USER MANUAL

FlexScan3D

Document revision: A

Copyright

Contact Information
For more information, please contact LMI Technologies.

LMI Technologies, Inc. 1673 Cliveden Ave.
Delta, BC V3M 6V5
Canada

Telephone: +1 604 636 1011
Facsimile: +1 604 516 8368

www.lmi3D.com

End User License Agreement

LMI Technologies SOFTWARE END USER LICENCE AGREEMENT

By agreeing to have any of LMI Technologies Software Products installed on to your computer equipment and by subsequent use of the Software, you agree to comply with the terms of this general End User Licence Agreement ("EULA") where no specific agreement is in place between LMI Technologies and the user of the software. If you do not agree to the terms of this EULA, do not install or use the Software but return it for a full refund. This EULA applies to any upgrades and supplements to the original Software provided.

1. The Licensed Software is owned and copyrighted by LMI Technologies. The Software is licensed, not sold, only on the terms of this EULA. Acceptance and installation of the software indicates your acceptance of the terms and conditions of this EULA.

Upon receipt and installation of the software and payment of the licence fee, you will acquire the right to use the Software in object code form, directly from LMI Technologies although the product may be distributed by a Value Added Reseller (VAR). You assume responsibility for the selection of the program to achieve your intended results, and for the installation, use and results obtained from the Licensed Software.

2. In consideration of your acceptance of the terms and conditions contained in this EULA, LMI Technologies grants you a non-exclusive license to use the Licensed Software and the associated documentation for your own needs on one Server. You are not licensed to rent, lease, or distribute the Software.

3. Title and copyright to the Software, including object code media and documentation, remain with LMI Technologies. You may not copy, reproduce or make data transmissions, in whole or in part, except as is necessary for back-up or archival purposes. You may not reverse engineer, translate, disassemble or decompile the Software, in whole or in part.

4. The licence is effective upon acceptance and installation of the Licensed Software and shall continue until terminated. You may terminate it at any time by destroying the Licensed Software media. LMI Technologies has the right to terminate this Agreement if you fail to comply with any term or condition of this EULA. Upon termination you shall stop all use of the Software and return the Licensed Software and all copies and documentation to LMI Technologies or destroy the Licensed Software and provide LMI Technologies with a statutory declaration signed by you declaring that the Licensed Software and the documentation and all copies have been returned or destroyed and the copy of the Licensed Software on the hard disk has been removed.

5. Copyright and confidentiality of the Software will survive any termination of this EULA in perpetuity.

6. LMI Technologies warrants for a period of ninety (90) days from the date of delivery that the LMI Technologies Software object code will perform the functions of the Software as set out in any LMI Technologies Software Reference Material in effect on the date of delivery.

Except for the warranty stated herein, LMI Technologies disclaims all warranties with regard to the Software, including the implied warranty of merchantability and fitness for a particular purpose.

Some Countries/States have laws which require warranty and liability rights different from those stated herein. In those areas the required warranty and liability terms will apply.

7. LMI Technologies' entire liability and your exclusive remedy, if the VAR from whom you acquired the Licensed Software is unable to deliver acceptable replacement media, is limited to your purchase price, which shall be paid to you upon return of the Licensed Software and the statutory declaration required above certifying complete return.

In no event will LMI Technologies be liable for any loss of profits, loss of use, or indirect, special, incidental or consequential damages in any way related to or arising out of the use of the Software. LMI Technologies maximum liability shall in no event exceed the amounts paid to LMI Technologies for the Licensed Software.

8. The prevailing party in any action or proceeding between LMI Technologies and End-User Licensee arising out of or related to this Agreement shall be entitled to recover reasonable legal fees and costs, including lawyers' fees, which may be incurred.

9. This Agreement shall be construed and enforced in accordance with the laws of British Columbia, Canada and each party agrees to be subject to those relevant laws.

Copyright law protects the Licensed Software and accompanying documentation. Except as specifically authorized in writing by LMI Technologies, copying, duplication, sale, distribution or other use of the Licensed Software is prohibited.

It is understood and acknowledged that LMI Technologies has the absolute right to obtain injunctive relief to protect LMI Technologies' proprietary rights.

By using the Software, you further agree that this is the complete and exclusive statement of the Agreement which supersedes any proposal or prior agreement, oral or written, and any other communications relating to the subject matter of this EULA.

If any provision of this Agreement is held to be invalid or unenforceable the remaining provisions will not be affected.

This software contains Autodesk® FBX® code developed by Autodesk, Inc. Copyright 2010 Autodesk, Inc. All rights, reserved. Such code is provided "as is" and Autodesk, Inc. disclaims any and all warranties, whether express or implied, including without limitation the implied warranties of merchantability, fitness for a particular purpose or non-infringement of third party rights. In no event shall Autodesk, Inc. be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of such code.

Table of Contents

Introduction

Welcome to the FlexScan3D user manual!

FlexScan3D is an innovative 3D scanner software package that allows you to create digital 3D models directly from physical objects in seconds. It is completely scalable and generates high accuracy 3D geometry. This user's guide provides information about installing, calibrating, and using FlexScan3D. If you are a new user, we recommend that you read through the guide in the order presented to walk you through the process of getting set up and introducing important features.

Document revision: A

Getting Started

This sections provides information on system requirements and installation.

System Requirements

Computer Requirements

For optimum performance, follow the recommended requirements.

> FlexScan3D is not compatible with Netbooks or Macintosh computers.

	Minimum	Recommended
Operating System	Windows 7 (64-bit)	
CPU	Intel Pentium IV or AMD Athlon XP or equivalent 2 GHz	Quad-core Intel 2 GHz or better
Memory	4 GB	8 GB or greater
Video Card	DirectX 9.0c compatible, 64 MB (with two video outputs)	AMD or NVIDIA graphics adapter, 512 MB or greater
Free Disk Space	50 GB or more	1 TB or more; 7200 rpm
Ethernet Card	Gigabit Ethernet connection required for the HDI 100 series scanner	
FireWire Card	FireWire 800 Dual Bus card required for scanning with FireWire MV Cameras	
USB (HDI Advance)	USB 3.0	

HDI Scanners and FlexScan3D Software

HDI scanners and the FlexScan3D software are complete 3D scanning systems that include a projector, cameras, lenses, and other equipment to give you everything you need to scan objects. For more information, contact us at LMI Technologies.

Installation

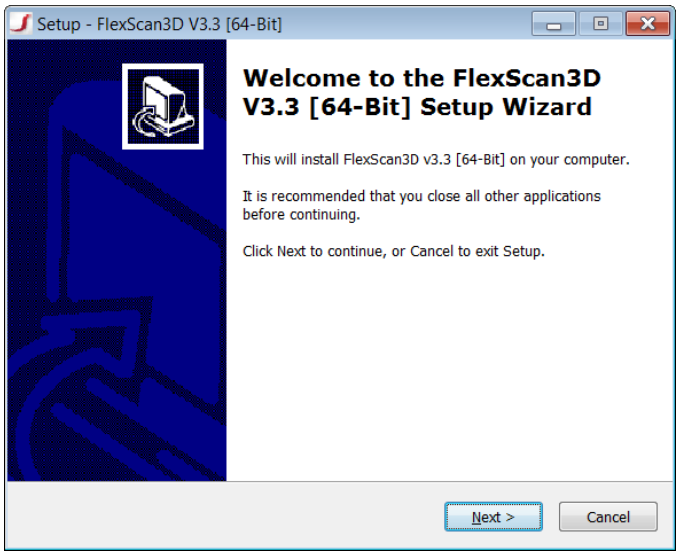
The FlexScan3D setup wizard easily guides you through the installation of the software. Although you can change several settings, the defaults are the recommended settings for maximum ease of use.

To install FlexScan3D:

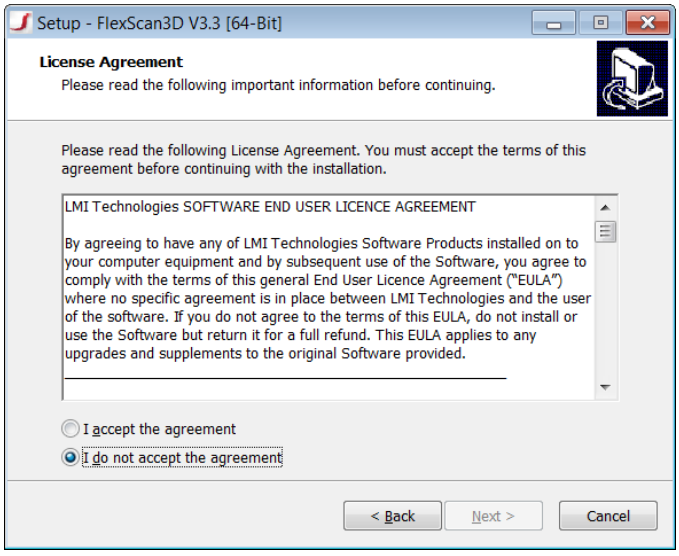
1. Log in to the LMI Technologies Download Center (downloads.lmi3d.com)
   To use this site, you need a download account. To create a download account, send your contact name, email address, company name, and scanner model to orders@lmi3D.com.

2. Click on **HDI Advance Files** or **HDI 100 Files**, and select the software for your version of Windows.
3. After the download is complete, double-click the application icon to start the setup wizard, which will guide you through the installation.
   If a message appears asking if you want to allow the program to make changes to the computer, click **Yes**.
4. Choose the language to use during the installation.

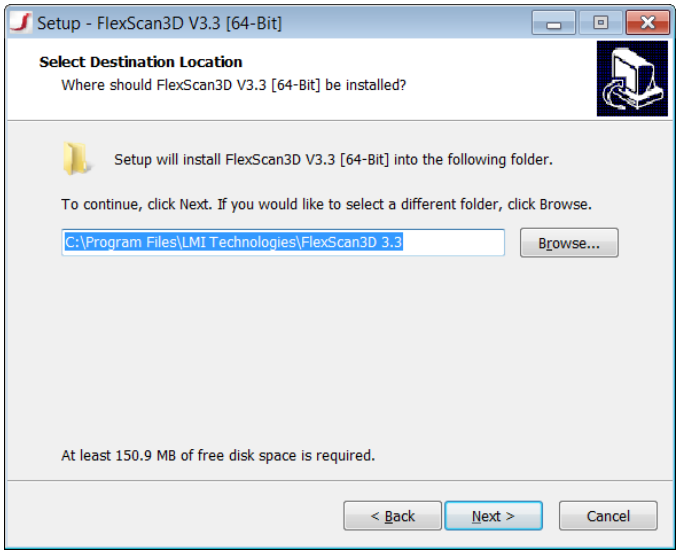
5. Click **Next** to open the **License Agreement** dialog box.



6. Read the agreement carefully. Then check **I accept the agreement** and click **Next** to open the **Select Destination Location** dialog box.

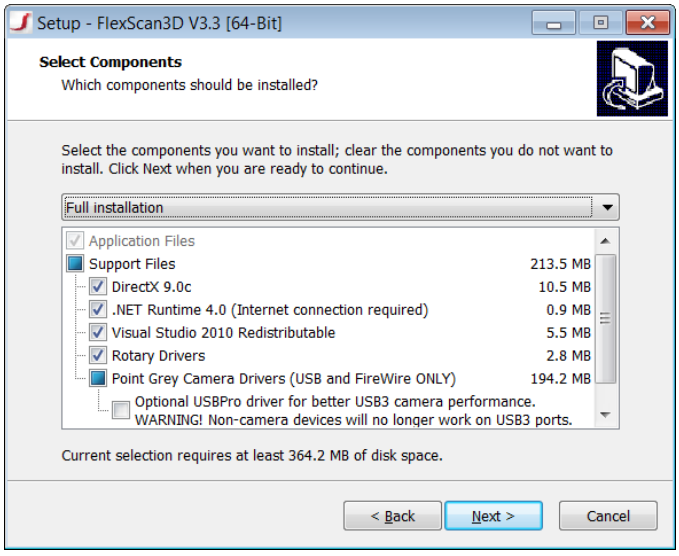


7. Type or browse to where you want to install the program. The default is the recommended installation folder. Click **Next** to open the **Select Components** dialog box.

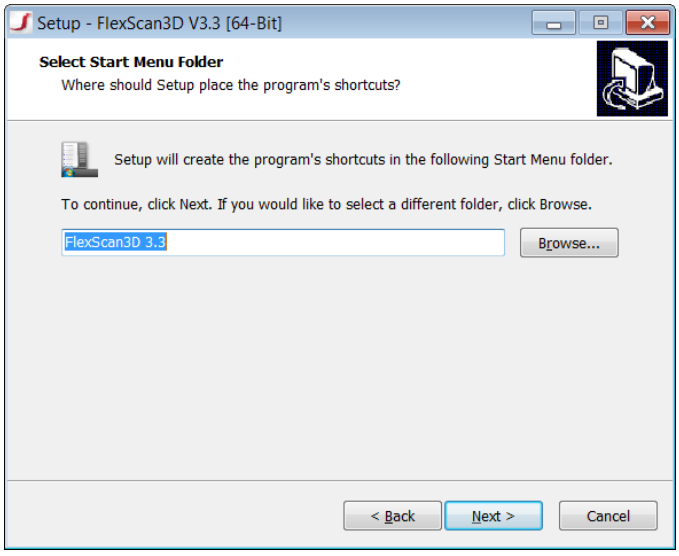


8. Select the components you want to install and click **Next** to open the **Select Start Menu Folder** dialog box.

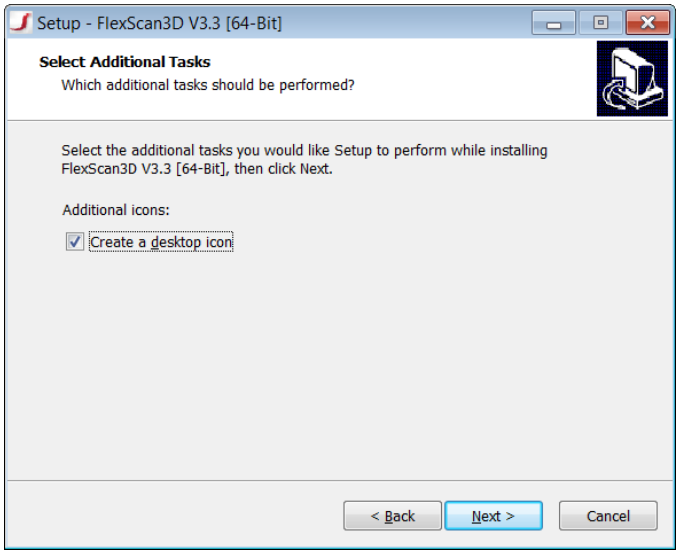
   You should install all of the components that are checked by default to ensure that FlexScan3D runs properly. The optional USBPro driver should only be installed if you need better USB 3.0 camera performance, as this driver prevents non-camera devices from working on USB 3.0 ports.



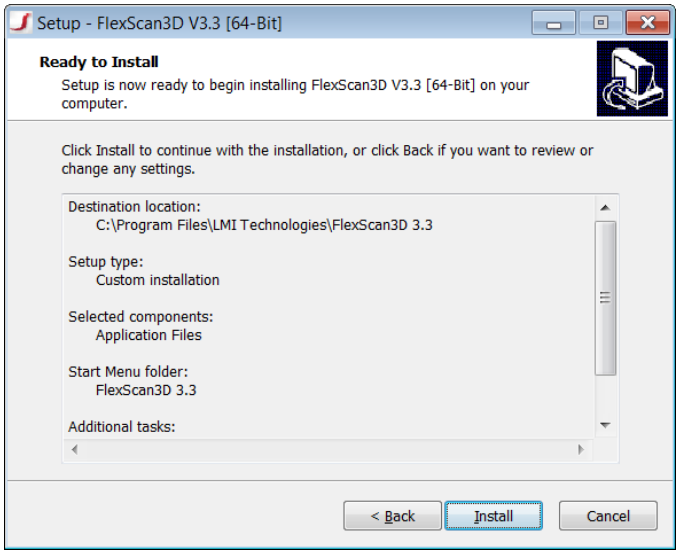
9. Type or browse to where you want to create shortcuts. Click Next to open the Select Additional Tasks dialog box.
   The default is the recommended **Start Menu** folder.



10. Select or clear the **Create a desktop icon** check box and click **Next** to open the **Ready to Install** dialog box.

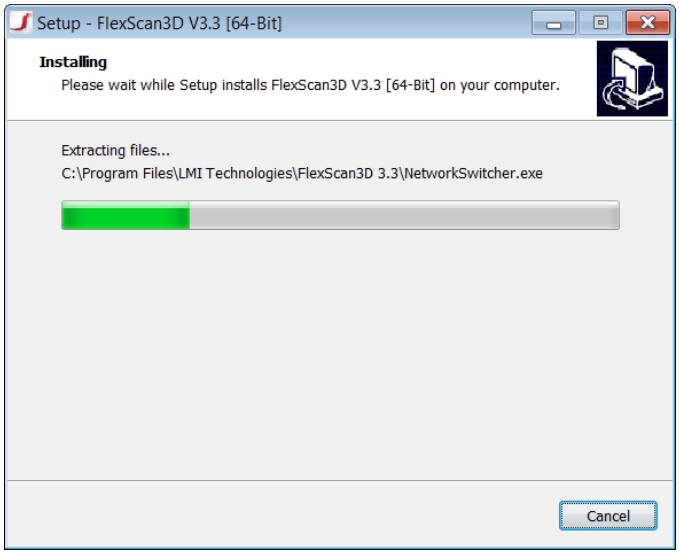


11. Click **Install**.

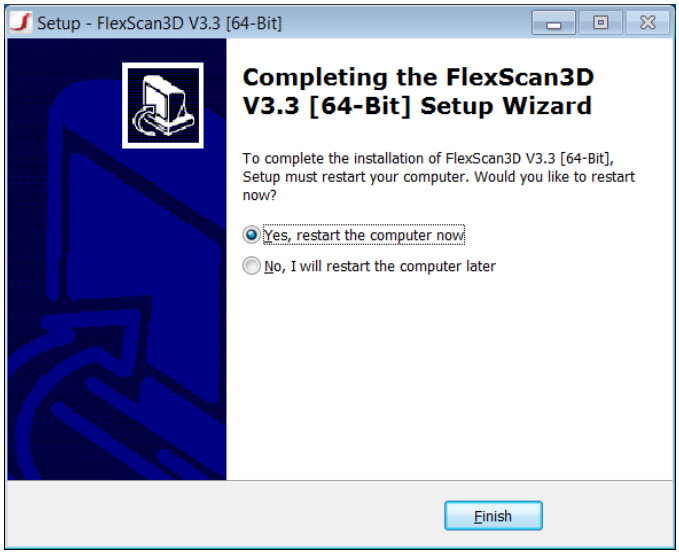


12. The **Installing** dialog box shows the progress of the installation.
    This process may take several minutes.

	Do not power off your operating system during installation.



13. Choose whether to restart your computer immediately or manually later and click **Finish** to exit the installer.
    Make sure to restart your computer before starting FlexScan3D.



To begin using the software, double-click the **FlexScan3D** shortcut on your desktop, or click **FlexScan3D** from the **Start** menu.

Activation and Upgrades

Activation

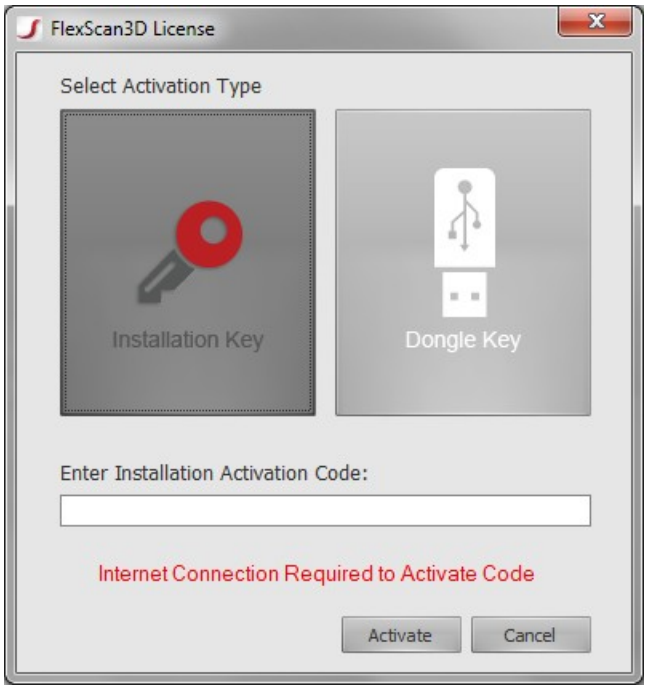
To activate the license for the software, you need an Internet connection. If you do not have access to the Internet, please contact us to discuss switching to a pre-activated USB dongle license.

Installation Key

You will need to activate the installation key the first time you run the software.

To activate the installation key:

1. Start FlexScan3D. The **FlexScan3D License** dialog box opens.
2. Click **Installation Key**.



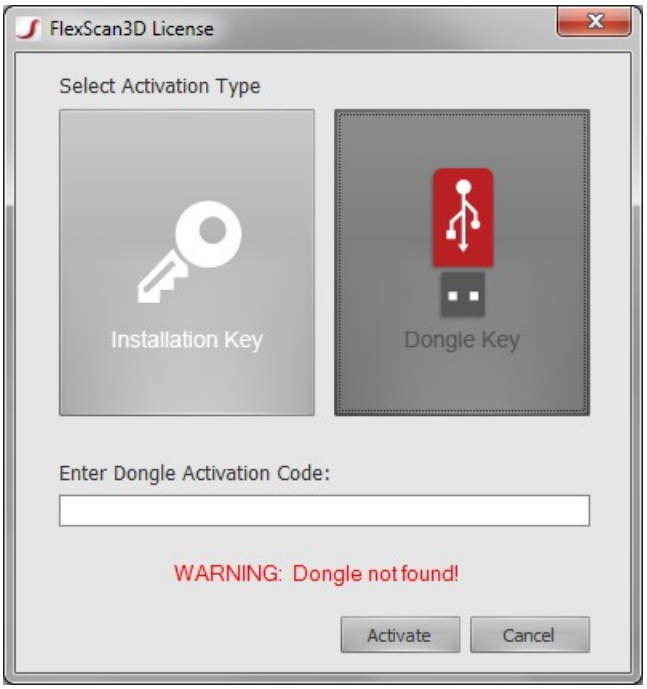
3. Type your activation code in **Enter Activation Code**.
4. Click **Activate**.
   You may need to wait several moments while the software communicates with the license server.
   After the license has been activated, you can start using FlexScan3D.

Dongle Key

To start FlexScan3D using a dongle:

1. Insert the dongle into a USB slot on the computer and wait for Windows to recognize the device.
2. Start FlexScan3D.
3. If a User Account Control (UAC) message appears, click **Yes** to begin using FlexScan3D.

If you try to start FlexScan3D without first inserting the dongle, the **FlexScan3D License** dialog box opens. This dialog box appears any time you start FlexScan3D without either a valid installation key or a dongle.



Insert the dongle. If the dongle contains a valid license, FlexScan will start and you can start using it.

Activating a Dongle

If the dongle does not contain a valid license, it can be activated using a dongle activation code (obtained from LMI Technologies).

To activate a dongle:

1. Click **Dongle Key**.
2. Enter the activation code.
3. Click **Activate**.
   You can now start using FlexScan3D.

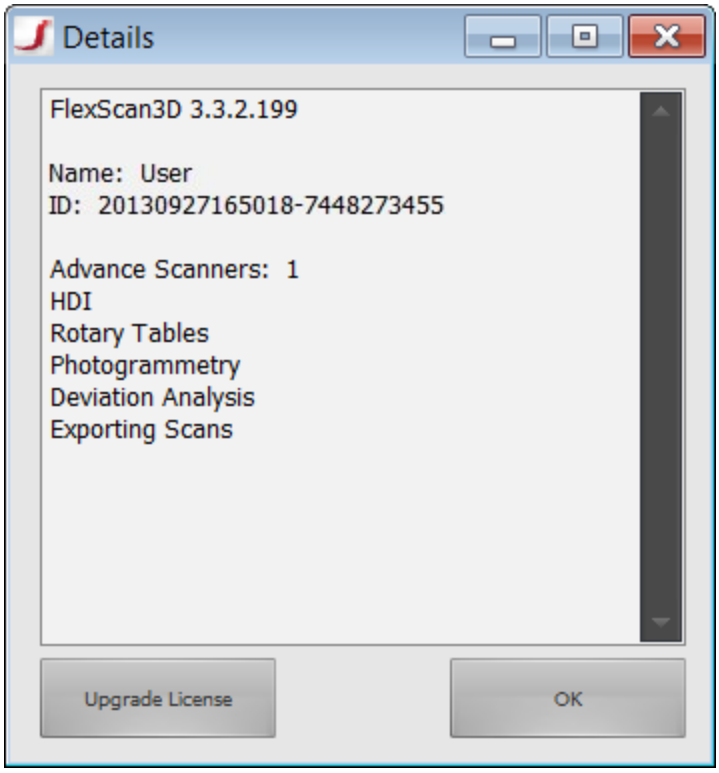
Licensing

Viewing Your License

You can view your license from within FlexScan3D.

To view your license:

1. Click the **Getting Started** tab.
2. Click **License** to open the **Details** dialog box.

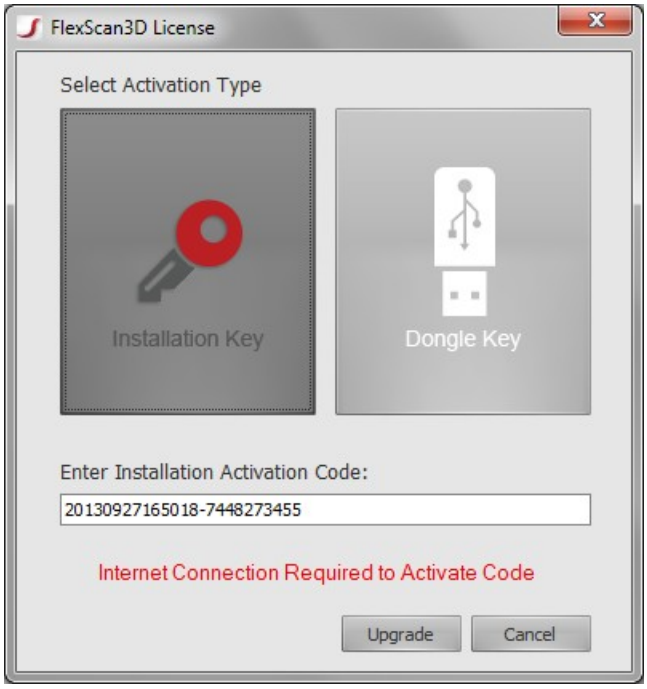


Upgrading Your Licence

To upgrade an existing license to a new one, you must be connected to the Internet.

To upgrade your license:

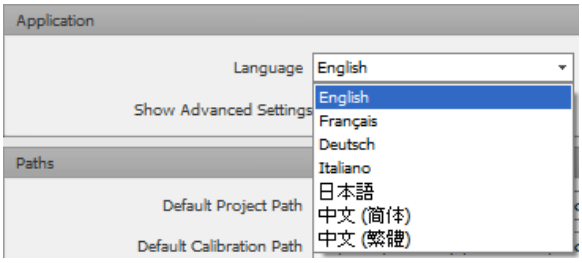
1. Click the **Getting Started** tab.
2. Click **License** to open the **Details** dialog box.
3. Click **Upgrade License** to open the **FlexScan3D License** dialog box.



4. Select either **Installation Key** or **Dongle Key** as the license activation type.
5. Type your new activation code in **Enter Activation Code**.
6. Click **Upgrade** to complete the process and view your new license details.

Interface Language

You can change the interface language of FlexScan3D.



To change the interface language:

1. Go to the **Settings** tab.
2. In the **Settings** tab, click on the **Language** drop-down box and choose the interface language you want.
3. Restart FlexScan3D for the language change to take effect.

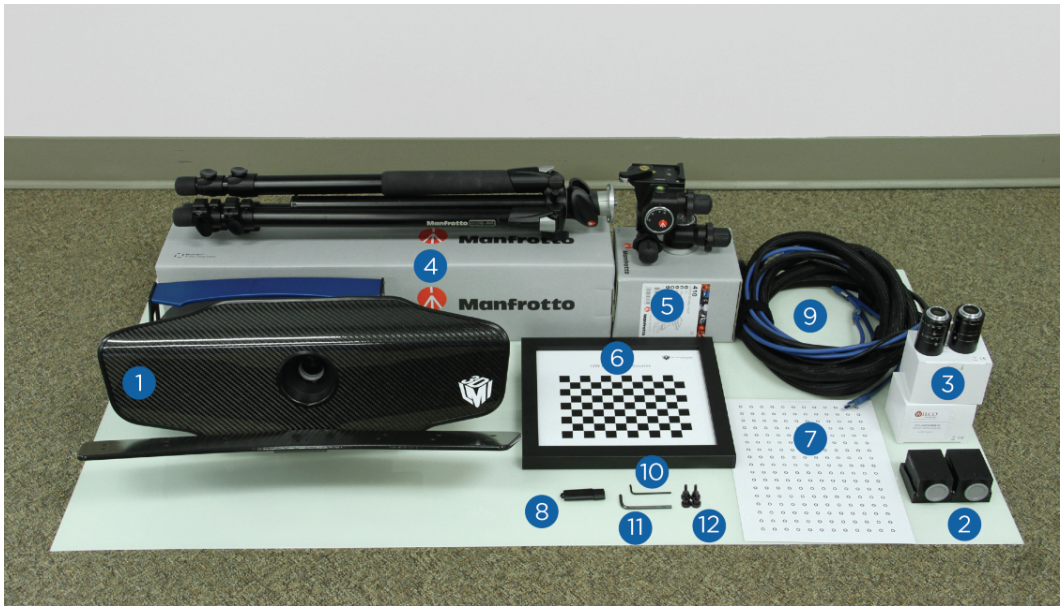
Setting Up: HDI Advance Scanners

This section only applies to the R1x and R3x models of the HDI Advance.

Congratulations on purchasing the HDI Advance 3D Scanner. Here are the steps to set up the capturing unit for 3D data acquisition. In this tutorial the HDI Advance R3x 3D Scanner was used. There are slight variations to the instructions depending on which model you have purchased. Please see the following instructions to assemble your 3D scanner, most of which will be pre-assembled.

Included Equipment and Setup

When you receive your HDI Advance 3D scanner, please verify that all components are included.



1. HDI Carbon Fiber Mount with Projector and Remote
2. 2 x USB 3.0 Cameras
3. 2 x Lenses (12 mm)
4. Tripod leg
5. Tripod head
6. 15 mm Glass Calibration Board
7. 1 Sheet of Photogrammetry Dots
8. FlexScan3D Dongle

9. Power cable, HDMI cable, USB 3.0 cable (3 m)
10. Hex key, 2 mm, 16x83 mm
11. Hex key, 5/32" Short Arm Key
12. 4 x Screws

Physical Setup of Your 3D Scanner

The following video provides an overview of the setup: http://www.youtube.com/embed/9GVQs0Ugbzg

Tripod Setup

Simply open up the tripod legs and place them firmly on the ground.

Attach the tripod head to the legs by firmly twisting the tripod head until it is hand tight.

Connecting the Scanner to the Tripod

Carefully lift up your pre-assembled scanner and locate mount adapter. Gently place the scanner onto the tripod head. This will snap into place when the scanner rests on top of the tripod head.

Attaching the cameras to the scanner

Attach the two cameras to the scanner on both sides and hand-tighten the screws until they are firm. Do not over tighten the screws.

Attaching the lenses to the cameras

Carefully remove the lenses from the packaging and take off the camera lens cap from each camera. The lenses will need to be screwed onto the cameras until they are hand tight.

Connecting the HDMI and power cables to the projector

Attach the HDMI cable and power cables to the back of your projector.

🗋	If your system was delivered with a VGA cable, use that cable or a separately purchased standard HDMI cable.

Connecting the USB 3.0 cables to the cameras

Attach the USB3.0 cables to the back of the cameras, and lock the cables with the locking screws.

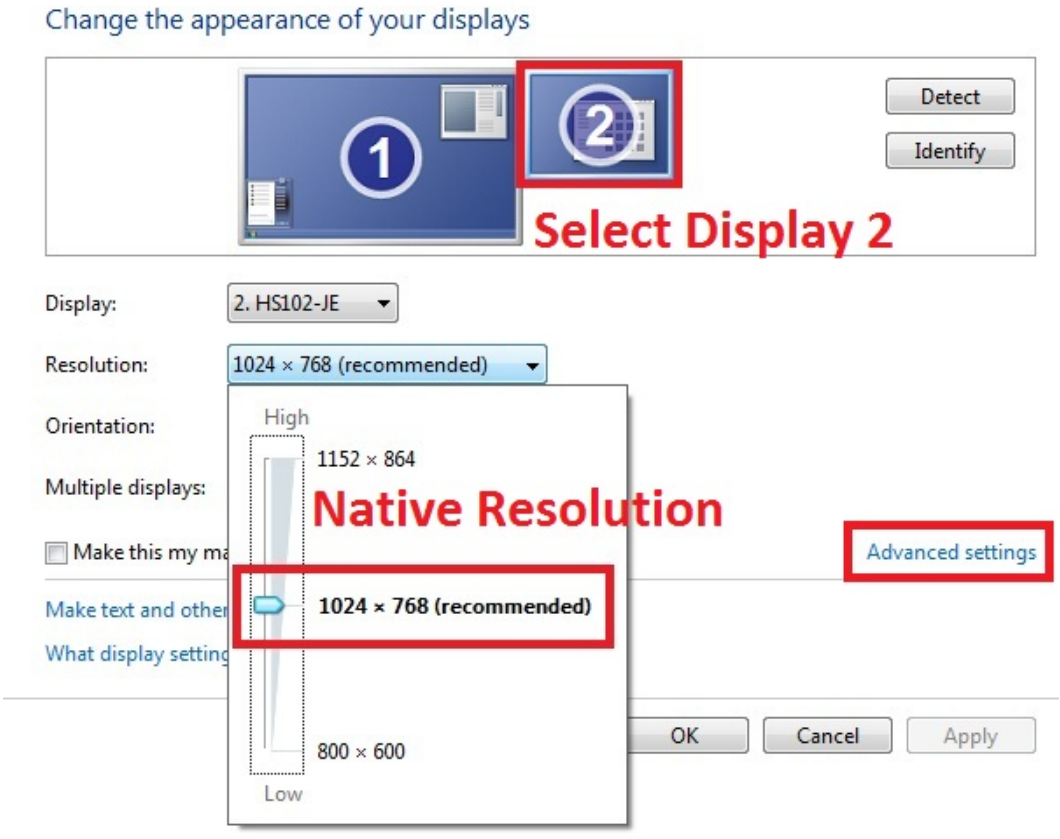
Connect USB 3.0 cables and VGA cable to your computer. Plug the power cable into an electrical outlet. The physical assembly of the HDI Advance 3D Scanner is now complete. Please follow the instructions in the next topic to connect the capturing unit to the computer for operation.

Connecting Your Computer and Projector

Windows 7 and 8

To connect your computer and projector:

1. Connect the projector to the computer using the HDMI cable.
   If your system was delivered with a VGA cable, connect the projector and the computer using that cable, or use a separately purchased standard HDMI cable.
2. Turn on the projector.
3. Right-click on the Windows desktop and click **Screen resolution** to open **Screen Resolution** in **Control Panel**.
4. Select the projector in **Display**.
5. Set the recommended screen resolution in **Resolution**. We recommend using the Native Resolution on your projector. This setting is easy to find within the projector resolution screen. Look for the setting in **BOLD**, which also usually has "Recommended" next to it.

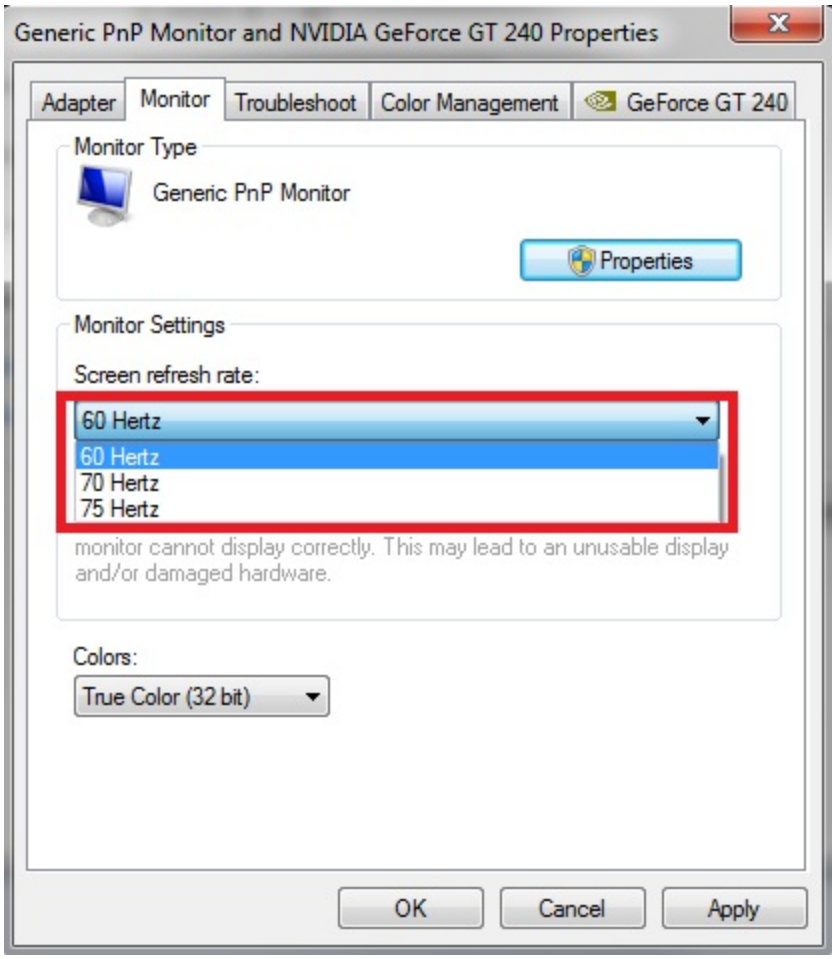


6. Click **OK**. The projector will be available in the **Projector** list in FlexScan3D.
7. Make sure to select **Extend These Displays** in the **Multiple Displays** drop-down list.

It is also important to set the projector refresh rate to 60Hz.

To set the projector refresh rate:

1. Open the **Screen Resolution** dialog box.
2. Select the projector in **Display**.
3. Click **Advanced settings** to open the settings dialog box for the projector.
4. Click the **Monitor** tab.



5. Select **60 Hertz** in **Screen refresh rate**.
6. Click **OK**.

Steps to Get Started on 3D Scanning

We recommend that you go through the following steps in sequence to familiarize yourself with the basics of 3D scanning:

1. Installing FlexScan3D (page 13)
2. Activating FlexScan3D (page 18)
3. Adding a scanner (page 35)

4. Creating a calibration (page 37)
5. Configuring calibration settings (page 37)
6. Capturing calibration images (page 52)
7. Confirming your calibration quality (page 57)
8. Setting up a scan project (page 72)
9. Processing scan data (page 78)
10. Manipulating and editing meshes (page 81)
11. Importing and exporting your scan data (page 92)

Setting Up Your 3D Scanning Environment

Make sure the projector light is brighter than the ambient light in the location you are scanning. For more information on how to set up your 3D scanning environment, please visit our blog post on 3D Scanning in Everyday Environments.

Final Thoughts

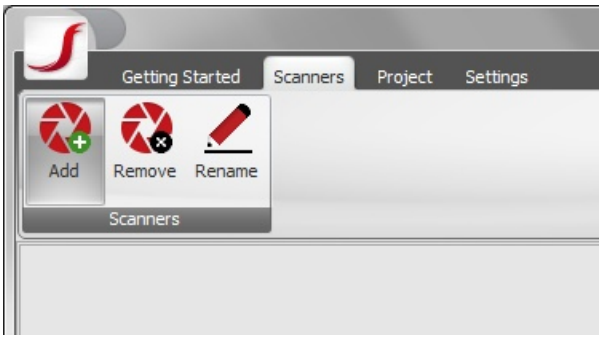
We hope the instructions have given you a good understanding of the 3D scanning process. If you are unsure of anything, see *FAQ / Troubleshooting* (page 179). Happy Scanning!

Adding a Scanner

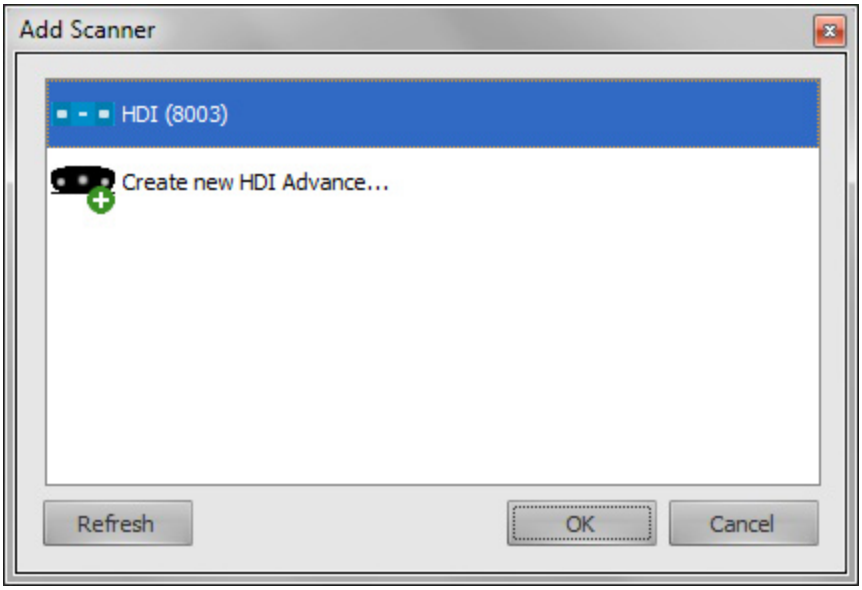
Before adding a scanner, make sure the scanner is connected and powered on. For an HDI Advance, this includes both the cameras and projectors, ensuring that the projector is turned on and is available in the Windows display settings.

To add a scanner:

1. Go to the **Scanners** tab and click the **Add** button on the ribbon.



2. In the list of calibrated scanners attached to the system that is displayed, select the desired scanner.



If you are setting up an HDI Advance for the first time, select **Create new HDI Advance...** from the list.

3. Click **OK**.

If this is a new HDI Advance, the next step is to calibrate the scanner. See *Calibration Overview* (below) for more details.

For a calibrated HDI Advance, see *3D Scanning Basics* (page 70), or alternatively, see *Setting Up Your Scan Project* (page 72).

Calibration Overview

A scanner calibration defines the conversion from 2D images into 3D geometry. Calibration is perhaps the most important part of setting up your 3D scanner. When calibrating for the first time, try not to rush the process. It may take some practice to achieve the desired calibration accuracy. An accurate calibration is the key prerequisite for capturing high-quality scan data. Note that this applies to the HDI Advance line of products only. HDI 100 series scanners are calibrated during the manufacturing process, and as a result they do not require user calibration.

Scanner Types

Calibration can be done with either one camera and a projector, referred to in this manual as a "Single" scanner, or two cameras and a projector, referred to as a "Duo" scanner. Duo scanner configurations are generally preferred due to the improved levels of accuracy. A "Multi" scanner simply refers to the use of multiple scan heads.

Duo

A duo calibration requires two identical cameras and one projector. It is calibrated by capturing images of a calibration board from both cameras. The images are then compared in order to determine the 3D coordinates of the cameras relative to the target scanning area. Duo scanner is recommended for users who require high accuracy.

Single

A single calibration requires one camera and one projector. It is calibrated by capturing images of projected patterns on a calibration board. The software analyzes these images in order to determine the 3D coordinates of the camera relative to the projector, as well as to the target area. Single scanner is recommended for users who require maximum coverage.

Scanner Type Quick Comparison Chart

Calibration Type	Number of Cameras	Number of Projectors	How to Calibrate	Advantages
Duo	2 (identical)	1	Capture images of a calibration board from both cameras	Provides higher accuracy
Single	1	1	Capture images of projected patterns on a calibration board	Provides maximum coverage
Multi	Multiple single and/or duo scanners as described above		Calibrate each scanner as described above	Captures the object from multiple locations and angles at the same time and allows for more coverage from a single press of the Scan button

Step-by-Step Instructions (Video)

Single Scanner Calibration

watch

Duo Scanner Calibration

watch

Configuring the Calibration

Capture Settings

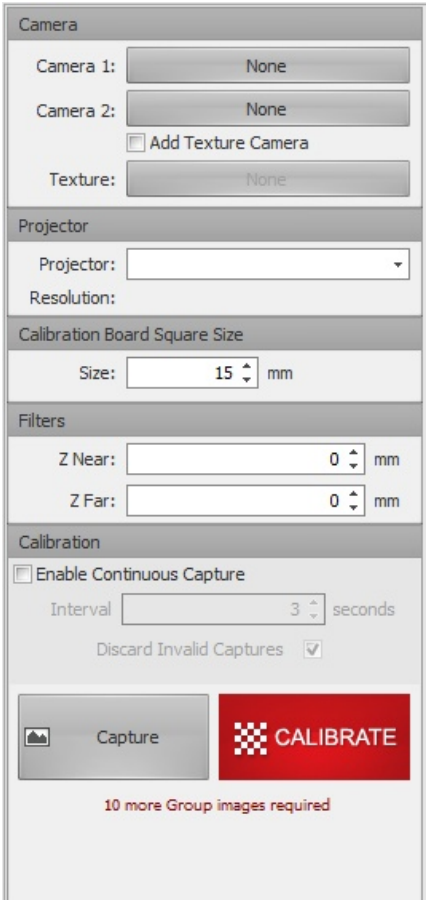
> It is HIGHLY recommended that you view our step-by step instruction videos (page 37). This is an essential step and must not be skipped. The information gained by viewing the tutorial videos will greatly help you understand this process.

Selecting Cameras

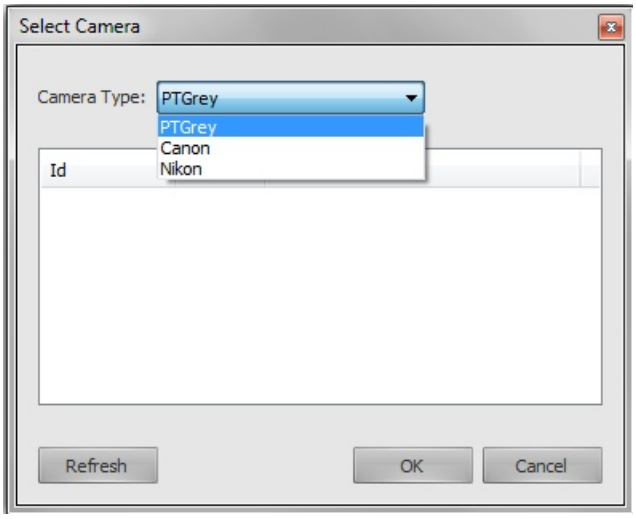
> Make sure your camera is connected and that you have installed camera drivers. We do not recommend manually installing the latest drivers from Point Grey Research, as they may be incompatible with the latest version of FlexScan3D and may cause performance issues.

To select cameras:

1. Click on the **Scanners** tab at the top of FlexScan3D.
   You will be making adjustments in the panels shown below.



2. Click the button next to **Camera 1** to open the **Select Camera** dialog box.



3. Select the camera brand in **Camera Type**.

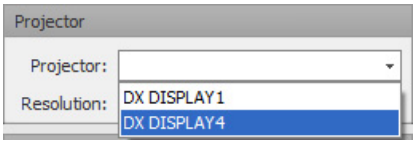
4. Select the camera from the list.
   If there are multiple cameras attached to the computer and you are unsure which one to choose, select the first one, then confirm which one it is by checking the live camera feed. If you selected the wrong

camera, repeat steps 2 though 4, but select the next camera in the list instead.

5. Click **OK**.
   After you select a camera, you can rename the camera to make it easier to distinguish from other cameras. To rename a camera, click the arrow to the right of the camera button and type the new name.
6. For a Duo scanner, do the following:
   a. Click the button next to **Camera 2** to open the **Select Camera** dialog box.
   b. Repeat steps 2 through 5.
7. If your system includes a separate texture camera, do the following:
   a. Select the **Enable Texture Camera** check box.
   b. Click the button next to **Texture** to open the **Select Camera** dialog box.
   c. Repeat steps 2 though 5.

Selecting a Projector

To select the projector, select it from the drop-down list:

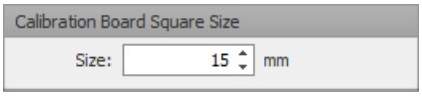


> The **Resolution** is set automatically after capturing the first image, and you cannot change it afterward.

Other Settings

Selecting the Calibration Board

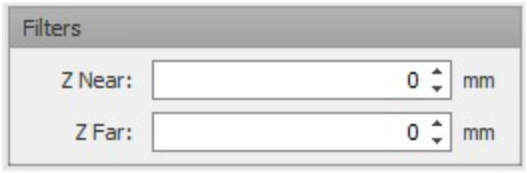
Calibration boards can vary in size. The Calibration Board is defined by the size of the square in mm. Smaller calibration boards are better for scanning small objects, while larger calibration boards are necessary to ensure accuracy for scanning large objects.



Setting Filters

Setting filter values allows you to clip near and far data points, and are based on the distance from the scanner to the target scanning area. For example, if the wall behind an object keeps showing up as geometry in the scan data, you can reduce the Z Far value so that any data corresponding to the wall is automatically removed.

To set filters, on the far left of the screen, on the **Settings** tab, adjust the distances in millimeters for the **Z Near** and **Z Far** values.



Recommended Next Step: Adjusting Your Equipment (page 40)

Adjusting Your Equipment

Scanner Menu Bar

The scanner equipment controls are accessed via the menu bar on the right side of FlexScan3D. Depending on the system, the following are some of the buttons present on the menu bar.

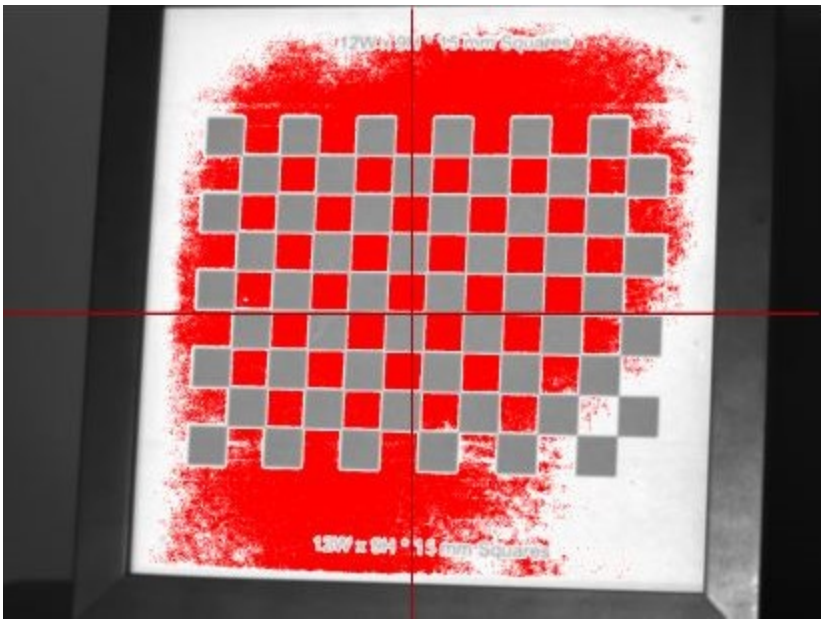
Command Icon	Description
	Show/Hide all scanners (for multi-scanner calibrations only).
	Show/Hide HDI Advance scanner control.

When using a multi-scanner calibration, each scanner is represented by its own button on the menu bar.

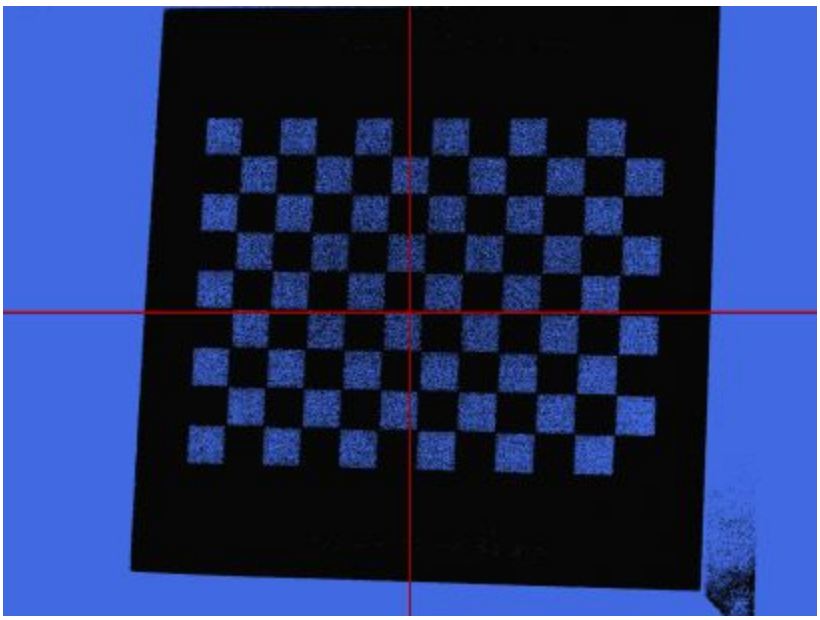
Camera Exposure

The exposure of the images captured by a camera depends on the projector brightness, camera shutter speed, and camera aperture. Shutter speed controls how long to let light pass through (duration), while aperture controls how much light passes through the camera lens (quantity).

If the camera live video feed is running, it will automatically indicate areas where the image is overexposed (in red) and/or underexposed (in blue).



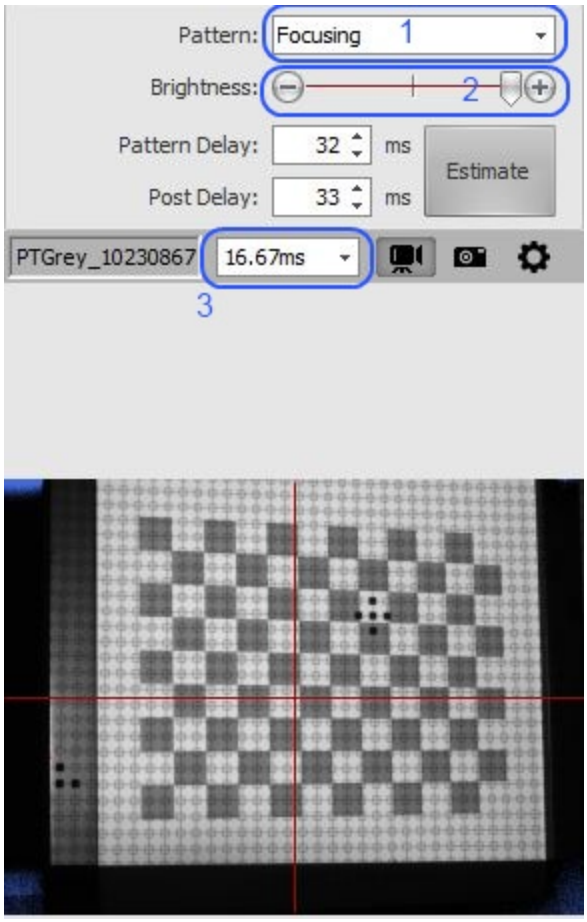
Over-exposed image in live feed



Under-exposed image in live feed

Exposure Calibration

To ensure your scanner hardware is optimally configured, position the calibration board in front of the scanner (in the target scanning area). Note that the use of a calibration board is not a requirement for setting the exposure; it is used here as an example object since it consists of easily distinguishable light and dark areas. Then follow the steps below.



To calibrate exposure:

1. Open the scanner control on the right side of the application.
2. Set the projector pattern to **Focusing**.
3. Slide the projector brightness slider all the way to the right.
4. Set the shutter speed to the fastest time possible (first item in the drop-down list).
5. Turn on live video if it is not already on. If live video is not available for your camera model, you will need to use screenshots to test each camera setting change.

Now adjust the camera lens aperture so that the calibration board is clearly visible without showing any exposure highlights.

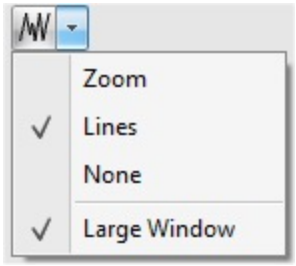
Fine-Tuning Exposure

Fine-tuning exposure will produce maximum scan quality.

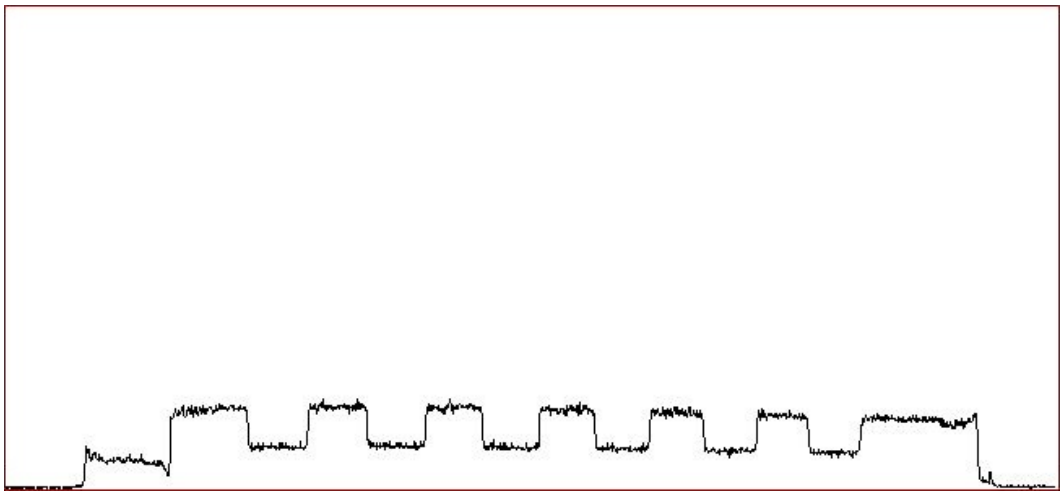
To fine-tune exposure:

1. Set the projector pattern to **White**.

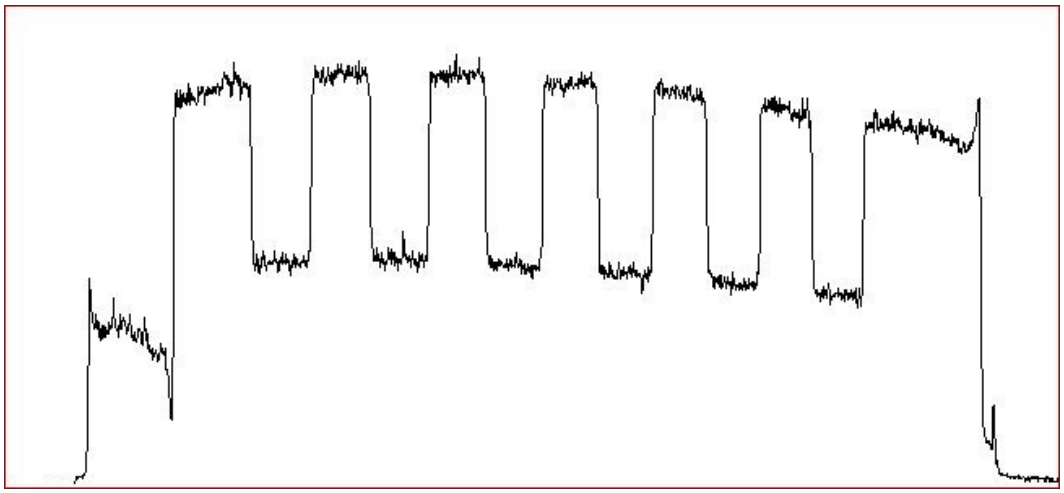
2. Place your cursor over the video feed from the camera.
3. Click on the icon that appears in the top-left corner of the video feed and click **Lines**.



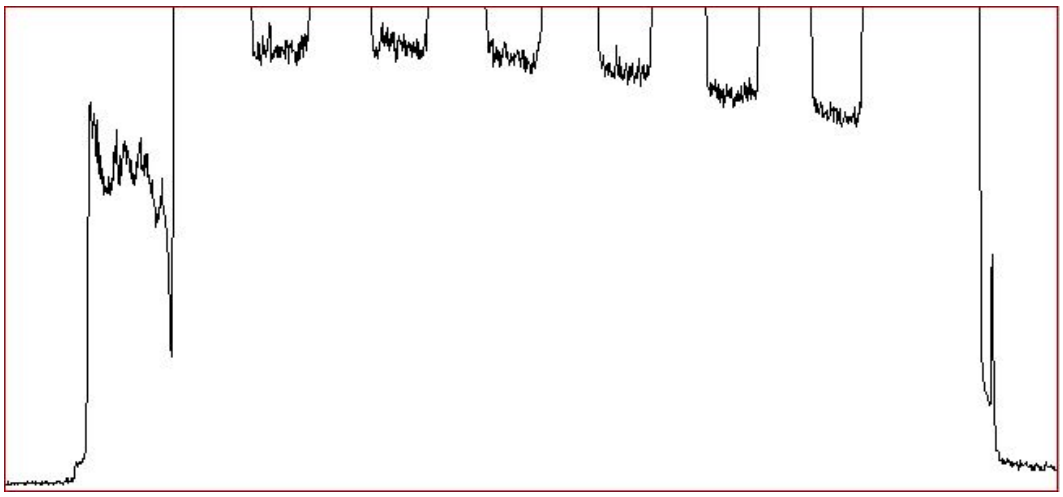
4. Hover over the video feed to view the horizontal brightness levels of the image.
5. Rotate the aperture ring while looking at the line viewer. You should notice the lines moving up and down depending on the direction of rotation. No values should touch the top of the line view window (overexposed), but the upper values should not be too low either (underexposed). The goal is to have good contrast between light and dark areas.



Under-exposed



Optimally exposed



Over-exposed

A Note About Aperture

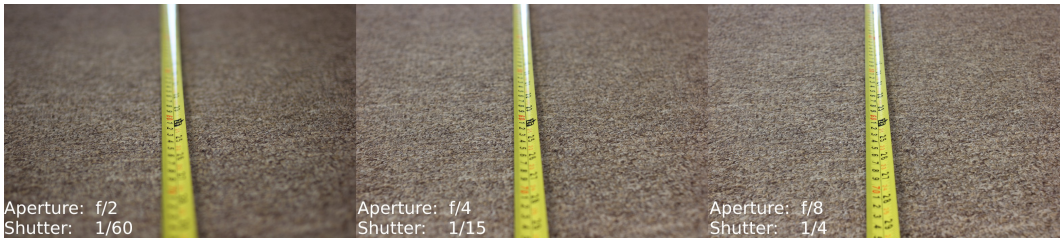
Aperture is usually measured in f-numbers or f-stops. An f-stop of 1.4 (larger hole) will result in a brighter image than an f-stop of 8.0 (smaller hole). Larger stops (smaller f-numbers) can use faster shutter speeds, which results in faster scanning. There is a drawback though: Smaller f-numbers result in a reduced depth of field.

A reduced depth of field means that there is a very narrow region in terms of the distance up to which you can scan an object. Larger objects may not fit within this focus window, which means that accuracy would be reduced for the near and/or far surfaces. This means that when adjusting the aperture for brightness, try to use as large an f-stop number as possible to maximize the focus area. This is why it's best to ensure the projector is as bright as possible before adjusting the aperture, which allows you to use a larger f-stop number to bring the exposure back down to an ideal range.

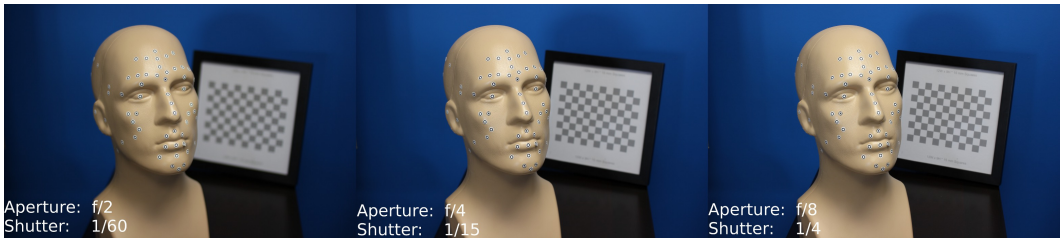
Generally, we recommend using an f-stop between 5.6 and 11, and most of the time we use an f-stop of 8.

Aperture Examples

The following examples illustrate the differences between aperture f-stops. With a smaller f-stop number, the depth of field is much narrower than the larger f-stop numbers. The shutter speed is adjusted in order to keep the overall exposure consistent across the f-stop. The shutter speed has no effect on the focus range.



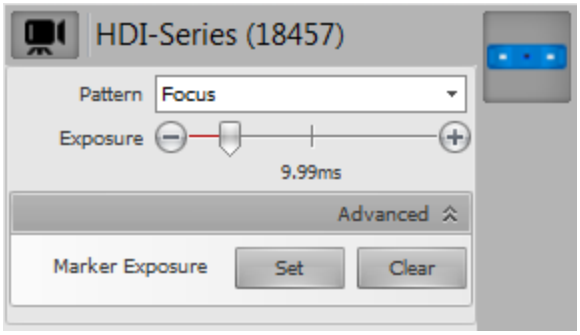
Tape measure example. The camera is focused on the '2F' mark on the tape. At f/2, the depth of field is so narrow the focus falls off almost immediately above and below the focus point. As the f-stop increases, so does the depth of field range.



Marker head example. The camera is focused on the tip of the nose. Note the calibration board in the background: It is blurry until the aperture is brought down to f/8. Also compare the ear between apertures: Blurry at f/2, slightly soft at f/4, and sharply in focus at f/8. (Look carefully at the marker sticker on the ear).

Marker Exposure

A separate exposure duration can be set to optimize the detection of markers, which is especially useful when there is a high contrast between the marker and the target's surface. Once marker exposure has been set, for each scan an additional image capture will be made using the marker exposure to help locate markers on the scan.



To set marker exposure:

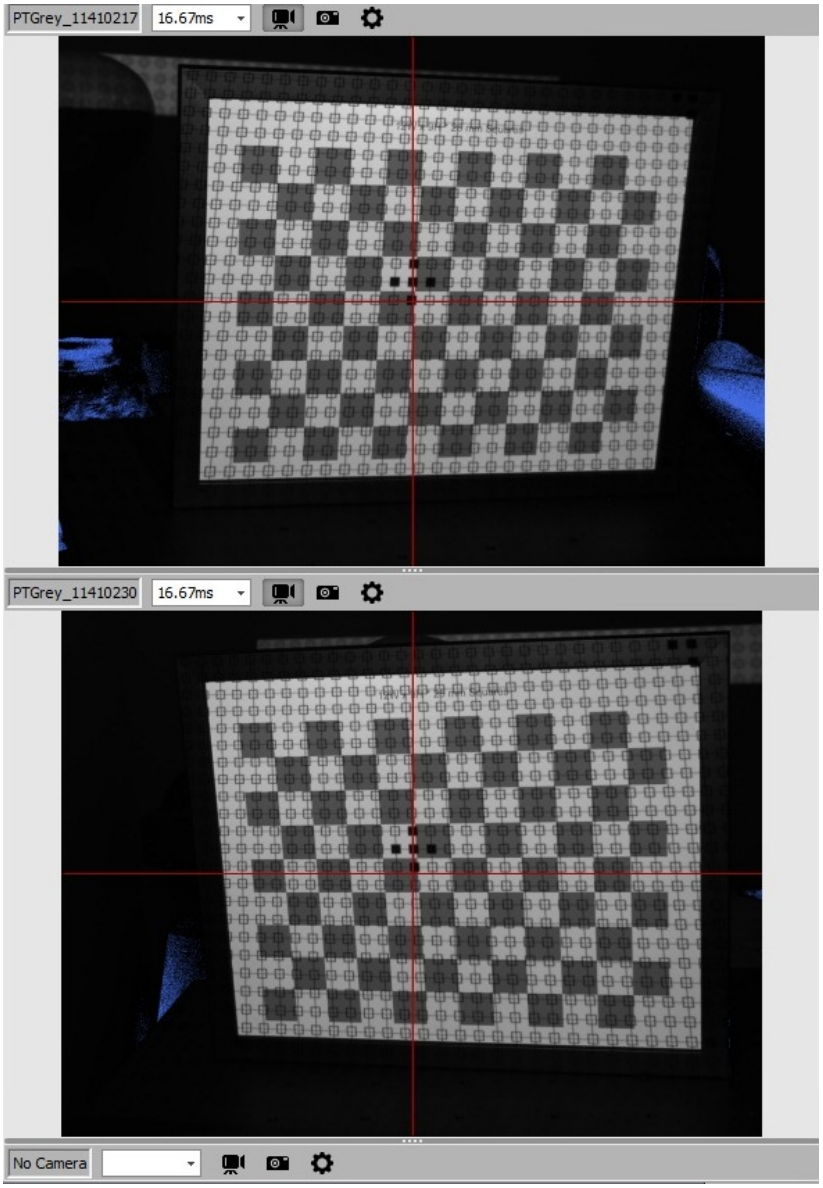
1. Ensure that the live video feed is on by clicking on Toggle Video  .
2. Set the exposure using the slider.
   Follow the guidelines described in Fine-Tuning Exposure above, but choose an exposure optimized for the markers and not the overall image of the target.
3. Click on **Advanced** below the slider.
4. Click on the **Set** button to set the marker exposure.

   A Marker Exposure indicator  will be displayed above the **Scan** button.

To clear marker exposure, click on the **Clear** button.

Setting the Focus

Blurry images produce poor scan data. You should properly focus both the projector and camera(s) to capture high-quality scans.



Position the calibration board so that the crosses (five black dots) are vertically aligned in live view.

Projector

To set projector focus:

1. On the far right of the screen, select **Focusing** in **Pattern**. This projects a pattern of lines and squares onto your target area.

2. Change the projector zoom based on the field of view (FOV) that you are using.
   For a FOV of 200 mm or lower, set the projector zoom to the minimum by rotating the zoom dial completely to the left (when standing behind the scanner). For a FOV of 400 mm and higher, set the projector zoom to the maximum by rotating the zoom dial completely to the right.
3. Place the calibration board in your target scanning area, with the back of the board facing the projector. You can use any object when focusing the projector, but it is better to use the calibration board. By using the calibration board, the focus point is where the standoff should be (where the 5 black dots of the cross are aligned).
4. While looking at the object, rotate the focus ring on the projector until the lines projected onto the sample object are thin and sharp.

Camera

To set camera focus:

1. On the scanner control, select **White** in **Pattern**.
2. Place the calibration board in your target scanning area, with the back of the board facing the camera. You can use any object when focusing the camera, but it is better to use the calibration board. By using the calibration board, the focus point is where the standoff should be (where the 5 black dots of the cross are aligned).
3. Place your cursor over the video feed from the camera.
4. Click on the icon that appears in the top-left corner of the video feed and click **Zoom**.
5. While looking at the zoomed-in view of the object and lines, rotate the focus ring on the camera until the video image is sharply in focus.
6. Repeat for all cameras.

> Do not use a projector pattern to focus the cameras. If the projector pattern itself is not in focus, then it is nearly impossible to properly focus the camera. Instead, use printed text as a focus object, such as the calibration board's text, a magazine cover, a bar code, or anything that has fine detail.

Calibrating the Scanner (Advanced Configuration)

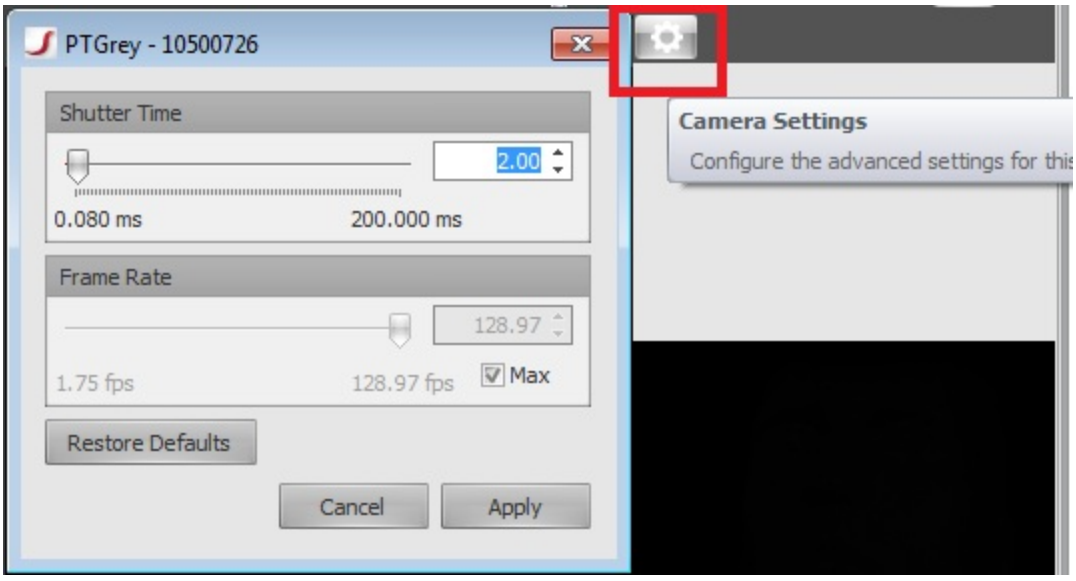
> Because cameras are shipped with proper configuration, we do not recommend that you change these settings unless you know what you are doing.

Point Grey Research (Flea or Grasshopper)

Point Grey camera

To configure settings for a Point Grey camera:

1. On the far right side of the window, click the Camera Settings icon (shown in the red square) to open the **Camera Settings** dialog box.



2. Adjust the settings and click **OK**. Refer to the table below for more information.

Point Grey Settings

Setting	Description
Shutter Time	Sets the camera's exposure time when capturing images. Faster shutter times result in faster scanning, but also result in lower brightness.

Setting	Description
	We strongly recommend that you set the shutter time to a multiple of the projector's refresh rate (usually 60Hz or 16.67ms, depending on the projector model). Otherwise, the captured image may contain uneven exposure, incomplete projector patterns, etc.
Frame Rate	Determines the minimum and maximum shutter time range. Use a lower frame rate if you want longer exposure times. Use a higher frame rate if you want shorter exposure times.
Restore Defaults	Resets the camera back to its pre-configured state.

Canon

To prepare a Canon camera:

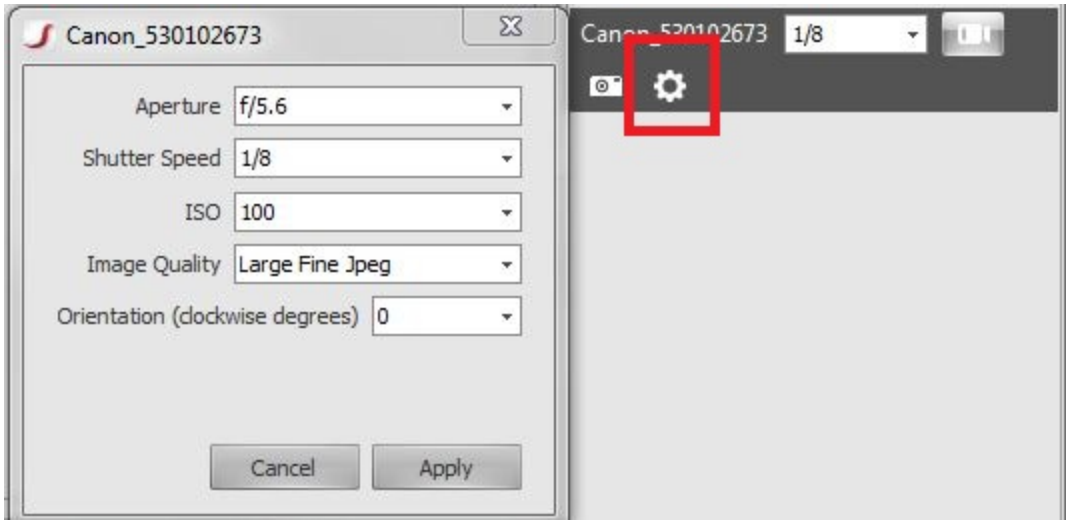
1. Set the camera to Manual Exposure mode by turning the dial on the top of the camera to **M**.
2. Set the lens to Manual Focus mode (be sure to focus first beforehand).
   Most lenses provide a switch to choose manual focus (usually labeled MF) or auto focus (usually labeled AF). Auto focus interferes with the exposure time during scanning. If you need to refocus after calibration, switch AF back on, capture an image (using the Screenshot button), then switch back to MF.

Avoid any physical contact with the camera after calibration. Touching the camera can disturb the calibration. To turn the camera on and off, use an AC adapter instead of the camera's power switch.

Canon camera

To configure settings for a Canon camera:

1. On the far right side of the window, click the Camera Settings icon (shown in the red square) to open the **Camera Settings** dialog box.



2. Adjust the settings and click **OK**. Refer to the table below for more information.

Canon Settings

Setting	Description
Aperture	Controls the size of the shutter inside the lens, which in turn controls how much light will reach the sensor. Although aperture can also be used to control the exposure, its primary purpose is to set the depth of field (range of acceptable focus). The smaller the f-number, the wider the aperture opening will be, and the depth of field will be narrower. Therefore, you must be careful to ensure the entire subject is in focus when using small f-numbers.
Shutter Speed	Controls how long the shutter will stay open to capture an image. This is the main way of controlling exposure. The values used are in seconds or fractions of a second.
ISO	Controls the sensor's sensitivity to light, and is another way of controlling exposure. If you use the flash, ISO also controls the flash range (the higher the ISO is, the farther the flash range will be). Higher ISO settings will also add noise (graininess) to the image, so it is recommended to keep this as low as possible.
Image Quality	Determines whether images are saved out as RAW and/or JPG, and in what resolution and/or quality (amount of lossy compression). FlexScan does not natively support RAW images, so you will need to use other software to process RAW images.
Orientation	Designates the mounting position relative to the projector. For example for portrait mode, you need to specify the orientation so that the resulting image will be auto-rotated to match the scanner for processing. The available options are 0, 90, 180, and 270 degrees clockwise.

Nikon

Due to the nature of the Nikon development kits, live video is not available for Nikon cameras. Use the Screenshot feature when testing Nikon settings.

To prepare a Nikon camera:

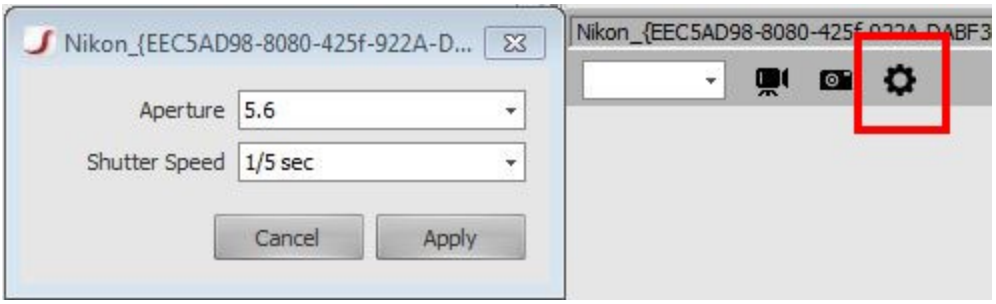
1. Set the camera to Manual Exposure mode by turning the dial on the top of the camera to **M**.
2. Set the lens to Manual Focus mode (be sure to focus first beforehand).
   Most lenses provide a switch to choose manual focus (usually labeled MF) or auto focus (usually labeled AF). Auto focus interferes with the exposure time during scanning. If you need to refocus after calibration, switch AF back on, capture an image (using the Screenshot button), then switch back to MF.

Avoid any physical contact with the camera after calibration. Touching the camera can disturb the calibration. To turn the camera on and off, use an AC adapter instead of the camera's power switch.

Nikon camera

To configure settings for a Nikon camera:

1. On the far right side of the window, click the Camera Settings icon (shown in the red square) to open the **Camera Settings** dialog box.



2. Adjust the settings and click **OK**. Refer to the table below for more information.

Nikon Settings

Setting	Description
Aperture	Controls the size of the shutter inside the lens, which in turn controls how much light will reach the sensor. Although aperture can also be used to control the exposure, its primary purpose is to set the depth of field (range of acceptable focus). The smaller the f-number, the wider the aperture opening will be, and the depth of field will be narrower. Therefore, you must be careful to ensure the entire subject is in focus when using small f-numbers.
Shutter Speed	Controls how long the shutter will stay open to capture an image. This is the main way of controlling exposure. The values used are in seconds or fractions of a second.

Capturing Calibration Images

> It is HIGHLY recommended that you view our step-by step instruction videos (page 37). This is an essential step and must not be skipped. The information gained by viewing the tutorial videos will greatly help you understand this process.

After you complete all the configuration steps, it is time to start capturing calibration images.

> These instructions assume that you still have the calibration open that you configured earlier. If not, click the **Calibration** tab, and click **Open Calibration** to select it.

To capture calibration images:

1. Ensure that the cameras and the projector are securely fastened. This is extremely important because any change in angle or position will ruin the calibration.
2. Ensure that the projector is turned on.
3. Place the calibration board in your target area.
4. Ensure that the projector illuminates the entire grid.
5. Use the live view to make sure that the grid is completely visible to all cameras.
6. On the far left of the screen, click **Capture**.
   FlexScan3D automatically detects the grid and overlays a series of colored lines and dots on top of the image. The resulting image file appears in the Images list.

If you take a bad image, an error message displays to indicate that the calibration image is invalid. You can either delete the image immediately or keep it.

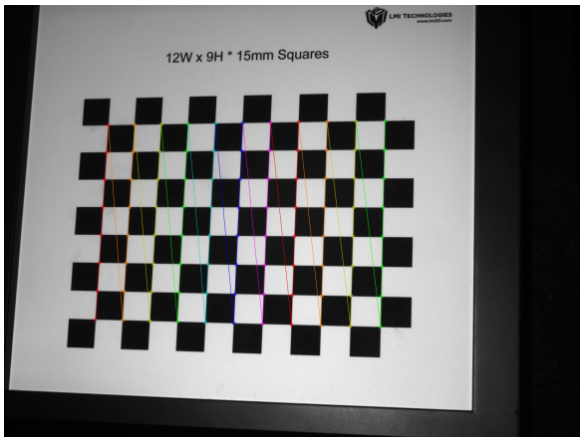
7. Click the image file to view it.
8. Ensure that the lines are straight and that the dots are on the corners of the squares. Look at the examples given below as a reference.
9. Repeat steps 6 though 8 so that you have calibration images of the entire target area, adjusting the location of the calibration board each time.
   You will need to capture images of the calibration board from multiple positions and various angles.
   - Ensure calibration board is in the center of both cameras.
   - Move the calibration board vertically and horizontally to cover the entire field of view.
   - Adjust the angle of the calibration board for variations.
   - Move the calibration board forward and backward for depth variation.

   FlexScan3D allows you to calibrate with as few as five images, but this is usually not enough if you require high accuracy. For high-accuracy scans, you should consider capturing 30 to 40 or more calibration images.

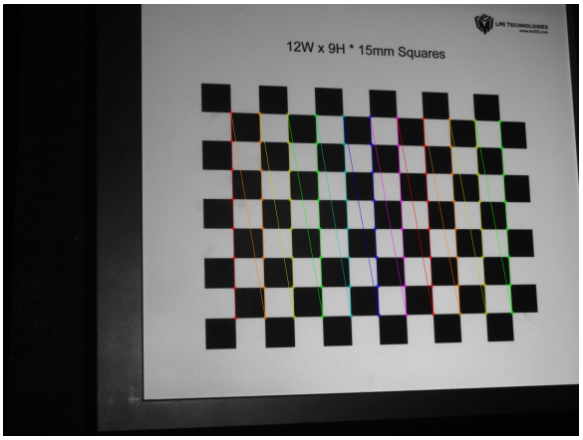
See *Tutorial Videos* (page 184) (Single and Duo Camera Calibrations) for more information.

Examples

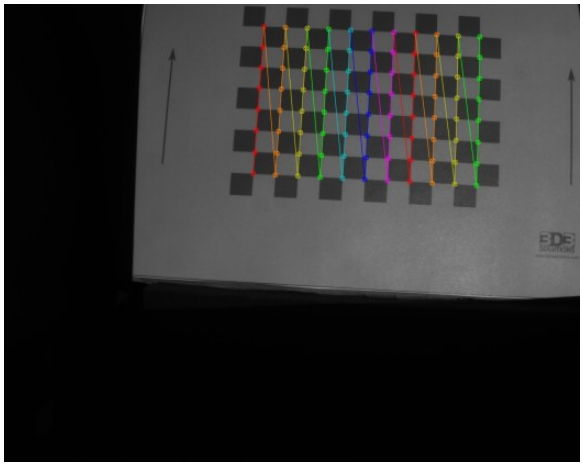
Good Calibration Images



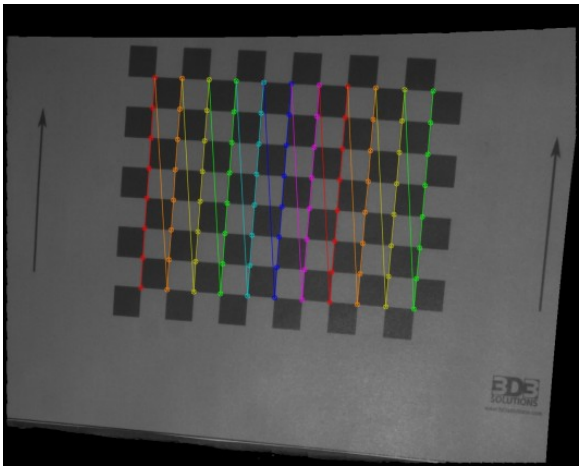
Duo calibration, left camera



Duo calibration, right camera

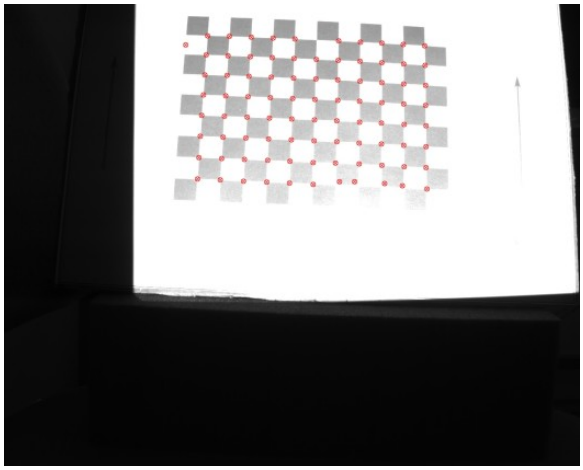


Single calibration, camera view

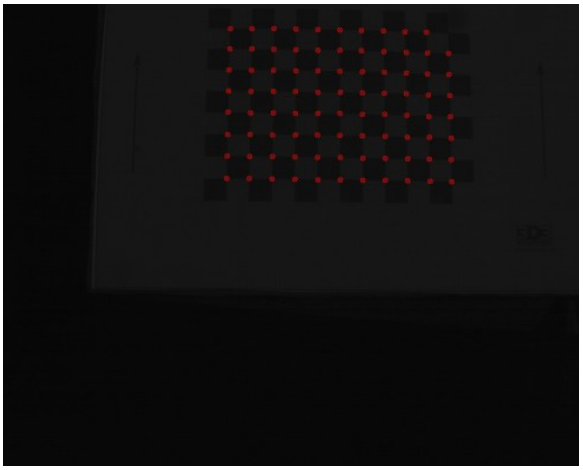


Single calibration, projector view

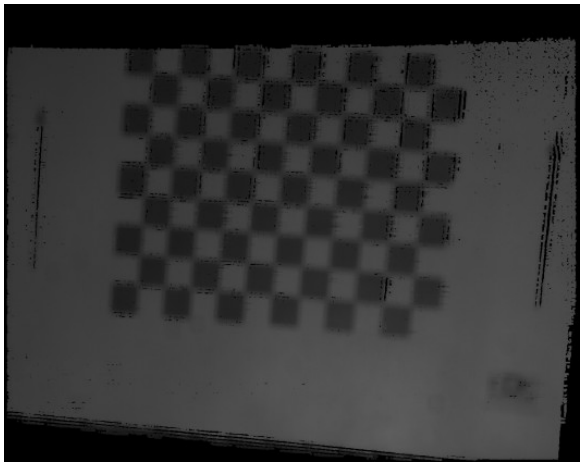
Bad Calibration Images

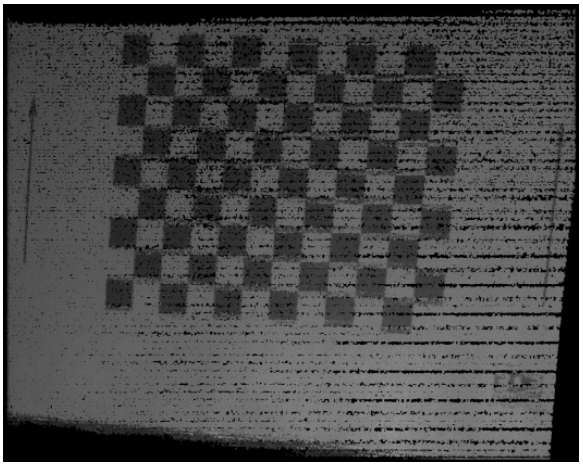


Overexposed

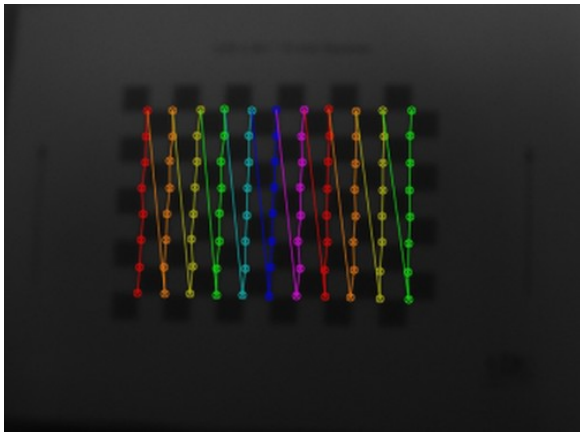


Underexposed



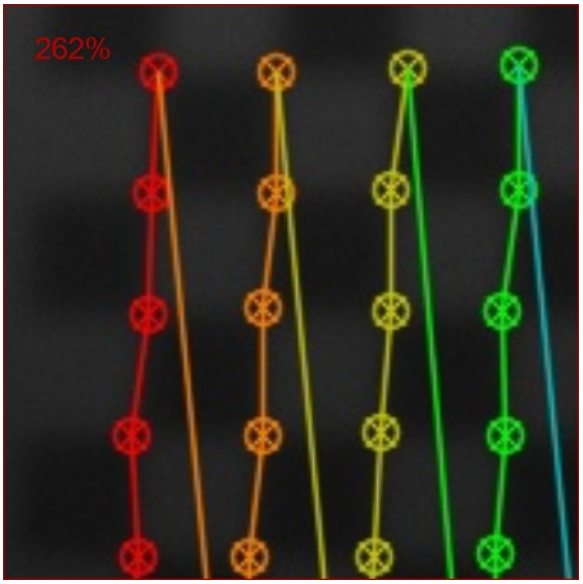


Single calibration, projector view, bad camera focus

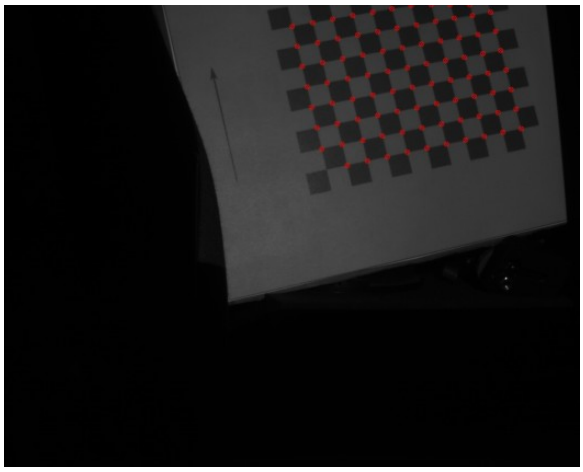


Crooked grid lines due to bad focus

Single calibration, projector view, bad projector focus



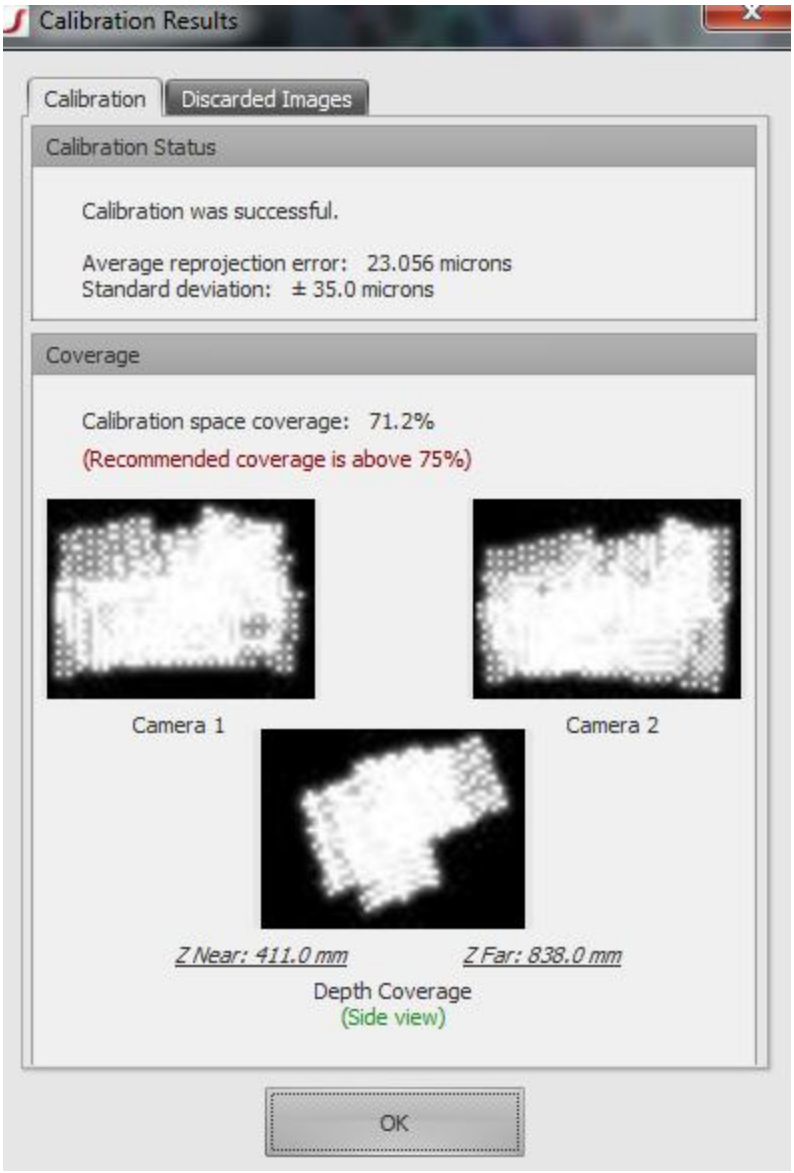
Crooked grid lines due to bad focus, zoomed in



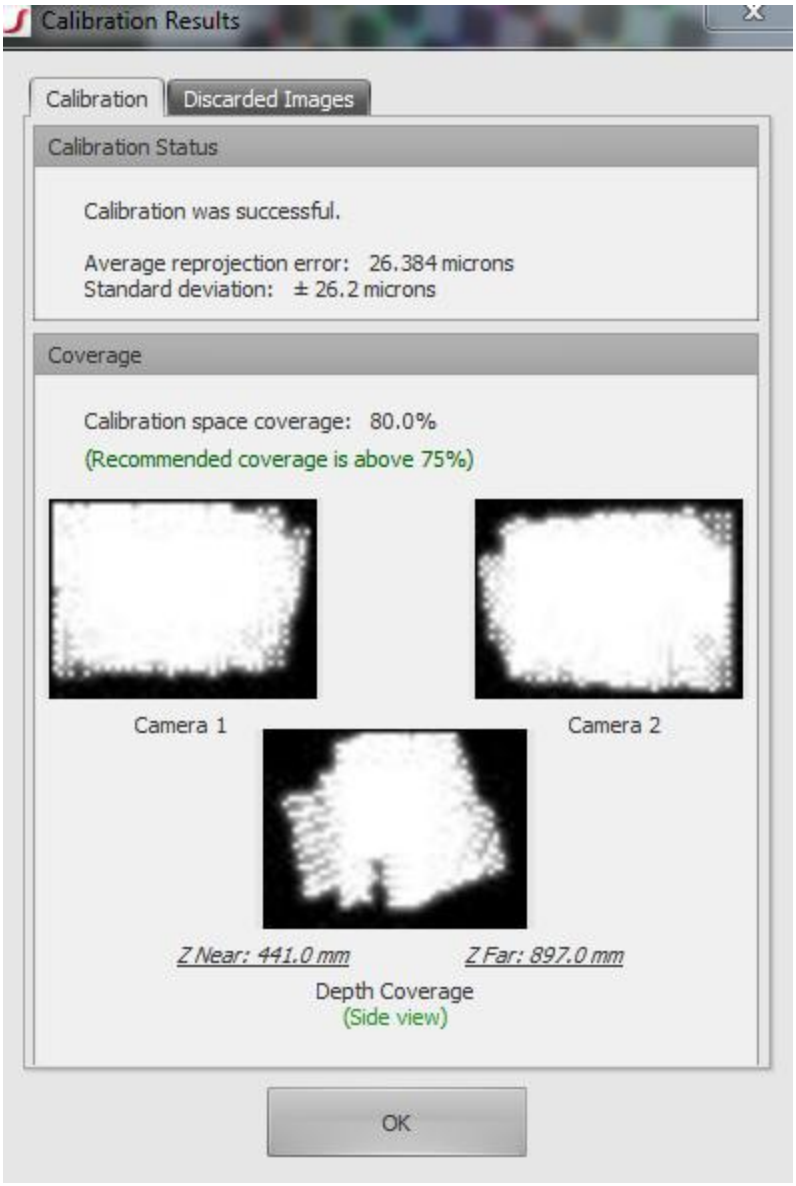
Missing grid squares

Finishing Calibrating

After you capture all the calibration images, on the far left side of the screen, click **Calibrate**. FlexScan3D uses the calibration images to define the scanner calibration. After it completes, a message displays that indicates the accuracy of the calibration and your coverage.



Calibration results after 15 scans



Calibration results after 40 scans

If you are not happy with your accuracy or coverage you can simply press OK and continue to capture images. Otherwise, go to the **Project** tab.

Your scanner should now be calibrated and ready for scanning.

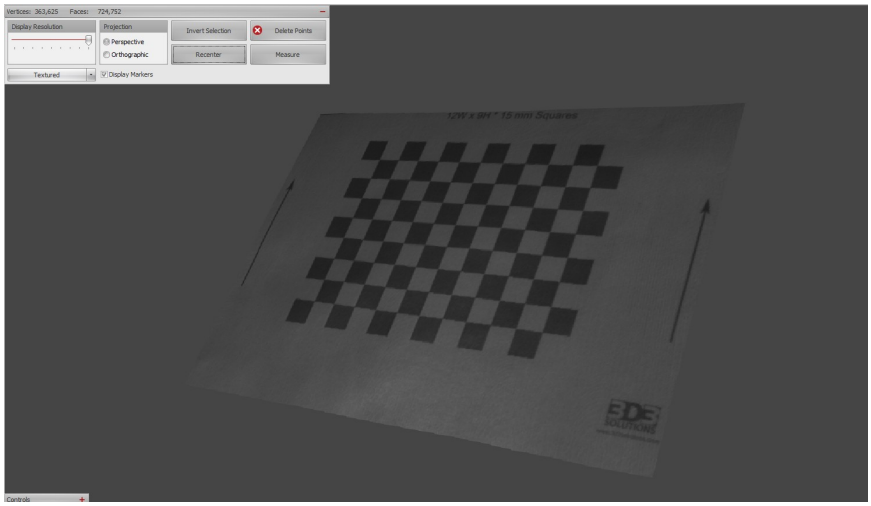
Recommended Next Step: Confirming the Calibration (page 57)

Confirming the Calibration

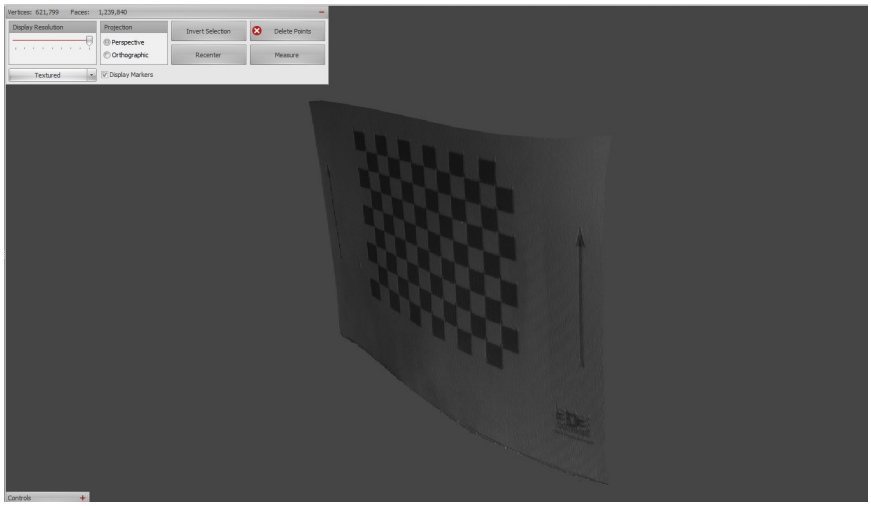
To get a clearer definition of the accuracy of your calibration, you must test actual scans.

To test actual scans:

1. Click the **Project** tab for testing.
   Refer to the section on capturing scan data for more information about creating a project, scanning, and processing scan data.
2. Scan the calibration board in three or four different positions and angles.
   Scanning the calibration board provides a measurable way to verify your calibration accuracy.
3. Inspect the 3D geometry produced by each scan individually.
   Is the geometry flat? Or does the board appear to curve? If the geometry is not flat, go through the check list below.



Good shape, flat



Bad shape, curved

If it looks like the calibration is bad, click on the **Scanners** tab and check the following:

- Is the projector set to its native resolution?
- Do the calibration images appear focused and properly exposed?
- Are the colored grid lines straight? Use the zoom view to confirm that none of your calibration images have crooked grid lines.
- Is the calibration board positioned to cover the entire scanning area? Make sure to include near/far, top/bottom, left/right, in key combinations at various angles.
- Did you take enough calibration images?

If everything looks good, create a new project, and start scanning!

Setting Up a Rotary Table

Using a rotary table streamlines the scanning process. Only minimal user interaction is required to achieve a 360° scan of a static object.

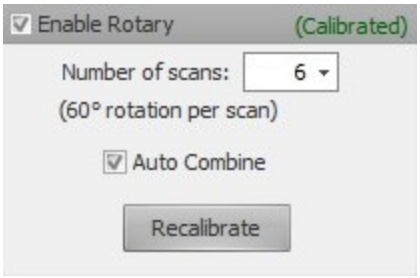
Setting Up the Hardware

To set up the hardware:

1. Ensure FlexScan3D has been installed, including the rotary driver component (installed by default).
2. Connect the cable from the table motor to one of the available ports on the controller box.
3. Plug in the power cable for the controller box.
4. Connect the USB cable between the controller box and the PC.

Using the Software

Before the rotary table can be used in FlexScan3D, it must be calibrated.



Calibrating the Rotary Table

Once the rotary is calibrated, FlexScan3D will automatically align meshes based on the rotary position. To calibrate the rotary, you will first need to make sure you already have a scanner connected and open. For HDI Advance, the scanner must also be calibrated.

To calibrate the rotary table:

1. Create a new project or open an existing one.
2. On the **Project** tab, click the **Scan** button (top-left of the application window) to switch to Scan mode. Note that a valid calibration must be open and the scanner connected. Otherwise, this button will be disabled.
3. Click the **Enable Rotary** checkbox.
   The software will attempt to connect to the rotary. If it succeeds, you will immediately be prompted to calibrate. If it does not succeed, please make sure all cables are connected properly to the controller box and PC, and also ensure that the power cord is plugged in.
4. Place a calibration board on the rotary, ensuring that it can be seen by the camera(s) and that it is not overexposed or underexposed.
   For HDI Advance scanners, the calibration board must be the exact same size as the one used during calibration of the scanner. Otherwise, alignment will not work properly.
   For HDI 120 scanners, use a 10 mm calibration board. For HDI 109 scanners, use a 5 mm calibration board.
5. Click **OK** to start this calibration process.
   The rotary will rotate as needed and capture images of the calibration board at various angles. A message will be displayed informing you whether or not the calibration was successful.

> If you find that scans are not aligning properly (for example, if the wrong board was used), you can click **Recalibrate** to run the calibration process again.

360° Scanning

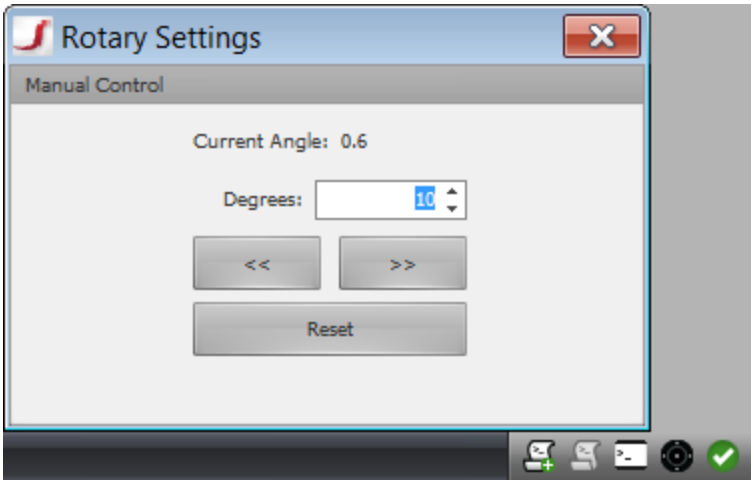
The most common use for a rotary is to achieve a complete 360° scan of an object. If the rotary is not yet calibrated, first follow the steps for calibrating the rotary.

To do a 360° scan of an object:

1. If the **Enable Rotary** checkbox is not checked, check it now.
2. Choose the number of desired scans.
   More scans will result in better coverage, but setting this value too high may result in a lot of excess data, which will slow down scanning and mesh operations. We recommend doing 6 to 12 scans.
3. Choose whether or not to automatically combine the scans by checking/unchecking the **Auto Combine** checkbox.
   Automatically combining the scans will simplify subsequent operations on the mesh, but may not be desired in all cases. (For example, if the scans are intended to be exported for alignment in third-party mesh processing software.)
4. Make sure the object is on the rotary table facing the scanner, and then click **Scan**.
   Scanning, processing, and alignment may take several minutes to complete.
5. Often, you will also need to place the object on its side and run another scan to capture the bottom and top of the object. Align to the other scans as needed.

Manual Jogging

The rotary can also be used manually. Click the rotary icon to bring up the rotary controls. If the rotary is calibrated, new scans will get aligned based on the rotary position.

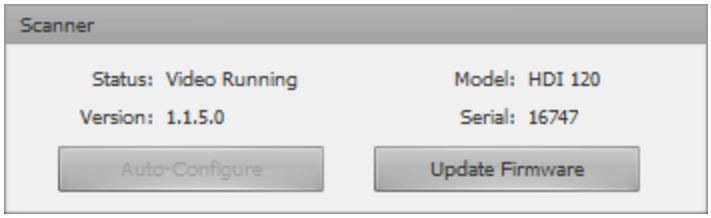


Setting Up: HDI 100 Series Scanners

The Quick Start Guide and Hardware Manual for HDI 100 series scanners can be downloaded from the HDI 100 Series download area.

HDI 100 Series Scanner Configuration

After adding an HDI 100 series scanner, basic sensor information and the network connection details will be displayed.



Updating the Firmware

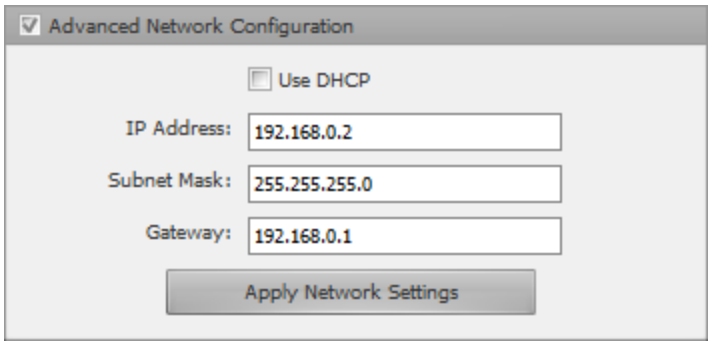
The FlexScan3D package includes the latest firmware for the sensor. If the sensor firmware is incompatible, an **UPDATE REQUIRED** message will be displayed in red in the **Scanner** panel. Click the **Update Firmware...** button to update the sensor. The update will take a couple of minutes to complete.

Network Configuration

If the scanner is connected and the network is configured properly, the networking panel will contain the current network information for the scanner.

If there is an issue with the network connection, a **CONFIGURATION REQUIRED** message will be displayed in red. Click the **Auto-Configure** button to attempt to configure the network automatically. Note that if the scanner is connected to an Ethernet jack on the computer, the Ethernet adapter may be modified to use a static IP address.

Network settings can also be set up manually. Check the **Advanced Network Configuration** option, enter the desired values for the IP address, subnet mask, and gateway, and then click **Apply Network Settings**.



Adjusting Your Equipment

Scanner Menu Bar

The scanner equipment controls are accessed via the menu bar on the right side of FlexScan3D. Depending on the system, the following are some of the buttons present on the menu bar.

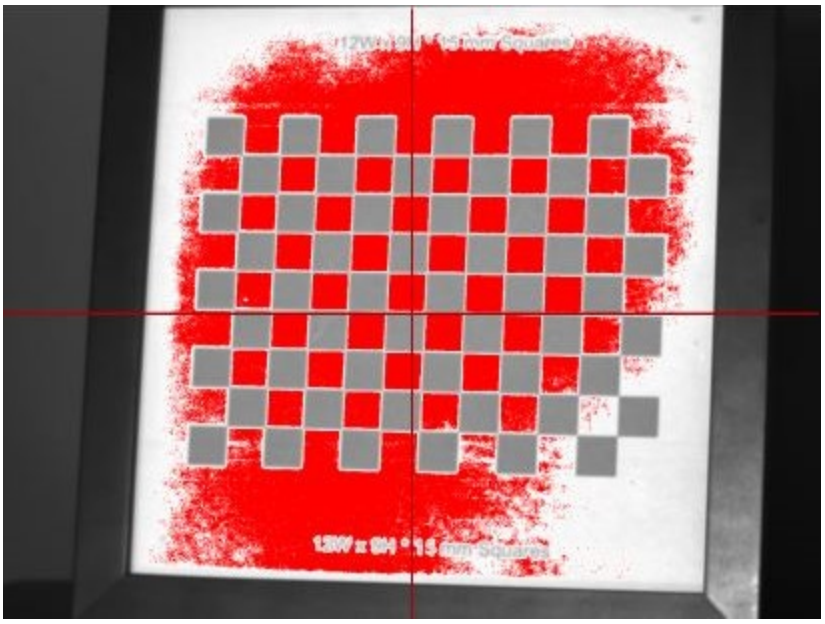
Command Icon	Description
	Show/Hide all scanners (for multi-scanner systems only).
	Show/Hide HDI scanner control.

When using a multi-scanner system, each scanner is represented by its own button on the menu bar.

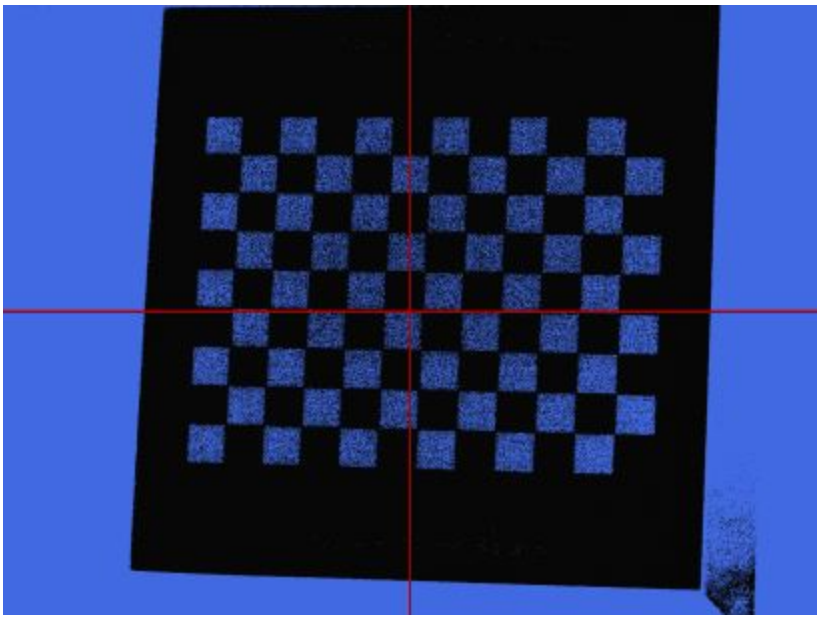
Camera Exposure

The exposure duration of the images captured by the scanner's cameras can be adjusted.

If the camera live video feed is running, it will automatically indicate areas where the image is overexposed (in red) and/or underexposed (in blue).



Over-exposed image in live feed



Under-exposed image in live feed

To ensure your scanner hardware is optimally configured, position an object in front of the scanner (in the target scanning area). Note that the use of a calibration board is not a requirement for setting the exposure; it is used here as an example object since it consists of easily distinguishable light and dark areas.

To set exposure:

1. Open the scanner control on the right side of the application.
2. Set the projector pattern to **Focusing**.
3. Slide the projector brightness slider all the way to the right.
4. Set the shutter speed to the fastest time possible (first item in the drop-down list).
5. Turn on live video if it is not already on. If live video is not available for your camera model, you will need to use screenshots to test each camera setting change.

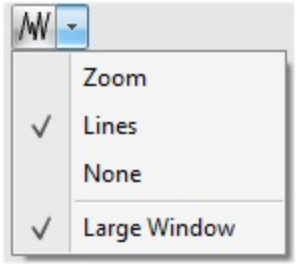
Now adjust the camera lens aperture so that the object in the target scanning area is clearly visible with no exposure highlights.

Fine-Tuning Exposure

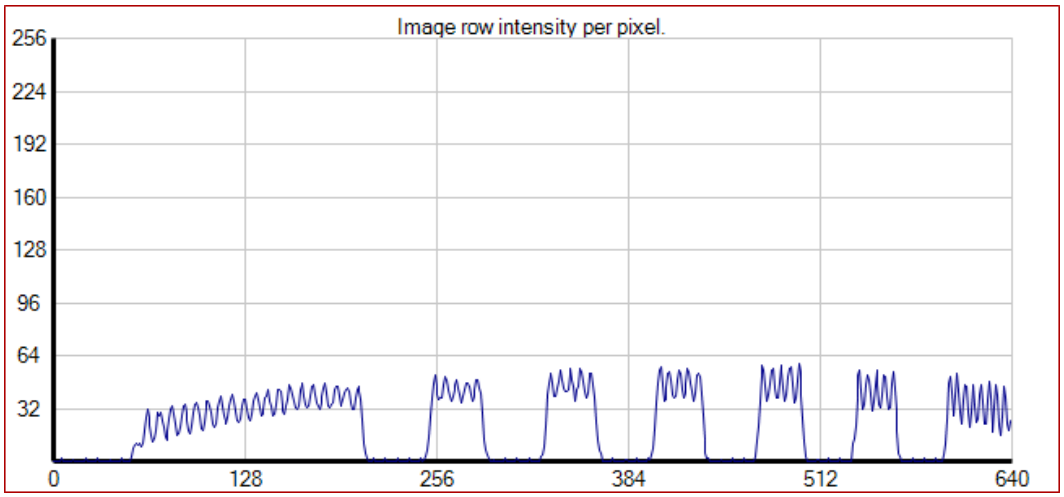
Fine-tuning exposure will produce maximum scan quality.

To fine-tune exposure:

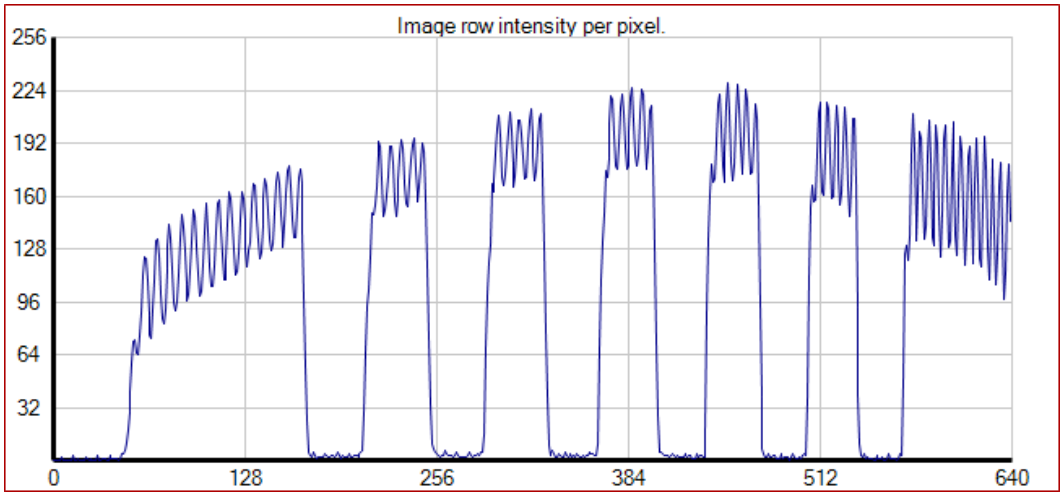
1. Ensure that the live video feed is on by clicking on **Toggle Video** 
2. Choose **Phase** in the **Pattern** drop-down list.
3. Place your cursor over the video feed from the camera.
4. Click on the icon that appears in the top-left corner of the video feed and click **Lines**.



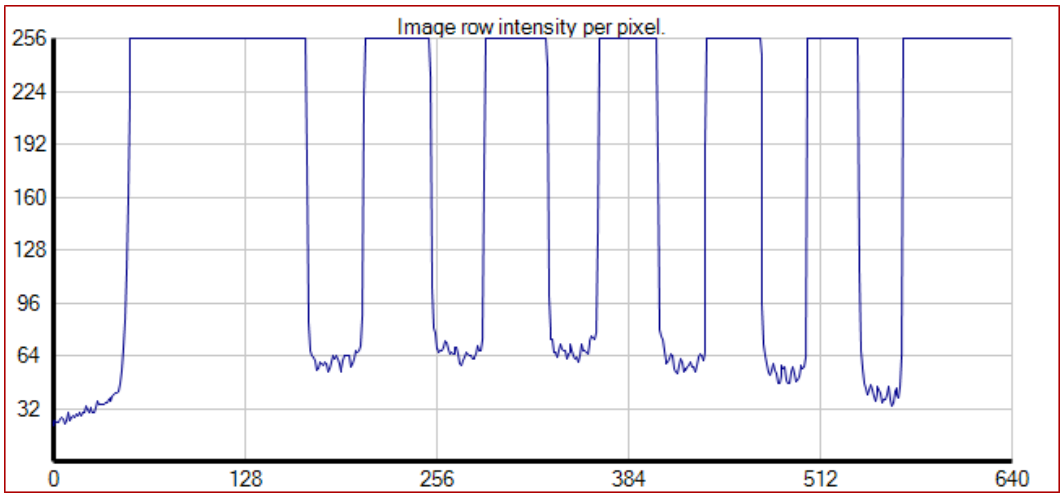
5. Hover the mouse pointer over the video feed to view the brightness level of the row of pixels under the mouse pointer.
6. Adjust the exposure duration using the **Exposure** slider.
   The line should not touch the top of the line view window (overexposed), but should not be too low either (underexposed). The goal is to have good contrast between light and dark areas. Peaks should generally be close to an intensity level of 192. When **HDR** is checked under **Scan Mode**, the slider can be used to set upper and lower exposure limits.



Under-exposed



Optimally exposed

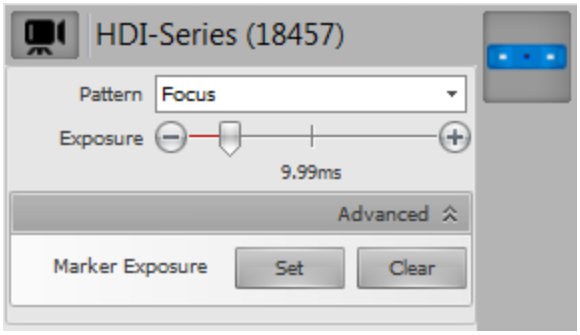


Over-exposed

If multiple scanners are connected, the current exposure can be applied to all of the scanners by right-clicking on the **Exposure** slider and choosing **Apply to all scanners**.

Marker Exposure

A separate exposure duration can be set to optimize the detection of markers, which is especially useful when there is a high contrast between the marker and the target's surface. Once marker exposure has been set, for each scan an additional image capture will be made using the marker exposure to help locate markers on the scan.



To set marker exposure:

1. Ensure that the live video feed is on by clicking on Toggle Video  .
2. Set the exposure using the slider.
   Follow the guidelines described in Fine-Tuning Exposure above, but choose an exposure optimized for the markers and not the overall image of the target.
3. Click on **Advanced** below the slider.
4. Click on the **Set** button to set the marker exposure.
   A Marker Exposure indicator  will be displayed above the **Scan** button.

To clear marker exposure, click on the **Clear** button.

Setting Up a Rotary Table

Using a rotary table streamlines the scanning process. Only minimal user interaction is required to achieve a 360° scan of a static object.

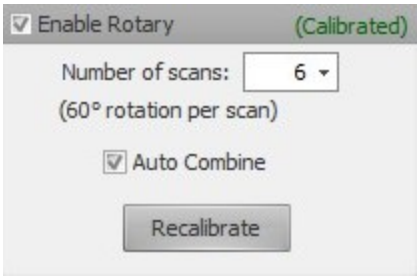
Setting Up the Hardware

To set up the hardware:

1. Ensure FlexScan3D has been installed, including the rotary driver component (installed by default).
2. Connect the cable from the table motor to one of the available ports on the controller box.
3. Plug in the power cable for the controller box.
4. Connect the USB cable between the controller box and the PC.

Using the Software

Before the rotary table can be used in FlexScan3D, it must be calibrated.



Calibrating the Rotary Table

Once the rotary is calibrated, FlexScan3D will automatically align meshes based on the rotary position. To calibrate the rotary, you will first need to make sure you already have a scanner connected and open. For HDI Advance, the scanner must also be calibrated.

To calibrate the rotary table:

1. Create a new project or open an existing one.
2. On the **Project** tab, click the **Scan** button (top-left of the application window) to switch to Scan mode. Note that a valid calibration must be open and the scanner connected. Otherwise, this button will be disabled.
3. Click the **Enable Rotary** checkbox.
   The software will attempt to connect to the rotary. If it succeeds, you will immediately be prompted to calibrate. If it does not succeed, please make sure all cables are connected properly to the controller box and PC, and also ensure that the power cord is plugged in.
4. Place a calibration board on the rotary, ensuring that it can be seen by the camera(s) and that it is not overexposed or underexposed.
   For HDI Advance scanners, the calibration board must be the exact same size as the one used during calibration of the scanner. Otherwise, alignment will not work properly.
   For HDI 120 scanners, use a 10 mm calibration board. For HDI 109 scanners, use a 5 mm calibration board.
5. Click **OK** to start this calibration process.
   The rotary will rotate as needed and capture images of the calibration board at various angles. A message will be displayed informing you whether or not the calibration was successful.

> If you find that scans are not aligning properly (for example, if the wrong board was used), you can click **Recalibrate** to run the calibration process again.

360° Scanning

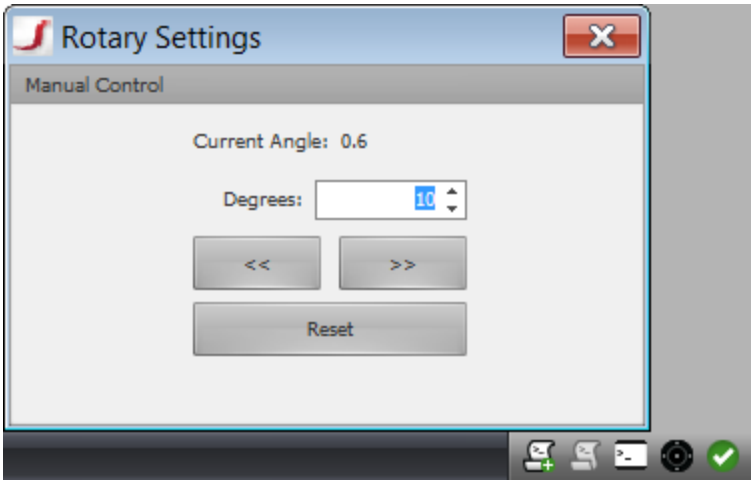
The most common use for a rotary is to achieve a complete 360° scan of an object. If the rotary is not yet calibrated, first follow the steps for calibrating the rotary.

To do a 360° scan of an object:

1. If the **Enable Rotary** checkbox is not checked, check it now.
2. Choose the number of desired scans.
   More scans will result in better coverage, but setting this value too high may result in a lot of excess data, which will slow down scanning and mesh operations. We recommend doing 6 to 12 scans.
3. Choose whether or not to automatically combine the scans by checking/unchecking the **Auto Combine** checkbox.
   Automatically combining the scans will simplify subsequent operations on the mesh, but may not be desired in all cases. (For example, if the scans are intended to be exported for alignment in third-party mesh processing software.)
4. Make sure the object is on the rotary table facing the scanner, and then click **Scan**.
   Scanning, processing, and alignment may take several minutes to complete.
5. Often, you will also need to place the object on its side and run another scan to capture the bottom and top of the object. Align to the other scans as needed.

Manual Jogging

The rotary can also be used manually. Click the rotary icon to bring up the rotary controls. If the rotary is calibrated, new scans will get aligned based on the rotary position.



Capturing Scan Data

The following sections describe how to capture st.

3D Scanning Basics

Scan Quality

Scan quality is influenced by the following factors:

Scanner calibration
The more calibration scans you take, the better the accuracy of the scanner.

Environmental conditions
Light and vibration can cause inaccuracies. Ensure the system is stable and the exposure is set to the conditions.

Stability of the scanner and the object
If either the scanner or the object move during the scan, the data will be inaccurate. Ensure both are stable and secure.

Shape and size of the object
If the part you want to scan has limited features, you must use markers or spheres to align the data. You also must calibrate the scanner with an appropriate lens and calibration board to suit the item.

Scan Preparation

When you set up the scanner, you should:

Understand how the scan data will be used
This determines the mechanics of the scanning process and which features, if any, should be hard probed.

Know who will use the scanned image and what they will need captured
This eliminates unnecessary work and potential rescanning.

Scanner Positioning

When positioning the scanner to acquire the most accurate data, consider the following items:

Scanner position
Positioning the scanner perpendicular to the surface being scanned optimizes the amount data the scanner collects. When the data is at an angle, the light can fade out as it goes farther back and the data becomes less accurate. Also, when you are determining how to position the scanner and object, place the

scanner so it can capture as much of the object as possible while remaining perpendicular to the object's surface.

Line of sight
The scanner can capture only what is in its line of sight. If you scan a propeller, for example, it will need to be placed so that the scanner can capture the curve of the blades and in between the blades. If the part needs to be held at a particular location, you must decide the best method for holding the part while scanning. Any solution should allow you to fully capture the part in one setup to minimize the amount of repositioning.

Scanner volume and area
Depending on the lens and calibration board used, the scanner will have a defined area that can be captured. A large part may require several scans to capture all sides and then will be stitched together using Flexscan, Leios, or Geomagic. Each scan needs to overlap the previous section to be able to stitch them together or there needs to be common targets on each scan set for alignment. Be sure there are enough features to align the data.

Part Preparation

Before scanning there are two steps for part preparation: placing reference targets and coating the part.

Placing Reference Targets

For a part that has little geometry, place photogrammetric dots on the part in order to use these features during alignment. See *Scanning with Markers* (page 100) for more information

Coating a Part

Reflective surfaces scatter the scanner light and create noise artifacts in the final data. Laser and white-light scanners do better when scanning matte, white surfaces. Products called "developer", such as Magnaflux Spotcheck Developer, which is a powder-based spray, can be used to create a flat white surface for scanning and is available at welding supply stores. Other products you can use include some athlete's foot treatments, spray deodorant with titanium oxide, and tempera paint from a compressor. You can also use white spray paint if it is acceptable to create a more permanent coating.

You must spray the part with a uniform, light coat. The thinner the coating, the better. Adding more than a thin coat of developer can impact a part's thickness when scanning. Developer comes off easily and is prone to fingerprints. Mounting an object to a surface (such as a bolt through the center of a threaded section) can provide a hand-hold for moving the part after spraying. You can also spray a portion of the part (leaving a section as a hand-hold), complete the first scan, clean the part, respray the area used as a hand-hold, and then rescan as needed.

To apply developer to a part (general procedure):

1. Apply targets before spraying.
   Otherwise, the targets will slip off the white part.
2. If necessary, mount the part or decide which section to use as a hand-hold.
3. Hold the spray can of developer about 8 inches from the part and spray a light first coat.

4. Set the object aside to dry for a few minutes.
5. Re-spray any areas that need it.
6. Clean any targets with a cotton swab before scanning if necessary.

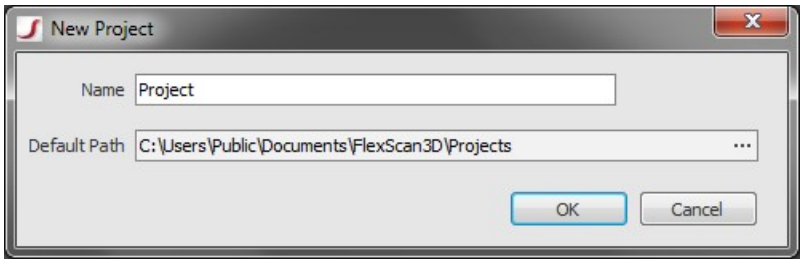
Setting Up Your Scan Project

After you connect and calibrate your scanner, you can create or open a project, set up the project, and then begin capturing data.

Creating a New Project

To create a new project:

1. Click the **Project** tab.
2. Click **New Project** to open the **New Project** dialog box.

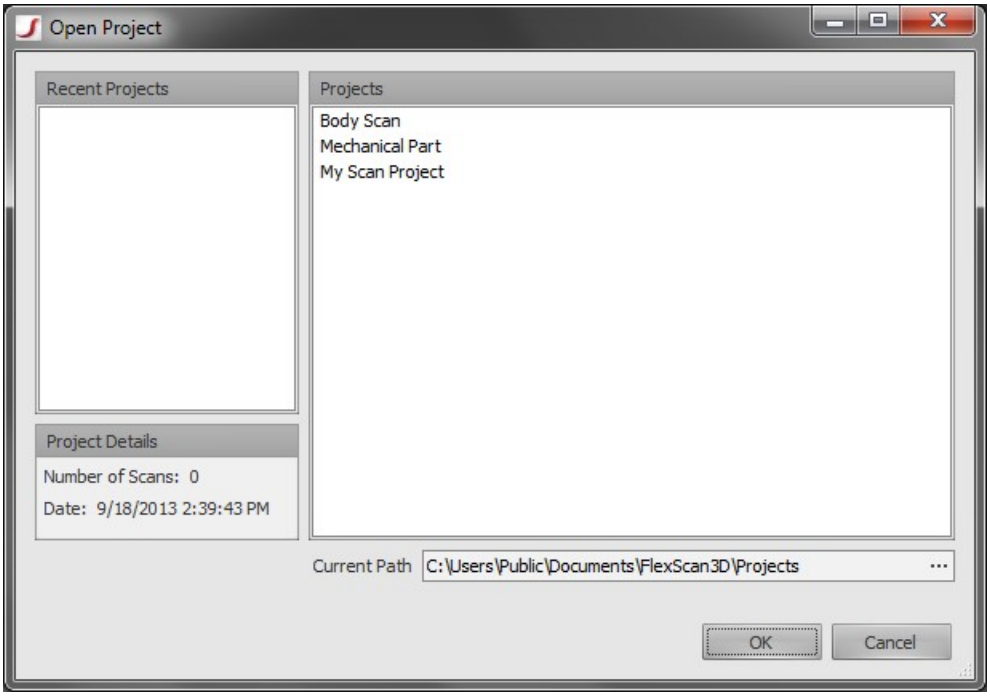


3. Type a descriptive name in **Name**.
4. Click **OK**.

Opening an Existing Project

To open an existing project:

1. Click the **Project** tab.
2. Click **Open Project** to open the **Open Project** dialog box.

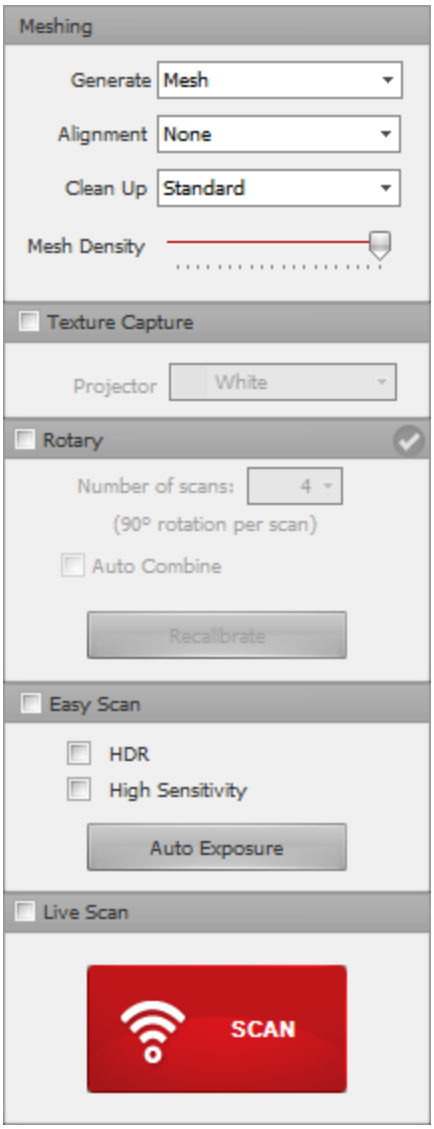


3. Select the project you want to open in the **Projects** list.
4. Click **OK**.

Project Settings

You use the various panels when in **Scan Mode** to adjust the scan settings. See *Processing Scan Data* (page 78) for more details.

To display these panels, click on **Scan** in the **Mode** group.



The following table describes the settings in these panels:

Setting	Description
Meshing - Generate	Selects whether you want to process the data as a **Mesh**, to process the data as **Points**, or in the case of scanning, to capture images only and postpone the processing step until later (**Capture Only** option).
Meshing - Alignment	Selects the type of alignment for the data. The options are: **None**: Use this option if you want to align manually. **Mesh Geometry**: The geometry of the meshes will be used to automatically align the meshes. You may still need to align meshes manually if FlexScan3D is not able to align the meshes automatically. **Marker**: Use this option if you are using markers on your target.

Setting	Description
Meshing - Clean Up	Selects the type of clean-up to use on the data. The options are: **None**: Not recommended. **Relaxed**: Recommended for scanning hair. **Standard**: Recommended for most scanning jobs. **High**: Recommended for a field of view below 200 mm. Useful for scanning mechanical components and other detailed objects. **Extreme**: Recommended for a small field of view, that is, under 75 mm. Useful for scanning coins and other finely detailed objects.
Meshing - Mesh Density	Settings less than 100% will run a decimation pass to reduce the number of vertices in the final scan.
Texture Capture	Captures a separate texture image using the specified projector brightness (HDI Advance only).
Rotary	Activates the rotary table and enables it for calibration. A green checkmark icon indicates that the rotary table is calibrated. A red exclamation point icon indicates that the rotary table is not calibrated.
Easy Scan	Determines the optimum exposure to use for scans and whether HDR and High Sensitivity are needed. When this option is checked, the HDR and High Sensitivity settings are ignored. After performing a scan with Easy Scan, the exposure and HDR settings are updated based on the optimum exposure.
HDR	Scans multiple times at varying exposure levels in order to capture objects with large contrast variances.
High Sensitivity	Special scanning mode which allows for capturing of difficult-to-capture noisy surfaces, such as hair.
Auto Exposure	Determines the optimum exposure to use for scans.
Scan	Causes the connected scanners to scan. The **Scan** button changes to **Easy Scan** when the **Easy Scan** option is checked. If a cut plane, alignment, or marker exposure has been set, a  ,  , or  indicator will be displayed, respectively. Right-click on an icon to display an option to remove the setting.
Live Scan	Causes the connected scanners to monitor the live feed for motion, which triggers an automatic scan (Toggle Video  must be on). An initial scan must be performed by clicking on the Scan button before motion will cause a scan to be triggered. The motion-triggered scan is rejected if no changes from the previous scan are detected. After clicking on Scan, the button changes to Stop. Click on Stop to stop monitoring the live feed.

Checking the System Connection

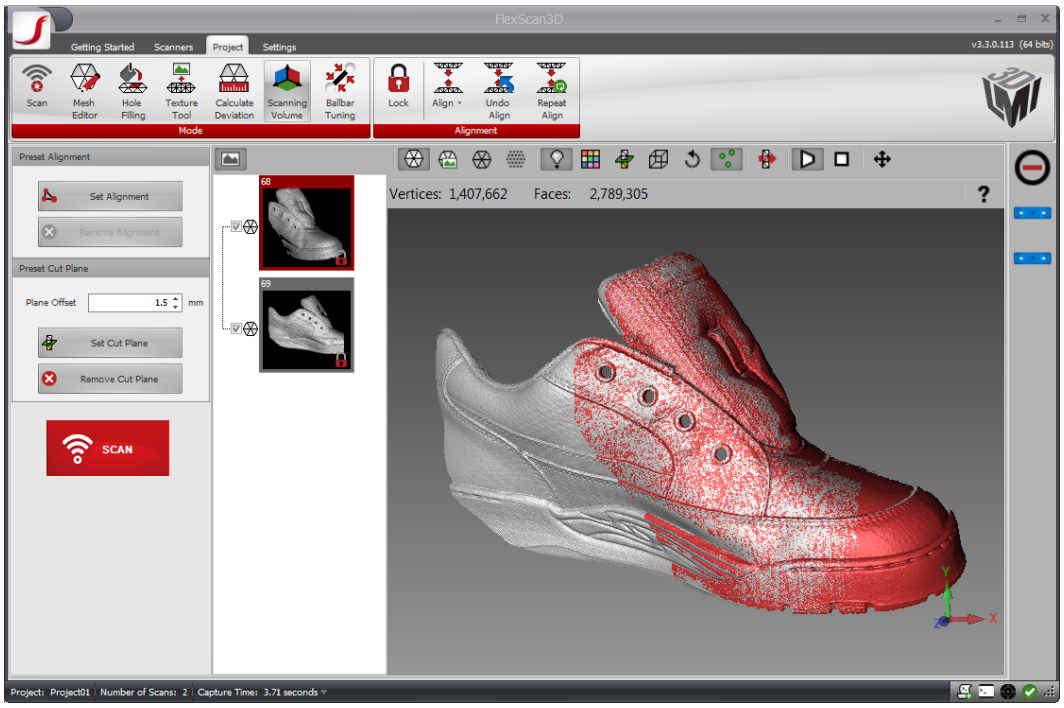
The scanner system must be connected and active to scan using FlexScan3D. When the software is successfully connected to the system, there will be a green light at the bottom-right of the application status bar. If you see a red exclamation mark instead, click it to attempt to connect to the system. If you've set up everything properly, this icon will turn green, indicating that the system is ready to scan.

Setting the Scanning Volume

Before you begin capturing data, you can optionally set the alignment of the scanners if you are using two or more scanners. You can also set a cut plane, which causes data below the plane to be discarded in subsequent scans.

Setting Camera Alignment for Multi-Scanner Setups

When you use two or more scanners with FlexScan3D, you can set their alignment so that future scans are automatically aligned.



To set the alignment:

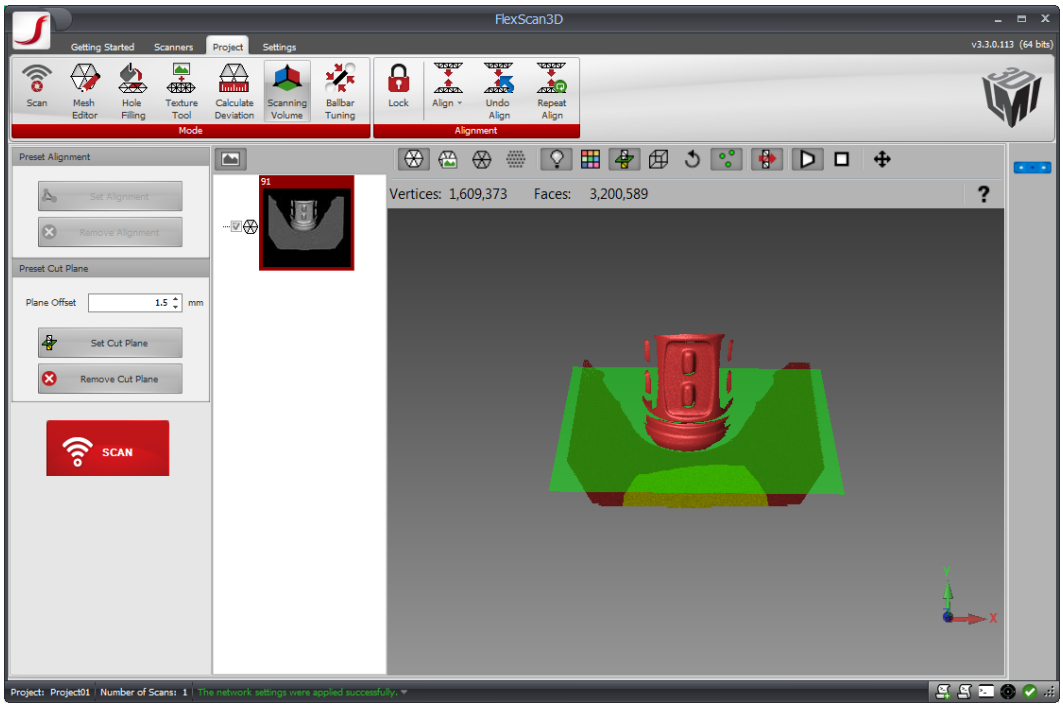
1. In the **Project** tab, click the **Scanning Volume** button.
2. Set up your target.
3. Click on the **Scan** button. A pair of scans, one from each scanner, will be displayed.
4. Align the scans manually.
5. In the **Preset Alignment** panel, click the **Set Alignment** button.

To remove the alignment, click on **Remove Alignment**.

Setting a Cut Plane

A cut plane is typically used to exclude the surface under your target from scans. This is especially useful when scanning a target whose overall color resembles the surface the target is on. You can also set a cut plane to exclude part of a target below the plane. For HDI 100 series scanners, cut planes are stored based on scanner ID.

Once a cut plane has been set, it is applied to subsequent scans.



To set the cut plane:

1. In the **Project** tab, click the **Scanning Volume** button.
2. Click the **Scan** button.
   If you are trying to exclude the scanning surface and you find that it's difficult to get a good scan of the surface, remove the target from the surface and scan again.
3. In the resulting scan, select the part of the surface, or the part of the target itself, that you want to exclude.
   The selected area turns yellow.
4. In the **Preset Cut Plane** panel, set the **Plane Offset** (optional).
5. Click on the **Set Cut Plane** button.

   The cut plane will appear in green. Make sure that **Display Cut Planes**  is active by clicking on the option's button above the 3D window.

To remove a preset cut plane, click on **Remove Cut Plane.**

Scanning

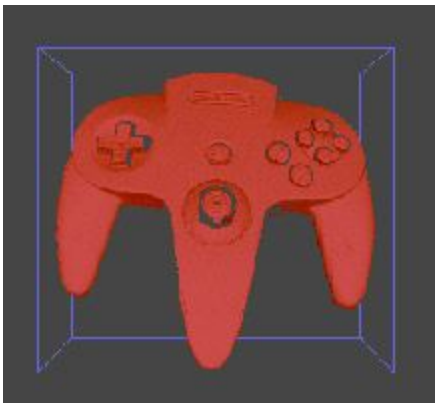
After you connect your system and have the correct project settings (and, for HDI Advance systems, have calibrated the system), you can begin scanning.

To scan:

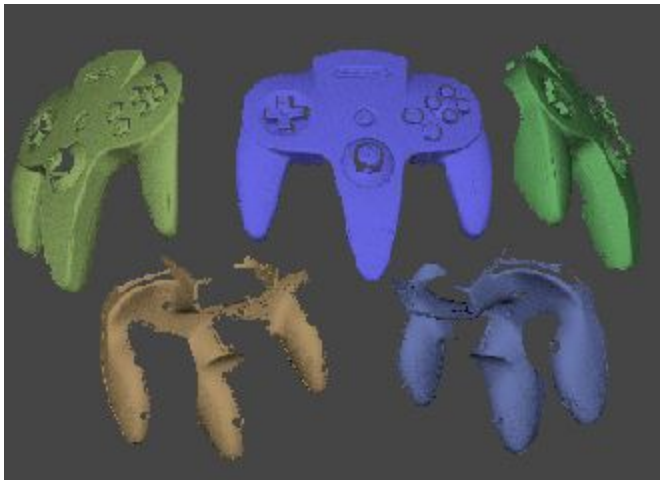
1. Click the **Scan** button.
   After you acquire the scan data, a thumbnail image of the scan appears in the list of scans. If you selected **Mesh** in the **Generate** option, the corresponding mesh loads automatically.

	Avoid disturbing the scanner. Even small vibrations can affect the data quality.

   A single scan:

2. Take multiple scans of the object in various positions in order to capture all of the geometry.

Processing Scan Data

After you have scanned your target, you can process the scan data by choosing **Mesh Editor** in the **Project** tab, **Mode** group.

If you selected **Mesh** or **Points** from the **Generate** option in **Meshing** (when in Scan mode), captured image data is processed automatically after each scan.

When **Capture Only** is selected, you will need to process the scan later. Click **Build** to manually generate a mesh for a selected scan.

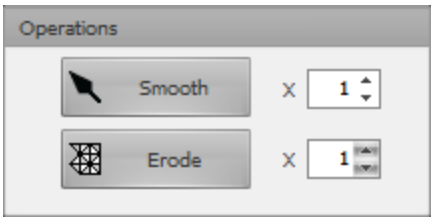
> Processing a previously processed scan overwrites its mesh data.

Operations

Operations only apply to selected meshes. The selected meshes must be loaded for these operations to be available.

These operations do not support point clouds and combined scans.

You can "undo" these operations by selecting the changed meshes and clicking on the **Build** button in the **Meshing** panel.



Smooth

The Smooth operation reduces or removes noise in selected meshes by averaging the angles between the individual polygons. Several passes can be done with one button click by adjusting the **Smoothing Passes** value next to the button.

Erode

The Erode operation removes one polygon strip from all edges in selected meshes. This is useful when the meshes have rough edges. Several passes can be done with one button click by adjusting the **Erosion Passes** value next to the button.

The operation causes all existing holes to grow larger.

Decimation

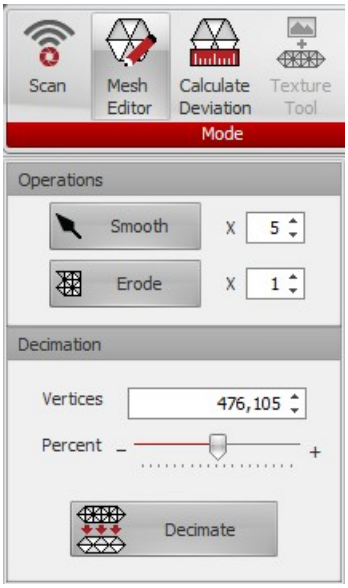
Decimation intelligently discards vertices from the currently loaded mesh, often without any noticeable change in mesh quality. A decimated mesh requires less memory and is faster to work with than a full-resolution mesh.

You can "undo" decimation by selecting the changed meshes and clicking on the **Build** button in the **Meshing** panel.

> Decimating a scan will remove the color texture from the scan.

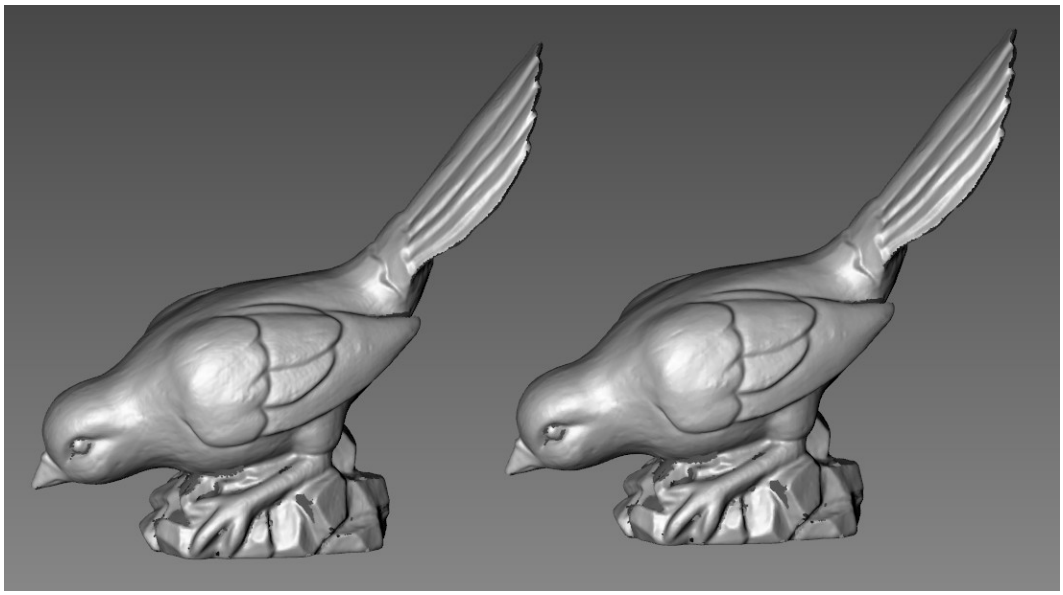
To apply decimation:

1. In the **Project** tab, click the **Mesh Editor** button.



2. Load the mesh you want to decimate, then make sure it is selected.
3. Adjust **Vertices** and **Percent** to the target values for your resulting mesh.
4. Click **Decimate**.

Here is an example of a decimated mesh next to the original mesh. The mesh on the left contains over 1.8 million faces. The mesh on the right is much smaller, containing just over 600k faces. As you can see, the meshes look roughly identical, with only very minor blemishes missing in the smaller mesh.



Using 3D Window Display Commands

You can control how data is displayed in the 3D window by using the command buttons along the top of the window.

Command Icon	Description
	Displays the data as a solid object.
	Displays the data with texture information.
	Displays the data as a wireframe.
	Displays the data as a point cloud.
	Enables specular highlighting. This setting can help see the "shape" of the object in the 3D window.
	Applies different colors to each mesh to help distinguish different meshes in the 3D window.
	Displays cut planes if they have been set.
	Displays a 3D box around the scan. This setting can be useful to find stray noise in a scan, as the bounding box will be larger than expected.
	Displays a widget that helps move the scan in the 3D window.
	Displays markers on the scan if markers have been used.
	Toggles between selecting all the way through the model and selecting on the surface of the model.
	Displays meshes with correction for distance from the camera (creates a slight warping at the edges). Looks more natural.
	Displays meshes with no correction for distance from the camera. Useful when comparing two similar objects side by side, but it looks somewhat unnatural.
	Recenters on the selected scan in the 3D window, or recenters on all scans if no scans are selected.
	Displays a list of scan movement commands and their corresponding mouse and keyboard controls.

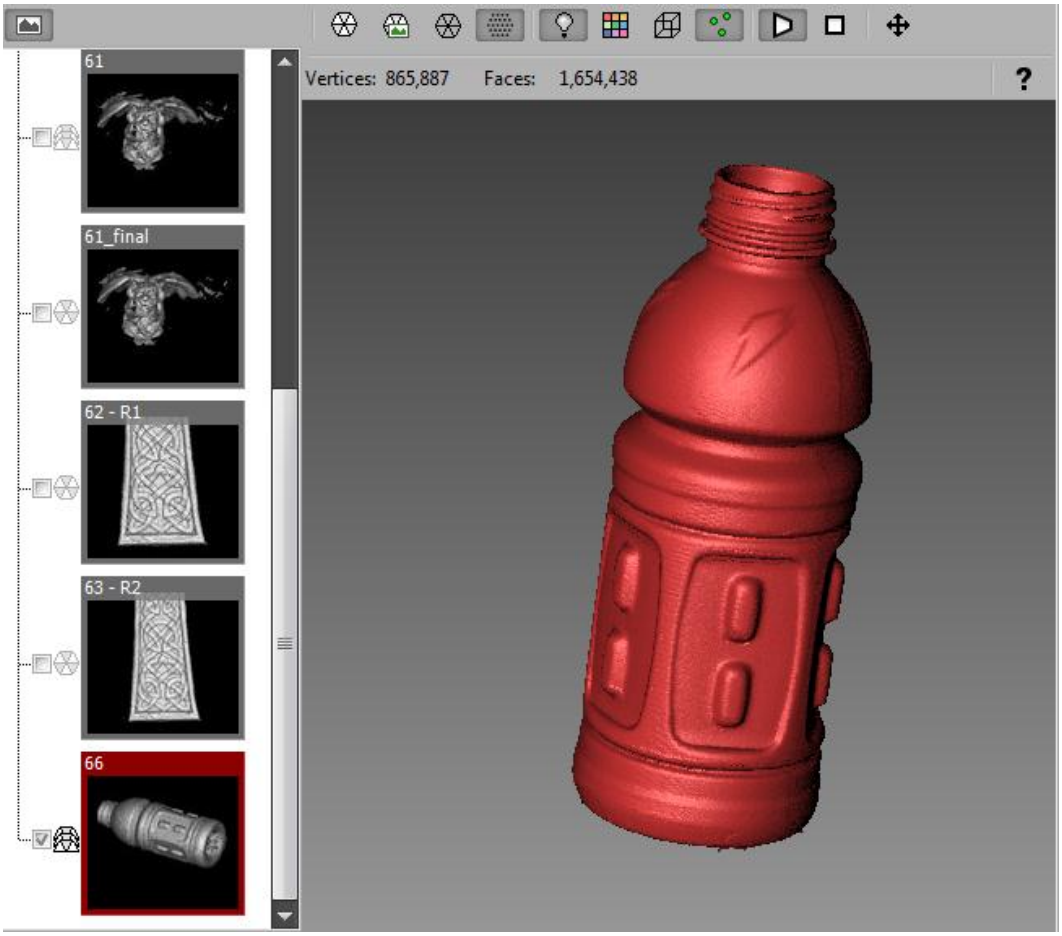
The number of vertices and faces of the data is displayed below the command buttons.

Manipulating and Editing Meshes

You can move and rotate a mesh in 3D using the mouse. The **Controls** menu in the 3D window lists how to use the mouse to manipulate the mesh.

Selection

To select a mesh, click the mesh thumbnail in the list of scans or click anywhere on the mesh geometry of a scan. When selected, a 3D mesh turns red.



Movement

To move everything in the 3D window, hold down both the left and right mouse buttons while moving the mouse.

To move a single mesh, select the mesh and hold down the ALT key and both the left and right mouse buttons while moving the mouse.

Rotation

To rotate everything in the 3D window, hold down only the left mouse button while moving the mouse.

To rotate a single mesh, select the mesh and hold down the ALT key and the left mouse buttons while moving the mouse.

> The rotation is not fixed to any particular axis. It rotates around the center of the mesh. This may take some practice to get used to.

Removing Unwanted Geometry

Sometimes you will see some "extra" geometry in a processed scan, such as sections of a wall behind the target object, the surface the object is on, etc.

To remove unwanted geometry:

1. Click on the checkbox next to a thumbnail to load the mesh that contains unwanted geometry.
2. Hold down the CTRL key and click and drag the mouse to select an area to remove.
3. Release the mouse button to select the area within the outline.
   The selected area turns yellow.
4. Press the DELETE key.
   You can also right-click and choose **Delete**.
5. Repeat steps 1 though 4 for any other unwanted geometry.
6. Select the thumbnails of the scans you modified, right-click on them, and choose **Save**.

Helpful Hints

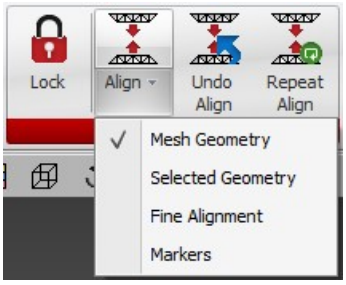
- To revert to the original mesh if you have not yet saved the changes, unload the mesh by clicking on its checkbox and choose **No** in the **Save** dialog.
- To revert to the original mesh if you unloaded a mesh and chose **Yes** in the **Save** dialog, select the mesh, switch to **Mesh Editor** mode, and click on the **Build** button .
- To add to an existing selection, hold down the CTRL key while selecting areas you want to add.
- It may be useful to click **Recenter** after deleting part of a mesh.
- Depending on the mesh, it is sometimes simpler to select the parts of the mesh that you want to keep, right-click in the 3D window, choose **Invert Selection**, right-click again, and then choose **Delete**.

Aligning and Merging Scan Data

Aligning Meshes

When you align meshes in **Mesh Editor** mode, FlexScan3D tries to move and rotate meshes so that their geometries line up. After you align meshes, they are also automatically "locked." A locked mesh is in a fixed position and orientation. That is, it cannot be moved, rotated, or edited. To move, rotate, or edit a locked mesh (for example, if a misalignment happens during alignment), you must first unlock it. To unlock a mesh, select it and click the Lock icon on its thumbnail to toggle the locked status. You can also unlock scans by double clicking on the thumbnail.

There are various methods you can use to align meshes in **Mesh Editor** mode (see below). The type of alignment can be specified by clicking the arrow below the line and selecting **Mesh Geometry**, **Selected Geometry**, **Markers**, or **Fine Alignment**, which are described below. This selection will be used every time you click the **Align** button above the drop-down.



Mesh Geometry

When aligning a mesh based on the geometry itself, you must rotate the mesh until it is in roughly the same position and orientation as any currently locked meshes.

To align a series of meshes:

1. Load the first mesh by clicking on the checkbox next to it in the list of scans.
2. Lock this first mesh by double-clicking on the scan in the list of scans.
3. Load and select another scan.
4. While holding down the ALT key, click on a mesh and drag it to rotate the mesh to the proper orientation.

   > If you need to also move the mesh, holding down the ALT key, drag the mouse while holding down both the left and right mouse buttons.

5. Click **Align** after the mesh is close to the same position.
   If the meshes are close enough, the mesh snaps into place, aligning with the locked mesh, and becomes locked itself.
6. Repeat steps 1 through 5 for the other meshes.
7. Select the thumbnails of the scans you modified, right-click on them, and choose **Save**.

To align a mesh to the closest mesh, click on the mesh and then click on the **Align** button in the **Alignment** group.

Selected Geometry

In some cases, aligning using the entire mesh geometry may fail. For example, if 90% of the data in a pair of scans is flat, normal geometry alignment may not work: everything matches so well in the flat areas that the scans may not snap together properly. In cases like these, you should align based on selected geometry.

To align based on selected geometry:

1. Load a scan by clicking on the checkbox next to the scan in the list of scans.
2. Lock the scan by double-clicking on its thumbnail.
3. Load and select another scan.

4. Load another scan and roughly align it to the first scan.
5. Change the alignment type by clicking on the **Align** button dropdown and choosing **Selected Geometry**.
6. Select an area of the unlocked mesh by holding down the CTRL key, and clicking and dragging over the part you want to select.
   You will want to select the "interesting" geometry on the mesh, that is, an overlapping area that is not completely flat.
7. Click **Align** to snap the mesh into place.
8. Select the thumbnails of the scans you modified, right-click on them, and choose **Save**.

Markers

With the proper setup for markers, alignment will be automatic. However, if marker alignment fails, you can either try using Mesh Geometry alignment, or take another scan. See *Scanning with Markers* (page 100) for more information.

Rotary

If a rotary table is present and has been calibrated, the rotary alignment is automatically applied to each mesh.

Preset

The Preset Alignment feature enables you to have two or more multiple scan heads' scans (all scan heads are controlled by one computer) to align immediately after scanning. You must have multiple scanners active to use this feature.

To set up the preset alignment:

1. Click the **Scan** button, ensuring a mesh is created by each scanner.
2. Align the scans.
   Follow the instructions in Mesh Geometry or Selected Geometry.
3. Select the thumbnails of the scans you modified, right-click on them, and choose **Save**.
4. While the scans are highlighted, click **Set Preset Alignment**.

For subsequent multi-head scans, the preset alignment will be applied automatically.

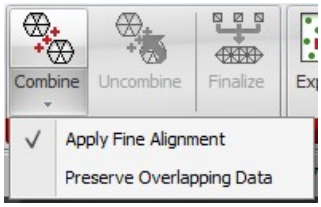
Fine Alignment

The fine alignment feature provides a tighter and more accurate alignment. You typically would use fine alignment when you have manually aligned your multiple scans. Generally this will be done when you are ready to combine your meshes.

Combining Meshes

Combining meshes merges several meshes into a single mesh. You must complete this step to finalize your model, which will allow you to resample, decimate, and hole fill. Two options are available to control

how meshes are combined.



The **Apply Fine Alignment** option causes FlexScan3D to perform a fine alignment before combining meshes.

By default, when combining meshes, FlexScan3D averages overlapping data points and then discards redundancies. For highly accurate scans, this is ideal for attaining the best results. For certain cases where accuracy is not quite as good, for example when using **High Sensitivity mode**, enable the **Preserve Overlapping Data** option by clicking on the drop-down on the **Combine** button. During the Finalize step using a combined mesh with overlapping data, the **Smoothed Merge** option can be used to average out the overlapping geometry to provide a smooth fused result.

To combine meshes:

1. Load all of the source scans by clicking on the checkbox next to each scan in the list of scans.
2. If you want to change the options, click on the **Combine** button's drop-down.
3. Click on the **Combine** button.
   A thumbnail for the newly combined mesh will be displayed. The "multiple mesh" icon next to the thumbnail indicates that it has been combined.

Uncombining Meshes

To un-merge several meshes into their original meshes:

1. Load the combined scan from the list of scans.
2. In the main ribbon on the right, click the **Uncombine** button.

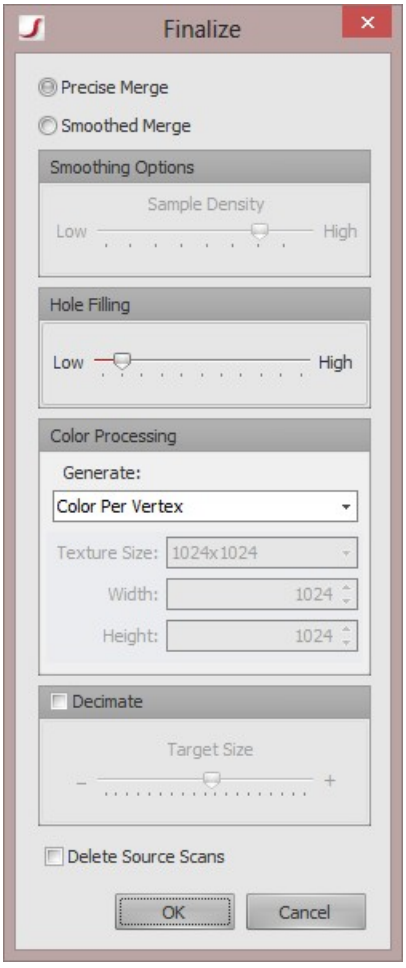
   You will now see all of your original meshes.

Finalizing Meshes

To finalize a mesh:

1. Select a combined mesh from the list of scans.
2. On the far right of the main ribbon click the **Finalize** button.

3. The **Finalize** options window will appear.



4. Select the desired options (see table below), then click **OK** to start the finalize process.

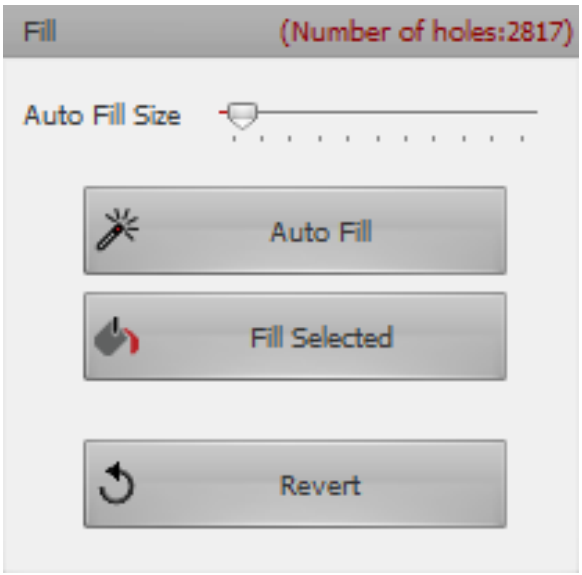
Finalize options

Option	Description
Precise Merge	Selected by default. This mode assumes the input scans are highly accurate, and makes only minor adjustments during the merging process. Note that this mode is unavailable for meshes which were combined with the Preserve Overlapping Data

Option	Description
	option enabled.
Smoothed Merge	Takes in all the input points and outputs a smoothed average of the data. The **Smoothing** slider adjusts how many points to include for creating the final mesh. Lower sample densities will be faster to finalize and will give result in a smoother mesh; higher densities will take longer to complete, but will also result in more accurate results.
Hole Filling	Combined meshes often contain small holes or gaps due to occlusions, where the scanner cannot reliably see a portion of the object. This feature fills in those holes based on the neighboring geometry. Using higher hole-filling settings will allow larger holes to be filled.
Color Processing	Provides the following options: **None**: Skips the colour processing step. The final mesh will not contain any colour information. **Color Per Vertex**: Adds colour data to each point in the final mesh. Note that the mesh density will affect the visual result, and will not look good in low density final meshes. **Generate Single Texture**: Generates complete UV-mapped texture data at the specified resolution. A single bitmap image will be associated with the final mesh. Mesh density does not affect the texture result. For applications which require good texture mapping, this is the best option.
Decimate	This allows for intelligent decimation of the mesh as part of the finalize process. Moving the slider to the left will produce fewer vertices for the final mesh, and vice versa. Decimated meshes will appear very similar to the original high-resolution mesh despite containing significantly fewer points.
Delete Source Scans	This will delete the combined mesh, including all the source scans. This may be useful in cases where many objects are being scanned and processed, and hard drive space is a concern. Generally though, it is recommended to leave this unchecked, since it may be necessary to run through the finalize process with slightly different settings to achieve the desired results. If the source scans were deleted, re-running the finalize process would not be possible.

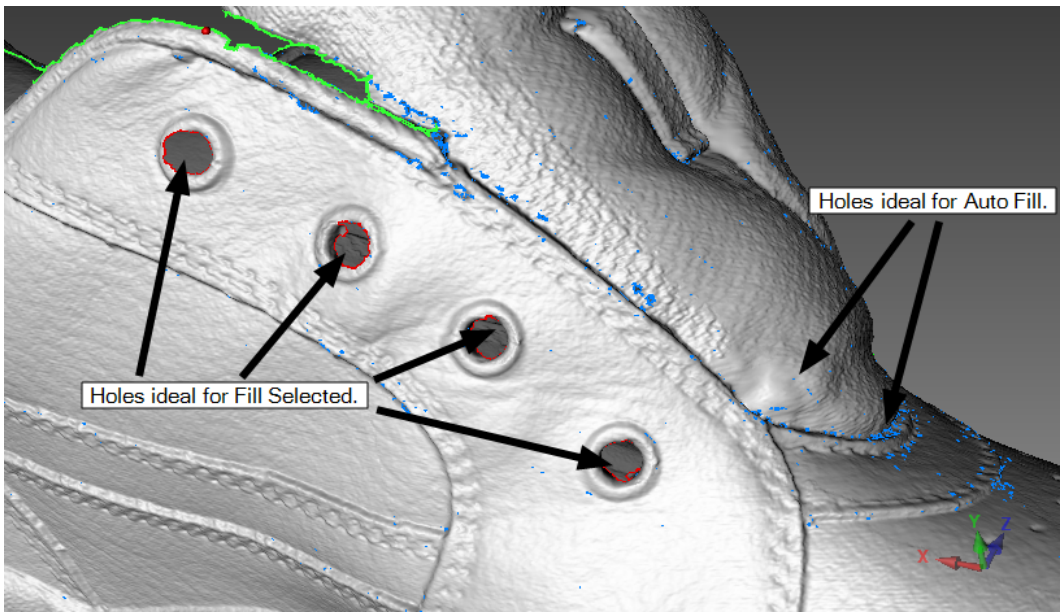
Hole Filling

You can interactively choose which holes are filled in Hole Filling mode. You can choose to fill holes using **Auto Fill** or **Fill Selected**. The number of holes in the scan is displayed in the fill panel.



The image below shows holes that are ideal for **Auto Fill**, which are generally small, and larger holes that are ideal for manual filling using the **Fill Selected** function. Very large holes should not be filled using either function. FlexScan3D uses the following hole outline colors.

- Green: Non-selected hole.
- Blue: Hole selected using Auto Fill Size slider.
- Red: Hole selected by clicking on hole outline.



Auto Fill

The **Auto Fill Size** slider lets you choose the number of holes that are selected, based on the size of the holes. Moving the slider all the way to the right selects all of the holes. Moving the slider to the left

lowers the upper hole size limit and selects fewer and fewer holes. The outline of holes that are selected are displayed in blue in the 3D window. Unselected hole outlines are displayed in green. Click on the **Auto Fill** button to automatically fill the selected holes. Click on the **Revert** button to undo the last hole fill.

Fill Selected Holes

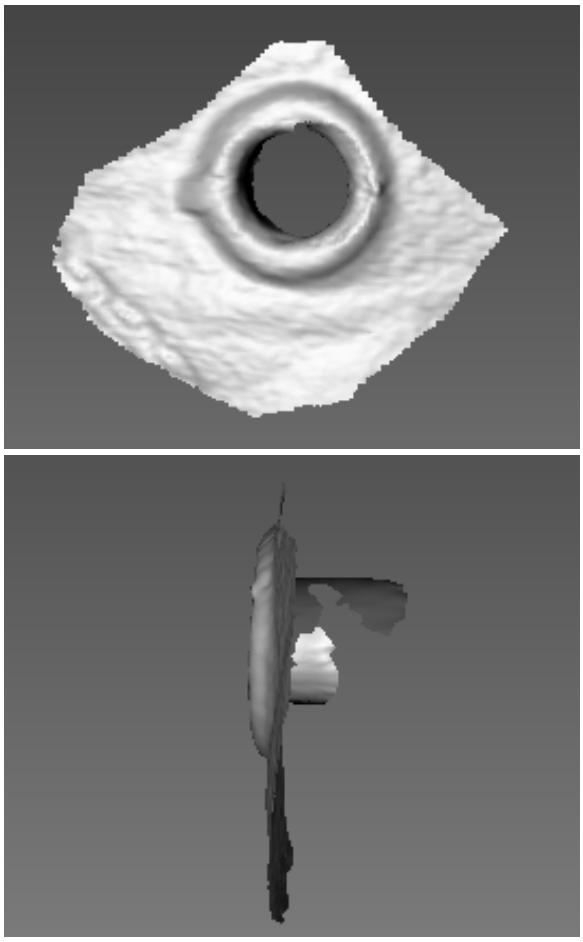
You can also select individual holes by moving the mouse pointer over a hole outline and clicking on it. Its outline will turn red. You can select multiple holes by holding the Shift key and clicking on additional hole outlines. Then click on the **Fill Selected** button to fill the selected holes. Click on the **Revert** button to undo the last hole fill.

Clicking on anything other than an outline in the 3D window deselects any selected holes.

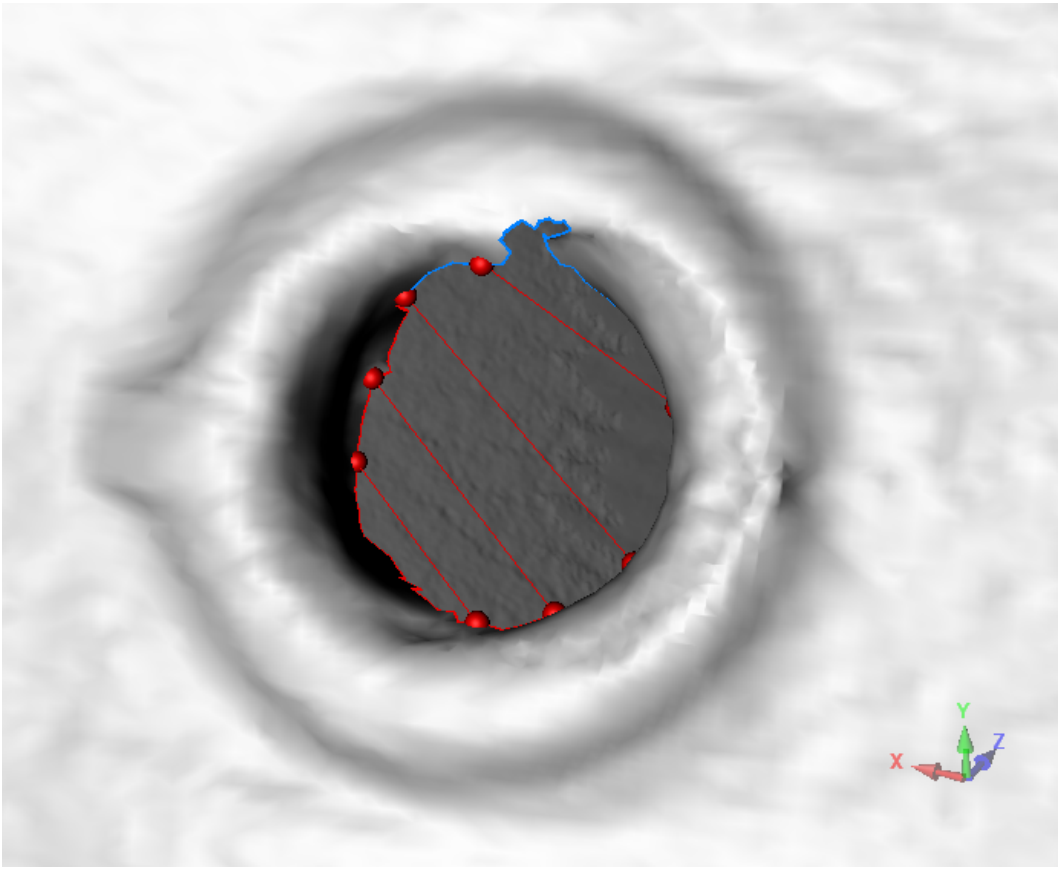
Bridges

Bridges divide a hole into smaller sections and can be used to simplify overly complex holes. A hole can be overly complex if it provides insufficient information for FlexScan3D or if its boundaries goes too far below the plane of the majority of the hole's boundary. If FlexScan3d encounters a hole that it can't fill, it will display an error message in the status bar.

The image below shows an example of a hole that is too complex for FlexScan3D. The side view shows that the hole's boundary actually goes "into" the shoe model.



To help FlexScan3D fill these holes, you can use bridges to break the hole down into smaller, simpler sections.



To add a bridge:

1. Move the mouse pointer over the outline of a hole that FlexScan3D can't fill because it is too complex.
2. Click on the outline with the mouse, drag the pointer to another part of the outline, and release the mouse button to create a smaller section.
3. Continue dividing the hole into small sections.
4. Click on the outline contained in a section to select it. Hold shift and select other sections if you have created any others.
5. Click on **Fill Selected**.

It may take a few tries to find the right number and size of sections for FlexScan3D to successfully fill the hole.

To delete a bridge:

1. Select one of the bridge's points.
2. Right-click on the point.
3. Choose **Delete**.

Importing and Exporting

FlexScan3D allows you to import and export meshes.

Importing

File Formats

FlexScan3D supports importing meshes from the following formats:

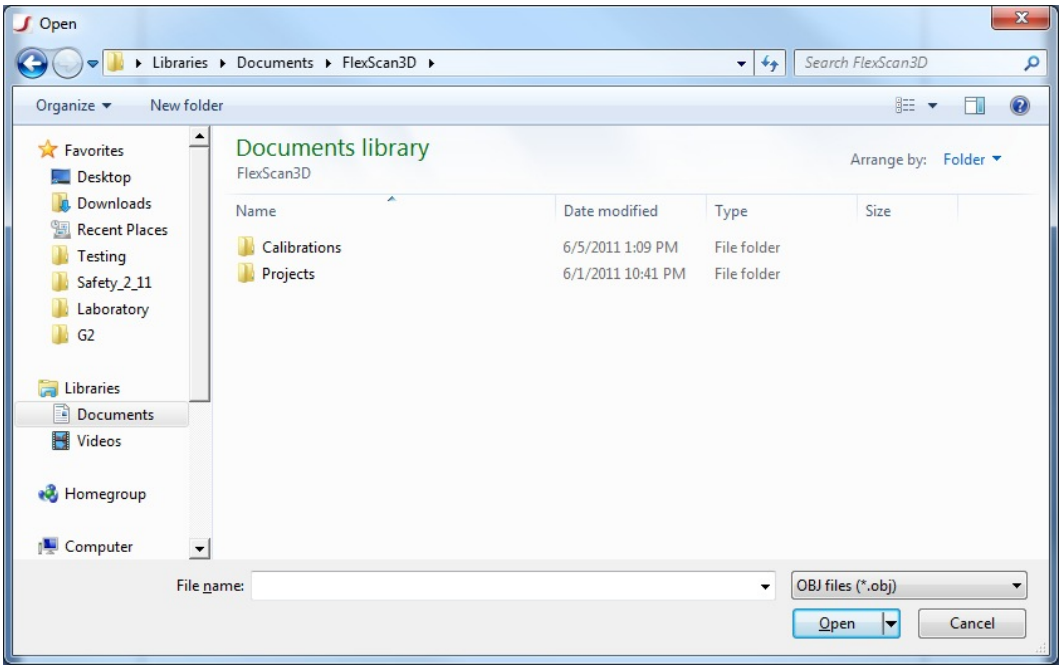
- .3d3
- .stl
- .obj

Importing a Mesh

You can import a mesh into your current project. You can then use mesh-related operations such as alignment and deviation analysis.

To import a mesh:

1. If you do not have a current project, click the **Project** tab and either create or open a project.
2. Click **Import** to open the **Open** dialog box.



3. Choose the file type for the model you wish to import.
4. Navigate to and select the file you want to import.
5. Click **Open**.

Exporting

Exporting a mesh converts your scan data to a format compatible with 3D modelling software in case you need to do more advanced post-processing of the geometry. The export applies to all meshes currently loaded into the 3D window

File Formats

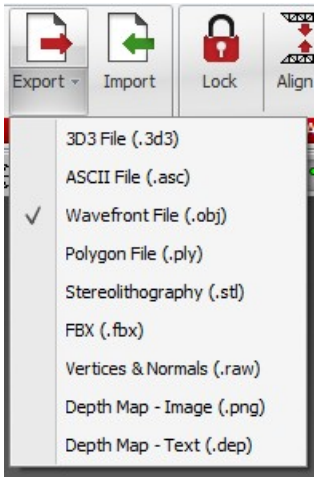
FlexScan3D supports exporting a mesh to the following formats:

- .3d3
- .asc (ASCII)
- .obj
- .ply
- .stl
- .fbx
- .raw
- .png
- .dep

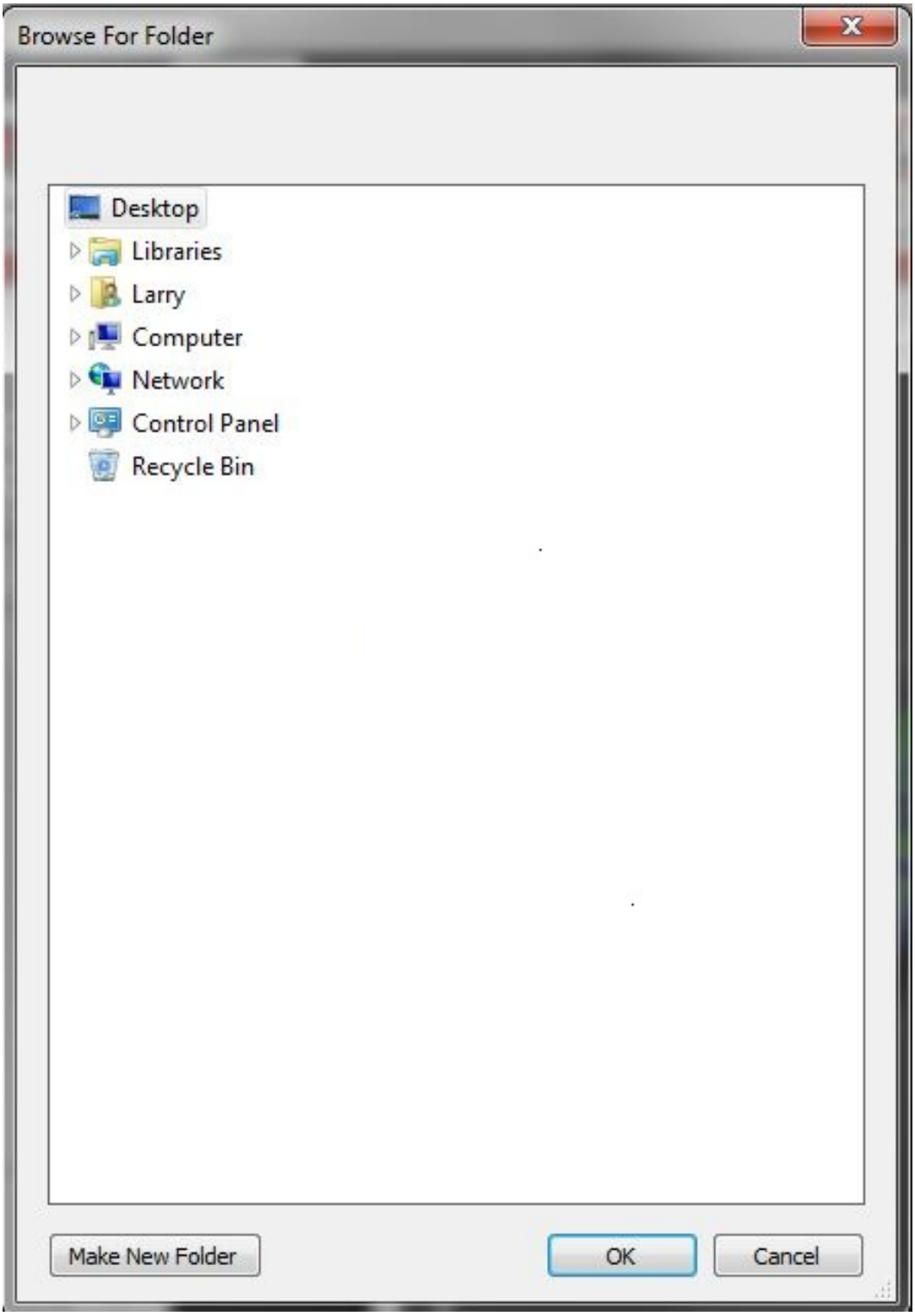
Exporting Meshes

To export a mesh:

1. If you do not have a current project opened, click the **Project** tab and open a project.
2. Load one or more meshes by clicking on the checkbox next to the meshes.
3. Click the drop-down arrow next to the **Export** button and select the file type.



4. Click **Export** to open the **Browse For Folder** dialog box.



5. Navigate to a location to store the meshes in an existing folder or click **Make New Folder** to create a new one.
6. Click **OK**. The names of the exported files are the same as their respective scan names.

Advanced Scanning Techniques

The following topics describe scanning techniques for various scenarios. Please note that the topics are tailored for scanning using HDI Advance scanners, and may not apply to scanning with HDI 100 series scanners.

Scanning with Texture

Capturing textured 3D scan is possible with just colored machine vision cameras. However, if you need a much higher resolution scan, a digital SLR camera is recommended. This provides greater detail and brighter color compared to machine vision cameras. Below you will discover a bit more about this method of scanning.

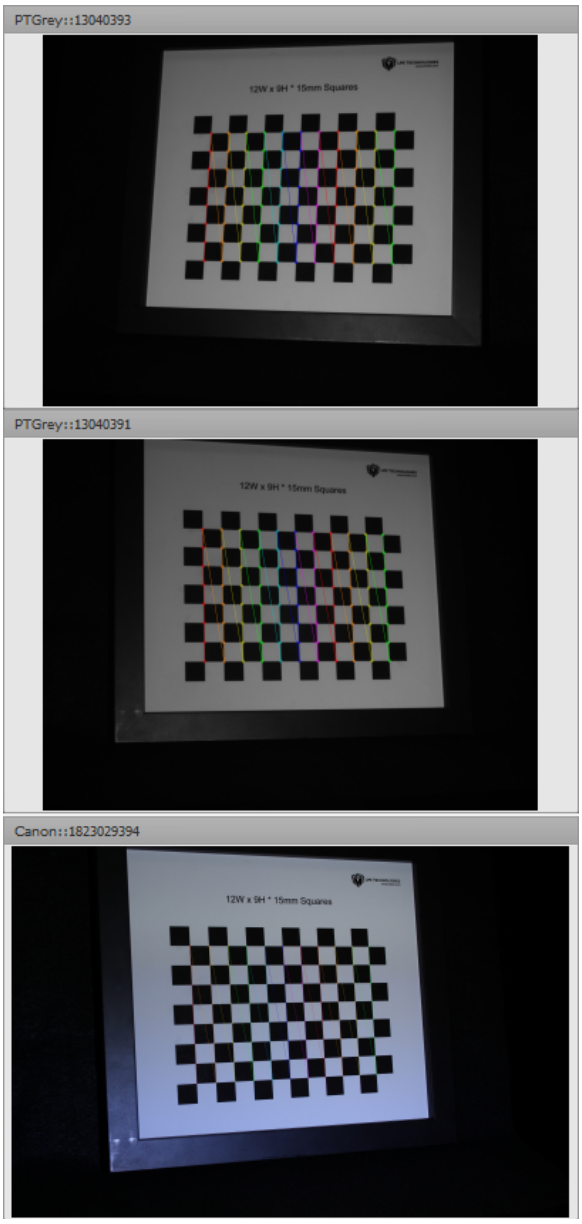
Setup

In order to get the most out of using a digital camera, please follow the instructions below as seen in the image. Ensure that your digital camera is placed just above "Cam 1". The DLSR does have to be perfectly lined up with Cam 1. It is also recommended that they share a common field of view (FOV). In the following example, 2 machine vision cameras plus a Canon camera are used

Texture camera setup

Attaching the digital camera to your scanner is quite easy. A "Micro Ball Head" is used to mount the camera onto the scanner, carefully placing it as close to "Cam 1" as possible. You will notice in the image above that the facing direction of both Cam 1 and the digital camera are very similar. This is needed to properly overlay the texture onto the mesh.

Micro ball head



Machine vision camera and digital camera comparison

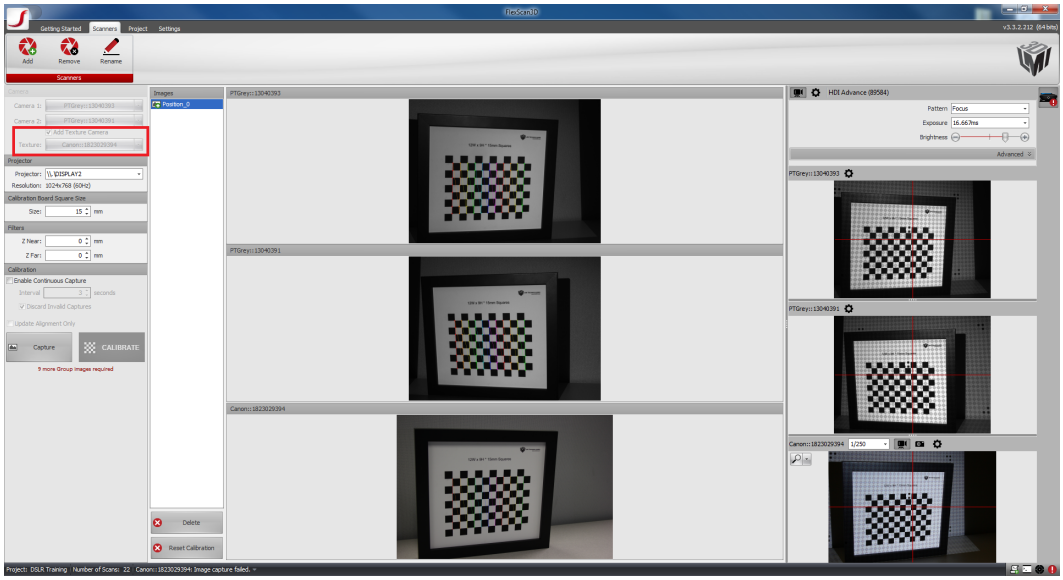
When you attach the digital camera to your computer, the drivers should automatically install.

> There is a known incompatibility between Canon camera drivers and Windows 7 "N" (no media) which is available in Europe. To resolve this, install Windows Media Player, which will also install drivers that Canon drivers are dependent on.

Calibration

Once the hardware has been properly set up, calibration can begin. Calibrating with a texture camera is very similar to Duo or Single scan mode; see *Step-by-Step Instructions (Video)* (page 37). Check to see if the digital camera is in good focus just as you would check the machine vision cameras. Lighting plays a

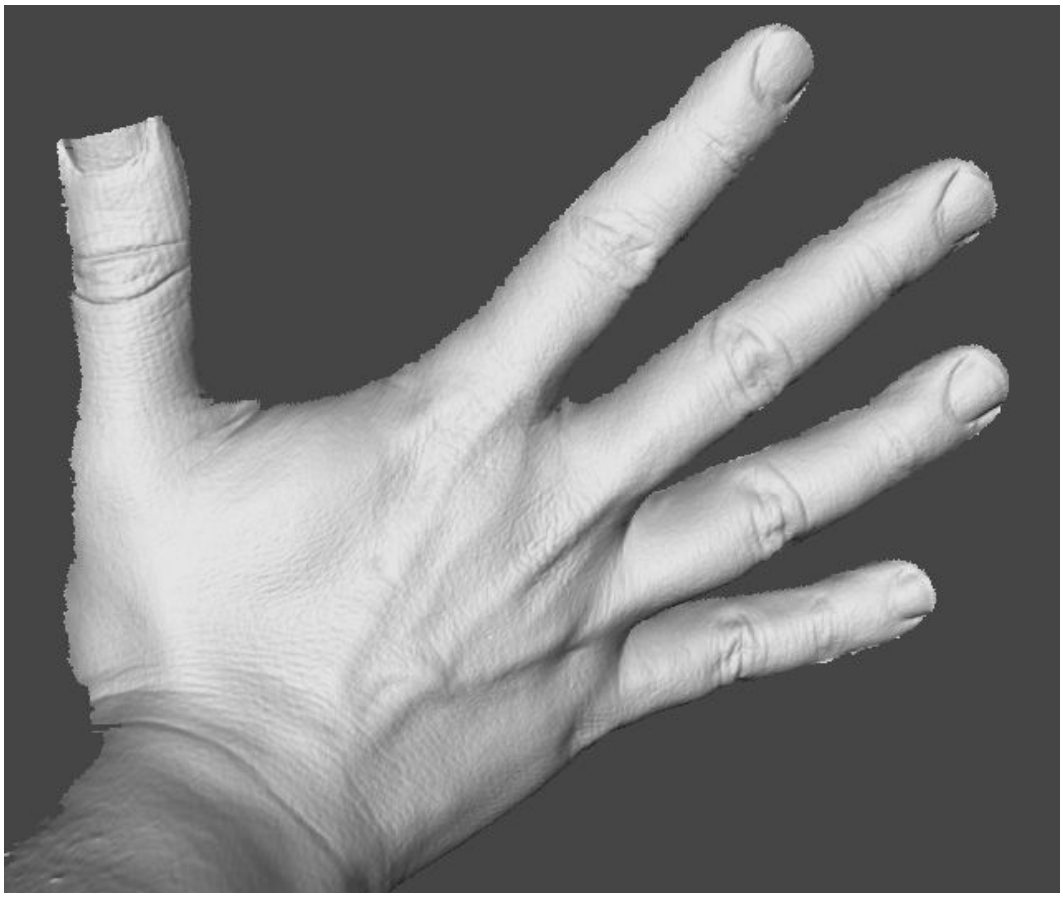
big role as well, so ensure that your digital camera is bright enough to pick up the calibration board patterns. The red square in the image below marks the section where you will choose to enable a texture camera, as well as choose which camera to use. We recommend that you turn on the flash when calibrating and scanning, so please use an A/C adapter that plugs the camera directly into an electrical socket. Switching batteries while calibrating or scanning will cause you to lose that calibration due to physically having to touch and remove the battery. See *Calibrating the Scanner (Advanced Configuration)* (page 47) to adjust your aperture, shutter speed, and other settings.



Calibration view with texture camera

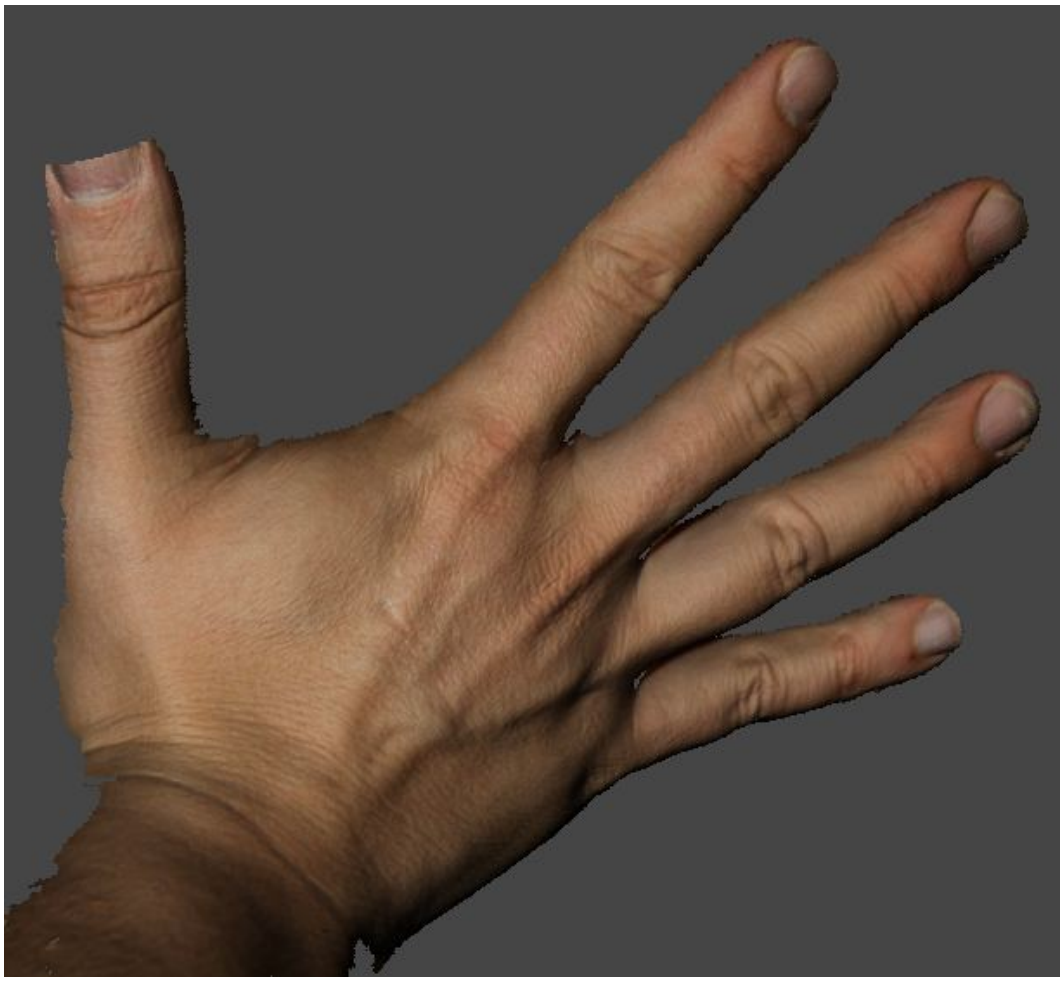
Scanning

Once calibrated you are able to do your first scan. This process is similar to scanning with a Single or Duo mode. You will notice that the digital camera will flash before the machine vision cameras are activated. The example below shows a hand scanned with just machine vision cameras:



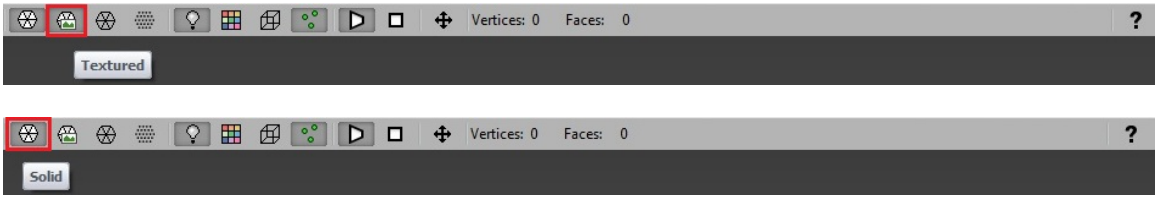
Hand scan using machine vision cameras

The second example shows the same hand scan with texture enabled. Notice the in-depth detail when compared to a regular scan:



Hand scan using texture

In FlexScan3D, you can switch between these two views. Simply click the button on the tool bar, indicated by red squares below, to switch to a different view mode.



Scanning with Markers

> Scanning with markers requires a duo camera configuration. It will not work with single scanners.

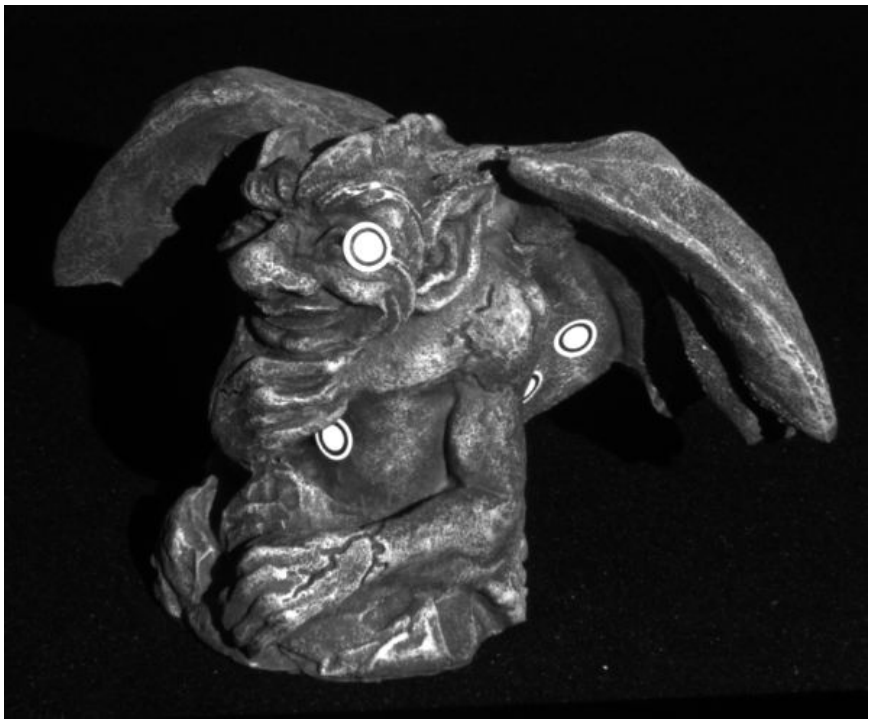
Scanning with markers provides a faster and more convenient way of aligning multiple scans since the alignment algorithm is greatly sped up by having far fewer reference points to search for and match up.

Markers are generally small, round stickers (provided by LMI Technologies) that you place either on the subject or on a movable surface the subject rests on. After scanning the subject and generating a mesh, the scanner then searches for the markers and attempts to align them with a previous scan. If enough matches are found, the alignment is successful and you are ready for a new scan. If not enough matching markers are found, the alignment fails and you can either retry or use mesh geometry alignment.

Setup

The placement of markers should be in an unique pattern, similar to stellar constellations, to make it easy for the scanner to identify markers. If you place the markers in a regular pattern (such as a grid), the scanner may have difficulty matching markers and fail to align scans. Please ensure the markers are placed so both cameras can see them.

Good marker placement



Bad marker placement

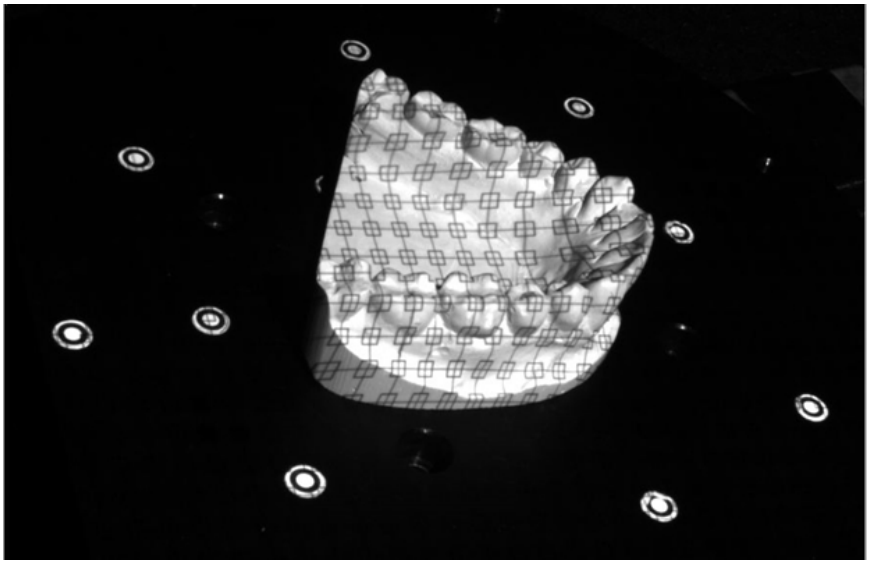
There are two placement strategies that may be used: Direct (directly on the subject) or Indirect (on a surface the subject rests on, such as a rotary table). Advanced users may use a mix of these strategies plus mesh alignment to create a complete 3D reconstruction of the subject. To align two scans, at least 4 common markers must be matched between the two scans. This means that regardless of which strategy you use, you must use enough markers so that there are always at least four markers visible from both cameras during a scan.

Direct Placement

Object with markers

For direct placement, you place the markers directly on the subject. This is an appropriate strategy if the subject has enough smooth surfaces for the markers to stick to, and if the markers will not cover up any necessary fine details. This is most useful for when you wish to create a complete reconstruction of the subject from all angles because this does not require that the subject is fixed to a surface.

Indirect Placement



Dental mold with markers on rotary table

If the subject has a lot of fine detail that cannot be concealed, or is textured in such a way that direct placement will not work, then it is best to use indirect placement. This works by attaching the subject to a movable surface (such as a rotary table), then place the markers on the table. The advantage of this approach is you get to preserve maximum detail in the subject while still being able to use fast marker alignment. The subject must remain static relative to the markers, so you can only move the surface and never the subject itself. This creates the drawback that you will not be able to scan the underside of the subject without starting a new mesh fragment.

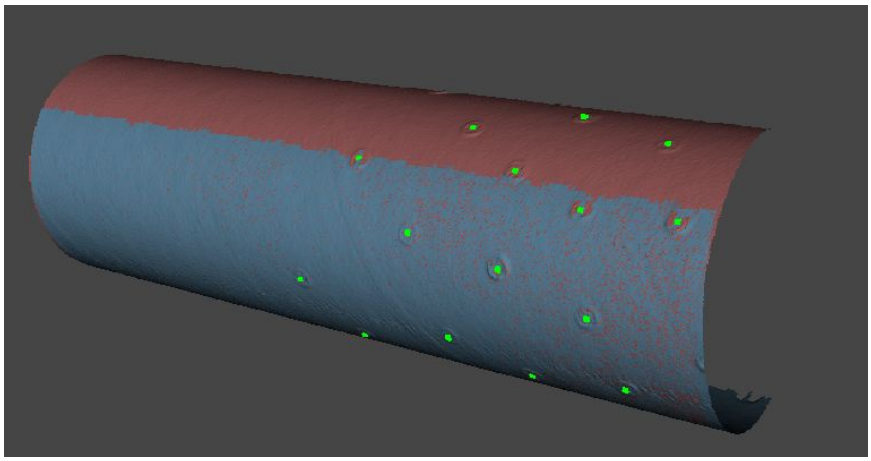
Creating a complete reconstruction of a subject using indirect placement requires that multiple mesh fragments are created that are then later aligned using mesh alignment.

Scanning

When scanning a subject with markers, the most important thing to keep in mind is that at least 4 markers must be visible to both cameras, and that alignment requires that at least 4 common markers are found between scans. This means that during multiple scans, do not rotate the subject too far so that the scanner is unable to match markers between consecutive scans for alignment. Therefore it is advisable that the subject is rotated by less than 30 degrees in any direction between scans, depending on the subject shape and marker configuration.

When using a movable surface for indirect placement, care must be taken to not allow the subject to slip or move out of position relative to the markers.

Below is an example of 2 separate meshes merged into one using markers:



Two meshes merged using markers

Alignment

After each scan, markers will be detected and identified. If there are previous scans made using the same markers, the scanner will automatically attempt to align the new scan with the previous ones. If it succeeds, you may rotate the subject (or surface) and proceed to the next scan.

If it fails, then you have several options:

- Delete the scan, and rotate the subject back nearer to its previous position for a better alignment attempt.
- Keep the scan, and either merge it with the previous scans using mesh alignment, or create a new scan with the subject repositioned in between the last two attempts.

Scanning a Large Object

Scanning a large object simply requires the cameras to be placed on the outer slot of the HDI Advanced Scanner or manually adjust the angle of the cameras for custom scanners.

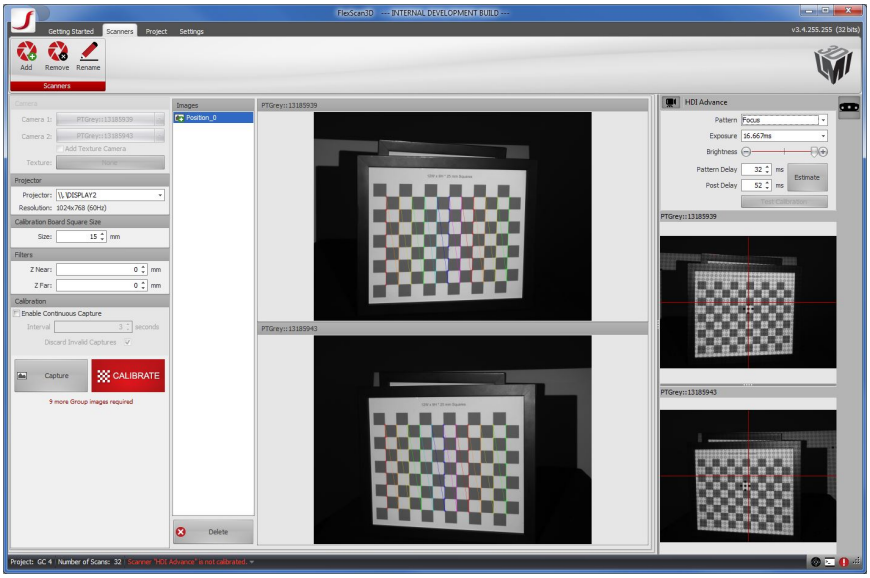
Setup

The setup is similar to the typical HDI Advanced Scanner setup with the only difference of camera placement. The cameras should be placed on the outer slot of the HDI mount, and a pair of 12 mm lenses should be sufficient. Depending on the size of the object, you can manually change the camera's field of view by using a single bolt, removing the one closest to the projector. (R1 up to 1.5 meter field of view, and R2 up to 2 meter field of view.)

Large field of view with cameras on the outer slots

Calibration

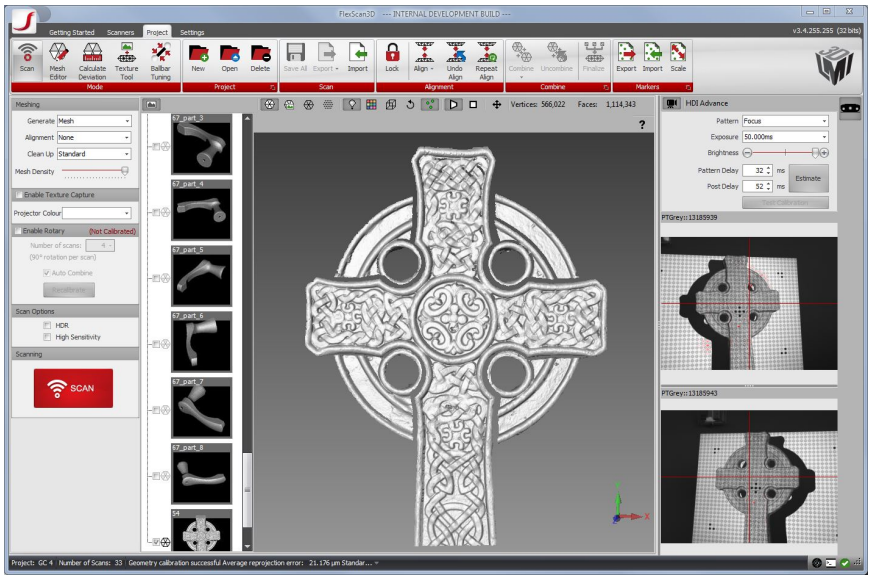
In general, a larger calibration board is recommended for larger field of views. A 25 mm calibration board should be used in place of the 15mm calibration board. For custom angles with even larger field of view, a 40mm calibration board is recommended. Once the hardware has been properly set up, calibration can begin.



Calibration view

Scanning

Once your scanner is calibrated, you are able to do your first scan.



Scan with a larger field of view

Scanning a Small Object

Scanning a small object simply requires the cameras to be placed on the inner slot of the HDI Advanced Scanner or manually adjust the cameras for custom scanners.

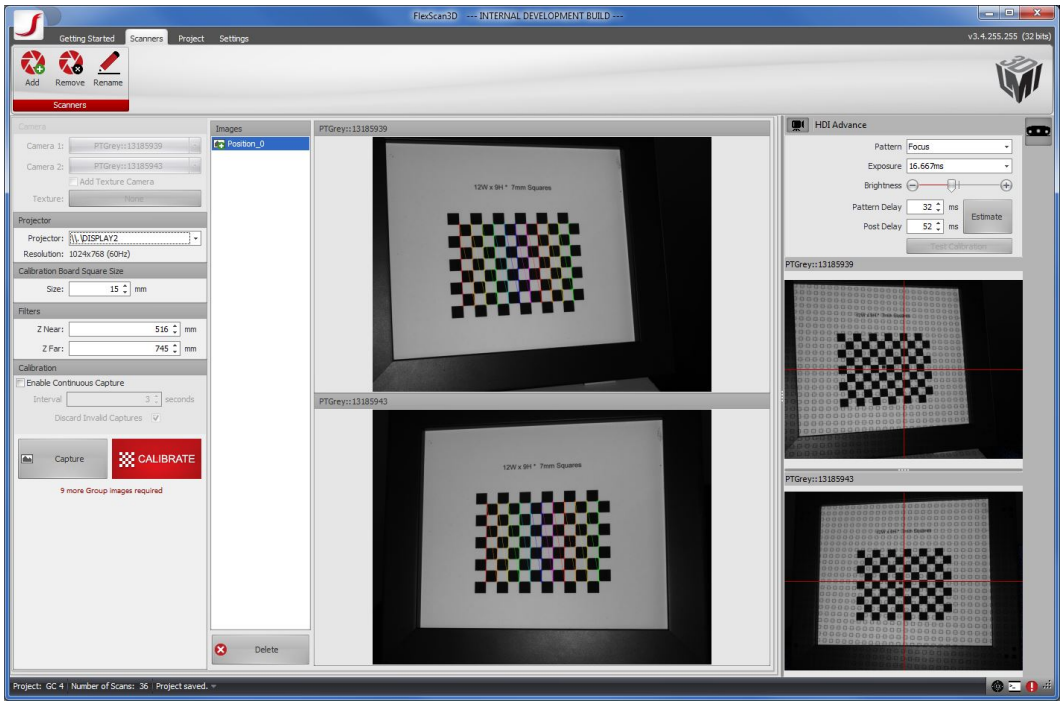
Setup

The setup is similar to the typical HDI Advanced Scanner setup with the only difference of camera placement. The cameras should be placed on the inner slot of the HDI mount. Depending on the object size, 12 mm, 16 mm, or 25 mm lenses are recommended.

Small field of view with cameras on the inner slots

Calibration

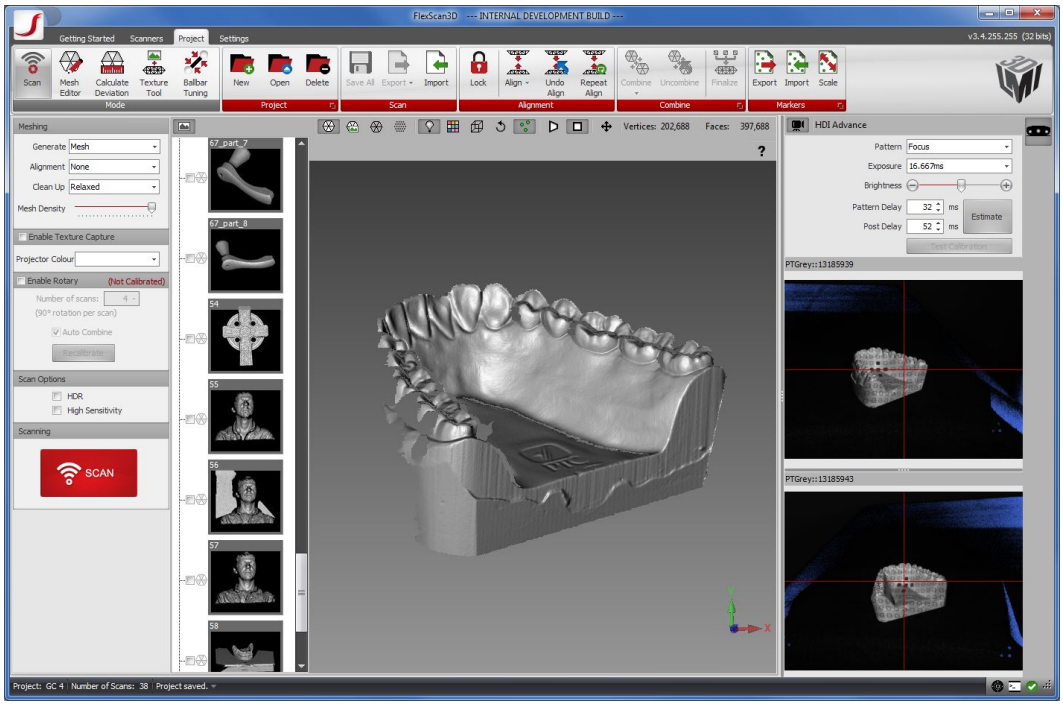
In general, a small calibration board is recommended for small field of views. A 5 mm or 10 mm calibration board should be used in place of the 15 mm calibration board. Once the hardware has been properly set up, calibration can begin.



Calibration view

Scanning

Once your scanner is calibrated, you can do your first scan.



Scan with a smaller field of view

Scanning a Human Face

Scanning a human face simply requires the cameras to be placed on the outer slot of the HDI Advanced Scanner.

For more tips on 3D scanning please refer to the blog post, 3 Simple Tips to Improve 3D Face Scanning Results: http://blog.3d3solutions.com/bid/52375/3-simple-tips-to-improve-3d-face-scanning-results.

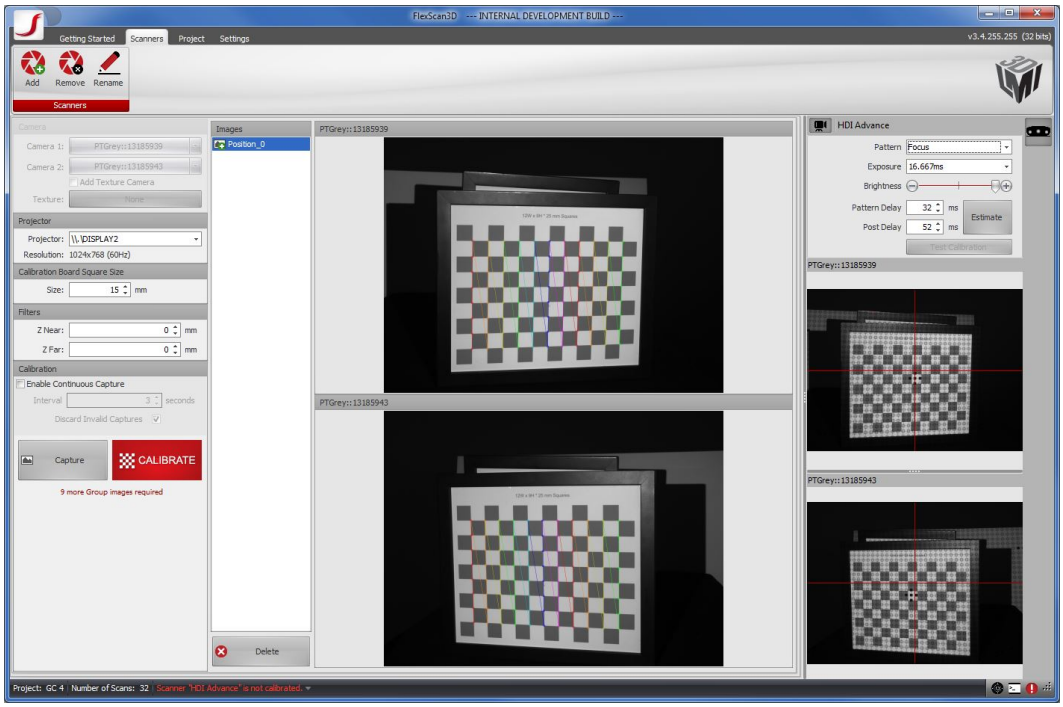
Setup

The setup is similar to the typical HDI Advanced Scanner setup with the only difference of camera placement. The cameras should be placed on the outer slot of the HDI mount, and a pair of 12 mm lenses should be sufficient.

Large field of view with cameras on the outer slots for a face scan

Calibration

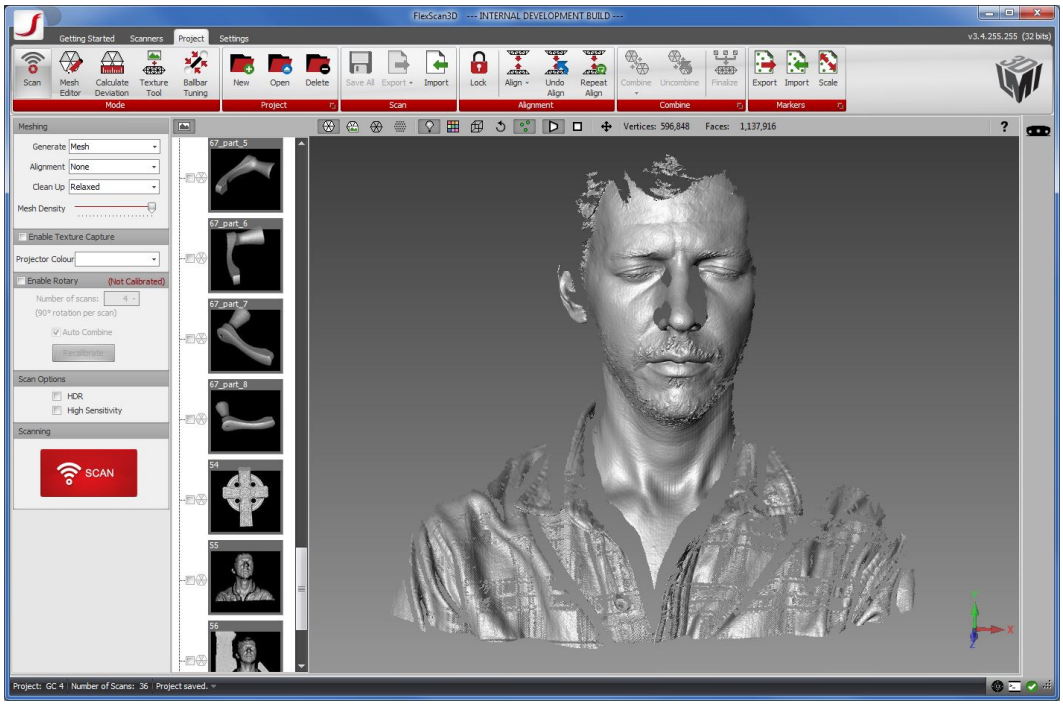
In general, a larger calibration board is recommended for larger field of views. A 25 mm calibration board should be used in place of the 15 mm calibration board. Once the hardware has been properly set up, calibration can begin.



Calibration view

Scanning

Once your scanner is calibrated, you can do your first scan. If possible, try to lean the head against a wall or sit in a chair which can turn and get someone else to turn the person in different positions. To do full coverage of a face, a left, front and right face scan is needed. (Or even just a left and right scan.) To get the best results, try to maximize the field of view to capture as much as possible in a single shot.



Face scan with a larger field of view

Data Cleanup/Alignment

3D face scanning can be difficult to align accurately because there is movement in between each scans: the neck invariable changes position. By deleting the neck data and performing an alignment on just the face will help to align the face scans more accurately.

Scanning a Mechanical Part

Scanning a mechanical part, depending on the size of the part, users may need to adjust the positions of the cameras accordingly. For smaller parts, the cameras should be placed on the inner slot of the HDI Advanced Scanner. And for larger parts, the cameras should be placed on the outer slot of the HDI Advanced Scanner.

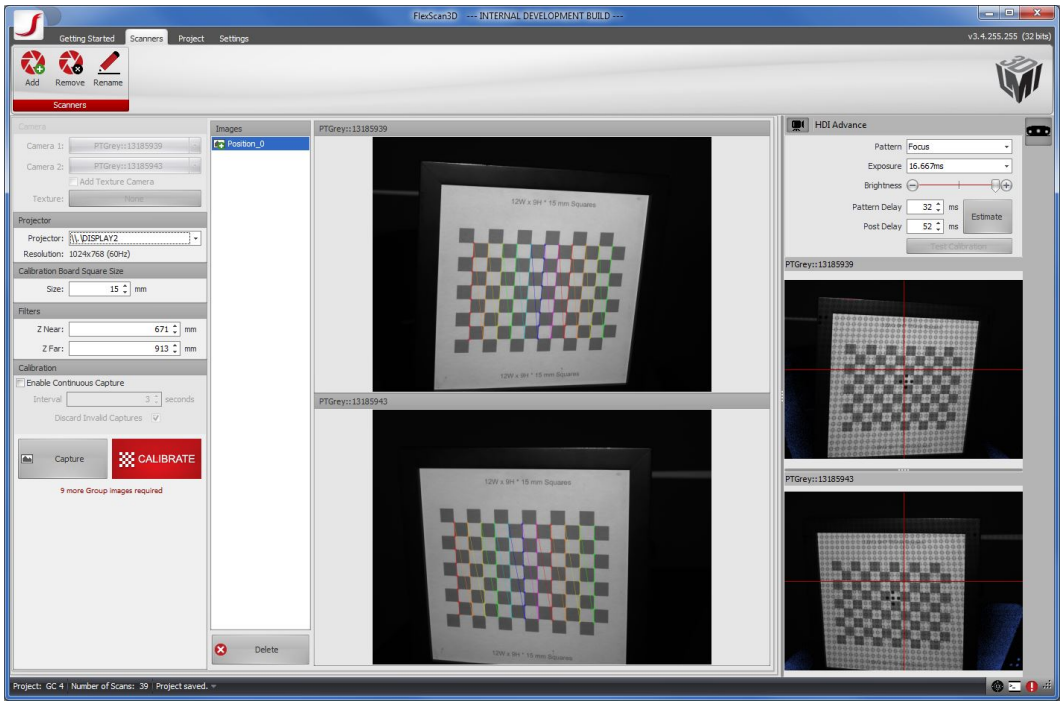
Setup

The setup is similar to the typical HDI Advanced Scanner setup with the only difference of camera placement. For this particular setup, the cameras are placed on the middle slot of the HDI mount, and a pair of 12 mm lenses are being used. Depending on the size of the object, users can move the cameras to the inner or outer slots for a more suitable field of view.

Cameras on the middle slots of the HDI mount

Calibration

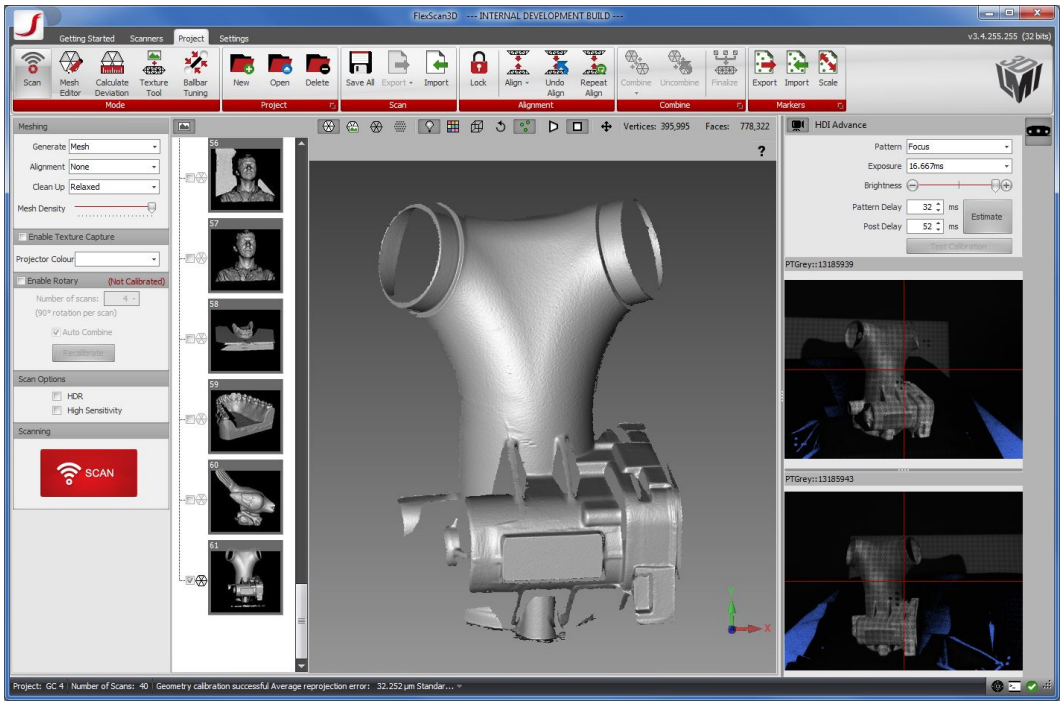
Depending on the field of view, different calibration board should be used accordingly. For this particular setup, a 15 mm calibration is being used. Once the hardware has been properly set up, calibration can begin.



Calibration view

Scanning

Once your scanner is calibrated, you can do your first scan.



Scan with a larger field of view

Scanning Hair

In the past, capturing hair was an exercise in futility. For a head scan, the facial features would look great, but the person's hair would barely be visible or it would not show up at all. Hair can be a challenge for high-accuracy structured light scanners. It consists of very fine and partially translucent filaments which scatter the light and are often interpreted as noise during 3D reconstruction. As a result, these areas are usually removed from typical scan data. The High Sensitivity mode was introduced to address this shortcoming.

This article provides some tips on acquiring usable hair scans. While it is mainly targeted at scanning human hair, it also applies to scanning other objects, such as fuzzy clothing, stuffed animals, or dense vegetation.

Steps

Scan

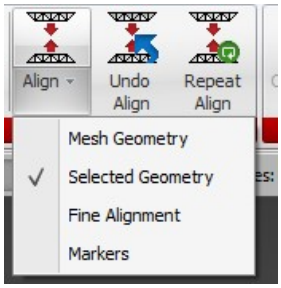
Before scanning, ensure the **High Sensitivity** checkbox above the **Scan** button is checked. High sensitivity mode will fill in the holes and cracks in a scan with best guesses based on the surrounding geometry. Also ensure that the **Clean Up** dropdown menu is set to **Relaxed**. The scan data will most likely contain some jagged areas, but with some minor editing we can end up with a great final mesh.

Raw scans, aligned

Align

Since hair scans are usually quite rough, geometry-based alignment may be unreliable for scans mainly consisting of hair. Instead of relying on the rough hair scan data, first ensure that each scan contains other surface information. For body scans, this can simply be the person's torso. Use the **Selected Geometry** alignment feature to align to the torso. For a scan of an object which has no smooth

surfaces, such as a teddy bear or a fuzzy sweater, easily scannable items (ones with smooth and opaque surfaces) should be strategically placed around the object. **Selected Geometry** alignment can then be done by selecting the geometry of these external items.



Selected alignment setting

Combine

Save and Duplicate

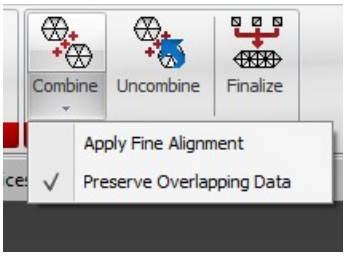
Once alignment is complete, click the **Save All** button to write the changes to disk. Next, select all of the meshes in the thumbnail view. (Hint: select any mesh and press CTRL-A on the keyboard to select all.) Right-click anywhere in the thumbnail list and select **Duplicate** from the pop-up menu. All of the duplicated scans will be selected. This is what we want for the next couple of steps.

Rebuild

With all of the duplicated meshes selected, click on the **Mesh Editor** button. Change the **Clean Up** dropdown to the **High** setting, then click **Build**. This will generate a new model which only includes the most accurate data. As a result, much of the hair will be removed, but the face should look much better. Now we have two full copies of the data, one which we will use for the hair, and one which we will use for the face and skin.

Initial Combine

First, ensure that **Preserve Overlapping Data** is checked, and that **Apply Fine Alignment** is unchecked in the **Combine** button dropdown menu.



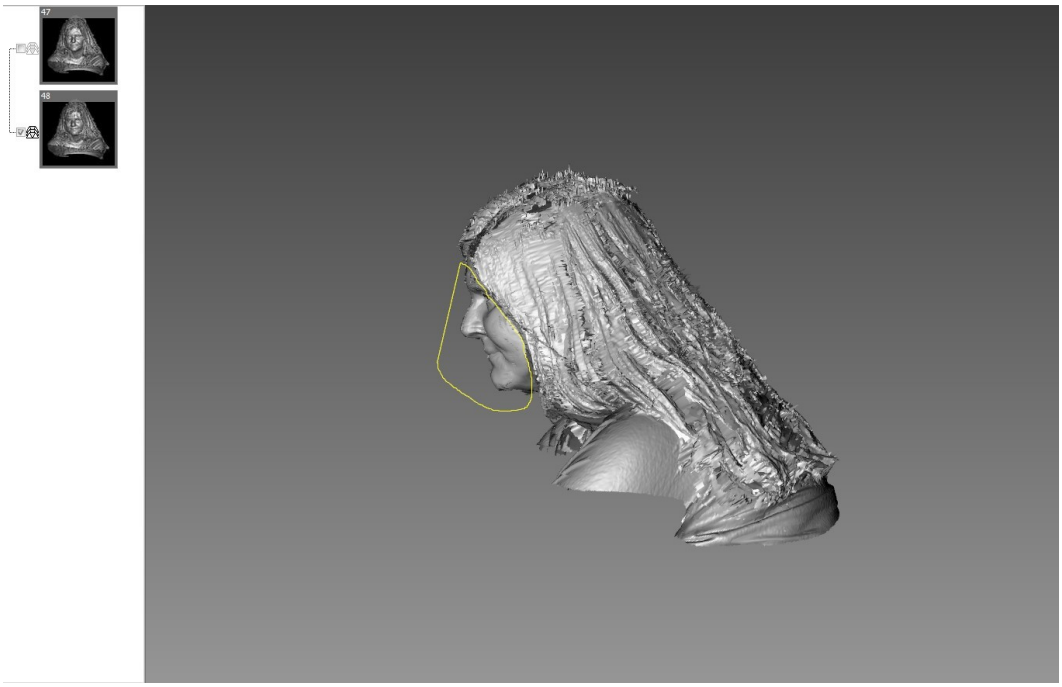
Combine settings

Combine all of the duplicated scans. Now, select all of the original scans and Combine those as well. You should end up with two almost identical meshes, with the duplicated mesh containing less hair but better facial features.

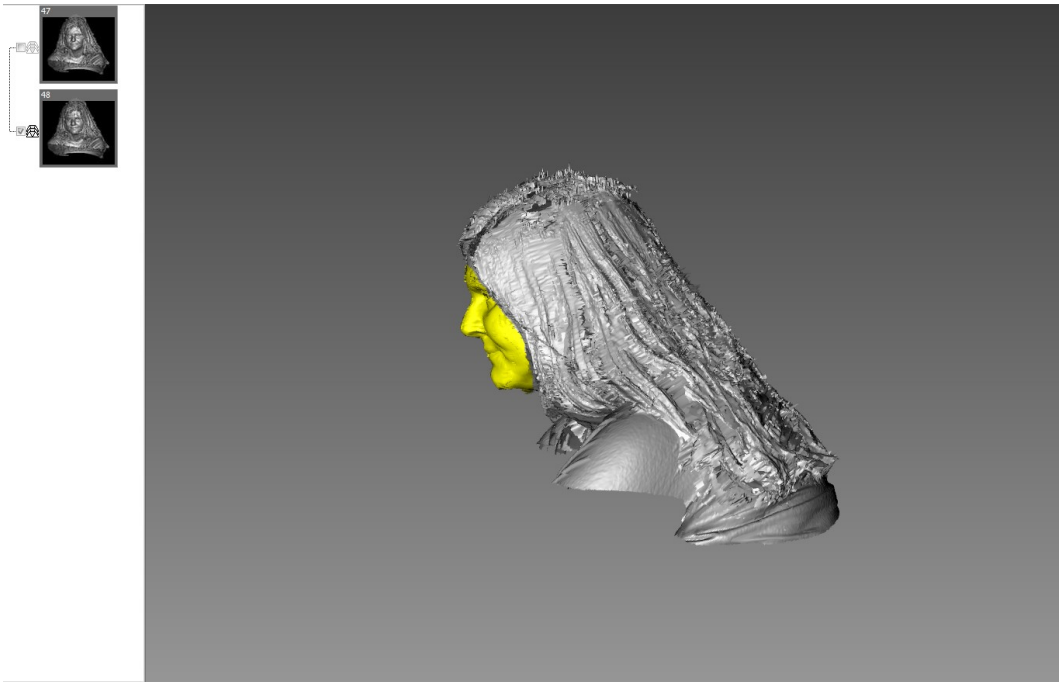
Left scan: Cleanup set to High. Right scan: Cleanup set to Relaxed.

Mesh Editing

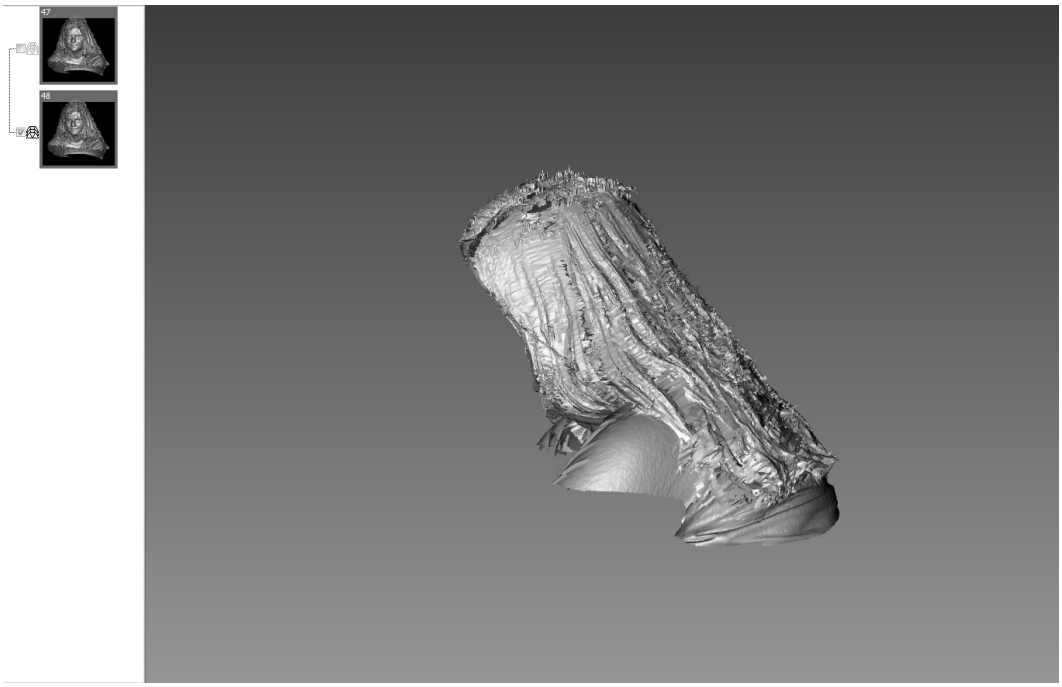
Load the combined mesh from the original data (it should be the second item in the thumbnail list) and unload any other meshes. This mesh most likely has some oddities around the face, but since we have the higher-quality version that we created in the previous steps, we can remove the offending geometry from this one. When deleting the face geometry, make sure to avoid doing the sub-selection head-on, otherwise the hair geometry may end up getting selected too. A side view of the face makes it much easier to select without accidentally selecting the hair. If there are some other surfaces that look pretty rough, remove those as well. When the geometry deletion is complete, click **Save All**.



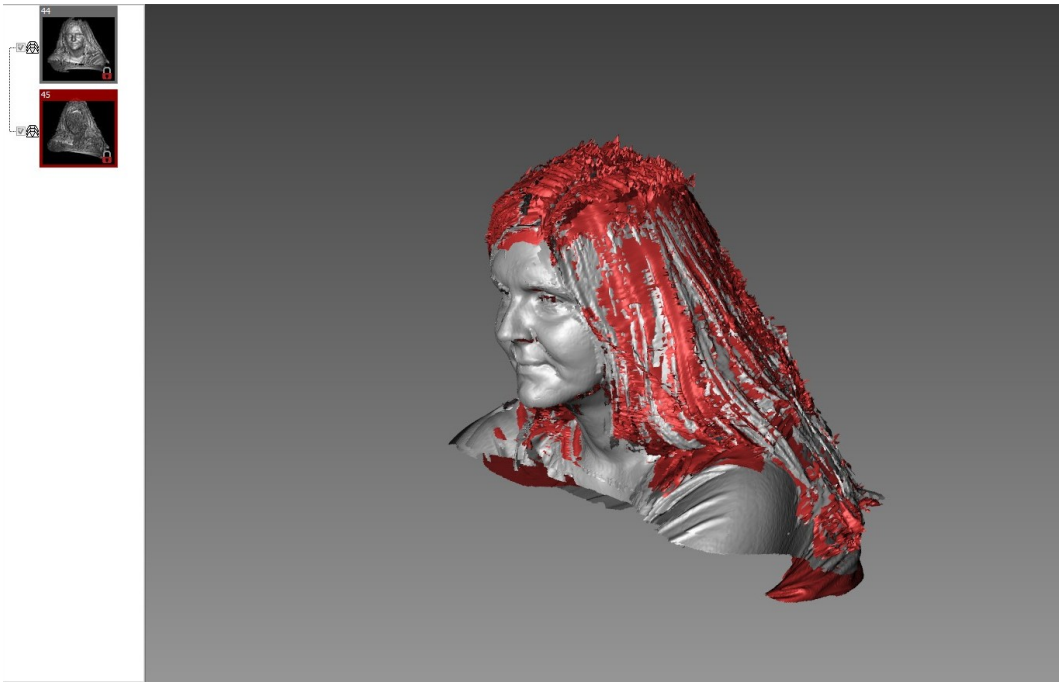
Sub-selection outline



Sub-selection highlighting



Facial features deleted



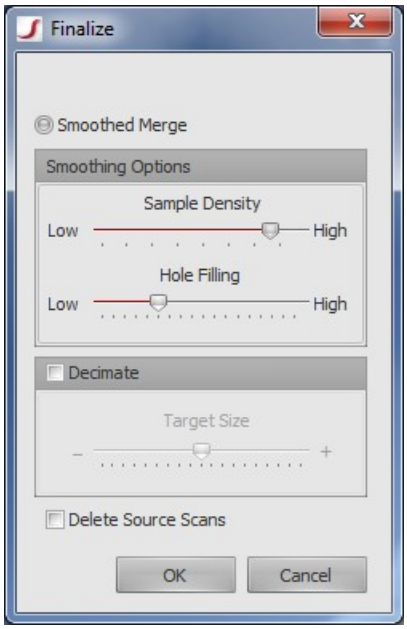
Facial features removed from the relaxed cleanup mesh

Secondary Combine

There should now be 2 combined meshes. Select both of them, load them, then click the Combine button. Once that process is complete, the mesh is ready to be finalized.

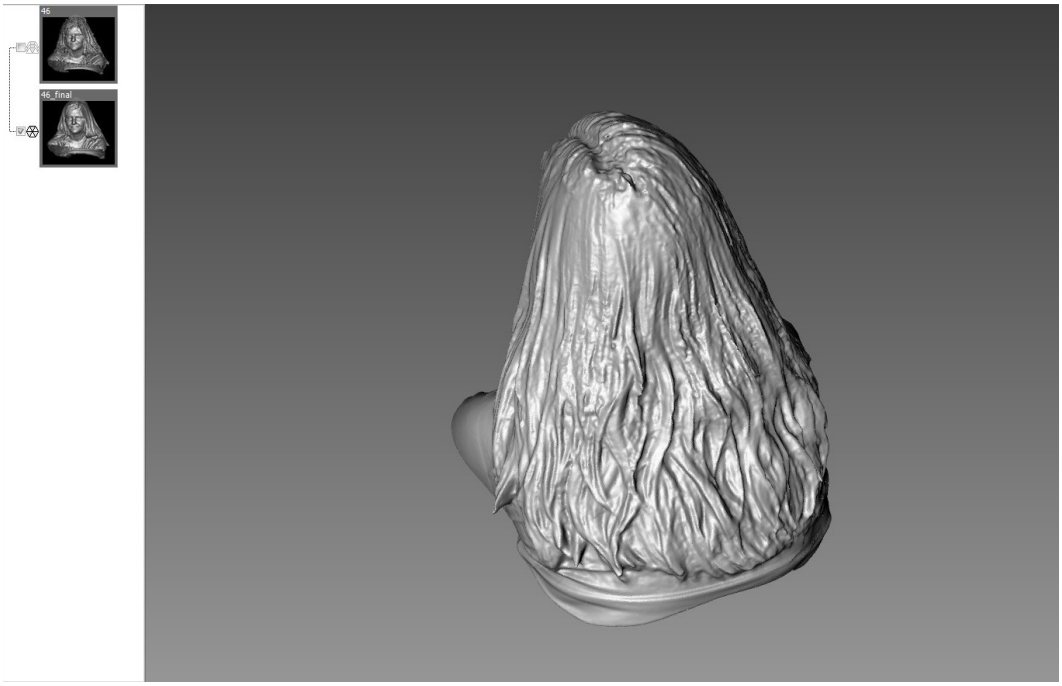
Finalize

Select the Combined mesh and click the Finalize button. Since **Preserve Overlapping Data** was enabled during the Combine phase, Finalize is limited to **Smoothed Merge** only (but this is what we want anyway). For smoother results, use a lower sample density.

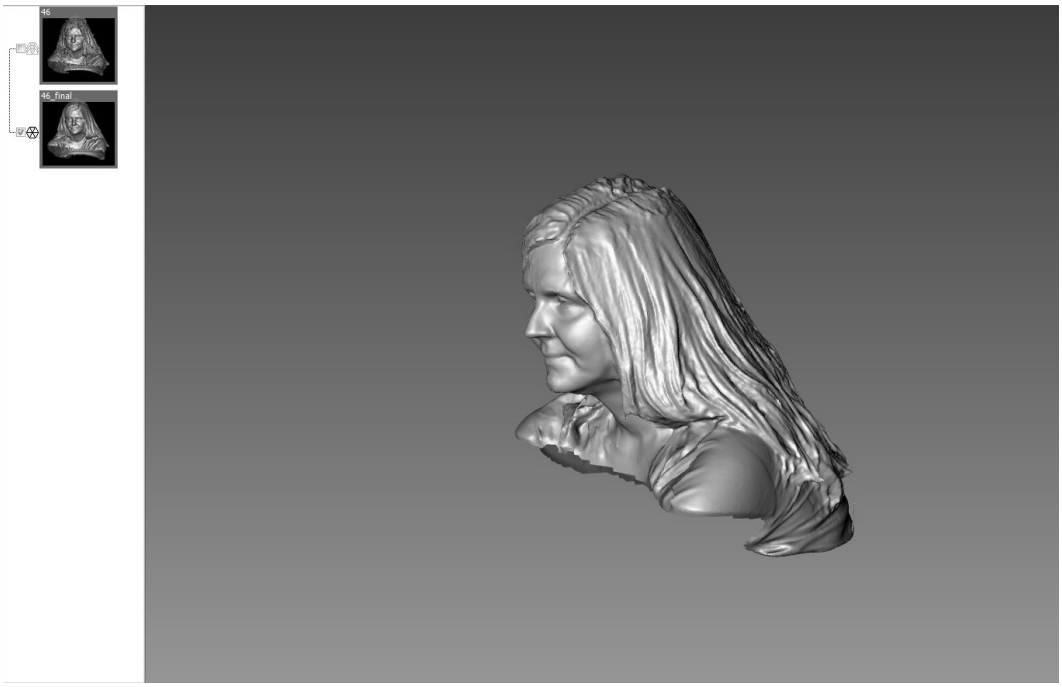


Suggested Finalize settings

Sample Results



Finalized mesh, rear view



Finalized mesh, front view

Other Notes

High-Contrast Scans

For high-contrast scans (light-skinned with dark hair or dark-skinned with light hair), the camera exposure used for capturing the skin will probably not work for capturing the hair. In these cases, either use FlexScan3D's HDR-mode scanning, or capture the skin and the hair separately, then merge the results afterwards.

Accuracy

While **High Sensitivity** mode is very useful for scanning hair, local accuracy is not guaranteed when it is enabled. The 3D reconstruction algorithm smooths over holes and cracks in the source data, and may fill in certain areas erroneously. As a result, it is not recommended for scanning typical surfaces.

API/SDK and Automation

The following sections describe the command line interface, the Rotary SDK, automation, and the FlexScan3D DLL interface.

FlexScan3D Command Line Interface

The FlexScan3D command line interface is accessible by calling FlexScan3D.exe on the command line or through the API DLL. A path that contains spaces must be enclosed in double quotes.

Example: Flexscan3d.exe script C:\Users\User\Documents\Scripts\MyScript.script\

To communicate from the script to the calling application, variables can be set in the script which can then be queried using the FS3D_ScriptQuery() function after the script has completed. For more about writing scripts, see *Automation* (page 127)

interactive

Usage

interactive [console]

Example

Flexscan3d.exe interactive

Starts FlexScan3D in Interactive mode to allow multiple commands to be sent without starting and stopping all the time. Including the word "console" causes FlexScan3D to open up a command-line like console for the user to type in. Otherwise the calling application is responsible for managing the input and output pipes. Programming languages that can read / write the standard console input / output streams can use this interface (just ensure that all commands end with a newline (\n) character).

To test the interface we recommend using "FlexScan3D.exe interactive console"

script

Usage

script <script file>

Example

Flexscan3d.exe script C:\Users\User\Documents\Scripts\MyScript.script

Runs a script file. The path and file name can contain Unicode characters.

scriptline

Usage

scriptline <script>

Example

Flexscan3d.exe scriptline "a = 4"

Runs a line of script. Note: backslashes in path names will need to be escaped to work properly (replace "\" with "\\").

scriptquery

Usage
scriptquery <script variable>

Example
Flexscan3d.exe scriptquery "a"

Returns the value of the specified script variable.

exit

Usage
exit

Example
Flexscan3d.exe exit

Requests a clean shutdown from FlexScan3D and the end of interactive mode.

FlexScan3D DLL Interface

The FlexScan3D DLL allows the calling application to start an interactive session via any programming or scripting language that supports the loading up of DLL interfaces. The DLL also supports callbacks that allow users to receive event and mesh data asynchronously.

You can find the include file, library, and example projects under C:\Program Files\LMI Technologies\FlexScan3D 3.3\SDK\ScanInterface after installation.

Callbacks

Callbacks are supported in FlexScan3D version 3.3.2.178 and higher.

Initializing FlexScan3D

For basic applications, FS3D_Init should be called at the beginning to start FlexScan3D and FS3D_Exit at the end to exit FlexScan3D. If the basic command I/O is handled separately through standard input/output, then FlexScan3D should be started separately and the application calls FS3D_Attach and FS3D_Detach to connect to the already running FlexScan3D instance. See below for more information on FS3D_Init and FS3D_Exit.

Either FS3D_Init or FS3D_Attach must be called before FS3D_RegisterCallback.

Registering Callbacks

Call FS3D_RegisterCallback to register callbacks. The callback to be registered is specified by name. The following callbacks are supported:

Callback Name	Description
ScanProcessed	Triggers immediately after a scan has been processed. Allows users to receive vertices or faces.
MotionDetected	Active when live scanning is enabled. Triggers when live scanning is enabled and motion is initially detected.
MotionStopped	Active when live scanning is enabled. Triggers when live scanning is enabled and motion is no longer detected.

Call this function multiple times to register multiple callbacks.

To avoid writing to disk during scanning, the Scanning_WriteToDisk advanced setting must be set to False before scanning.

Processing Callbacks

The callbacks return a FS3D handle, which is container of multiple items. The items contained depend on the callbacks. Users can use the FS3D_Get<Property type> functions to access the contained items.

Callback: ScanProcessed

Item Name	Item Type	Description
gridHeight	int	Height of the scan grid
gridWidth	int	Width of the scan grid
nVertices	int	Number of vertices
nFaces	int	Number of faces
nGrid	int	Number of grid indices (should be the same as nVertices)
vertices	double array	Vertices in X,Y,Z sequence
faces	int array	Triangle faces, with 3 vertex index values per face
grids	int array	Grid indices with a H,W sequence per vertex

Callback: MotionDetected

Item Name	Item Type	Description
scannerName	string	Name of the scanner which detected motion

Callback: MotionStopped

Item Name	Item Type	Description
scannerName	string	Name of the scanner which detected motion
motionDetected	int	Set to 1 if motion was detected, 0 if no motion was detected

Memory returned through various FS3D_Get*() function parameters are accessible throughout the scope of the callback function. There is no need to explicitly allocate or free memory.

Error Handling

Error FS3D_RESULT_WRONGTYPE will be returned if the item names or type do not match.

Callback Functions

See below.

C API command functions

int FS3D_Init(const char* a_PathName)

a_PathName
: Full path for the FlexScan3D executable.

Starts FlexScan3D in Interactive mode and connects to the input and output pipes to allow direct control over it. Must be called before any FS3D_Command calls. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure.

int FS3D_Command(const char* a_Command)

a_Command
: Any interactive command for FlexScan3D.

Sends the specified command to FlexScan3D and waits for it to reply that the task is completed. Returns FS3D_RESULT_OK if the command was successful, FS3D_RESULT_ERROR if the command was unsuccessful, and FS3D_RESULT_UNKNOWN if the command was not recognized.

int FS3D_CommandAsync(const char* a_Command)

a_Command
: Any interactive command for FlexScan3D.

Sends the specified command to FlexScan3D and returns immediately. Always returns FS3D_RESULT_OK.

int FS3D_AsyncResult()

Used to check the result of an asynchronous command from FS3D_CommandAsync(). While the command is still running, this function will return FS3D_RESULT_EXECUTING. For multi-threaded applications, a "while(FS3D_CommandAsync()==FS3D_RESULT_EXECUTING)" loop can be used to determine when the command has completed.

const char* FS3D_ScriptQuery(const char* a_Query)

a_Query
: A script variable name.

Queries FlexScan3D for the current value of a script variable. Returns the value of the variable as a string, or 0 (NULL) if the query failed. String variables are returned as-is, numbers are converted to a string representation before being returned, and nil values are returned as "nil".

Example: For script "a = 4 * 5", FS3D_ScriptQuery("a") returns "20".

int FS3D_Attach()

Opens a direct communication pipe with a running instance of FlexScan3D. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure.

int FS3D_Detach()

Closes the communication with FlexScan3D. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure.

int FS3D_RegisterCallback(const char* a_FunctionName, void* userContext, void (*a_Callback)(void* userContext, FS3D_Handle handle))

a_FunctionName
: The internal function name.

userContext
: Context pointer which will get passed through to the callback function.

a_Callback
: A pointer to a custom function.

Registers a callback with FlexScan3D. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure.

int FS3D_UnregisterCallback(const char* a_FunctionName)

a_FunctionName
: The internal function name.

Unregisters a callback with FlexScan3D. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure.

int FS3D_GetNumItems(const FS3D_Handle handle, int* numItems)

handle
: A handle to FlexScan3D data.

numItems
: Returns the number of items.

Gets the number of available named data items. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure.

int FS3D_GetItem(const FS3D_Handle handle, const int itemIndex, char** itemName, char** itemType)

handle
: A handle to FlexScan3D data.

itemIndex
: An index into the entire item list (0-based).

itemName
: Returns a text string containing the name of the item.

itemType
: Returns a text string containing the internal item type.

Gets information pertaining to the item at a particular index. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure.

int FS3D_GetString(const FS3D_Handle handle, const char* itemName, char** value)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

value
Returns the item text string.

Gets a text string. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not a text string.

int FS3D_GetDouble(const FS3D_Handle handle, const char* itemName, double* value)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

value
Returns the item number value.

Gets a 64-bit floating-point value. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not a 64-bit floating-point value.

int FS3D_GetFloat(const FS3D_Handle handle, const char* itemName, float* value)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

value
Returns the item number value.

Gets a 32-bit floating-point value. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not a 32-bit floating-point value.

int FS3D_GetInt(const FS3D_Handle handle, const char* itemName, int* value)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

value
Returns the item number value.

Gets a 32-bit integer value. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not a 32-bit integer value.

int FS3D_GetDoubleArray(const FS3D_Handle handle, const char* itemName, int* numValues, double** values)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

numValues
Returns the number of values in the array.

values
Returns the item values.

Gets an array of 64-bit floating-point values. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not an array of 64-bit floating-point values.

int FS3D_GetFloatArray(const FS3D_Handle handle, const char* itemName, int* numValues, float** values)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

numValues
Returns the number of values in the array.

values
Returns the item values.

Gets an array of 32-bit floating-point values. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not an array of 32-bit floating-point values.

int FS3D_GetIntArray(const FS3D_Handle handle, const char* itemName, int* numValues, int** values)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

numValues
Returns the number of values in the array.

values
Returns the item values.

Gets an array of 32-bit integer values. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not an array of 32-bit integer values.

int FS3D_GetByteArray(const FS3D_Handle handle, const char* itemName, int* numValues, unsigned char** values)

handle
A handle to FlexScan3D data.

itemName
The name of the desired item.

numValues
Returns the number of values in the array.

values
Returns the item values.

Gets an array of byte values. Returns FS3D_RESULT_OK on success, FS3D_RESULT_ERROR on failure, or FS3D_RESULT_WRONGTYPE if the item defined by itemName is not an array of byte values.

int FS3D_Abort()

Used to abort a command while it is being executed, which can be useful for long operations such as 360-degree rotary scans. This can be used in multi-threaded applications when using FS3D_CommandAsync(). Always returns FS3D_RESULT_OK.

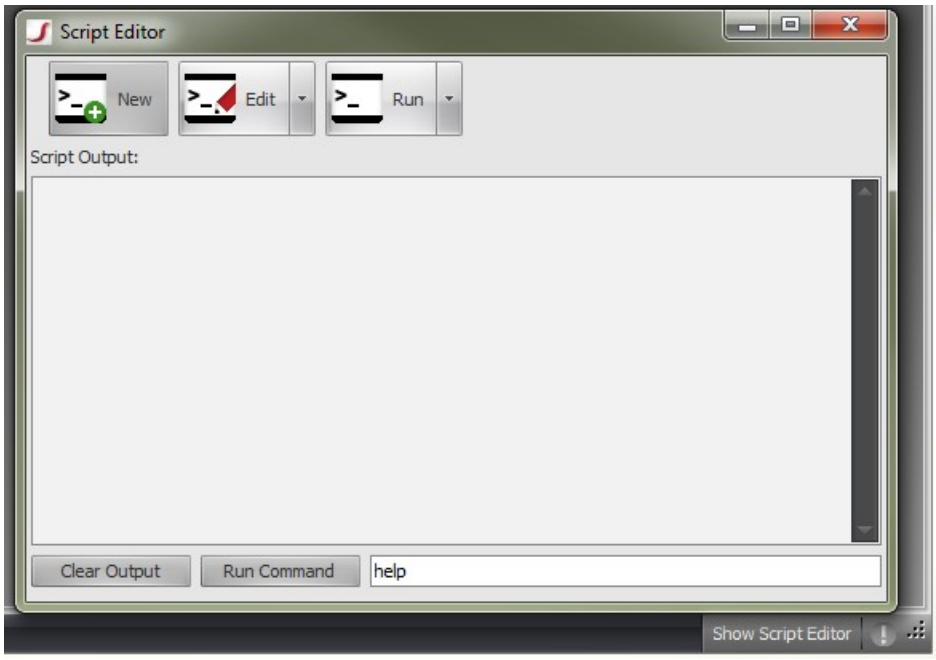
int FS3D_Exit()

Shuts down the communication pipes with FlexScan3D and terminates the application. Failure to call before terminating the calling application may result in data loss and/or error messages from FlexScan3D. Always returns FS3D_RESULT_OK.

Automation

You can automate certain features in FlexScan3D by using scripts. Scripts are written in the Lua scripting language. Lua allows for feature-rich scripting, including more advanced features such as loops and custom functions. You can control calibration, scanning, and rotary movement using scripts.

To open the Script Editor, click the **Show Script Editor** button at the bottom-right of the status bar. From here, you can create new scripts, edit them in a text editor (default is Notepad), and run the scripts. The log window will let you know if any errors occurred. To see a list of available functions, type "help" in the text box and press the Enter key (or click the **Execute** button).



See *Working with Scripts* (below) for more information on general script operations.

See *Functions* (page 135) for a complete list of the functions.

See *LUA Basics* (page 131) for basic information on LUA, and Examples (page 134) for some simple examples.

Working with Scripts

Running an Individual Command

Running an individual command is an easy way to quickly test a command without saving a new script.

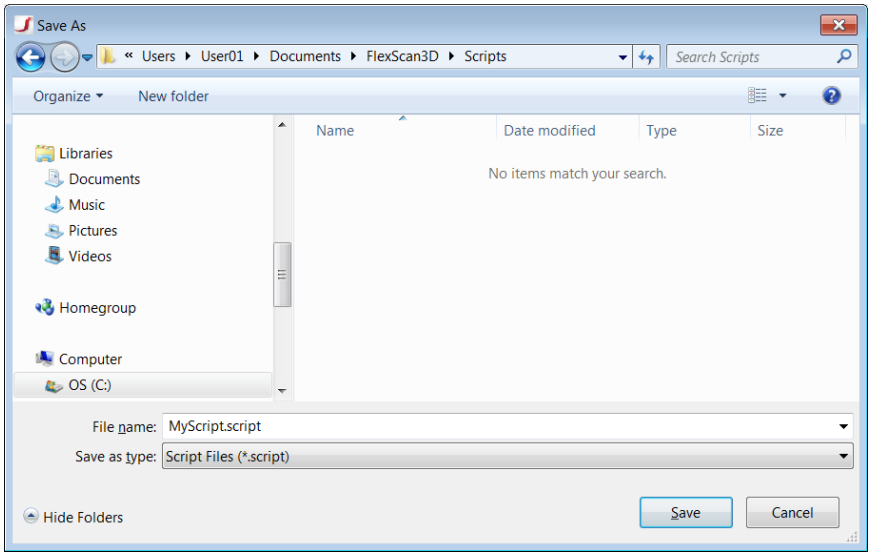
To run an individual command:

1. Type the script commands into the textbox at the bottom of the script editor.
2. Click **Execute** or press ENTER to execute the command.

Creating a New Script

To create a new script:

1. Click **New**.
   A **Save As** dialog will open. By default, the FlexScan3D*Scripts* folder will be used, but you can choose a different folder.

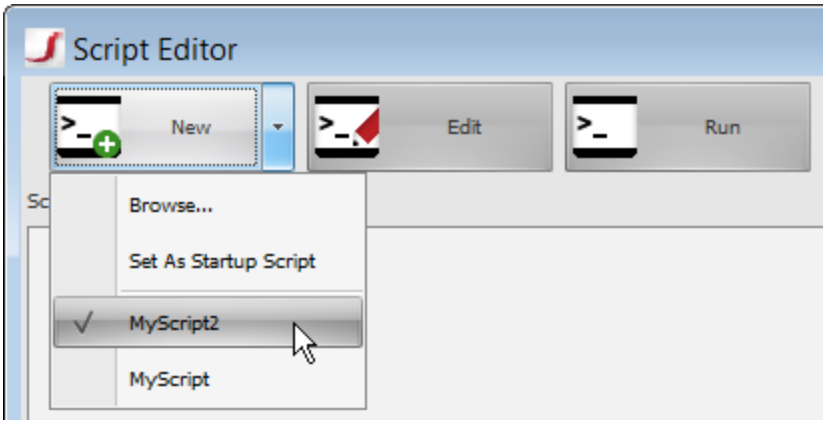


2. Type a file name for your script and click **Save**.
3. Edit your script in the text editor that will open.
4. Save your script when you have finished.
   The script will automatically be added to the list of scripts in the **New** drop-down.

Editing an Existing Script

To edit an existing script:

1. Click the drop-down arrow next to the **New** button and select a script to load it.
   You can also browse for a script by clicking **Browse...** in the drop-down.

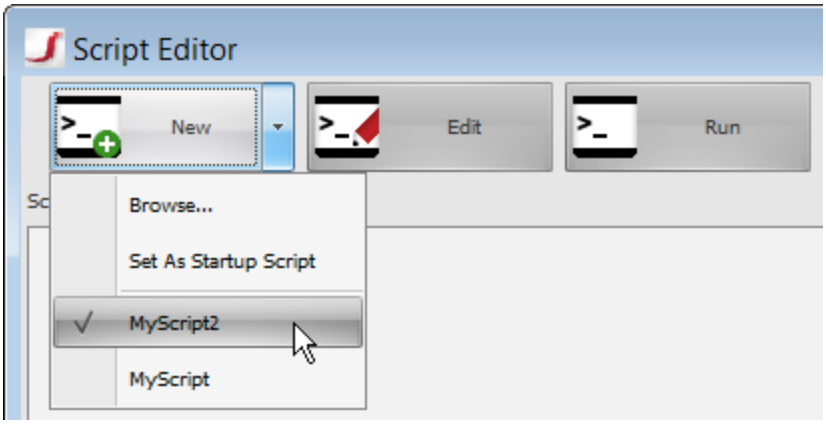


2. Click **Edit** to open the selected script in your default text editor.
   Each time you click the **Edit** button, the script currently selected in the **New** drop-down will open for editing.

Running a Script

To run a script:

1. Click the drop-down arrow next to the **New** button and select a script to load it.
   You can also browse for a script by clicking **Browse...** in the drop-down.



2. Click **Run** to run the selected script.
   Each time you click the **Run** button, the script currently selected in the **New** drop-down will run.
3. Check the **Script Output** window for logged output results.

> You can also run scripts from a command line. Refer to the script command-line arguments for more information (page 121).

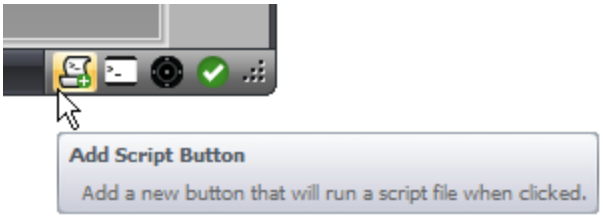
Setting Script Buttons and Hot Keys

You can add buttons to the FlexScan3D interface to launch functions quickly and easily. You can also set hot keys to launch scripts. See *Working with Scripts* (page 128) for more information on functions.

Script Buttons

To add a script button:

1. Click on the **Add Script Button** button in the lower right corner of FlexScan3D.



2. Navigate to the location of the script for which you want to add a button. A button will be added in the right side of the status bar. Clicking on the button will launch the script.

To remove a script button, right click on the button and choose **Remove Button**.

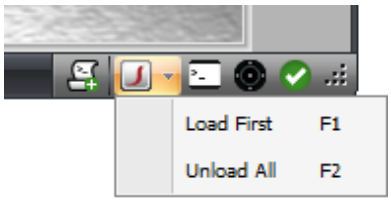
Hot Keys

To set hot keys to launch scripts:

1. Create a script that contains one or more calls to the SetHotKey(name, key, script, description) function.
2. Run the script.

Alternatively, you can run the function in the Script Editor.

A button and drop-down list will be added in the right side of the status bar. The drop-down list contains a list of all active hot keys. Each time you run a script that calls the SetHotKey(name, key, script, description) function, new hot keys will be added to the list.



Hot keys are not persistent. When you relaunch FlexScan3D, the hot keys will no longer be active. However, you can set a script that adds hot keys to run automatically whenever FlexScan3D launches.

To automatically adds hot keys whenever FlexScan3D launches:

1. Open the Script Editor.
2. Load the script that sets hotkeys using SetHotKey(name, key, script, description).
3. In the **New** drop-down list, select **Set As Startup Script**.

Each time that FlexScan3D is launched, the script will be run and the hot keys will be added.

To remove a hot key, use the UnsetHotKey(key) function.

LUA Basics

This section covers the basics of scripting with Lua and FlexScan3D. It is not meant to be a comprehensive guide. For more in-depth information, refer to the Lua reference manual here: Lua Language Reference

Debugging

To check the value of a variable, use the PrintValue() function.

```
a = 4
PrintValue( a )
```

That would display *PrintValue(variable="4")* in the log window. Note that some variables cannot be displayed, such as lists. To display the contents of a list, loop through and print the values by their index in the list (*PrintValue(myList[2])*).

Comments

To add a comment line into the script, place "--" as the first non-whitespace characters on a line.

```
-- this is a comment
```

Variables

Variables in Lua can be defined without needing to specify the type - Lua will automatically set the variable to the correct type as needed.

Global

The default scope in Lua is global. These variables are persistent, so setting a variable in one script will allow that variable to be accessed in another script.

```
a = true
a = 9.37
```

Local

For variables which will only be used within the context of the current script, the local keyword can be specified.

```
local a = true
local a = 9.37
```

Conditionals/Booleans

To check if a statement is true or false, we can use conditional statements. Keywords include if, then, else, elseif, and not.

```
a = 4 * 6
if a < 20 then
    PrintValue("a is smaller than 20")
else
    PrintValue("a is larger than 20")
end


a = false
```

```
if not a then
    PrintValue("a is false")
end
```

Loops

for

Runs for a specific number of iterations in the format "for i=startIndex,endIndex,increment". If the increment is absent, it is presumed to be 1.

```
for i=1,10,1 do
    PrintValue( i )
end
```

while

Runs until a condition is no longer true.

```
count = 10
while count > 0 do
    PrintValue( i )
    count = count - 1
end
```

nil

Some functions will return nil. This is a null value and usually indicates that the function failed.

Strings

Text and other variables can be concatenated using ".." characters.

```
timeString = "The time taken was: "
totalTime = 937
PrintValue( timeString .. totalTime .. " milliseconds" )
```

Lists

Several functions return lists. These lists are based on the .NET System.Collections.Generic.List<string>, and can be accessed as such.

```
groupList = GetAllGroups()
PrintValue( "There are " .. groupList.Count .. " scans in the current project." )
```

New lists can be created from scratch using the NewListString() function. Here is an example using Add() (to add a single string) and AddRange() (to add a list of strings).

```
myList = NewListString()
```

```
myList:Add( "Item 1" )
myList:Add( "Item 2" )
newList = NewListString()
newList.AddRange( myList )
PrintValue( newList.Count )
-- displays: PrintValue(variable="2")
```

Examples

You can copy the following examples into a blank .script file.

This very basic script is all you need to scan.

```
Scan()
```

This script uses a loop to do a 360° scan using a rotary and an angle increment of 90° (4 scans).

```
increment = 90
for i=1,360,increment do
    Scan()
    RotaryRotate(1, increment)
end
```

> Note that an easier way to accomplish the above example would be to use the Rotary360Scan() function.

Often, you will want to create a new project before each scan session. To ensure a unique name, you can use the built-in Lua date and time functions from the "os" module. The result is also checked to make sure the project was created.

```
-- creates a project in the format "2012-03-23[1332532193]"
result = NewProject(os.date("%Y-%m-%d") .. "[" .. os.time() .. "]" )
if not result then
    PrintValue( "The project was not created successfully." )
    return
end
```

Here's a more advanced example of running several capture-only scans with a pause in between to allow a user time to change the position of the object before the next scan - useful if a rotary is not available. Afterwards, the script loops through all of the scan groups to process the data and create a mesh. The built-in "os.clock()" function is also used to track the amount of time it took to complete the script.

```
-- start the timer
startTime = os.clock()
-- enable capture-only mode
Set("Scanning_Generation_Type", 2)
-- capture 10 scans and add them to a list
groupList = NewListString()
```

```
for i=1,10 do
    result, newGroups = Scan()
    -- check to make sure the scan went OK - if not, we stop the script execution
    if not result then
       -- show a custom error message
       PrintValue( "Scan #" .. i .. " failed." )
       return
    end
    -- add the new groups to the list (usually there will only be one, but there could be
more for multi-scanner setups)
    groupList:AddRange( newGroups )
    -- pause for 7 seconds to allow the object to be moved/rotated
    Wait(7)
end
-- get the final count of groups
groupCount = groupList.Count
-- loop through and process the new scan groups into meshes - note that list indexing
starts at 0
for i=0,groupCount-1 do
    result = Process( groupList[i], 0 )
    -- check the result
    if not result then
       PrintValue( "Failed to process group: " .. groupList[i] );
       return
    end
end
totalTime = os.clock() - startTime
PrintValue( "Script completed successfully. Time taken: " .. totalTime .. " seconds" )
```

Functions

> Functions that pass *file* and *folder* names support Unicode in these names, letting you use localized names containing non-ASCII characters.

Calibrating

AddScanner(scannerID)

Adds a scanner.

Parameters

scannerID: The ID of the scanner.

Example

AddScanner(GetScannerIDs()[0])

Returns

True if scanner was added successfully.

AddScannerByType(scannerType, serialNumber)

Adds a scanner by a given type.

Parameters

scannerType: The type of the scanner. Current valid values are "HDI" or "HDI Advance".
serialNumber: The serial number of the scanner.

Example

AddScannerByType("HDI", 12078)

Returns

True if scanner was added successfully, along with the name of the scanner added.

AutoSetExposure()

Automatically selects the exposure for all scanners and enables HDR if necessary.

Returns

true if successful, otherwise false, as well as a boolean indicating whether HDR is recommended.

ExportScanner(scannerName, fileName, preserveImages)

Parameters

scannerName: The name of a scanner.
fileName: Name of file to export scanner to. Supports Unicode characters.
preserveImages: true if calibration images should be preserved, otherwise false to minimize space.

Example

ExportScanner("Scanner-001", "C:\Exports\Scanner.7z", true)

Returns

true if successful, otherwise false.

GetPattern(scannerName)

Gets the focus pattern of the named scanner.

Parameters

scannerName: The name of a scanner.

Example

GetPattern("Scanner-001")

Returns

The name of the current focus pattern.

GetScannerIDs()

Gets a list of all available scanner IDs.

Returns

A list of scanner IDs.

GetScannerIndexFromName(scannerName)

Gets the internal scanner index based on its name.

Parameters

scannerName: The name of a scanner.

Example
GetScannerIndexFromName("Scanner-001")

Returns
The index of the scanner in the current calibration, or -1 if it does not exist.

GetScannerNameFromIndex(scannerIndex)

Gets the name of the scanner based on its internal index in the current calibration.

Parameters
scannerIndex: A zero-based index into the list of scanners.

Example
GetScannerNameFromIndex(0)

Returns
The scanner name if successful, nil if unsuccessful.

HDI_Advance_CalculateDelayTiming(scannerName)

Auto calculate the delay timing values for a scanner.

Parameters
scannerName: The name of the scanner.

Example
HDI_Advance_CalculateDelayTiming(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_Advance_CalculateWhiteBalance(scannerName)

Auto calculate the white balance for a scanner.

Parameters
scannerName: The name of the scanner.

Example
HDI_Advance_CalculateWhiteBalance(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_Advance_Calibrate(scannerName)

Calibrates a scanner based on its calibration images.

Parameters
scannerName: The name of the scanner.

Example
HDI_Advance_Calibrate(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_Advance_CaptureCalibrationImage(scannerName)

Captures a calibration image.

Parameters
scannerName: The name of the scanner.

Example
HDI_Advance_CaptureCalibrationImage(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_Advance_DeleteCalibration(scannerName)

Deletes all calibration data in a scanner.

Parameters
scannerName: The name of the scanner.

Example
HDI_Advance_DeleteCalibration(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_Advance_DeleteCalibrationImage(scannerName, imageID)

Delete a calibration image.

Parameters
scannerName: The name of the scanner.
imageID: The id of the image to delete.

Example
HDI_Advance_DeleteCalibrationImage(GetScannerNameFromIndex(0), 15)

Returns
true if successful, false if unsuccessful.

HDI_Calibrate(scannerName)

Calibrates a scanner based on its calibration images.

Parameters
scannerName: The name of the scanner.

Example
HDI_Calibrate(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_CaptureCalibrationImage(scannerName)

Captures a calibration image.

Parameters
scannerName: The name of the scanner.

Example
HDI_CaptureCalibrationImage(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_DeleteCalibrationImage(scannerName, imageID)

Deletes a calibration image.

Parameters
scannerName: The name of the scanner.
imageID: The id of the image to delete.

Example
HDI_DeleteCalibrationImage(GetScannerNameFromIndex(0), 15)

Returns
true if successful, false if unsuccessful.

ImportScanner(fileName)

Imports a scanner from a previously exported scanner file.

Parameters
fileName: Name of file to import scanner from. Supports Unicode characters.

Example
ImportScanner("C:\Exports\Scanner.7z")

Returns
true if successful, otherwise false.

IsScannerEnabled(scannerName)

Get a scanner's enabled state. Note that the scanner must be connected first using ScannerConnnect().

Parameters
scannerName: The existing scanner name.

Example
IsScannerEnabled("Scanner1")

Returns
true if enabled, otherwise false.

RemoveScanner(scannerName)

Removes a scanner from the current calibration and deletes the local scanner data.

Parameters
scannerName: The existing scanner name.

Example
RemoveScanner("Bottom")

RemoveScanners()

Deletes the current scanner configuration, including all associated local files.

RenameScanner(scannerName, newScannerName)

Renames a scanner in the current calibration. Note that the calibration must have been created as type "Multi".

Parameters
scannerName: The existing scanner name.

newScannerName: The new scanner name.

Example
RenameScanner("Scanner1", "Bottom")

Returns
true if successful, false if unsuccessful.

SetScannerEnabled(scannerName, enabled)

Enable or disable a scanner.

Parameters
scannerName: The name of the scanner.
enabled: true if scanner should be enabled, false for disabled.

Example
SetScannerEnabled("Scanner1", true)

Returns
true if successful, otherwise false.

ShowPattern(scannerName, patternName)

Sets the projector pattern.

Parameters
scannerName: The name of a scanner.
patternName: The name of the pattern to project.

Example
ShowPattern("Scanner-001", "Focus")

Returns
true if successful, otherwise false.

StartVideo(scannerName)

Starts live video for a scanner.

Parameters
scannerName: The name of a scanner.

Example
StartVideo("Scanner-001")

Returns
true if successful, otherwise false.

StopVideo(scannerName)

Stops live video for a scanner.

Parameters
scannerName: The name of a scanner.

Example
StopVideo("Scanner-001")

Returns
true if successful, otherwise false.

TestCalibration(scannerName)

Calculate reprojection error for a specific board.

Parameters
scannerName: The name of a scanner.

Example
TestCalibration("Scanner-001")

Returns
true if successful, false if unsuccessful, followed by the scanner and board reprojection error values respectively (in microns).

Cameras

AttachVideoWindow(scannerName, cameraID, windowHandle)

Attaches the live video feed for the specified camera to an application window handle.

Parameters
scannerName: The name of a scanner.
cameraID: The camera ID.
windowHandle: The application window handle.

Example
AttachVideoWindow("Scanner-001", 0, 51347692)

Returns
true if successful, false if unsuccessful.

DetachVideoWindow(scannerName, cameraID)

Detaches the live video feed for the specified camera.

Parameters
scannerName: The name of the scanner to get the camera from.
cameraID: The camera ID.

Example
DetachVideoWindow("Scanner-001", 0)

Returns
true if successful, otherwise false.

Configuration

HDI_AutoUpdateScanner(scannerName)

Updates to the latest scanner firmware included with this software distribution.

Parameters
scannerName: The name of the scanner.

Example
HDI_AutoUpdateScanner(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_CheckScanner(scannerName)

Determines whether the scanner firmware needs to be updated.

Parameters

scannerName: The name of the scanner.

Example

HDI_CheckScanner(GetScannerNameFromIndex(0))

Returns

0 = OK
-1 = Scanner Not Found
-2 = Scanner Not Discovered
-3 = Network Configuration Required
-4 = Firmware Update Required
-5 = Model Unsupported

HDI_GetFirmwareVersion(scannerName)

Gets the firmware version of the scanner.

Parameters

scannerName: The name of the scanner.

Example

HDI_GetFirmwareVersion(GetScannerNameFromIndex(0))

Returns

true if successful, false if unsuccessful, as well as the firmware version of the scanner.

HDI_GetScannerHealth(scannerName)

Gets comprehensive details regarding the current state of the scanner.

Parameters

scannerName: The name of the scanner.

Example

HDI_GetScannerHealth(GetScannerNameFromIndex(0))

Returns

true if the scanner exists, false if the scanner does not exist, as well as the following details:
- temperature (degrees C)
- memory used (bytes)
- memory capacity (bytes)
- storage used (bytes)
- storage capacity (bytes)
- CPU used (%)
- netOutUsed
- netOutCapacity
- uptime (seconds)
- current state (1: Conflict, 2: Ready, 3: Running)
- camera frame error count
- camera frame drop count
- messageInDrops
- messageOutDrops.

HDI_GetScannerModel(scannerName)

Gets the internal model name of a scanner.

Parameters
scannerName: The name of the scanner.

Example
HDI_GetScannerModel(GetScannerNameFromIndex(0))

Returns
true if the scanner exists, false if the scanner does not exist, as well as the model name of the scanner.

HDI_GetScannerOptionCode(scannerName)

Gets the internal option code of a scanner.

Parameters
scannerName: The name of the scanner.

Example
HDI_GetScannerOptionCode(GetScannerNameFromIndex(0))

Returns
true if the scanner exists, false if the scanner does not exist, as well as the model name of the scanner.

HDI_IsUpdateRequired(scannerName)

Determines whether or not the scanner firmware needs to be updated.

Parameters
scannerName: The name of the scanner.

Example
HDI_IsUpdateRequired(GetScannerNameFromIndex(0))

Returns
true if the scanner is connected and needs a firmware update. false in all other cases.

HDI_UpdateScanner(scannerName, firmwarePath)

Updates the scanner using the firmware file at the specified path.

Parameters
scannerName: The name of the scanner.
firmwarePath: The path to the firmware file. Supports Unicode characters.

Example
HDI_UpdateScanner(GetScannerNameFromIndex(0), C:\firmware\upgrade.dat)

Returns
true if successful, false if unsuccessful.

General

DisplayString(text)

Displays a string in the status bar and/or the log file.

Parameters
text: The string to display.

Example
DisplayString("Scanners are ready.")

Get(settingName)

Gets an application setting value by name.

Parameters
settingName: An application setting name.

Example
Get("Calibrating_MinCalibrationImages")

Returns
The value of the setting as text if successful, nil if unsuccessful.

NewListString()

Utility function to allow for the creation of lists compatible with other FlexScan3D script functions. Items (such as scan group IDs) can be added to the list using myListName:Add("Scan01").

Returns
An empty list.

PrintValue(variable)

Displays the current variable value in the log. Useful for debugging purposes.

Parameters
variable: Any variable returned from a script function.

Example
PrintValue(groupID)

QuietModeOff()

Disables quiet mode when running scripts from the script editor.

QuietModeOn()

Enables quiet mode when running scripts from the script editor. This suppresses pop-up messages that only has an OK button. Yes/No prompts will still appear. Note that each QuietModeOn() call must be paired with QuietModeOff().

QuietModeStackSize()

Returns the size of the quiet mode stack. This can be used to determine how many times QuietModeOff () must be called to completely disable quiet mode.

Returns

size of quiet mode stack.

Run(fileName, arguments)

Launches an external application and waits until it closes.

Parameters

fileName: The complete path to an application. Supports Unicode characters.

arguments: Any arguments to the application.

Example

Run("C:\Windows\notepad.exe", "C:\My Scripts\My Scan.script")

Returns

true if successful, false if unsuccessful.

Set(name, value)

Sets an application setting value by name.

Parameters

name: An application setting name.

arguments: An application setting value, as text.

Example

Set("Calibrating_MinCalibrationImages", "10")

Returns

true if successful, false if unsuccessful.

SetHotKey(name, key, script, description)

Sets a hotkey to trigger a script sequence.

Parameters

name: The display name of the hotkey.

key: The hotkey to assign.

script: The script to run.

description: Further information that describes this hotkey function.

Example

SetHotKey("Scan", "Ctrl-F1", "Scan()", "Perform a scan and process.")

Returns

true if successful, otherwise false.

UnsetHotKey(key)

Unsets a hotkey.

Parameters

key: The hotkey to unassign.

Example

UnsetHotKey("Ctrl-F1")

Returns

true if successful, otherwise false.

Wait(seconds)

Pauses the script for a specified delay.

Parameters
seconds: The delay time (in seconds).

Example
Wait(1.33)

Groups

Copy(groupID, suffix)

Makes a copy of a scan group.

Parameters
groupID: A scan group ID.
suffix: A suffix to append to the new group ID.

Example
Copy("1", "copy")

Returns
true if successful, false if unsuccessful, as well as the new group ID.

DeleteAllGroups()

Deletes all scan groups in the current project.

DeleteGroup(groupName)

Deletes a scan group.

Parameters
groupName: The name of a scan group.

Example
DeleteGroup("Scan01")

DeleteSelectedGroups()

Deletes all selected scan groups in the current project.

DeselectAll()

Deselects all scan groups.

DeselectGroup(groupID)

Deselects a scan group.

Parameters
groupID: A scan group ID.

Example
DeselectGroup("1")

GetAllGroups()

Gets a list of all scan groups in the current project.

Returns
A list of all scan groups.

GetGroupAliasFromID(gid)

Gets the group alias its corresponding ID.

Parameters
gid: A group ID.

Example
GetGroupAliasFromID("1")

Returns
The group alias if found, nil if not found.

GetGroupIDFromAlias(alias)

Gets the group ID its corresponding alias.

Parameters
alias: A group ID.

Example
GetGroupIDFromAlias("Side Scan 1")

Returns
The group ID if found, nil if not found.

GetSelectedGroups()

Gets a list of selected scan groups.

Returns
A list of selected scan groups.

IsGroupLoaded(groupID)

Returns whether a scan group is loaded.

Parameters
groupID: A scan group ID.

Example
IsGroupLoaded("1")

Returns
true if the specified group is loaded, otherwise false.

IsGroupLocked(groupID)

Returns whether a scan group is locked.

Parameters
groupID: A scan group ID.

Example
IsGroupLocked("1")

Returns
true if the specified group is locked, otherwise false.

IsGroupSelected(groupID)

Returns whether a scan group is selected.

Parameters
groupID: A scan group ID.

Example
IsGroupSelected("1")

Returns
true if the specified group is selected, otherwise false.

LoadAll()

Loads all scan groups.

LoadGroup(groupID)

Loads a scan group.

Parameters
groupID: A group ID.

Example
LoadGroup("1")

Returns
true if successful, false if unsuccessful.

LoadSelected()

Loads all selected scan groups.

Returns
true if successful, false if unsuccessful.

LockGroup(groupID)

Locks a scan group.

Parameters
groupID: A scan group ID.

Example
LockGroup("1")

SaveGroup(groupID)

Saves a scan group.

Parameters
groupID: A scan group ID.

Example
SaveGroup("1")

Returns
true if successful, false if unsuccessful.

SaveGroups(groupIDs)

Saves a list of scan groups.

Parameters

groupIDs: A list of scan group IDs.

Example
SaveGroups({ "1", "2", "3", "4" })

Returns
true if successful, false if unsuccessful.

SelectAll()

Selects all scan groups.

SelectGroup(groupID)

Selects a scan group.

Parameters
groupID: A scan group ID.

Example
SelectGroup("1")

SetGroupAlias(groupID, alias)

Sets the alias of a group.

Parameters
groupID: A scan group ID.
alias: The new group alias.

Example
SetGroupAlias("1", "Side Scan 1")

Returns
true if successful, false if unsuccessful.

UnloadAll()

Unloads all scan groups.

UnloadGroup(groupID)

Unloads a scan group.

Parameters
groupID: A scan group ID.

Example
UnloadGroup("1")

UnloadSelected()

Unloads all selected scan groups.

UnlockGroup(groupID)

Unlocks a scan group.

Parameters
groupID: A scan group ID.

Example

UnlockGroup("1")

Networking

HDI_AutoConfigureNetwork(scannerName)

Automatically configures the network address of the scanner (and network adapter, if necessary).

Parameters
scannerName: The name of the scanner.

Example
HDI_AutoConfigureNetwork(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful.

HDI_GetScannerAddress(scannerName)

Changes the network address of the scanner.

Parameters
scannerName: The name of the scanner.

Example
HDI_GetScannerAddress(GetScannerNameFromIndex(0))

Returns
true if successful, false if unsuccessful, as well as the IP address, subnet mask, gateway, and DHCP state (true/false).

HDI_SetScannerAddress(scannerName, ipAddress, subnetMask, gateway, useDHCP)

Changes the network address of the scanner.

Parameters
scannerName: The name of the scanner.
ipAddress: A network IP address.
subnetMask: A network subnet mask.
gateway: A network gateway IP address.
useDHCP: Set to true to enable DHCP on the scanner (useful when the scanner is connected to a router, for example), or false to use a static IP.

Example
HDI_SetScannerAddress(GetScannerNameFromIndex(0), 192.168.61.51, 255.0.0.0, 192.168.61.1, false)

Returns
true if successful, false if unsuccessful.

Processing

Align()

Aligns one or more unlocked meshes to any locked meshes. Meshes must be loaded. Note that the alignment method used is based on the Processing_Alignment_Type setting.

Returns

true if successful, false if unsuccessful.

AlignFastICP(calibDir)

Fast ICP all selected.

ClipGroup(groupID, xMin, xMax, yMin, yMax, zMin, zMax)

Removes all points outside of the specified clipping boundaries. Please note that transformations are ignored, and combined groups are not supported.

Parameters

groupID: A scan group ID.
xMin: A minimum value for X.
xMax: A maximum value for X.
yMin: A minimum value for Y.
yMax: A maximum value for Y.
zMin: A minimum value for Z.
zMax: A maximum value for Z.

Example

ClipGroup("10", -50.0, 50.0, -50.0, 50.0, -50.0, 50.0)

Returns

true if successful, false if unsuccessful.

Combine(groups)

Combines multiple groups of one or more meshes into a single group.

Parameters

groups: A list of one or more scan group IDs.

Example

Combine({ "1", "2", "3", "4" })

Returns

true if successful, false if unsuccessful, as well as the combined group ID.

Decimate(groupList)

Applies mesh decimation to all selected scan groups. The decimated resolution is defined by the Processing_MeshDecimationResolution application setting, where 25.0 results in approximately 25% of the original number of vertices, and 100.0 = 100% of the original (no decimation).

Returns

true if successful, false if unsuccessful.

Deviation(referenceGroupID, targetGroupID, exportFile, pointIDs, targetPoints)

Performs a deviation between 2 meshes and exports the results to a file. This only works on single scans, and assumes the scans are already aligned.

Parameters

referenceGroupID: The reference group ID.
targetGroupID: The target group ID.
exportFile: The file name for the exported ASCII deviation data. Supports Unicode characters.

pointIDs: Whether or not to include the point IDs in the exported data.
targetPoints: Whether or not to include the target points in the exported data.

Example
Deviation("10", "11", C:\Deviations\dev.txt, false, false)

Returns
true if successful, false if unsuccessful, as well as the maximum, minimum, maximum average, minimum average, absolute average, and standard deviation values, respectively.

ErodeSelected()

Applies mesh erosion to all selected scan groups. The number of erosion passes is defined by the Processing_Erosion application setting.

Returns
true if successful, false if unsuccessful.

Export(outputDir, ext)

Exports all loaded groups except for combined groups.

Parameters
outputDir: The complete path to an existing output directory. Supports Unicode characters.
ext: The exported file type/extension. Can be ".3d3", ".asc", ".obj", ".ply", ".stl", ".png", or ".dep". Note that when using a texture camera, OBJ, PLY, and 3D3 types will also export the texture images, and in the case of OBJ, accompanying ".mtl" files.

Example
Export("C:\Exported Meshes\", ".obj")

Returns
A list of complete paths to the exported meshes. Any mesh which failed to export will result in a blank entry ("").

ExportGroups(outputDir, ext, groups)

Exports all specified groups.

Parameters
outputDir: The complete path to an existing output directory. Supports Unicode characters.
ext: The exported file type/extension. Can be ".3d3", ".asc", ".obj", ".ply", ".stl", ".png", or ".dep". Note that when using a texture camera, OBJ, PLY, and 3D3 types will also export the texture images, and in the case of OBJ, accompanying ".mtl" files.
groups: A list of one or more scan group IDs.

Example
ExportGroups("C:\Exported Meshes\", ".obj", { "1", "2", "3", "4" })

Returns
A list of complete paths to the exported meshes. Any mesh which failed to export will result in a blank entry ("").

Finalize(groups)

For each group, creates a unified mesh based on one or more combined meshes. This is affected by the Processing_Merging_Type setting: if it is 0, then the existing data is used as-is; if it is 1, then the data is

resampled based on Poisson surface reconstruction. Also note that if the Processing_Finalize_ RemoveSourceData setting is false, a new scan group will be created for each group ID.

Parameters
groups: A list of one or more scan group IDs.

Example
Finalize({ "1", "2", "3", "4" })

Returns
true if successful, false if unsuccessful, as well as a list of one or more output group IDs.

FineAlign(groups, type)

Runs a thorough global alignment. Note that the meshes must already be loaded.

Parameters
groups: A list of one or more scan group IDs.
type: Alignment type. Must be 3.

Example
FineAlign({ "1", "2", "3", "4" }, 3)

Returns
true if successful, false if unsuccessful.

GetMarkers(groupID)

Gets the markers for the specified scan group.

Parameters
groupID: A scan group ID.

Example
GetMarkers("10")

Returns
true if successful, false if unsuccessful, as well as a list of marker points (untransformed). Each item in the list consists of an X/Y/Z coordinate, stored as a 3-value array.

GetMeshDetails(groupID)

Gets information about the 3D mesh represented by the groupID.

Parameters
groupID: A scan group ID.

Example
GetMeshDetails("1")

Returns
true if successful, false if unsuccessful, as well as the number of vertices, faces, and markers, respectively.

GetTransformation(groupID)

Gets the current transformation of the specified scan group.

Parameters
groupID: A scan group ID.

Example

GetTransformation("10")

Returns

A 4x4 3D transformation matrix, returned as an array containing all 16 values, or nil if an error occurred.

Import(fileName, markers)

Imports a mesh.

Parameters

fileName: The complete path to a mesh file. Acceptable formats are 3D3, OBJ, and STL. Supports Unicode characters.

markers: If this flag is true, then the imported points will be treated as a set of markers instead of a mesh.

Example

Import("C:\Models\Car.obj", false)

Returns

The new group ID if successful, nil if unsuccessful.

MeshClean()

Cleans up.

NewTransformationMatrix()

Creates a transformation matrix.

Returns

A 4x4 3D transformation matrix, returned as an array containing 16 values.

Process(groupID, generateType)

Generates 3D data from 2D scan images.

Parameters

groupID: The scan group ID.

generateType: 0 for Mesh, 1 for Points.

Example

Process("16", 0)

Returns

true if successful, false if unsuccessful.

ProcessGroups(groups, generateType)

Generates 3D data from 2D scan images.

Parameters

groups: A list of one or more scan group IDs.

generateType: 0 for Mesh, 1 for Points.

Example

ProcessGroups({ "1", "2", "3", "4" }, 0)

Returns

true if successful, false if unsuccessful.

ReprojectUVTexture(referenceID, targetID, txtWidth, txtHeight)

Projects the texture from one group onto another.

Parameters

referenceID: A source scan group ID. The mesh must contain texture coordinates.
txtWidth: The width of the texture.
txtHeight: The height of the texture.

Returns

true if successful, false if unsuccessful.

SetCleanUpType(cleanUpType)

Set the type of clean-up on mesh generation.

Parameters

cleanUpType: A type of clean-up on meshing: 0 - None, 1 - Relaxed, 2 - Standard, 3 - High, 4 - Extreme.

Example

SetCleanUpType(1)

SetPresetTransform(groups)

Set preset alignment based on specified prealigned scans. Assumes each group has only 1 scan.

Returns

true if successful, false if unsuccessful.

SetTransformation(groupID, matrix)

Sets the transformation of the specified scan group.

Parameters

groupID: A scan group ID.
matrix: A 4x4 matrix representing a 3D transformation.

Example

SetTransformation("10", {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1})

Returns

true if successful, false if unsuccessful.

SmoothSelected()

Applies mesh smoothing to all selected scan groups. The number of smoothing passes is defined by the Processing_Smoothing application setting.

Returns

true if successful, false if unsuccessful.

UnCombine(groupID)

Splits a combined group into its component groups. The output groups only contain a single mesh each.

Parameters

groupID: A combined group ID.

Example
UnCombine("10")

Returns
true if successful, false if unsuccessful, as well as a list of one or more output group IDs.

Projector

HDI_Advance_SetProjectorBrightness(scannerName, brightness)

Sets the brightness of the projector for a specific scanner.

Parameters
scannerName: The name of a scanner in the current calibration.
brightness: A brightness level from full black (0.0) to full white (1.0).

Example
HDI_Advance_SetProjectorBrightness("AdvancePlugin::1", 0.85)

HDI_Advance_ShowImage(scannerName, imageFileName)

Displays an image on the projector.

Parameters
scannerName: The name of a scanner in the current calibration.
imageFileName: The name of an image to display on the projector. Supports Unicode characters.

Returns
true if successful, false if unsuccessful.

Projects

CloseProject()

Closes the current project.

Returns
true if successful, false if unsuccessful.

DeleteProject(name)

Deletes a project by name.

Parameters
name: A project name. Supports Unicode characters.

Example
DeleteProject("My Project")

Returns
true if successful, false if unsuccessful.

DeleteProjectPath(dir)

Deletes a project by complete path.

Parameters
dir: A project directory. Supports Unicode characters.

Example
DeleteProjectPath("C:\Users\User\Documents\FlexScan3D\Projects\My Project\")

Returns
true if successful, false if unsuccessful.

GetProjectNames()

Gets a list of project names.

GetProjectsPath()

Gets the projects directory (the base path where projects are stored).

LoadProject(name)

Loads a project by name.

Parameters
name: A project name. Supports Unicode characters.

Example
LoadProject("My Project")

Returns
true if successful, false if unsuccessful.

LoadProjectPath(dir)

Loads a project by complete path.

Parameters
dir: A project directory. Supports Unicode characters.

Example
LoadProjectPath("C:\Users\User\Documents\FlexScan3D\Projects\My Project\")

Returns
true if successful, false if unsuccessful.

NewProject(name)

Creates a new project by name.

Parameters
name: A project name. Supports Unicode characters.

Example
NewProject("My Project")

Returns
true if successful, false if unsuccessful.

NewProjectPath(dir)

Creates a new project by complete path.

Parameters
name: A project directory. Supports Unicode characters.

Example
NewProjectPath("C:\Users\User\Documents\FlexScan3D\Projects\My Project\")

Returns
true if successful, false if unsuccessful.

SaveProject()

Saves the current project.

Returns
true if successful, false if unsuccessful.

Rotary

GetNumMotors()

Returns
True if rotary is connected and number of motors.

IsRotaryCalibrated(scannerName)

Determines whether or not a rotary is calibrated.

Returns
true if a rotary is calibrated, false if a rotary is not calibrated.

IsRotaryConnected()

Determines whether or not a rotary is connected.

Returns
true if a rotary is connected, false if a rotary is not connected.

Rotary360Scan(motor, nScans, HDR)

Runs a 360-degree rotary scan.

Parameters
motor: The rotary motor index (1-4).
nScans: The number of scans. Ideally, this should be a factor of 360.
HDR: Scan using HDR technology.

Example
Rotary360Scan(1, 12, false)

Returns
true if successful, false if unsuccessful, as well as a list of new scan group IDs.

RotaryAlignScanner(scannerName, motor)

Does a rotary alignment on the specified scanner. Note that a calibration board must first be placed on the rotary, facing the scanner.

Parameters
scannerName: The name of a scanner.
motor: Axis number.

Example
RotaryAlignScanner("Scanner-001", 1)

Returns
true if successful, false if unsuccessful.

RotaryCalibrate(scannerName, axis)

Calibrate the rotary table using the available rotary calibration images.

Parameters
scannerName: The name of a scanner.
axis: The axis/motor to calibrate for.

Example
RotaryCalibrate("Scanner-001", 1)

Returns
true if successful, false if unsuccessful.

RotaryCaptureCalibrationImage(scannerName)

Capture images of the calibration board for rotary table calibration.

Parameters
scannerName: The name of a scanner.

Example
RotaryCaptureCalibrationImage("Scanner-001")

Returns
true if successful, false if unsuccessful.

RotaryDeleteCalibration(scannerName)

Resets the rotary calibration.

Parameters
scannerName: The name of a scanner.

Example
RotaryDeleteCalibration("Scanner-001")

Returns
true if successful, false if unsuccessful.

RotaryGetCurrAngle(motor)

Parameters
motor: The rotary motor index (1-4).

Example
RotaryGetCurrAngle(1)

Returns
True if rotary is connected and current position of the rotary in the form of angle in degrees.

RotaryGetCurrStep(motor)

Parameters
motor: The rotary motor index (1-4).

Example

RotaryGetCurrStep(1)

Returns
True if rotary is connected and current position of the rotary in the form of steps.

RotaryGetStepsPerTurn(motor)

Parameters
motor: The rotary motor index (1-4).

Example
RotaryGetStepsPerTurn(1)

RotaryIDs()

Get a list of all available rotaries.

Returns
Autodetect all rotary tables and returns a list of rotary IDs.

RotaryMove(motor, steps)

Moves the rotary.

Parameters
motor: The rotary motor index (1-4).
steps: The number of steps to move the rotary.

Example
RotaryMove(1, 150)

Returns
true if successful, false if unsuccessful.

RotaryReset()

Resets the rotary position.

Returns
true if successful, false if unsuccessful.

RotaryRotate(motor, degrees)

Rotates the rotary.

Parameters
motor: The rotary motor index (1-4).
degrees: The number of degrees to rotate the rotary.

Example
RotaryRotate(1, 15)

Returns
true if successful, false if unsuccessful.

RotarySet(ID)

Instead of using any available rotary use one specified by ID. Set to null to revert to default behaviour.

Parameters

ID: ID of the rotary table, usually one returned by RotaryIDs().

Example
RotarySet(PluginRotaryWR::COM3)

RotarySetStepsPerTurn(motor, steps)

Parameters
motor: The rotary motor index (1-4).

Example
RotarySet(PluginRotaryWR::COM3)

Scanning

ClearMarkerExposure(scannerName)

Clears the marker exposure. Markers can still be detected using the scan exposure.

Parameters
scannerName: The name of a scanner.

Example
ClearMarkerExposure("Scanner-001")

Returns
true if successful, false if unsuccessful.

EasyScan()

Determines exposure settings, then does a scan. HDR will be used if necessary.

Returns
true if successful, false if unsuccessful, as well as a list of new scan group IDs.

GetMarkerExposure(scannerName)

Gets the exposure time for markers.

Parameters
scannerName: The name of the scanner to query the marker exposure from.

Example
GetMarkerExposure(Scanner001)

Returns
The exposure time for markers.

GetScannerExposure(scannerName)

Gets the exposure for this scanner's geometry camera(s).

Parameters
scannerName: The name of the scanner to query the exposure from.

Example
GetScannerExposure(Scanner001)

Returns
The current exposure time.

GetScannerGroup(scannerName)

Gets the scanner group.

Parameters
scannerName: The scanner name.

Example
GetScannerGroup("Scanner1")

Returns
true if successful, false if unsuccessful, as well as the name of the scanning group.

IsScannerConnected(scannerName)

Checks to see whether the specified scanner is connected.

Parameters
scannerName: The scanner name.

Example
IsScannerConnected("Scanner 1")

Returns
true if the scanner is connected, otherwise false.

Scan()

Scans into the project using the scanner(s) defined in the current calibration. Note that the Scanning_Generation_Type setting will affect whether the resulting scan is automatically processed or not, and if so, whether to generate a complete mesh or just a point cloud.

Returns
true if successful, false if unsuccessful, as well as a list of one or more output group IDs.

ScanHDR()

Scan the object at different exposures and combine them into one scan to get maximum data.

Returns
true if successful, false if unsuccessful, as well as a list of new scan group IDs.

ScannerConnect()

Ensures that all scanners in the current calibration are connected and working properly.

Returns
true if successful, false if unsuccessful.

SetMarkerExposure(scannerName)

Sets the marker exposure according to the current exposure value.

Parameters
scannerName: The scanner name.

Example
SetMarkerExposure("Scanner-001")

Returns

true if successful, false if unsuccessful.

SetScannerExposure(scannerName, time)

Sets the exposure for this scanner's geometry camera(s).

Parameters

scannerName: The name of the scanner to apply the exposure to.
time: The exposure time to apply (in milliseconds).

Example

SetScannerExposure(Scanner001, 16.6)

Returns

true if successful, false if unsuccessful.

SetScannerExposureSize(scannerName, size)

Sets the exposure size for this scanner's geometry camera(s).

Parameters

scannerName: The name of the scanner to apply the exposure to.
size: The exposure delta size from the minimum to maximum exposure for HDR scans (in milliseconds).

Example

SetScannerExposureSize(Scanner001, 33.3)

Returns

true if successful, false if unsuccessful.

SetScannerGroup(scannerName, groupName)

Sets the scanner group. Scanners within the same group will all start capturing simultaneously.

Parameters

scannerName: The scanner name.
groupName: A logical scanning group.

Example

SetScannerGroup("Scanner1", "1")

Returns

true if successful, false if unsuccessful.

StartLiveScan()

Enable live scan mode, then begin scanning using current scan options.

Returns

true if successful, false if unsuccessful.

StopLiveScan()

Stop live scan mode, if it was enabled.

Returns

true if successful, false if unsuccessful, as well as a list of new scan group IDs acquired during live scanning.

StopLiveScan()

Stop live scan mode, if it was enabled.

Returns
true if successful, false if unsuccessful, as well as a list of new scan group IDs acquired during live scanning.

UI

UI_InvertSelection()

Inverts the mesh selection.

UI_Recenter()

Recenters the 3D scene.

Various

GetMemoryUsage()

Gets the memory usage of FlexScan3D.

Returns
private memory usage of FlexScan3D, in bytes.

TranslucencyCompensation(groupID, k)

Scale the scan in z direction as: z = z + k * abs(nz).

Parameters
groupID: A combined group ID.
k: Constant multiplier which is the max shift in millimeters.

Example
TranslucencyCompensation("10", 0.75)

Returns
true if successful, false if unsuccessful.

Rotary Plugin Module

The plugin SDK is included with each FlexScan3D installation starting with release 3.3.2.x. The plugin is included in the directory C:\Program Files\LMI Technologies\FlexScan3D 3.3\SDK\PluginRotary by default.

Use the PluginRotaryCore project to implement the communication in C/C++. For C#, use the PluginRotaryWR project; this project implements the protocol described in the Rotary Protocol section.

Setup

You must develop your plugin using Visual Studio 2010 or later.

API

The following describes the API functions in general terms.

Required Functions

int PRC_BuildRotaryList()

Builds the list of all the available rotaries. This function is first called when FlexScan3D loads the rotary module.

The implementation should communicate with the actual rotary to determine if the rotary is connected.

In a multi-axis system, each axis should be represented as an independent motor within a rotary module.

Returns
The number of rotaries detected.

char* PRC_GetRotaryID(int index)

Gets the rotary name from the list of rotaries.

Parameters
index: Index of the rotary name to retrieve.

Returns
Unique rotary ID string. For example: "PluginRotaryABC::COM3", "PluginRotaryABC::IP:68.179.21.245", "PluginRotaryABC::MyRotary1".

BOOL PRC_IsConnected(const char* ID)

Determines if a rotary is connected.

Parameters
ID: Unique rotary ID string returned by PRC_GetRotaryID.

Returns
True if the function succeeds. False if the function fails.

BOOL PRC_GetNumMotors(const char* ID, int& motors)

Gets the number of motors or axes.

Parameters
ID: Unique ID string.
motors: Pointer to return the number of motors.

Returns
True if the function succeeds. False if the function fails.

BOOL PRC_GetCurrStep(const char* ID, int motor, int& step)

Gets the current step position.

Parameters
ID: Unique ID string.
motors: The number of motors.
step: The current step position. The value is between -StepsPerTurn to StepsPerTurn.

Returns
True if the function succeeds. False if the function fails.

BOOL PRC_GetStepsPerTurn(const char* ID, int motor, int& steps)

Gets the number of steps per one revolution (360 degree).

Parameters

ID: Unique ID string.
motors: Pointer to return the number of motors.
step: Pointer to return the number of steps per turn.

Returns

True if the function succeeds. False if the function fails. Returns 0 for displacement motors.

BOOL PRC_SetStepsPerTurn(const char* ID, int motor, int steps)

Sets the number of steps per one revolution (360 degree).

Parameters

ID: Unique ID string.
motors: Pointer to return the number of motors.
step: Number of steps per turn.

Returns

True if the function succeeds. False if the function fails.

BOOL PRC_Move(const char* ID, int motor, int steps)

Moves the motor forward by a fixed number of steps.

Parameters

ID: Unique ID string.
motors: Pointer to return the number of motors.
step: Number of steps to move.

Returns

True if the function succeeds. False if the function fails.

void PRC_Stop()

Stops the rotary. The function is called when FlexScan3D detaches the module.

Returns

nothing

BOOL PRC_GetMaxSpeed(const char* ID, int motor, double& speed)

Get the maximum speed of the rotary.

Parameters

ID: Unique ID string.
motors: Index to motor/axis.
speed: Maximum rotary speed. Speed should be between 0 and 1.0.

Returns

True if the function succeeds. False if the function fails.

BOOL PRC_SetMaxSpeed(const char* ID, int motor, double speed)

Set the maximum speed of the rotary.

Parameters

ID: Unique ID string.
motors: Index to motor/axis.
speed: Maximum rotary speed. Speed should be between 0 and 1.0.

Returns

True if the function succeeds. False if the function fails.

Optional Functions

These functions are already implemented, but you can override them if required.

BOOL PRC_Rotate(const char* ID, int motor, double degrees)

Moves the motor forward by a fixed number of degrees.

Parameters

ID: Unique ID string.
motors: Pointer to return the number of motors.
degrees: Number of degrees to rotate. Degrees should be between -360.0 and 360.0.

Returns

True if the function succeeds. False if the function fails.

BOOL PRC_GetCurrAngle(const char* ID, int motor, double& angle)

Returns the current position as an angle.

Parameters

ID: Unique ID string.
motors: Pointer to return the number of motors.
degrees: Current position as an angle. Angle returned is between -360.0 and 360.0.

Returns

True if the function succeeds. False if the function fails.

BOOL PRC_Reset(const char* ID)

Returns to the 0 position.

Returns

True if the function succeeds. False if the function fails.

C/C++ Specifics

The compiled PluginRotaryCore.dll should be put in the SDK\PluginRotary\PluginRotaryWrapper folder. The C/C++ API only supports one custom rotary plugin per FlexScan3D installation.

C# Specifics

The compiled DLL should be put in the "Rotary Modules" folder in the FlexScan3D installation folder. Multiple plugin DLLs for different motion controllers can be placed in the directory. FlexScan3D will call the BuildRotaryList in each DLL to determine which one is applicable.

The C# implementation inherits the PluginRotary interface under the PluginRotaryInterface namespace. The project should be setup to include the PluginRotaryInterface.dll from the FlexScan3D directory.

The following is the definition of the PluginRotary interface:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Configuration; namespace PluginRotaryInterface
{
    public abstract class PluginRotary
    {
       ////////////////////////////////////////////////////////////////////////
       // Default pluginID is the class name itself.
       ////////////////////////////////////////////////////////////////////////
       public virtual string pluginID { get { return GetType().Name; } }

////////////////////////////////////////////////////////////////////////////////////////
////
       // MANDATORY: Abstract functions are used by FlexScan3D and must implemented.

////////////////////////////////////////////////////////////////////////////////////////
////
       ////////////////////////////////////////////////////////////////////////
       // Builds the list of all the available rotaries. This function is first
       // called when FlexScan3D loads the rotary module.
       //
       // The implementation should communicate with the actual rotary to determine
       // if the rotary is connected.
       //
       // Parameters:
       // None
       // Returns:
       // List of rotary IDs
       // Format: "pluginID::rotaryID"
       //
       // example:
       // "MyPluginClass::Rotary1"
       // "MyPluginClass::Rotary2"
       //
       ////////////////////////////////////////////////////////////////////////
       public abstract List<string> rotaryIDs { get; }
       ////////////////////////////////////////////////////////////////////////
       // Determines if a rotary is connected.
       // Possible to try to reconnect or connect here.
       //
       // Parameters:
       // ID - Unique rotary ID string returned by rotaryIDs
```

```
// Returns:
// True if rotary is connected. False if rotary is not connected.
////////////////////////////////////////////////////////////////////////////
public abstract bool IsConnected(string ID);
////////////////////////////////////////////////////////////////////////////
// Gets the number of motors/axis
//
// Parameters:
// ID - Unique ID string
// motors - Returns the number of motors.
// Returns:
// True if the function succeeds. False if function fails.
//
////////////////////////////////////////////////////////////////////////////
public abstract bool GetNumMotors(string ID, out int motors);
////////////////////////////////////////////////////////////////////////////
// Gets the current step position.
//
// Parameters:
// ID - Unique ID string
// motors - Index to motor/axis
// step - Return the current step position.
// The value is between [-StepsPerTurn, StepsPerTurn]
//
// Returns:
// True if the function succeeds. False if the function fails.
//
////////////////////////////////////////////////////////////////////////////
public abstract bool GetCurrStep(string ID, int motor, out int step);
////////////////////////////////////////////////////////////////////////////
// Gets or sets the number of step per one revolution (360 degree).
// Return 0 for displacement motors.
//
// Parameters:
// ID - Unique ID string
// motors - Index to motor/axis
// step - Get or Set the number of steps per turn
//
// Returns:
// True if the function succeeds. False if the function fails.
//
////////////////////////////////////////////////////////////////////////////
public abstract bool GetStepsPerTurn(string ID, int motor, out int steps);
public abstract bool SetStepsPerTurn(string ID, int motor, int steps);
```

```
////////////////////////////////////////////////////////////////////////////////////////
////
        // generic motor steps interface

////////////////////////////////////////////////////////////////////////////////////////
////
        ////////////////////////////////////////////////////////////////////////
        // Move the motor forward by fixed number of steps.
        //
        // Parameters:
        // ID - Unique ID string
        // motor - Index to motor/axis
        // step - Number of steps to move per revolution
        //
        // Returns:
        // True if the function succeeds. False if the function fails.
        //
        ////////////////////////////////////////////////////////////////////////
        public abstract bool Move(string ID, int motor, int steps);
        ////////////////////////////////////////////////////////////////////////
        // Stops the rotary.
        // Emergency brake
        ////////////////////////////////////////////////////////////////////////
        public abstract void Stop();

////////////////////////////////////////////////////////////////////////////////////////
////
        // OPTIONAL: override if necessary

////////////////////////////////////////////////////////////////////////////////////////
////
        ////////////////////////////////////////////////////////////////////////
        // status functions
        //
        // By default FlexScan3D doesn't use these, but you can call them from scripting
        // to change rotary behavior.
        // // Gets or sets the maximum speed of the rotary.
        //
        // Parameters:
        // ID - Unique ID string
        // motor - Index to motor/axis
        // speed - Get or set max rotary speed.
        // Speed should be between 0 and 1.0.
        //
        // Returns:
```

```
        // True if the function succeeds. False if the function fails.
        //
        ////////////////////////////////////////////////////////////////////////
        public virtual bool GetMaxSpeed(string ID, int motor, out double speed) { speed =
0; return false; }
        public virtual bool SetMaxSpeed(string ID, int motor, double speed) { return
false; }
        ////////////////////////////////////////////////////////////////////////
        // exposing advanced plugin settings to Flexscan (optional)
        //
        // public override ApplicationSettingsBase settings { get { return
Properties.Settings.Default; } }
        ////////////////////////////////////////////////////////////////////////
        public virtual ApplicationSettingsBase settings { get { return null; } }
        ////////////////////////////////////////////////////////////////////////
        // Moves the motor forward by fixed number of degrees.
        // Same as Move() but in degrees.
        //
        // Parameters:
        // ID - Unique ID string
        // motor - Index to motor/axis
        // degrees - Number of degrees to turn.
        // Degrees should be between -360.0 to 360.0
        //
        // Returns:
        // True if the function succeeds. False if the function fails.
        //
        ////////////////////////////////////////////////////////////////////////
        public virtual bool Rotate(string ID, int motor, double degrees)
        {
           int stepsTurn;
           if (!GetStepsPerTurn(ID, motor, out stepsTurn)) return false;
           if (stepsTurn <= 0) return false; // displacement motor or not set
           int steps = (int)((degrees / 360.0) * stepsTurn);
           return Move(ID, motor, steps);
        }
        ////////////////////////////////////////////////////////////////////////
        // Returns the current position as an angle.
        //
        // Parameters:
        // ID - Unique ID string
        // motor - Index to motor/axis
        // angle - Current position in degrees.
        // Angle returned is between -360.0 to 360.0
        //
```

```
        // Returns:
        // True if the function succeeds. False if the function fails.
        //
        ////////////////////////////////////////////////////////////////////////
        public virtual bool GetCurrAngle(string ID, int motor, out double angle)
        {
           angle = 0;
           int step;
           if (!GetCurrStep(ID, motor, out step)) return false;
           int stepsTurn;
           if (!GetStepsPerTurn(ID, motor, out stepsTurn)) return false;
           if (stepsTurn == 0) return false; // displacement
           angle = step * 360.0 / stepsTurn;
           //tmp[i] = 360.0f; // assume step is already clamped
           return true;
        }
        ////////////////////////////////////////////////////////////////////////
        // Returns to the 0 position.
        //
        // Parameters:
        // ID - Unique ID string
        //
        // Returns:
        // True if the function succeeds. False if the function fails.
        //
        ////////////////////////////////////////////////////////////////////////
        public virtual bool Reset(string ID)
        {
           int nMotors;
           if (!GetNumMotors(ID, out nMotors)) return false;
           for (int i = 1; i <= nMotors; ++i)
           {
               int step;
               if (!GetCurrStep(ID, i, out step)) return false;
               //int stepsTurn = StepsPerTurn(ID, i);
               //if (stepsTurn > 0) step = stepsTurn;
               if (!Move(ID, i, -step)) return false;
           }
           return true;
        }
        ////////////////////////////////////////////////////////////////////////
        //
        ////////////////////////////////////////////////////////////////////////
        }
}
```

Rotary Protocol

General Rotary Communication Protocol

All commands sent to the rotary table use a simple character format, including the motor numbers. Parts in commands marked as xxx are passed to the table as byte data. For example, if you want table 1 to rotate 4 steps, instead of passing "I1M004" you pass "I1M" + (char)0 + (char)0 + (char)4

In general all commands get a reply in the form of ^XXXXXX.

Commands

V

Request the status of the rotary table. Usual reply would be ^R1R2R3R4 indicating rotary 1 ready, rotary 2 ready, etc. ^B1xxxR2R3R4 means rotary 1 is busy where xxx are 3 bytes indicates how many steps the rotary still has to perform.

SmMxxx

Sets the speed of the motor m to xxx, where xxx is a 3 bytes of data indicating the speed. Example code: port.Write("S1M" + (char)0 + (char)6 + (char)255); // set motor 1 to speed 1791. The standard speed range of our rotary table is: 0x000001 to 0x0012FF (1 to 4863). Controller will respond with ^mxx mirroring the motor number and 2 last bytes of speed setting.

ImMxxx

Turns motor m xxx number of steps. Controller will acknowledge with ^Bmxxx.

DmCWLO

Set motor number m to rotate clockwise. Each consecutive command to rotate the motor m will rotate it clockwise.

DmCWHi

Sets rotary m to rotate counterclockwise.

EmHALT

Rotary m stop.

Rotary Sample Command Sequence

Motor numbers are passed as characters but the number of steps and speed are passed as 3 bytes of binary for simplicity.

send: V reply: ^R1R2R3R4

send: S1M1791 reply: ^191

sned: D1CWLO reply: ^

send: I1M100 reply: ^B1100

3D3 File Format

Version 8 file format

Introduced texture per face as a separate array of uv coordinates and an indexed array of triplets for every face referencing those coordinates. This way every face can have an individual set of texture coordinates.

```
int verNum3d3; // version number (should be 8)
// header info ////////////////////////////////////////
int gridHeight; // camera pic height
int gridWidth; // camera pic width
int textureHeight; // texture cam height
int textureWidth; // texture cam width
int traceStepH; // ignore the following 2 integers should be 1
int traceStepW;
double ccDist; // distance between the cameras in mm
int decimated; // flag 1 or 0, indicates if the mesh has been decimated at creation
// if 0 the mesh is a perfect grid and can be processed more efficiently
double edgeTol; // all edges bigger than this tolerance were discarded during mesh
generating process
// as part of the mesh cleanup procedure
// sizes ///////////////////////////////////////////////
int nVertices; // number of vertices
int nFaces; // number of faces (triangles)
int nGrid; // every point has an associated height & width of the pixel it was traced
from
// nGrid is a number of height/width coordinates nGrid == nVertices || nGrid == 0
int nUVs; // number of texture coordinates
int nFaceUVs; // number of triplets of indices for texture per face nFaceUVs == 0 ||
nFaceUVs == nFaces
// if supplied for every face there is a corresponding triplet of indices pointing to uvs
array for
// a corresponding texture coordinate
int nConfidence; // during combining all the vertices get assigned a confidence value of
how reliable
// every vertex is [0, 1] == [bad, good]
int nVertColors; // number of vertex colors nVertColors == nVertices || nVertColors == 0
// Vertices ////////////////////////////////////////////
```

```
double points[nVertices*3]; // coordinates of all points xyz
// Faces /////////////////////////////////////////////
int indices[nFaces*3]; // indices of all the triangles (0 indented)
// grid //////////////////////////////////////////////
int grid[nGrid*2]; // for every point height & width of the pixel it was traced from
// UVs ///////////////////////////////////////////////
float uvs[nUVs*2]; // u/v texture coordinates for each vertex
// FaceUVs ///////////////////////////////////////////
int faceUVs[nFaceUVs*3]; // 3 indices pointing to uvs for corresponding texture
coordinates
// Confidence ////////////////////////////////////////
float confidence[nConfidence]; // confidence value [0, 1] for each vertex
// Colors ////////////////////////////////////////////
byte colors[nVertColors*4]; // 4 bytes (alphe,red,green,blue)[0..255] per each vertex
```

Version 7 file format

Introduced color per vertex values as four bytes per vertex of alpha, red, green, blue.

```
int verNum3d3; // version number (should be 7)
// header info ///////////////////////////////////////
int gridHeight; // camera pic height
int gridWidth; // camera pic width
int textureHeight; // texture cam height
int textureWidth; // texture cam width
int traceStepH; // ignore the following 2 integers should be 1
int traceStepW;
double ccDist; // distance between the cameras in mm
int decimated; // flag 1 or 0, indicates if the mesh has been decimated at creation
// if 0 the mesh is a perfect grid and can be processed more efficiently
double edgeTol; // all edges bigger than this tolerance were discarded during mesh
generating process
// as part of the mesh cleanup procedure
// sizes /////////////////////////////////////////////
int nVertices; // number of vertices
int nFaces; // number of faces (triangles)
int nGrid; // every point has an associated height & width of the pixel it was traced
from
// nGrid is a number of height/width coordinates nGrid == nVertices || nGrid == 0
int nUVs; // number of texture coordinates nUVs == nVertices || nUVs == 0
int nConfidence; // during combining all the vertices get assigned a confidence value of
how reliable
// every vertex is [0, 1] == [bad, good]
int nVertColors; // number of vertex colors nVertColors == nVertices || nVertColors == 0
// Vertices //////////////////////////////////////////
```

```
double points[nVertices*3]; // coordinates of all points xyz
// Faces /////////////////////////////////////////////
int indices[nFaces*3]; // indices of all the triangles (0 indented)
// grid //////////////////////////////////////////////
int grid[nGrid*2]; // for every point height & width of the pixel it was traced from
// UVs ///////////////////////////////////////////////
float uvs[nUVs*2]; // u/v texture coordinates for each vertex
// Confidence ////////////////////////////////////////
float confidence[nConfidence]; // confidence value [0, 1] for each vertex
// Colors ////////////////////////////////////////////
byte colors[nVertColors*4]; // 4 bytes (alphe,red,green,blue)[0..255] per each vertex
```

Version 6 file format

Introduced vertex confidence values. From now on every vertex has a confidence value attached to it indicating how reliable given vertex is. Confidence values are used in the combining algorithm for determining the true surface and discarding the false data points. Added a new value edgeTol for internal use, description below.

```
int verNum3d3; // version number (should be 6)
// header info ///////////////////////////////////////
int gridHeight; // camera pic height
int gridWidth; // camera pic width
int textureHeight; // texture cam height
int textureWidth; // texture cam width
int traceStepH; // ignore the following 2 integers should be 1
int traceStepW;
double ccDist; // distance between the cameras in mm
int decimated; // flag 1 or 0, indicates if the mesh has been decimated at creation
// if 0 the mesh is a perfect grid and can be processed more efficiently
double edgeTol; // all edges bigger than this tolerance were discarded during mesh
generating process
// as part of the mesh cleanup procedure
// sizes /////////////////////////////////////////////
int nVertices; // number of vertices
int nFaces; // number of faces (triangles)
int nGrid; // every point has an associated height & width of the pixel it was traced
from
// nGrid is a number of height/width coordinates nGrid == nVertices || nGrid == 0
int nUVs; // number of texture coordinates nUVs == nVertices || nUVs == 0
int nConfidence; // during combining all the vertices get assigned a confidence value of
how reliable
// every vertex is [0, 1] == [bad, good]
// Vertices //////////////////////////////////////////
double points[nVertices*3]; // coordinates of all points xyz
```

```
// Faces ////////////////////////////////////////////
int indices[nFaces*3]; // indices of all the triangles (0 indented)
// grid /////////////////////////////////////////////
int grid[nGrid*2]; // for every point height & width of the pixel it was traced from
// UVs //////////////////////////////////////////////
float uvs[nUVs*2]; // u/v texture coordinates for each vertex
// Confidence ///////////////////////////////////////
float confidence[nConfidence]; // confidence value [0, 1] for each vertex
```

Version 5 file format

Added a flag *decimated* indicating if the mesh was decimated during creation. Non decimated meshes can be processed more efficiently.

```
int verNum3d3; // version number (should be 5)
// header info //////////////////////////////////////
int gridHeight; // camera pic height
int gridWidth; // camera pic width
int textureHeight; // texture cam height
int textureWidth; // texture cam width
int traceStepH; // ignore the following 2 integers should be 1
int traceStepW;
double ccDist; // distance between the cameras in mm
int decimated; // flag 1 or 0, indicates if the mesh has been decimated at creation
// if 0 the mesh is a perfect grid and can be processed more efficiently
// sizes ////////////////////////////////////////////
int nVertices; // number of vertices
int nFaces; // number of faces (triangles)
int nGrid; // every point has an associated height & width of the pixel it was traced
from
//nGrid is a number of height/width coordinates nGrid == nVertices || nGrid == 0
int nUVs; // number of texture coordinates nUVs == nVertices || nUVs == 0
// Vertices /////////////////////////////////////////
double points[nVertices*3]; // coordinates of all points xyz
// Faces ////////////////////////////////////////////
int indices[nFaces*3]; // indices of all the triangles (0 indented)
// grid /////////////////////////////////////////////
int grid[nGrid*2]; // for every point height & width of the pixel it was traced from
// UVs //////////////////////////////////////////////
float uvs[nUVs*2]; // u/v texture coordinates
```

Version 4 file format

```
int verNum3d3; // version number (should be 4)
// header info //////////////////////////////////////
```

```
int gridHeight; // camera pic height
int gridWidth; // camera pic width
int textureHeight; // texture cam height
int textureWidth; // texture cam width
int traceStepH; // ignore the following 2 integers should be 1
int traceStepW;
double ccDist; // distance between the cameras in mm
// sizes ////////////////////////////////////////////////
int nVertices; // number of vertices
int nFaces; // number of faces (triangles)
int nGrid; // every point has an associated height & width of the pixel it was traced
from
//nGrid is a number of height/width coordinates nGrid == nVertices || nGrid == 0
int nUVs; // number of texture coordinates nUVs == nVertices || nUVs == 0
// Vertices /////////////////////////////////////////////
double points[nVertices*3]; // coordinates of all points xyz
// Faces ////////////////////////////////////////////////
int indices[nFaces*3]; // indices of all the triangles (0 indented)
// grid /////////////////////////////////////////////////
int grid[nGrid*2]; // for every point height & width of the pixel it was traced from
// UVs //////////////////////////////////////////////////
float uvs[nUVs*2]; // u/v texture coordinates
```

FAQ / Troubleshooting

Licensing

My license failed to activate, what do I do?

First, verify that the date on your system is correct. Then e nsure that you are logged in as the Administrator before attempting to activate your license. Just to be safe we recommend that you right-click on the FlexScan3D.exe and select "Run as Administrator", then simply input your provided license ID into the input box and select "Activate".

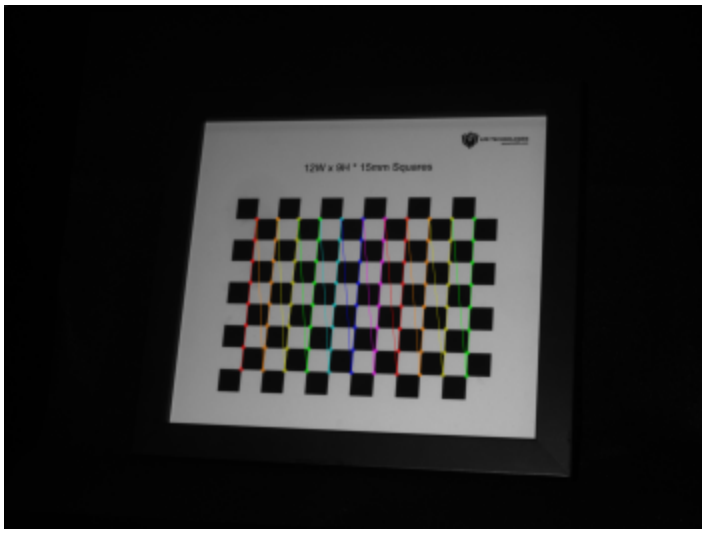
Setup and Calibration

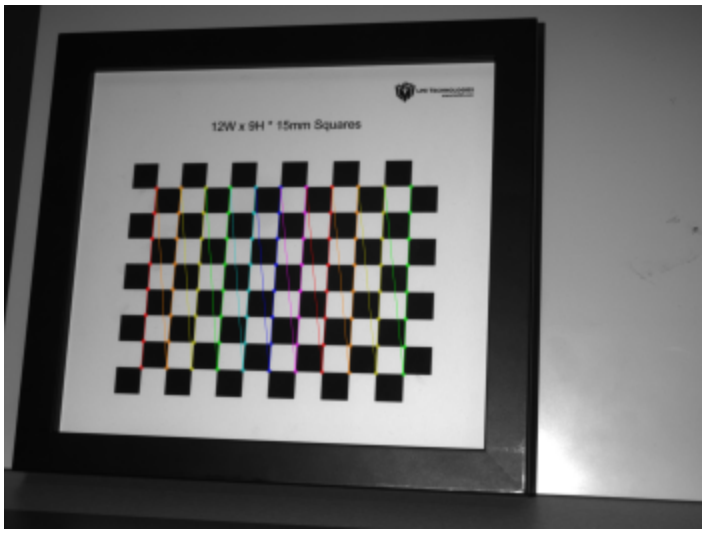
Why does it take a long time during the "Finding Corners" phase of a lens capture?

There are actually several things that can affect the finding corners speed: background, exposure, camera resolution, and CPU load.

Background

The choice of background behind the calibration board may affect the corner finding speed. Corner finding is slightly faster against a black background than against a white background.





Exposure
This is actually related to background because the average exposure level may be directly affected by the scanner's exposure time and lens aperture settings. There is one additional factor to consider: the type of material used for the calibration board. If the board is glossy or slightly reflective, then the exposure level on the board may be uneven and/or some of the squares may get washed out. This forces the algorithm to take longer to find all the corners.

Camera Resolution
Higher resolution images take longer to process than low-resolution images, especially when it comes to high-resolution DSLR texture cameras. 18-megapixel and higher images may be overkill for many scanning needs, in which case they will only serve to slow down the calibration and scanning process. Unless you absolutely require the highest resolution texture image, consider selecting the Medium or Small image size from the camera's setup menu.

CPU Load
A slower computer may not be able to multitask as efficiently as a higher-end computer with faster CPU and/or more memory. We recommend that you close all unnecessary applications while calibrating and scanning.

Calibration board reflects light onto the camera(s)

At times your calibration board may rest at a certain angle, which will cause the projector light to be reflected onto the cameras. This will cause failed calibration images. Simply angle the calibration board towards the cameras in a way where the light will not reflect onto the camera(s).

When doing close scans, my exposure is too bright even with the lens f-stop and cameras exposure set to the darkest settings

In this case you will need to lower the brightness on your projector. Overexposed areas will appear red in the live camera feed. Adjust the projector brightness slider until the exposure is at an acceptable level.

Which type (black-and-white or gray) of calibration board should I use to calibrate my Single and Duo scanner?

We recommend you use only the gray scale calibration board for calibrating in Single scan mode. However for Duo scan mode you are able to use either the black-and-white or the grayscale calibration board.

Cameras

Can I 3D scan with digital DSLR cameras, HD camcorders, or web cameras?

We do not support 3D scanning with these devices.

In the past, we have had hundreds of users attempt to use a wide variety of cameras but have found that the results were often unusable. Even with good results, the difficulty in achieving those results was not practical for everyday scanning. Given that a good pair of machine vision cameras are relatively inexpensive, work in a stable manner, and return high quality results, using other types of equipment is strongly discouraged.

Can I scan with a non-matching pair of cameras?

For Duo camera mode, we always suggest using a pair of the same cameras. You could consider different cameras as long as the size of the image returned is exactly the same.

Why can't I just set the aperture to the widest setting (lowest f-number) possible, and just adjust the exposure in the software?

Changing the aperture also affects the focus depth of field. The wider the aperture is, the narrower the depth of field will be. You need to consider this when choosing an object to scan. If the object is too long (either towards or away from the camera), not all of it may be in focus and it will be blurry in the images. This will reduce the mesh quality and add undesirable results.

My Canon Digital Camera is not being recognized by Windows 7 N 64-bit

The 64-bit version of Windows 7 N does not include the media pack required to properly install the Canon drivers. Please download and install the Windows Media Player pack. After the installation please reboot your system and connect your Canon camera to your PC. The drivers should now properly install.

My uEye camera is functioning slower than expected and is not performing properly

Support for uEye cameras has been removed in FlexScan3D 3.2. If your scanner uses a pair of uEye cameras, please use FlexScan3D 3.1.

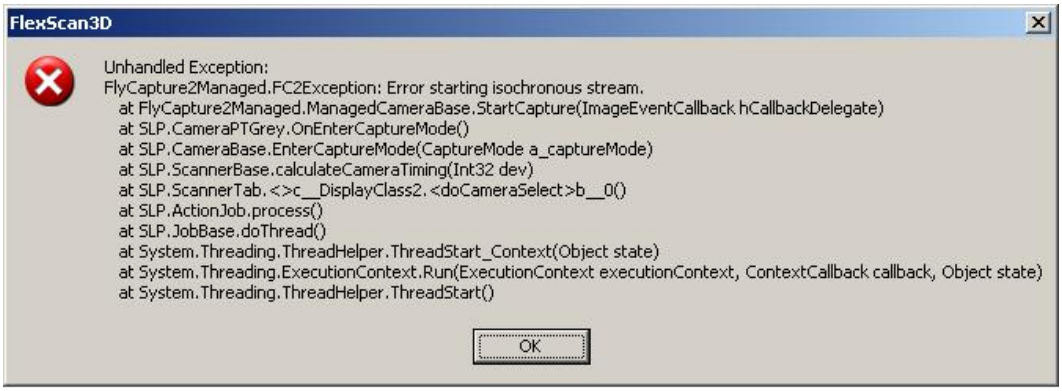
FlexScan3D crashes or my cameras won't respond when I plug in Firewire cameras

The most common cause of this is there is insufficient bandwidth on the Firewire card (or adapter for laptops) for streaming data from the camera(s). This can be fixed by reducing the framerate setting on the camera(s).

For a duo scanner, we highly recommend that you have a **dual bus** Firewire card. This means each camera will have its bus to the system, and will not interfere with each other. Most single bus Firewire cards may have multiple ports, but all devices share the same bus, causing bandwidth to be split between the devices. Avoid single bus cards!

If you do have a dual bus Firewire card, but are still having issues with a duo scanner system: Plug in only one camera, set its framerate to a lower setting (such as 7.5Hz if available) via the Advanced Camera Settings. Remove the camera and plug in the other camera and repeat. Now, you should be able to have both cameras plugged in and working.

Below is an example of the error message you may see:



PointGrey bandwidth error

Scanning

Below is a list of common 3D problems that people run in to and a list of likely causes.

No Data

- The projector was off
- The cameras did not capture any images
- Images were captured out of sync with the projector

The majority of the time, the camera images were out of sync with the projector. Your best bet is to increase the delays for the camera timings so that there is more time to synchronize the cameras to the projector.

Noisy Data

- Camera underexposure or overexposure (examples).
- Reflective, transparent, or very dark material surface.
- Sensor noise (High ISO, gain, or gamma settings).

Wave Patterns

This is one of the most difficult problems to fix because its cause can be any one of several completely different issues. The wavy patterns occur because the patterns that are projected and then decoded have been corrupted along the way, sometimes in subtle ways that are not visible to the naked eye. This causes the decoding process to partially fail and then artifacts appear as small wave patterns in your scan data.

- Out of focus projector (Solution)
- Camera underexposure or overexposure
- Subject moved during scanning; even the slightest movement will affect the scan
- Not enough distance between the cameras
- Camera's exposure was not a multiple of the projector refresh rate
- Hot air from the projector blows in front of one or more of the cameras
- Images were captured out of sync with the projector (Solution)

Misaligned Textures

When scanning with a texture camera you may experience misaligned textures over-top of your mesh. The root cause for most of this is simply that the texture camera may have been moved slightly during or after calibration or scanning. This is why we recommend you use a power adapter plugged into an outlet to provide a constant supply of power to the camera. For more information on using your texture camera, see See *Scanning with Texture* (page 95).

Markers

At times the markers on your object will not be seen by FlexScan3D, here are a few causes and solutions:

Marker alignment failed

FlexScan3D requires a minimum of 4 markers to be seen by BOTH cameras, if 1 of your cameras fails to see 1 of the 4 markers you will be informed that the minimum amount required was not found. Simply make sure that both cameras see at least 4 markers in clear view.

FlexScan3D is taking a long time finding the markers

This is caused by placing too many markers on your scan object. We recommend a minimum of 4 and a maximum of 10 to 15. Simply remove the extra markers and re-scan.

Tutorial Videos

Live Scanning: New Feature in FlexScan3D v3.3 Software
watch

Automated Mesh Geometry Alignment: New Feature in FlexScan3D v3.3 Software
watch

Cut Plane: New Feature in FlexScan3D v3.3 Software
watch

Easy Scan: New Feature in FlexScan3D v3.3 Software
watch

Interactive Hole Filling: New Feature in FlexScan3D v3.3 Software
watch

Automating the 3D Scanning Process Using Multiple HDI 120 3D Scanners
watch

FlexScan3D 3.1 3D Scanner Software: Improvements
watch

Single Scanner Calibration
watch

Duo Scanner Calibration
watch

FlexScan3D Version 3.0 Overview
watch

Tune Calibration
watch

Photogrammetry Alignment
watch

High Resolution Color Texture Capture
watch

Deviation Analysis
watch

Glossary

A

aperture
: Controls the size of the hole through which light can pass. (See Aperture on Wikipedia for more information.)

C

calibration
: Uses a known measurement (calibration board) to define the correspondence between the camera(s) and the projector(s).

calibration board
: A highly accurate image usually consisting of a grid of alternating black and white squares (checkerboard) of a known size.

D

decimation
: This is used in order to decrease the polygon count of a mesh while preserving key details.

E

erosion
: This removes polygons at the edges of a mesh in cases where the edge data is noisy.

M

megapixels (camera resolution)
: A rough estimate of how many pixels are contained in each image taken by the camera. For example, a "2 megapixel" camera has 1600x1200 (1,920,000) pixels.

multi-scanner
: A system of scanners which sequentially acquires scan data of a single object from multiple positions.

S

scanner
: A system consisting of a projector and one or more cameras in fixed positions.

shutter speed
: Controls the duration that the shutter remains open while capturing an image. (Wikipedia - Shutter Speed)

smoothing
: Reduces the intensity of bumps and spikes on mesh surfaces.

Z

Z-near / Z-far

Defines a distance envelope from the scanner to the scanning target area. Geometry points closer than the Z-near value or farther away than the Z-far value are discarded.