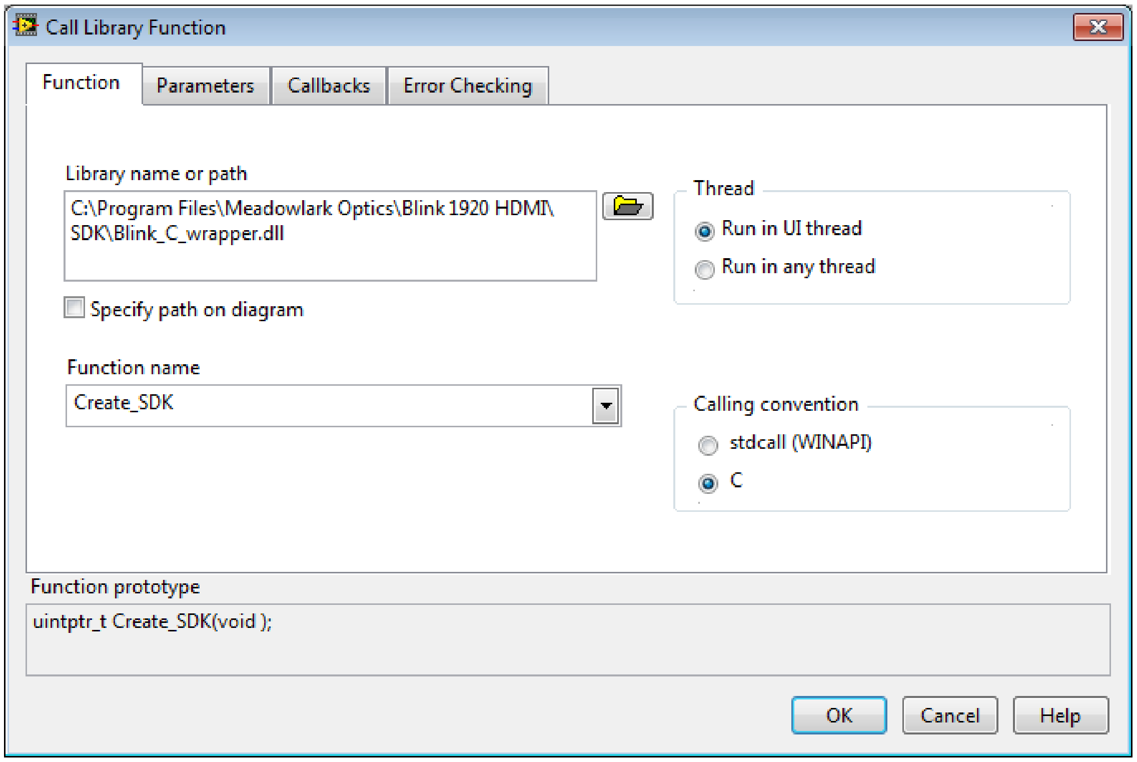


Figure 17 Function Call Order of Operations: initialization (green), looping through images (blue), and shutdown (red)

4.2 Function Summary and Usage

From LabVIEW functions are accessed through Call Library Function Nodes. By double clicking on the function nodes LabVIEW allows the user to edit the function call and parameters, as shown in Fig. 18.



Library name or path
C:\Program Files\Meadowlark Optics\Blink 1920 HDMI\SDK\Blink_C_wrapper.dll

Specify path on diagram [ ]

Function name
Create_SDK

Thread
[✓ Run in UI thread]
Run in any thread

Calling convention
stdcall (WINAPI)
[✓ C]

Function prototype
uintptr_t Create_SDK(void);

Figure 18 LabVIEW Call Library Function Node to interface to the Meadowlark Optics DLL

From Matlab the DLL is accessed through the loadlibrary function, and subsequent calls to calllib as shown to follow. Loadlibrary takes two parameters: the Blink_C_wrapper.dll, and the .h file. The calllib function takes 'Blink_C_wrapper' as the first parameter, the function name as the second parameter, and following that the function parameters as defined in the Blink_C_wrapper.h file. **Note that matlab requires a C compiler.** We recommend installing Microsoft SDK 7.1, then at the matlab command prompt run mex –setup to select the installed compiler.

```
if ~libisloaded('Blink_C_wrapper')
   loadlibrary('Blink_C_wrapper.dll', 'Blink_ C_wrapper.h');
end

calllib('Blink_C_wrapper', 'Create_SDK');
```

4.3 Function Definitions

All function definitions listed below are based on the .h file that Matlab, LabVIEW, C++, and Python can interface to.

void Create_SDK ();

Create SDK opens communication and initializes the hardware. When the constructor is called, a window to display images on the SLM will be created that matches the height and width of the SLM (either 1920 x 1152, or 1920 x 1200). If the SLM is not found, typically because the customer is dumping image data to a physical monitor as opposed to the SLM, then a 1920 x 1200 pixel window is created that will be located directly to the right of the primary monitor and 8-bit operation is utilized. A message will appear notifying the user that a SLM was not detected and that the software is running in simulation mode. **If a physical monitor is used in place of the SLM it is not necessary that the monitor dimensions match the size of the SLM. The monitor will just display a portion of the image intended for the SLM.**

void Delete_SDK ();

This function closes communication to the SDK. Note that in LabVIEW and Matlab calling this function will close LabVIEW and Matlab. We recommend that interaction with the SLM be broken into three separate steps: initialize communication with the SLM, then call the write functions as many times as desired, and last call the destructor when use of the SLM is complete.

int Load_LUT_file (char* file_path);

This function is used to load a calibration to the hardware that corrects for the nonlinear response of the liquid crystal to voltage. If the Look-Up Table (LUT) is correct, then the user can assume that linear increments in graylevel in user defined images translate to linear increments in phase delay on the SLM. Because images are processed through the LUT in hardware, it is important that the LUT file be loaded to the hardware prior to writing images to the SLM. The function takes a path to a LUT file and supports file types of: *.blt, *.lut, and *.txt.

int Write_image(unsigned char* image_data, int is_8_bit)

This function loads an image to the SLM. The function takes two parameters. The first parameter can either be a 1D 8-bit array of image data that has 1920*1152 or 1920*1200 elements or can be an RGB 1D 8-bit array that has 1920x1152*4 elements or 1920*1200*4. RGBA data is expected as follows: pixel 0 Red, pixel 0 green, pixel 0 blue, pixel 0 alpha, pixel 1 red, pixel 1 green, pixel 1 blue, pixel 1 alpha, and so on. **It is expected through the SDK that the array size will match the SLM dimensions.** If an RGBA array is passed, then the second parameter should be set to 0. Otherwise, the second parameter should be 1.

4.4 Optional Functions

int Get_Width();

This function returns the width of the SLM if it is found, otherwise it defaults to 1920.

int Get_Height();

This function returns the height of the SLM if it is found, otherwise it defaults to 1200.

int Get_Depth();

This function returns either 8 or 10 for 8-bit operation or 10-bit operation. If the SLM is not found the function returns 8.

int Set_channel(int channel)

By default the hardware reads the red bits for 8 bit operation, and a combination of red and blue bits for 10 bit operation.

float Get_SLMTemp();

Use this function to read the temperature of the SLM.

float Get_SLMVCom();

Use this function to read the voltage coverglass is set to.

int Set_SLMVCom(float volts);

It is generally not recommended that the user edit the coverglass voltage because improper settings could cause permanent damage to the SLM. Only if instructed by a Meadowlark Optics employee should this value be edited.

4.5 Optional Image Generation Functions

Image generation functions are accessed by including ImageGen.h and linking to ImageGen.dll or ImageGen.lib.

IMPORTANT NOTE: The code used to generate holograms and apply regional LUTs requires OpenCL to be installed on your computer, and requires a graphics card that supports OpenCL. PLEASE CHECK YOUR GRAPHICS CARD CAPABILITIES.

We recommend NVIDIA graphics cards, and use of OpenCL 1.2 or higher. To check your OpenCL version you currently have installed you can download and run GPU Caps Viewer. If you are having trouble finding this program, please contact slmsupport@meadowlark.com **and we will send you a link.**

void Generate_Stripe(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int PixelValOne, int PixelValTwo, int PixelsPerStripe, int vertical, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a stripe pattern consisting of two pixel values, and a user specified number of pixels per stripe.
The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. If vertical is set to 0 the stripes will be horizontal, otherwise they will be vertical. The RGB parameter is always set to true for the HDMI interface.

void Generate_Checkerboard(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int PixelValOne, int PixelValTwo, int PixelsPerCheck, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a checkerboard pattern consisting of two pixel values, and a user specified number of pixels per stripe.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_Solid(unsigned char* Array, unsigned char* WFC, int width, int height, int depth,int PixelVal, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a solid graylevel.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_Random(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a random phase pattern.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_Zernike(unsigned char* Array unsigned char* WFC, int width, int height, int depth, int CenterX, int CenterY, int Radius, float Piston, float TiltX, float TiltY, float Power, float AstigX, float AstigY, float ComaX, float ComaY, float PrimarySpherical, float TrefoilX, float TrefoilY, float SecondaryAstigX, float SecondaryAstigY, float SecondaryComaX, float SecondaryComaY, float SecondarySpherical, float TetrafoilX, float TetrafoilY, float TertiarySpherical, float QuaternarySpherical, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) using Zernike polynomials. The Zernikes center is defined by center x and center y. The radius defines the number of pixels over which one wave of phase change should occur.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_FresnelLens(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int CenterX, int CenterY, int Radius, double power, bool cylindrical, bool horizontal, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a lens function. The center of the lens is defined by center x and center y. The radius defines the number of pixels over which one wave of phase change should occur. Setting cylindrical will either generate a circular or cylindrical lens, if cylindrical the lens can be horizontal or vertical.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_Grating(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int Period, bool increasing, bool horizontal, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a repeating phase ramp. The user must specify the period of the ramp, if the ramp is increasing or decreasing, and if the grating is horizontal or vertical.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_Sinusoid(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int Period, bool horizontal, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a repeating phase ramp. The user must specify the period of the ramp, and if the grating is horizontal or vertical.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_LG(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int VortexCharge, int centerX, int center, bool fork, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a spiral phase. The user should specify the vortex charge (the number of waves of phase delay in the spiral), the x and y location of the center discontinuity, and if a tilt function should be superimposed with the grating.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_ConcentricRings(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int InnerDiameter, int OuterDiameter, int PixelValOne, int PixelValTwo, int centerX, int century, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with concentric rings to make a Bessel beam. The user should specify the inner and outer diameter of the repeating ring structure, the graylevel of the two rings, and the x and y location of the center discontinuity.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Generate_Axicon(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int PhaseDelay, int centerX, int centerY, bool increasing, int RGB);

This function will fill an array to define an image of user defined dimensions and bit depth (8 or 10) with a radially symmetric linear phase ramp. The user should specify the number of waves of phase delay, the x and y location of the center discontinuity, and if the phase delay is increasing or decreasing.

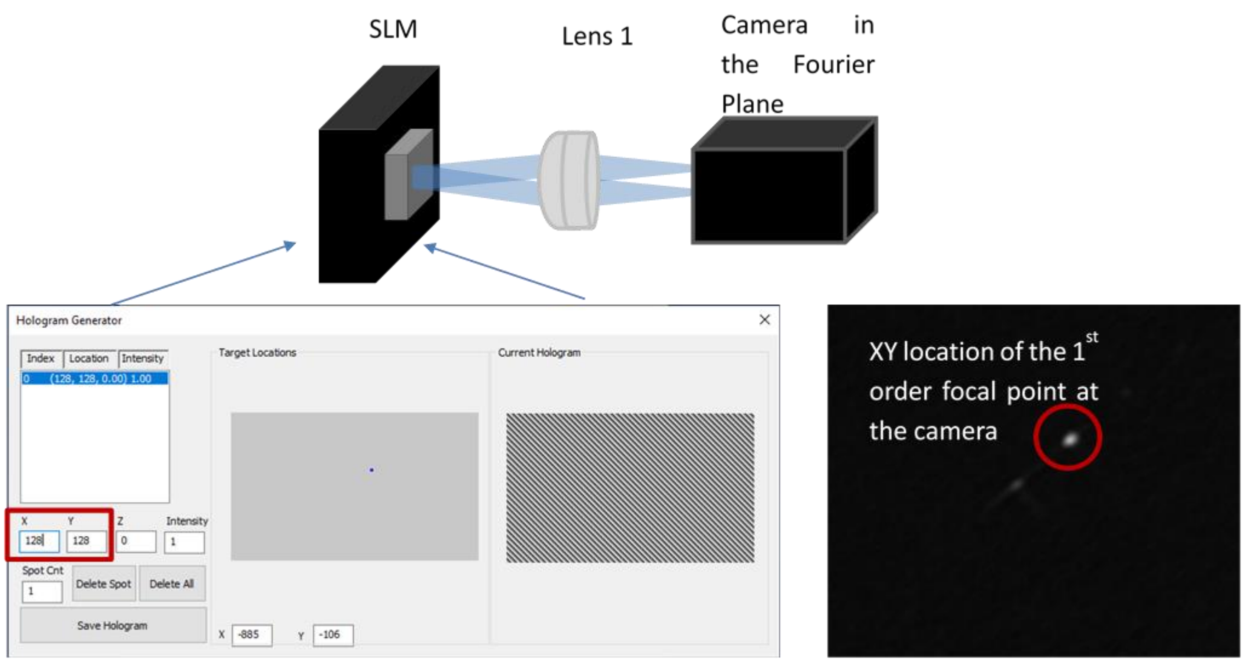
The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

bool Initialize_HologramGenerator(int width, int height, int depth, int iterations, int RGB);

The hologram generator relies on the GPU to complete the required computations. This function opens communication with the GPU, and specifies the height and width of the hologram that will be computed such that buffers can be allocated. This also allocates the number of iterations that will be used in computing the hologram. The RGB parameter is always set to true for the HDMI interface.

int CalculateAffinePolynomials(int SLM_X_0, int SLM_Y_0, int CAM_X_0, int CAM_Y_0, int SLM_X_1, int SLM_Y_1, int CAM_X_1, int CAM_Y_1, int SLM_X_2, int SLM_Y_2, int CAM_X_2, int CAM_Y_2);

An affine transformation can be applied to the hologram computation algorithm to apply a mapping between focal point locations in the image plane, and focal point locations generated by the SLM. To calculate the transformation the user should create a series of three focal points in XY locations in SLM coordinates and record the resulting location of the 1st order focal point at a detector in the Fourier Plane. For example, Figure 19 shows one of the three measurements that must be passed as parameters to CalculateAffinePolynomials: a focal point with a SLM XY location of 128, 128, and a corresponding first order focal point on a camera. The locations will be passed as SLM_X_0 (128), SLM_Y_0 (128), CAM_X_0 (the x location of the 1st order), and CAM_Y_0 (the y location of the 1st order). To get a good calibration it is recommended that focal point locations are recorded when steering a focal point to three different quadrants. For example, SLM XY locations of: (128,128), (-128, 128), and (-128, -128). If 128 is too close to the 0th order to get an accurate measurement a wider angle can be used, such as 256 or 512.



0 (128, 128, 0.00) 1.00

X 128 Y 128 Z 0 Intensity 1

Spot Cnt 1

X -885 Y -106

Figure 19 Typical diffractive calibration data

bool Generate_Hologram(unsigned char *Array, unsigned char *WFC, float *x_spots, float *y_spots, float *z_spots, float *I_spots, int N_spots, int ApplyAffine);

This function computes a hologram to create one or more focal points in the Fourier Plane using a Weighted Gerchberg Saxton. The user should supply an array to be filled with image data by the GPU, an array of x,y,z locations and corresponding focal point intensities, the number of focal points that the user is supplying, and indicate if the affine transformation should be applied. To apply the affine transformation, the user should have already made a call to CalculateAffinePolynomials to generate the required mapping of SLM locations to camera coordinates.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Destruct_HologramGenerator();

This must be called when done using the GPU to compute holograms.

int Initialize_GerchbergSaxton();

For some applications customers would not like to make a hologram to generate focal point locations but would instead like to make a hologram to create an image in the Fourier Plane. There are many algorithms to support calculation of holograms for images, one of which is a Gerchberg Saxton. This function must be called once to initialize the Gerchberg Saxton hologram generator.

int GerchbergSaxton(unsigned char *Phase, unsigned char* InputImg, unsigned char* WFC, int width, int height, int depth, int Iterations, int RGB);

This function computes a hologram of an image using a Gerchberg Saxton. The user should supply an array to be filled with image data by the GPU, an array containing the desired image to be created in the Fourier Plane, the height and width of the SLM, and the number of iterations used in the hologram computation.

The image will be automatically superimposed with the wavefront correction (WFC). The Phase, Input Image, and WFC parameters must be pre-allocated with either: 1920*1152*3 elements or 1920*1200*3 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

void Destruct_GerchbergSaxton();

This must be called when done using the GPU to compute holograms of images.

bool Initialize_RegionalLUT(int width, int height, int depth);

A look up table (LUT) calibration to linearize the optical response of the liquid crystal to applied voltage. A regional LUT is generated by characterizing the phase vs. voltage response regionally (see section 6) such that the calibration is spatially correct across the SLM. This calibration relies on the GPU to take a table of discrete measurements and generate a pixel-by-pixel phase calibration to be applied to the image. This function opens communication with the GPU, and specifies the height and width of the images that will be processed through the calibration such that buffers can be allocated.

bool Load_RegionalLUT(const char* const RegionalLUTPath, float* Max, float* Min);

Using the directions in Section 6 a regional calibration can be generated. This function loads the static table of discrete measurements to the GPU such that the pixel-by-pixel calibration map can be calculated. The function takes a path to the regional LUT file, and fills the contents of empty variables (Max and Min) with the maximum and minimum graylevels found in the regional LUT. The regional LUT is applied in software, and the global LUT applied in hardware should be defined by the user as a linear LUT that ranges from the minimum graylevel in the regional LUT to the maximum graylevel in the regional LUT.

bool Apply_RegionalLUT(unsigned char *Array);

This function processes images through the pixel-by-pixel phase calibration. This must be called prior to loading the image to the hardware.

void Destruct_RegionalLUT();

This must be called when done using the GPU to apply regional phase calibrations to images.

void Mask_Image(unsigned char* Array, int width, int height, int depth, int region, int NumRegions, int RGB);

This function is used in the calibration of a regional LUT to mask off all of an image to 0 except the specified region. For example, if NumRegions is 64, and region is 0, then the SLM is divided into 8x8 regions, and all image data outside region 0 is set to 0. The RGB parameter is always set to true for the HDMI interface.

The Array parameter must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled.

bool SetBESTConstants(int FocalLength, float BeamDiameter, float Wavelength, float SLMpitch, int SLMNumPixels, float ObjNA, float ObjMag, float ObjRefInd, float TubeLength, float RelayMag);

This function sets optical constants that are used in the BEST Bessel beam calculation. Focal Length is the focal length of the lens between the SLM and the amplitude mask. Beam diameter is the diameter of the beam incident on the SLM. Wavelength is the wavelength of excitation. SLM pitch is the pixel pitch of the SLM model used. SLM Num Pixels is the number of pixels along the short axis of the SLM. The objective specifications are the NA, Magnification, and refractive index (of air, water immersion if applicable, or oil immersion if applicable). The tube length depends on the objective manufacturer: Olympus objectives should use a tube length of 180, Nikon of 200, Zeiss of 165, and Leica of 200. The relay magnification is the total magnification from the amplitude mask to the objective.

bool GetBESTAmplitudeMask(float* AmplitudeY, float* Peaks, int* PeaksIndex, float Period);

This function calculates the lateral intensity profile of the excitation at the amplitude mask. The function takes an empty array. The elements in the array can be computed as follows:

Objective Pupil Radius = (Objective NA*Tube Length) / objective magnification
Center Mask = (effective NA * Objective Pupil Radius) / Total Optical Relay Magnification
Amplitude Y Array Length = (Center Mask * 1000 * 2) + 1

The Peaks and Peaks index arrays are arrays with five elements that are used to note the value of the peaks and nulls along the axial intensity profile, as well as the index in the array that contains the peaks and nulls. The indices are noted as follows:

0 – Inner Peak
1 – Inner Null
2 – Center Peak
3 – Outer Null
4 – Outer Peak

If using Case I as described in Rongwen et. al., then the amplitude mask rings should cut on the inner peak and the outer peak. If using Case II, then the amplitude mask rings should cut on the inner null and the outer null. For more information please read the following paper and corresponding supplemental information.

Lu, Rongwen, Wenzhi Sun, Yajie Liang, Aaron Kerlin, Jens Bierfeld, Johannes D. Seelig, Daniel E. Wilson et al. "Video-rate volumetric functional imaging of the brain at synaptic resolution." *Nature neuroscience* 20, no. 4 (2017): 620.

The last parameter passed to this function is the period of concentric ring pattern, which is a floating point number. This is calculated as:

Period = Focal Length Lens 1 * wavelength / SLM pixel pitch / Center Mask;

bool GetBESTAxialPSF(double* axialAmplitude, float* Intensity, float Period, float OuterDiameter, float InnerDiameter);

This function calculates the axial PSF of the Bessel excitation beam. The first parameter is an empty array that will be filled with the axial intensity profile. The length of this array is calculated as:

Axial Amplitude Length = Objective Pupil Radius / Total Optical Relay Mag. * 1000 * 2);

The Intensity array is also an empty array that is filled with the lateral intensity of the excitation at the objective. The length of this array is the same as the Axial Amplitude Length. Last the function takes the period of the concentric ring pattern (see 5.5.21) and the diameter of the amplitude mask cutoff.

If using Case I
Outer Diameter = 2.0 * Peaks[OuterPeak]; //mm, outer diameter of annular ring in the MASK
Inner Diameter = 2.0 * Peaks[InnerPeak];

If using Case II
Outer Diameter = 2.0 * Peaks[OuterNull]; //mm, outer diameter of annular ring in the MASK
Inner Diameter = 2.0 * Peaks[InnerNull];

void Generate_BESTRings(unsigned char* Array, unsigned char* WFC, int width, int height, int depth, int centerX, int centerY, float S, int RGB);

The function takes an empty array that will be filled with concentric ring image data, the width and height of the SLM, and x and y center of the SLM image (generally half the SLM height and half the SLM width), and a floating-point period for the concentric ring period.

The image will be automatically superimposed with the wavefront correction (WFC). The WFC and Array parameters must be pre-allocated with either: 1920*1152*4 elements or 1920*1200*4 elements depending on the SLM in use. The bit depth parameter will determine how the RGB bits are filled. The RGB parameter is always set to true for the HDMI interface.

5 Generating a Custom LUT Calibration

5.1 Background

All Meadowlark LCOS spatial light modulators ship with a custom calibration file called a LUT file (look-up table). The LUT files are generated with one of the standard laser wavelengths available at Meadowlark Optics (405 nm, 532 nm, 635 nm, 785 nm, 1064 nm, or 1550 nm). A LUT file is required because optical response of liquid crystal (LC) to applied voltage is nonlinear. If your optical system uses a different wavelength, or a different angle of incidence, or a different incident power then generating a calibration in your optical system is highly recommended. This document will walk you through the background of the process we recommend for calibrating the SLM and the steps you should follow to generate your own calibration.

5.2 Introduction to the Diffractive Calibration Method

The method Meadowlark utilizes to calibrate measures the optical response of the SLM as a function of applied voltage using diffraction. By tuning the frequency of the diffraction pattern used in the calibration to match the frequency of patterns in your experiments the calibration can be optimized for your experimental conditions. For example, if utilizing high frequency holograms it is recommended that a high frequency diffraction pattern be used to generate the calibration. The provided post processing software will read in the raw measurements you collect and map input graylevels to output phase. The output of the post processing software is either a regional or a global LUT file that is used to linearize the phase response between 0 and ‘n’π (in most cases 0 and 2π).

Figure 20 shows a typical optical setup for diffractive calibration. If a global calibration is being generated, then the SLM should be illuminated with a beam that covers the active area of the SLM without extending beyond the active area. If a regional calibration is being generated, then it is recommended that the incident illumination be as uniform as possible without reducing the regional intensity to the point that the 1st order diffraction pattern is in the noise. A lens should be selected with a long focal length (~100, or 150 mm) to provide enough separation between the diffraction orders to align the detector. An iris may be used in front of the detector to block detection of light from unwanted diffraction orders.

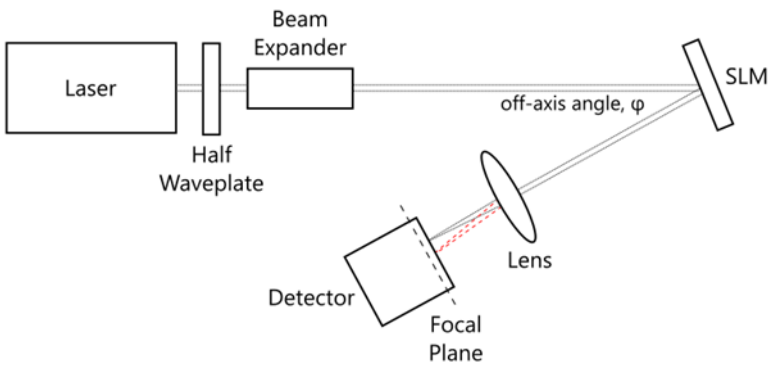


Figure 20 Typical optical setup for diffractive calibration

In the SDK folder Meadowlark Optics provides example programs written in C++, Matlab, and LabVIEW, which are named HDMIDiffractiveTest (.cpp, .m, or .vi). This example program is used to collect the raw measurements required by the post processing program (DiffractiveLUT.exe) to generate a LUT. The example program will generate and load a series of binary phase grating images to the SLM. These images can either be stripes or checkerboards with one value held constant at 0 for the 1920 x 1152, and 255 (8-bit) or 1023 (10-bit) for the 1920 x 1200 and the other varying from 0 to 255 for the 1920 x 1152 and 255 to 0 (8-bit) or 1023 to 0 (10-bit) for the 1920 x 1200. We recommend using stripes if measuring first order intensity to ease alignment. The width of the diffraction grating pattern written to the SLM during this test should be sufficiently small to clearly separate the 0th and 1st order. We typically use, and recommend, either 4 or 8 pixels per stripe. The customer should use a phototdetector and analog input board (such as the National Instruments DAQ board USB-6008) to measure the intensity of either the zeroth or first order diffraction spot as each pattern is loaded to the SLM.

Figure 21 shows typical raw measurements collected utilizing the diffractive method. In this case the 0th order was measured as the diffraction patterns were loaded to the SLM. Collection the 0th order is acceptable for a global calibration, but a regional calibration requires collection of the 1st order. In the example measurements below Grayscale Delta is the difference between the binary grating values. 0% is a fully black image (grayscale value 0), whereas 100% is a fully black and fully white grating (greyscale value 0 and grayscale value 255). The dotted lines represent the phase shift locations for π and 2π. The first null or peak (depending on if the 0th order, or 1st order is collected) is π and the second null or peak is 2π.

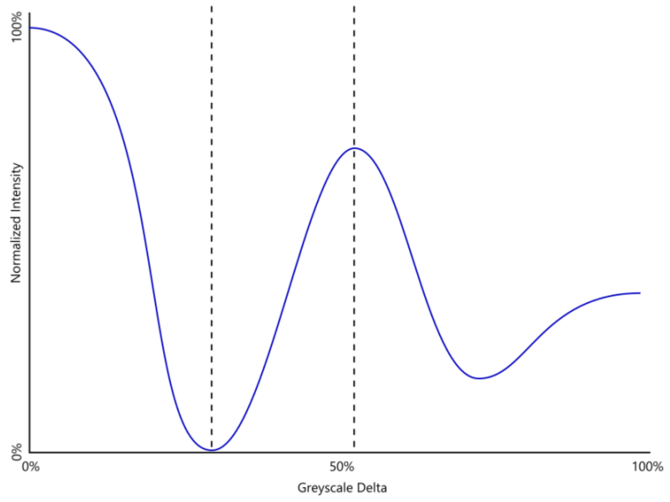


Figure 21 Typical diffractive calibration data

Figure 22 shows a simulated Fourier plane generated by the SLM when the diffractive calibration method is used. If writing binary phase grating stripes to the SLM a ±1st order will be generated about the 0th order. In order to generate a LUT, the customer will use the Meadowlark Optics provided example programs to load diffraction patterns to the SLM and read from an analog input board to store the 1st order intensity (or 0th order intensity) measurements and save the files in an appropriate format for generating a calibration. The contents of the collected files will look like that shown in Figure 21. After collecting the measurements, DiffractiveLUT.exe can be used to generate the calibration.

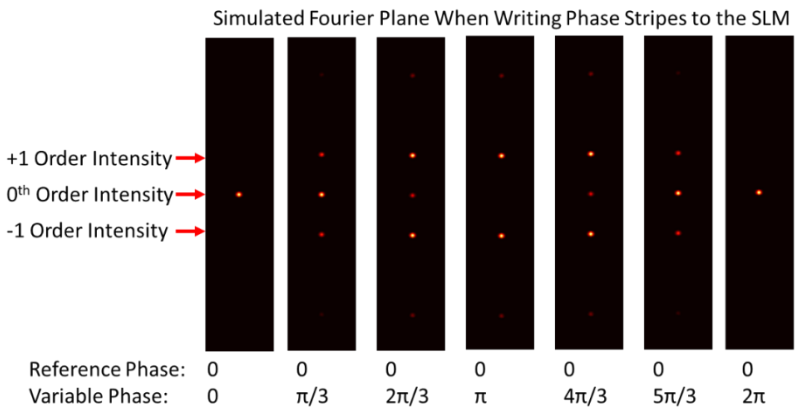

Simulated Fourier Plane When Writing Phase Stripes to the SLM
+1 Order Intensity
0th Order Intensity
-1 Order Intensity
Reference Phase: 0 0 0 0 0 0 0
Variable Phase: 0 π/3 2π/3 π 4π/3 5π/3 2π

Figure 22 Simulated Fourier Plane with stripe patterns

5.3 Collecting the Raw Measurements

Meadowlark optics provides example programs written in C++, Matlab, and LabVIEW that can be used to collect the raw measurements required to generate a LUT. The example programs are named HDMIDiffractiveTest (.cpp, .m, or .vi). The order of operations in the examples is as follows:

1. Open communication to the SLM

2. Load a linear LUT to the SLM – this effectively disables the LUT such that the raw optical response of the liquid crystal to applied voltage can be measured.
3. Loop through the number of regions in the calibration. A global LUT will use one region, or a regional LUT will use 64 regions. The measurements of regional data are built into a pixel-by-pixel 3D map to spatially linearize the phase response of the SLM from 0 – ‘n’π.
4. Loop through the 256 or 1023 diffraction patterns depending on the bit depth of your system. First, the example generates a stripe image. To maximize the SLM switching speed the calibration should be generated from the highest voltage available down to a voltage that gives a phase difference of 2π. However, in some cases customers will opt to calibrate at a lower starting voltage to minimize phase ripple. The table below summarizes the SLM model, grayscale, and corresponding pixel voltage.

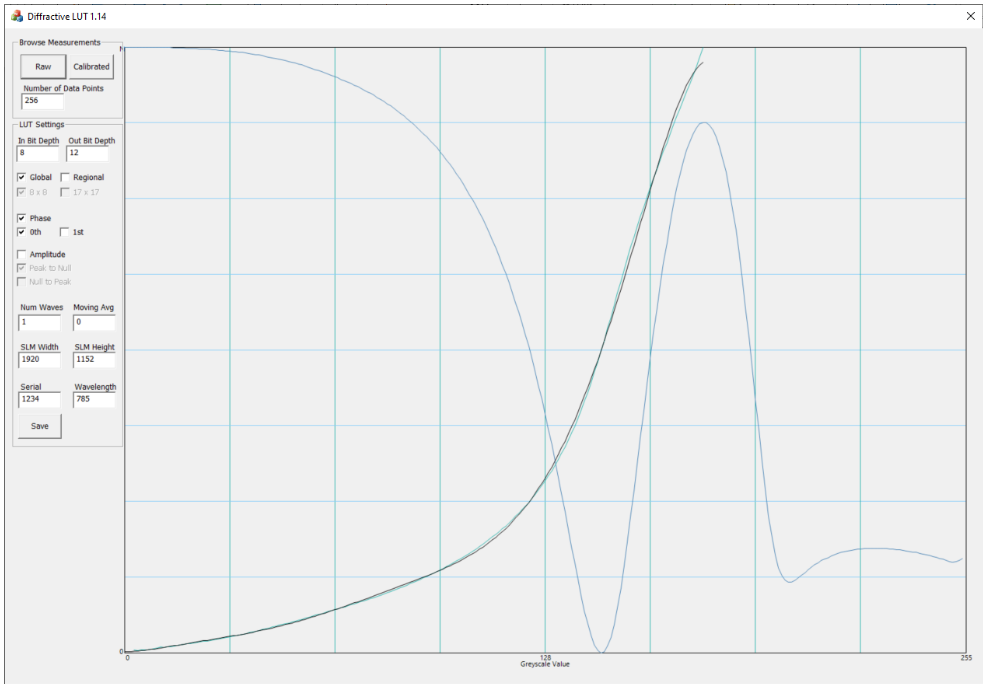
	Grayscale, Voltage	Grayscale, Voltage	Maximize Switching Speed	Minimize Ripple
1920 x 1152	Gray 0, 5 V Pixel	Gray 255, 0 V Pixel	Reference = 0, Variable Grayscale: 0 to 255	Ref > 0, Variable Grayscale: Ref to 255
1920 x 1200 8-bit	Gray 255, 5 V Pixel	Gray 0, 0 V Pixel	Reference = 255, Variable Grayscale: 255 to 0	Ref < 255, Variable Grayscale: Ref to 0
1920 x 1200 10-bit	Gray 1023, 5 V Pixel	Gray 0, 0 V Pixel	Reference = 1023, Variable Grayscale: 1023 to 0	Ref < 1023, Variable Grayscale: Ref to 0

Next, if generating a regional calibration, the image is masked off to graylevel 0 except the current region of interest. This masked image is written to the SLM. A delay allows the liquid crystal to settle into the new phase pattern.

5. After the phase pattern is written to the SLM, **you must fill in your own code to read from your particular analog input board.**
6. After each region is measured, the raw intensity vs. grayscale measurements are stored in Raw[region].csv to be post processed by DiffractiveLUT.exe. For example, if generating a global LUT there will be one file generated named **Raw0.csv**, and if generating a regional LUT there will be 64 files named **Raw0.csv** to **Raw63.csv**.

5.4 Generating a LUT from Raw Measurements

The raw measurements are post processed using DiffractiveLUT.exe. A screen capture of the program is shown for a global LUT in Figure 23, and for a regional LUT in Figure 24. Most of the controls are disabled when the program is first opened. The user can click on the Global LUT checkbox to indicate if they are generating a LUT with one region, or with 64 regions. After setting the number of regions by setting the state of the checkbox, the user can browse to a folder containing the Raw*.csv files. After the raw files have been read in the software will convert intensity to phase, and apply a curve fit to the phase versus grayscale trace.


Diffractive LUT 1.14

Browse Measurements

Raw | Calibrated

Number of Data Points: 256

LUT Settings

In Bit Depth: 8 | Out Bit Depth: 12

☑ Global ☐ Regional

☑ 8 x 8 ☐ 17 x 17

☑ Phase

☑ 0th ☐ 1st

☐ Amplitude

☑ Peak to Null

☐ Null to Peak

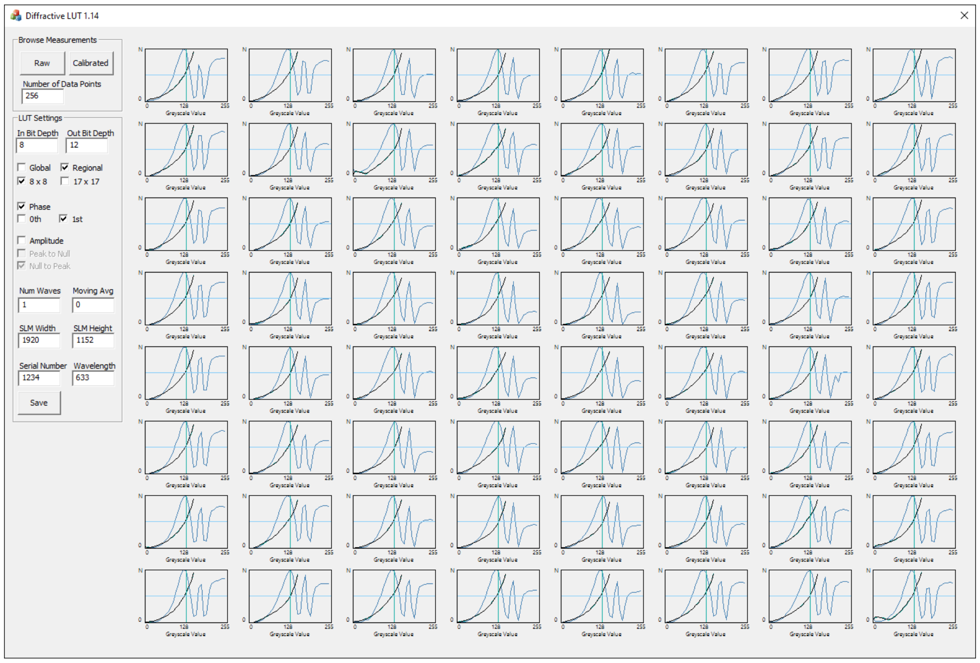
Num Waves: 1 | Moving Avg: 0

SLM Width: 1920 | SLM Height: 1152

Serial: 1234 | Wavelength: 785

Save

Figure 23 Global LUT: After the user browses for the raw measurement, the software will plot the intensity, the un-wrapped phase, and the curve fit. The user can scale the number of waves used in the calibration, and save the calibration.


Diffractive LUT 1.14

Browse Measurements

Raw | Calibrated

Number of Data Points: 256

LUT Settings

In Bit Depth: 8 | Out Bit Depth: 12

☐ Global ☑ Regional

☑ 8 x 8 ☐ 17 x 17

☑ Phase

☐ 0th ☑ 1st

☐ Amplitude

☐ Peak to Null

☑ Null to Peak

Num Waves: 1 | Moving Avg: 0

SLM Width: 1920 | SLM Height: 1152

Serial Number: 1234 | Wavelength: 633

Save

Figure 24 Regional LUT: After the user browses for the folder of raw measurements, the software will plot the intensity of each region as well as the corresponding regional un-wrapped phase, and the curve fit. The user can scale the number of waves used in the calibration, and save the calibration

There are several controls on the GUI that the user should be aware of:

Browse Measurement This portion of the GUI allows the user to specify the number of measurements collected for the calibration. If the input bit depth is 8-bits, then it is expected that the user collected 256 measurements to be post processed. If the input bit depth is 10-bits, then it is expected that the user collected 1024 measurements to be post processed.

Within this section of the GUI are two buttons to browse to a folder containing the measurement(s) to be post processed.

1. **Raw Button** Clicking this button will allow you to browse to a **folder of files** named **Raw[region].csv**. If a global LUT is being generated, then the folder should contain one file named Raw0.csv. If a regional LUT is being generated, then the folder should contain 64 files named Raw0.csv to Raw63.csv. The software will automatically convert the intensity measurements to phase, apply a curve fit, and update the graph. This will also generate excel csv files in the folder containing the raw files. The first file is a global normalization of the raw measurements. The second file is a normalization from 0 to ‘n’π. The third shows the unwrapped phase vs. grayscale measurements. The fourth is a curve fit of the phase vs. grayscale data. The fifth is the normalized LUT where 0 to 1 corresponds to 0 to ‘n’π. Seeing the output of each step of the LUT generation process allows you to see if the program failed to properly convert the raw measurements into a LUT. This can also be checked visually using the plots on the GUI. If the raw measurements are too noisy, the algorithm could fail to properly convert the intensity data to a calibration.
2. **Calibrated Button (optional)** Clicking this button will allow you to browse to Raw*_Calibrated.csv files. This is a way to validate your calibration. First, you can use the provided example programs to generate raw measurements with a linear LUT applied to the SLM and post process the raw files with DiffractiveLUT.exe to produce a calibration. Next, you can repeat the process with the calibrated LUT applied instead of using the linear LUT. Now the output files should be named **Raw[region]_Calibrated.csv**. By browsing to the calibrated raw files you can generate regional or global plots and excel files to validate that the calibrated phase vs. grayscale response is linear.

LUT Settings This portion of the GUI the user specifies details that the program needs to complete the post processing. If the user changes these settings it may be necessary to browse to the Raw file again to post process with the new settings applied.

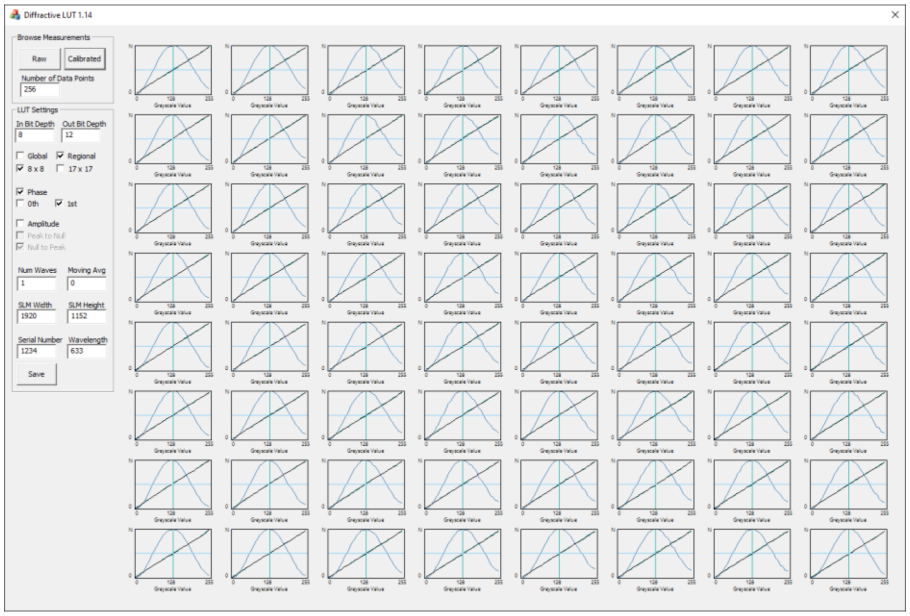
1. **In Bit Depth and Output Bit Depth** For the 1920x1152 pixel SLM and the 1920 x 1200 SLM with an 8-bit controller the input bit depth is 8, and the output bit depth is 12. For the 1920 x 1200 SLM with a 10-bit controller the input bit depth is 10, and the output bit depth is 12.
2. **Global LUT or Regional LUT** Check Global if calibrating the SLM as a single region or check Regional if calibrating regionally. The regional LUT supports 8 x 8 regions, or 17x17 regions.
3. **Phase or Amplitude** Check phase to generate a phase calibration or check amplitude for an amplitude calibration. For a phase calibration either check the 0th order or the 1st order to tell the software which order you measured to collect the raw measurements. For an amplitude calibration indicate if the LUT should be generated from a null to a peak, or a peak to a null.
4. **Num Waves** Using the Num Waves editbox is used for Phase calibrations. In most cases the user will calibrate over 1 wave (0 – 2 π). However, the user can dynamically set the number of waves

the calibration linearizes over: 0.5 waves = 0 to 1π, 1 wave = 0 to 2π, 1.5 waves = 0 to 3π, and 2 waves = 0 to 4π. After editing Num Waves the plot of the intensity to phase measurement and the curve fit will automatically update.

5. **Moving Average** If there is noise in the raw data it may be possible to reduce the noise by applying a moving average. Setting Moving Average to 0 will disable the moving average. Setting Moving Average to 1 will average one data point before, the current data point, and one data point after for each data point in each Raw*.csv file.
6. **SLM Width and SLM Height** Use the resolution of your SLM model.
7. **SLM Serial Number** This is used for naming the output calibration.
8. **Wavelength** This is also used for naming the output calibration.
9. **Save** When this button is clicked the Normalized LUT data is scaled to the appropriate output bit depth, and the user is prompted for a filename and location to save to. A global LUT is saved as a *.LUT, a regionals LUT is saved as a *.txt. Both file formats are accepted by Blink and the SDK functions.

5.5 Post Calibration Validation

If desired, the process can be repeated while applying either the regional or global calibrated LUT as opposed to using the linear LUT. The calibrated phase response should be linear over 'n' waves as shown in Figure 25.



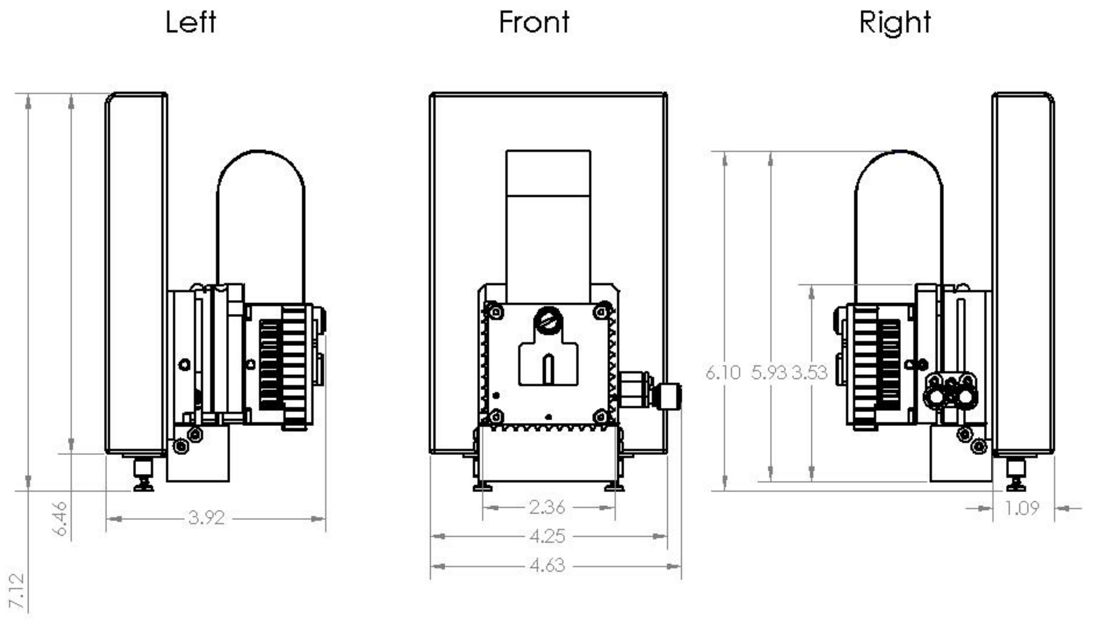
Number of Data Points: 256
LUT Settings
In Bit Depth: 8
Out Bit Depth: 12
Global [✓ Regional]
[✓ 8 x 8] 17 x 17
[✓ Phase]
0th [✓ 1st]
Amplitude
Peak to Null [✓ Null to Peak]
Num Waves: 1
Moving Avg: 0
SLM Width: 1920
SLM Height: 1152
Serial Number: 1234
Wavelength: 633

Figure 25 Post calibration regional LUT validation.

6 Mechanical Housing

The SLM Housing has vertical and horizontal micrometers for adjusting the tip and tilt up to ± 3°. The housing also allows the user to rotate the SLM about the optical axis by ~15°. A set screw (located between the tip/tilt knobs on the side of the SLM housing) locks the rotation about the optical axis.

HDMI 1920 x 1152 System Dimensions



HDMI 1920 x 1200 System Dimensions

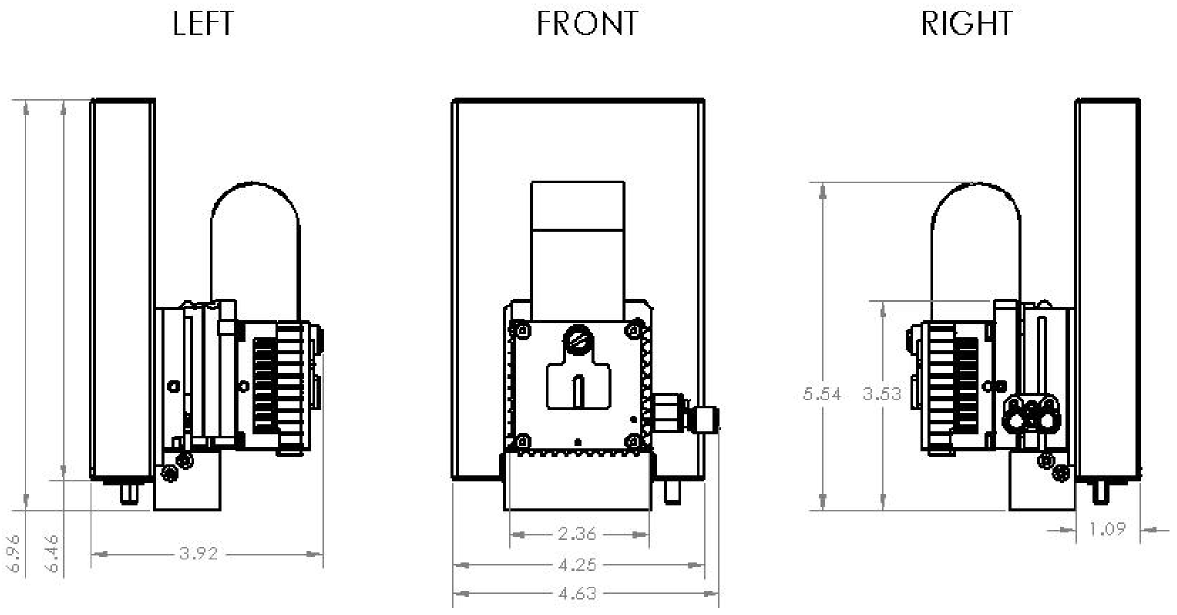


Figure 26 Outline drawing of front and side views of the 1920 x 1152 and 1920 x 1200 SLM Optical Heads. Dimensions in inches.

7 SLM care and cleaning

CAUTION

Failure to follow the recommendations included in this document may violate your warranty. Always ensure all personnel involved in the handling of your SLMs have appropriate tools, equipment, and training prior to cleaning any SLM.

Never use acetone to clean your SLM. Acetone will cause irreparable damage to your SLM.

Never use pressurized air or nitrogen to clean your SLM. Pressurized air or nitrogen will destroy the bond wires that provide the electrical interface to the SLM.

The SLM is shipped with a rotating aperture cover covering the SLM. This aperture, or a similar aperture, should be kept in place while the SLM is not in use to reduce the buildup of dust and debris on the SLM coverglass. When removing the aperture cover, or while the SLM coverglass is exposed, care should be taken to avoid any contact with the coverglass.

If dust or debris does contaminate the coverglass, gently remove them with a methanol soaked lens tissue. Take a small lens tissue or cleanroom wipe and soak a corner of it with methanol. Let the methanol evaporate until the lens tissue is just damp then gently wipe off the particles with the corner of the wipe. For more stubborn debris the methanol drag can be used. Fold a lens tissue several times until it is just small enough to fit inside of the SLM aperture. Soak the folded edge with methanol and let it evaporate until just damp. Line up the folded edge of the lens tissue with the edge of the glass, and drag the tissue straight across the glass. Never wipe the glass more than once with the same lens tissue as it will merely transfer dirt back onto the coverglass. Repeat as necessary.

If there is any dirt or contamination that cannot be removed, please contact the factory for additional cleaning instructions.

8 Troubleshooting

8.1 Image appears incorrect

When loading images to the SLM, make sure the image files are 1920 x 1152 8-bit bitmap images. If the image dimensions are incorrect, it will not be displayed properly on the SLM.

8.2 Computer boots with the SLM as the primary monitor

Shut down and disconnect the Meadowlark HDMI system from the graphics card. Then boot without the SLM attached. The primary should now appear correctly.

8.3 LEDs on the HDMI hardware are blinking

There is dual color LED on the front panel of the HDMI hardware, which indicates various states:

LED behaviour	State
RED	During initial power up, if the red LED is on then external power is present but one of the internal regulators is faulty.
AMBER	FPGA not configured
RED blinking	Temperature alarm, FPGA temperature is above 90C. *Note: the alarm is removed when FPGA cools down to 85C*
AMBER blinking	LCOS not in sync, for example LCOS disconnected or faulty
GREEN blinking	HDMI not in sync, for example HDMI cable disconnected
GREEN	Normal operation, HDMI is in sync, LCOS is in sync

Thank you for purchasing a Meadowlark Optics Spatial Light Modulator.

For additional product and company information, please contact:

Meadowlark Optics, Inc.
5964 Iris Parkway
P.O. Box 1000
Frederick, CO 80530

Telephone: 303-833-4333
Fax: 303-833-4335
Website: www.meadowlark.com

Please feel free to contact us with any questions you may have as well as to leave feedback about your device.

For questions regarding customer support for Meadowlark SLM products, please contact us by telephone or by e-mailing slmsupport@meadowlark.com

For questions regarding purchasing and pricing of additional Meadowlark SLM products, please contact us by telephone or by e-mailing sales@meadowlark.com