Contents

Introduction
About this User Guide 12
Contacting Support 12
Example Images 12
Installation 13
Installation on Windows 13
Installing Ocula from the command line 13
Installation on Mac 14
Installing Ocula silently from the command line 14
Installation on Linux 15
Licensing 16
About Licenses 16
Licensing on a Single Machine 17
Obtaining a License Key 17
Installing the License 17
On Windows and Mac 17
On Linux 18
Licensing over a Network 18
Obtaining Floating Licenses 18
Installing Floating Licenses 19
On Windows and Mac 19
On Linux 19
Telling the Client Machines Where to Find the Licenses 20
On Windows and Mac 20
On Linux 20
Further Reading 21
Other Foundry Products 21

DisparityGenerator
Description 23
Inputs 24
Generating Disparity Maps 25
NewView

Description 44
Inputs 44
Creating a New View 45
O_NewView Controls 46
Use GPU 46
Views to Use 46
View to Build 46
Occlusions 47
Edges 48
O_NewView Example 48
   Step by Step 49

ColourMatcher

Description 51
   Basic Mode 51
   Local Matching Mode 51
Inputs 52
Performing a Color Match 52
   Editing Color Match Results 53
O_ColourMatcher Controls 53
Use GPU 53
Views to Use 54
Match 54
Mode 54
Mask 54
View Menu Button 55
Local Matching 55
Occlusions 56
O_ColourMatcher Example 57
   Step by Step 58
FocusMatcher

Description
- Match Edges Mode
- Reconstruct Edges Mode

Inputs

Performing a Focus Match

O_FocusMatcher Controls

Use GPU

Views to Use

Match

Mode

Mask

Local Matching

Occlusions

O_FocusMatcher Example
  - Step by Step

Solver

Introduction

Inputs

Solving the Camera Relationship
  - Solving Using Python
  - Reviewing and Editing the Results

Adding User Matches
  - Feeding the Results to Other Ocula Nodes

O_Solver Controls
  - O_Solver Tab

Views to Use

Analysis

Display

Current Frame
  - Python Tab

O_Solver Example
Step by Step

**VerticalAligner**

Description 89
Global Alignment Mode 90
Local Alignment Mode 90
Fix Scale Mode 91
Fix Offset Mode 91
Inputs 91
Using O_VerticalAligner 92
  Analysing and Using Output Data 93
Scripting Analysis and CornerPin Creation 94
O_VerticalAligner Controls 95
  O_VerticalAligner Tab 95
Views to Use 95
Align 95
Filter 95
Output STMap 96
Global 96
Local Options 98
Fix Scale Options 99
Fix Offset Options 100
Analyze Sequence 100
  Output Tab 101
  Python Tab 101
O_VerticalAligner Example 102
  Step by Step 102

**VectorGenerator**

Description 104
Inputs 105
Generating Motion Vectors 106
  Writing Motion Vectors into a Clip 107
O_VectorGenerator Controls
Use GPU
Views to Use
Mask
Vector Detail
Strength
Consistency
Smoothness
O_VectorGenerator Example

Retimer
Description
Timing Methods
Inputs
Using O_Retimer
Retiming Stereo Footage Using Speed
Retiming Stereo Footage Using Source Frame
Varying the Retime Speed
Varying Retimes Using Speed
Varying Retimes Using Source Frames
O_Retimer Controls
Use GPU
Views to Use
Timing
Speed
Frame
Edges
O_Retimer Example
Step by Step
Generating Motion Vectors
Retiming the Sequence
InteraxialShifter
Description
Inputs
Using O_InteraxialShifter
O_InteraxialShifter Controls
Use GPU
Views to Use
Left Position
Right Position
Edges

DepthToDisparity
Description
Inputs
Generating a Disparity Map from Depth
O_DepthToDisparity Controls
Views to Use
O_DepthToDisparity Example
  Step by Step

DisparityToDepth
Description
Inputs
Using O_DisparityToDepth
  Using O_DisparityToDepth with DeepMerge
O_DisparityToDepth Controls
Views to Use
O_DisparityToDepth Example
  Step by Step
An Example of Using O_DisparityToDepth with ZDefocus
MultiSample

Description

Inputs

Using O_MultiSample

Using O_MultiSample with the Sample Input

Using O_MultiSample with the Sample and Mask Inputs

O_MultiSample Controls

Use GPU

Channels

Sample

Mask

O_MultiSample Example

Step by Step

Quality Control Tools

Description

DisparityViewer

Description

Displaying Disparity Vectors

Displaying Parallax Histograms

Displaying Parallax Violation Overlays

Inputs

O_DisparityViewer Controls

Views to Use

Display

Vectors

Histogram

Parallax

O_DisparityViewer Example

Step by Step

DisparityReviewGizmo

Description

Inputs
Using DisparityReviewGizmo 161
DisparityReviewGizmo Controls 162

Output 162
Background 163
Intensity 164
xDisparity 164
yDisparity 164
Occlusion 165
Limits 165

StereoReviewGizmo 166
Description 166
Inputs 166
Using StereoReviewGizmo 167
StereoReviewGizmo Controls 167

Output 167
Parallax Controls 169
Setup 170
Export Controls 170

Appendix A: Node Dependencies

Appendix B: External Software
Welcome to this User Guide for Ocula 4.0 on Nuke. Ocula is a collection of tools that solve common problems with stereoscopic imagery, improve productivity in post production, and ultimately help to deliver a more rewarding 3D-stereo viewing experience.

All Ocula nodes integrate seamlessly into Nuke. They are applied to your clips as any other node and they can all be animated using the standard Nuke animation tools.

About this User Guide

This User Guide tells you how to install and use the Ocula 4.0 nodes and tools. Each node or tool is described in detail in later chapters. Licensing Ocula is covered in the separate Foundry Licensing Tools (FLT) User Guide, which you can download from support.foundry.com.

This guide assumes you are familiar with Nuke and the machine it is running on.

**NOTE:** For the most up-to-date information, please see the Ocula on Nuke product page and the latest Ocula 4.0 user guide on our website at www.foundry.com.

Special thanks to Disney Enterprises, Inc. for use of the TRON: LEGACY images throughout this user guide.

Contacting Support

Should questions arise that the documentation fails to address, you can visit the Support Portal at support.foundry.com.

Example Images

Example images are provided for use with Ocula. You can download these images from our website at https://www.foundry.com/products/ocula/online-help and try Ocula out on them.
Installation

Installing Ocula 4.0 does NOT overwrite any versions of Ocula 3.x, Ocula 2.x, or Ocula 1.x.

To see the installation instructions for your operating system, go to:

- Installation on Windows
- Installation on Mac
- Installation on Linux

NOTE: You can put the Ocula nodes anywhere as long as you set the environment variable **NUKE_PATH** to point to it.

Installation on Windows

Ocula is distributed as a software download from our website at www.foundry.com. To install Ocula on a computer running Windows, follow these instructions:

NOTE: Throughout the following instructions, please replace `<version>` with the Nuke release you’re using.

1. Download the following file from our website at www.foundry.com:
   Ocula_4.0v6_Nuke_<version>-win-x86-release-64.zip
2. Unzip the file you downloaded.
3. Double-click on the exe file to launch the installer. Follow the on-screen instructions to install Ocula.
4. Proceed to Licensing.

Installing Ocula from the command line

To install Ocula from the command line, do the following:

NOTE: Throughout the following instructions, please replace `<version>` with the Nuke release you’re using.

1. Download and unzip the following file from our website at www.foundry.com:
   Ocula_4.0v6_Nuke_<version>-win-x86-release-64.zip
2. To open a command prompt window, select **Start** > **All Programs** > **Accessories** > **Command Prompt**.
3. Use the `cd` (change directory) command to move to the directory where you saved the installation file. For example, if you saved the installation file in C:\Temp, use the following command and press Return:

   `cd \Temp`

4. To install Ocula, do one of the following:
   - To install Ocula and display the installation dialog, type the name of the install file without the file extension and press Return:
     `Ocula_4.0v6_Nuke_<version>-win-x86-release-64`
   - To install Ocula silently so that the installer does not prompt you for anything but displays a progress bar, enter `/silent` after the installation command:
     `Ocula_4.0v6_Nuke_<version>-win-x86-release-64 /silent`
   - To install Ocula silently so that nothing is displayed, enter `/verysilent` after the installation command:
     `Ocula_4.0v6_Nuke_<version>-win-x86-release-64 /verysilent`

   **NOTE:** By running a silent install of Ocula, you agree to the terms of the End User License Agreement. To see this agreement, please see Foundry's [End User License Agreement](#) or run the installer in standard, non-silent mode.

**Installation on Mac**

Ocula is distributed as a software download from our website at [www.foundry.com](http://www.foundry.com). To install Ocula 4.0 on a Mac, follow these instructions:

   **NOTE:** Throughout the following instructions, please replace `<version>` with the Nuke release you're using.

1. Download the following file from our website at [www.foundry.com](http://www.foundry.com):
   `Ocula_4.0v6_Nuke_<version>-mac-x86-release-64.dmg`
2. Double-click on the downloaded dmg file.
3. Double-click on the pkg file that is created.
4. Follow the on-screen instructions to install Ocula.
5. Proceed to Licensing.

**Installing Ocula silently from the command line**

   **NOTE:** Throughout the following instructions, please replace `<version>` with the Nuke release you're using.
1. Download the following file from our website at www.foundry.com:
   Ocula_4.0v6_Nuke_<version>-mac-x86-release-64.dmg
2. Launch a Terminal window.
3. To mount the dmg installation file, use the `hdiutil` `attach` command with the directory where you saved the installation file. For example, if you saved the installation file in Builds/Ocula, use the following command:
   ```bash
   hdiutil attach /Builds/Ocula/Ocula_4.0v6_Nuke_<version>-mac-x86-release-64.dmg
   ```
4. Enter the following command:
   ```bash
   pushd /Volumes/Ocula_4.0v6_Nuke_<version>-mac-x86-release-64/
   ```
   This stores the directory path in memory, so it can be returned to later.
5. To install Ocula, use the following command:
   ```bash
   sudo installer -pkg Ocula_4.0v6_Nuke_<version>-mac-x86-release-64.pkg -target "/"
   ```
   You are prompted for a password.
6. Enter the following command:
   ```bash
   popd
   ```
   This changes to the directory stored by the `pushd` command.
7. Finally, use the following command to eject the mounted disk image:
   ```bash
   hdiutil detach /Volumes/Ocula_4.0v6_Nuke_<version>-mac-x86-release-64/
   ```

**NOTE:** By running a silent install of Ocula, you agree to the terms of the End User License Agreement. To see this agreement, please see Foundry's End User License Agreement or run the installer in standard, non-silent mode.

### Installation on Linux

Ocula is distributed as a software download from our website at www.foundry.com. To install Ocula 4.0 on a computer running Linux, follow these instructions:

**NOTE:** Throughout the following instructions, please replace `<version>` with the Nuke release you’re using.

1. Download the following file from our website at www.foundry.com:
   Ocula_4.0v6_Nuke_<version>-linux-x86-release-64.tgz
2. Move the downloaded file to the following directory (create the directory if it does not yet exist):
   `/usr/local/Nuke/
3. In the above mentioned directory, extract the files from the archive using the following command.
   ```bash
   tar xvzf Ocula_4.0v6_Nuke_<version>-linux-x86-release-64.tgz
   ```
This creates the `<version>/plugins/Ocula/4.0` sub-directory (if it doesn't already exist), and installs Ocula in that directory.

4. Proceed to Licensing.

Licensing

About Licenses

If you simply want to try out Ocula, you can obtain a trial license, which allows you to run Ocula for free for 15 days.

To use Ocula after this trial period, you need either a valid license key or a floating license and server running the Foundry Licensing Tools (FLT):

- **License Keys** - These can be used to install and activate node locked (also known as uncounted) licenses. Node locked licenses allow you to use Ocula on a single machine. This license does not work on a different machine and if you need it to, you'll have to transfer your license. Node locked licenses do not require additional licensing software to be installed. See Licensing on a Single Machine for more information.

- **Floating Licenses** - also known as counted licenses, enable Ocula to work on any networked client machine. The floating license should be put on the server and is locked to a unique number on that server. Floating licenses on a server require additional software to be installed. This software manages those licenses on the server, giving licenses out to client stations that want them. The software you need to manage these licenses is called the Foundry License Tools (FLT) and it can be freely downloaded from our website. Floating licenses often declare a port number on the server line and a port number on the vendor line. See Licensing over a Network for more information.

⚠️ **WARNING:** If there is an interruption between the license server and Ocula, rendering aborts with an exit code of 1. You can use the `--cont` command line argument to force Ocula to continue rendering on failure, producing black license failure frames rather than aborting the whole render.

The instructions below run through both licensing methods, and you can find a more detailed description in the Foundry Licensing Tools User Guide available on our website: support.foundry.com.
Licensing on a Single Machine

Obtaining a License Key

You can purchase license keys by:
- going to our website at www.foundry.com,
- e-mailing us at sales@foundry.com,

To generate a license key, we need to know your System ID. The System ID (sometimes called Host ID or rlmhostid) returns a unique number for your computer. We lock our license keys to the System ID.

To display your System ID, do the following:
- On Windows and Mac
  Download the Foundry License Utility (FLU) from support.foundry.com and run it. The System ID is displayed at the bottom of the window.
- On Linux
  Download the Foundry License Utility (FLU) from support.foundry.com and run it from the command line:
  
  `<download location>/FoundryLicenseUtility -i`

  ![NOTE: The <download location> refers to the location where you saved the Foundry Licensing Utility.]

Just so you know what a System ID number looks like, here’s an example: 000ea641d7a1.

Installing the License

Once a license has been generated for you, we e-mail you the license key and instructions on how to obtain the correct version of the Foundry License Utility (FLU). Gunzip or untar the file and save the FLU and your license key to a folder of your choice. The instructions below tell you what to do with these.

On Windows and Mac

Just drop the license key on the Foundry License Utility (FLU) application to install it. This checks the license key and copies it to the correct directory.
On Linux

1. Navigate to the location of the FLU_[version]_linux-x86-release-64.tgz file.
2. Type the following commands to extract and install the FLU. Note that you need to replace [version] with the version of FLU you are using and [my license] with the location of your license key.
   ```
   tar xvzf FLU_[version]_linux-x86-release-64.tgz
   cd FLU_[version]_linux-x86-release-64
   ./FoundryLicenseUtility -l [my license]
   ```

   For example, if you saved your license key to /tmp/Foundry.lic, the last line should be:
   ```
   ./FoundryLicenseUtility -l /tmp/foundry.lic
   ```
   This checks the license key and copies it to the correct directory.

Licensing over a Network

Obtaining Floating Licenses

You can purchase a floating license key by:
- going to our website at www.foundry.com,
- e-mailing us at sales@foundry.com,

To generate you a license key, we need to know the System ID of the machine that will act as the server. The System ID (sometimes called Host ID or rlmhostid) returns a unique number for the computer. We lock our license keys to the System ID. See Installing Floating Licenses.

To display your System ID, do the following:
- On Windows and Mac
  Download the Foundry License Utility (FLU) from support.foundry.com and run it. The System ID is displayed at the bottom of the window.
- On Linux
  Download the Foundry License Utility (FLU) from support.foundry.com and run it from the command line:
  ```
  <download location>/FoundryLicenseUtility -i
  ```

  📝 NOTE: The <download location> refers to the location where you saved the Foundry Licensing Utility.
NOTE: The System ID needs to be from the machine that will act as the server and not one of the client machines.

Just so you know what a System ID number looks like, here’s an example: 000ea641d7a1.

Installing Floating Licenses

Once a floating license has been created for you, we e-mail you a file containing the license key and instructions on how to obtain the correct version of the Foundry License Utility (FLU). Gunzip or untar the file and save the FLU and your license key to a folder of your choice.

Having installed a floating license key, you need to install some additional software (FLT) to manage the licenses on your network. Then you need to tell the client machines where to find the licenses.

On Windows and Mac

1. Just drop the license key on the Foundry License Utility (FLU) application to install it. This checks the license key and copies it to the correct directory.
   The license server address is displayed on screen:
   <number>@<license server name>
   You should make a note of the address as you’ll need it to activate the client machines.
2. In order for the floating license to work, you need to install the Foundry Licensing Tools (FLT) on the license server machine (not the client machines). For more information on how to install floating licenses, refer to the FLT user guide, which you can download from our website: support.foundry.com.
3. Once your license server is up and running, you need to direct your client machines to the server in order to obtain a license. See Telling the Client Machines Where to Find the Licenses.

On Linux

1. Navigate to the location of the FLU_[version]_linux-x86-release-64.tgz file.
2. Type the following commands to extract and install the FLU. Note that you need to replace [version] with the version of FLU you are using and [my license] with the location of your license key.
   tar xvzf FLU_[version]_linux-x86-release-64.tgz
cd FLU_[version]_linux-x86-release-64
./FoundryLicenseUtility -l [my license]

   For example, if you saved your license key to /tmp/Foundry.lic, the last line should be:
   ./FoundryLicenseUtility -l /tmp/Foundry.lic
This checks the license key and copies it to the correct directory.
The license server address is displayed on screen:
<number>@<license server name>
You should make a note of the address as you'll need it to activate the client machines.

3. In order for the floating license to work, you need to install the Foundry Licensing Tools (FLT) on the license server machine (not the client machines). For more information on how to install floating licenses, refer to the FLT user guide, which you can download from our website: support.foundry.com.

4. Once your license server is up and running, you need to direct your client machines to the server in order to obtain a license. See Telling the Client Machines Where to Find the Licenses.

Telling the Client Machines Where to Find the Licenses

In order for the client machines to get a license from the server, they need to be told where to look.

On Windows and Mac

1. Launch the Foundry License Utility (FLU).
2. Make sure you are viewing the License Install tab and copy and paste in an RLM server line:
   HOST <server name> any <port>
   For example: HOST red any 4101
   This creates and installs both a FLEXlm and an RLM client license.
3. Repeat this process for each machine you wish to have access to licenses on the server.

On Linux

1. Launch a shell and navigate to the location of the FLU_[version]_linux-x86-release-64.tgz file.
2. Type the following commands, replacing [version] with the version of FLU you are using:
   tar xvzf FLU_[version]_linux-x86-release-64.tgz
cd FLU_[version]_linux-x86-release-64
   ./FoundryLicenseUtility -c <port>@<server name>
   For example, the last line may be:
   ./FoundryLicenseUtility -c 4101@red
   This creates and installs both a FLEXlm and an RLM client license.
3. Repeat this process for each machine you wish to have access to licenses on the server.
Further Reading

There is a lot to learn about licenses, much of which is beyond the scope of this manual. For more information on licensing Ocula, displaying the System ID number, setting up a floating license server, adding new license keys and managing license usage across a network, you should read the Foundry Licensing Tools (FLT) User Guide, which can be downloaded from our website, support.foundry.com.

Other Foundry Products

Foundry is a leading developer of visual effects and image processing technologies for film and video post production. Its stand-alone products include Nuke, Modo, Mari, Hiero, Katana, and Flix. Foundry also supplies a suite of plug-ins, including Ocula, CameraTracker, Keylight, Kronos, and Furnace and FurnaceCore for a variety of compositing platforms, including Adobe® After Effects®, Autodesk® Flame®, Avid® DS™, and Apple's Final Cut Pro®. For the full list of products and supported platforms, visit our website at www.foundry.com.

Nuke is an Academy Award® winning compositor. It has been used to create extraordinary images on scores of feature films, including Avatar, District 9, The Dark Knight, Iron Man, Quantum of Solace, The Curious Case of Benjamin Button, Transformers, and Pirates of the Caribbean: At World's End.

Modo brings you the next generation of 3D modeling, animation, sculpting, effects and rendering in a powerful integrated package.

Mari is a creative texture-painting tool that can handle extremely complex or texture-heavy projects. It was developed at Weta Digital and has been used on films, such as District 9, The Day the Earth Stood Still, The Lovely Bones, and Avatar.

Hiero is a collaborative, scriptable timeline tool that conforms edit decision lists and parcels out VFX shots to artists, allowing progress to be viewed in context, and liberating your finishing systems and artists for more creative tasks.

Katana is a look development and lighting tool, replacing the conventional CG pipeline with a flexible recipe-based asset workflow. Its node-based approach allows rapid turnaround of high-complexity shots, while keeping artists in control and reducing in-house development overheads. Extensive APIs mean it integrates with a variety of renderers and your pre-existing shader libraries and workflow tools.

Flix is a collaborative, visual story-development tool. It allows directors, editors, cinematographers, storyboard artists, and pre-visualization artists to explore ideas quickly, saving valuable time, and to easily collaborate on the visual story development of a film.
Ocula is a collection of tools that solve common problems with stereoscopic imagery, improve productivity in post production, and ultimately help to deliver a more rewarding 3D-stereo viewing experience.

CameraTracker is an After Effects plug-in allowing you to pull 3D motion tracks and matchmoves without having to leave After Effects. It analyses the source sequence and extracts the original camera's lens and motion parameters, allowing you to composite 2D or 3D elements correctly with reference to the camera used to film the shot.

Keylight is an industry-proven blue/green screen keyer, giving results that look photographed, not composited. The Keylight algorithm was developed by the Computer Film Company who were honoured with a technical achievement award for digital compositing from the Academy of Motion Picture Arts and Sciences.

Kronos is a plug-in that retimes footage using motion vectors to generate additional images between frames. Utilising NVIDIA's CUDA technology, Kronos optimises your workflow by using both the CPU and GPU.

Furnace and FurnaceCore are collections of film tools. Many of the algorithms utilise motion estimation technology to speed up common compositing tasks. Plug-ins include wire removal, rig removal, steadiness, deflicker, degrain and regrain, retiming, and texture tools.
DisparityGenerator

Description

The O_DisparityGenerator node is used to create disparity maps for stereo images. A disparity map describes the location of a pixel in one view in relation to the location of its corresponding pixel in the other view. It includes two sets of disparity vectors: one maps the left view to the right, and the other maps the right view to the left.

NOTE: O_DisparityGenerator only requires an O_Solver node as one of its inputs if you intend to use the Alignment control. Alignment defaults to 0, but increasing it forces the disparity map to match the camera geometry to remove noise on the vertical component of disparity. You might want to do this if your plates don't contain much detail, such as bluescreen images with markers in the background or plates with a lot of featureless areas like sky.

The following Ocula nodes rely on disparity maps to produce their output:

- O_OcclusionDetector
- O_ColourMatcher
- O_FocusMatcher
- O_VerticalAligner (in Local Alignment mode)
- O_NewView
- O_InteraxialShifter
- O_DisparityToDepth, and
- O_DisparityViewer.

If you have more than one of these nodes in the Node Graph with one or more of the same inputs, they might well require identical disparity map calculations. O_DisparityGenerator is a utility node designed to save processing time by allowing you to create the disparity map separately, so that the results can then be re-used by other Ocula nodes.

The final disparity vectors are stored in disparity channels, so you might not see any image data appear when you first calculate the disparity map. To see the output inside Nuke, select a disparity channel from the channel controls in the top-left corner of the Viewer. Examples of what a disparity map might look like using the RGB and R channels, after adjusting the Viewer gain and gamma controls, are shown below. As you can see, the RGB layers on the left are harder to read than the single R channel.
In general, once you have generated a disparity map that describes the relation between the views of a particular clip well, it will be suitable for use in most of the Ocula nodes. We recommend that you insert a Write node after O_DisparityGenerator to render the original images and the disparity channels as a stereo .exr file (sometimes referred to as .srx). This format allows for the storage of an image with multiple views and channel sets embedded in it. Later, whenever you use the same image sequence, the disparity map is loaded into Nuke together with the sequence and is readily available for other Ocula nodes. For information on how to render disparity in to an .exr file, see Writing Disparity into a Clip.

If you have a CG scene with camera information and a z-depth map available, you can also create disparity maps using the O_DepthToDisparity node. For more information, see DepthToDisparity.

## Inputs

O_DisparityGenerator has the following inputs:

<table>
<thead>
<tr>
<th>Source</th>
<th>A stereo pair of images. If you intend to use the Alignment control, O_DisparityGenerator requires an O_Solver node as one of its inputs. Alignment defaults to 0, but increasing it forces the disparity map to match the camera geometry to remove noise on the vertical component of disparity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask</td>
<td>An optional mask that specifies areas to exclude from the disparity calculation. You can use this input to prevent distortions at occlusions or to calculate disparity for a background layer by ignoring foreground elements. Note that masks should exist in both views, and O_DisparityGenerator treats the alpha values of 1 as foreground and blurs to the 0 value using nearby disparity to recreate object boundaries, rather than image data. When you create a mask using Roto or RotoPaint, you can use the</td>
</tr>
</tbody>
</table>
feather control to extend the calculation. For example, the disparity map may have a sharper transition at depth edges with a binary mask, but applying feather on the mask can help smooth the resulting image.

The left view.  
A mask to select the dancers in the left view.

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Generating Disparity Maps

To generate a disparity map, do the following:

1. Launch Nuke and press S on the Node Graph to open the project settings. Go to the Views tab and click the Set up views for stereo button.

2. From the Toolbar, select Image > Read to load your stereo clip into Nuke. If you don’t have both views in the same file, select Views > JoinViews to combine them, or use a variable in the Read node’s file field to replace the name of the view (use the variable %V to replace an entire view name, such as left or right, and %v to replace an initial letter, such as l or r). For more information, refer to the Nuke User Guide.

3. Select Ocula > Ocula 4.0 > O_DisparityGenerator to insert an O_DisparityGenerator node after the Read node.

NOTE: If you intend to use the Alignment control, O_DisparityGenerator requires an O_Solver node as one of its inputs. Alignment defaults to 0, but increasing it forces the disparity map to match the camera geometry to remove noise on the vertical component of disparity. See Generating Disparity Using Alignment for more information.

4. Open the O_DisparityGenerator controls. O_DisparityGenerator renders using the Local GPU specified, if available, rather than the CPU. The output between the GPU and CPU is identical, but using the GPU can significantly improve processing performance.
If no suitable GPU or the required NVIDIA CUDA drivers are available, O_DisparityGenerator defaults to the CPU. You can select a different GPU Device, if available, by opening Nuke's Preferences and selecting an alternative card from the GPU Device dropdown.

**NOTE:** Selecting a different GPU requires you to restart Nuke before the change takes effect.

5. From the **Views to Use** menu or buttons, select which views you want to use for the left and right eye when creating the disparity map.

6. If there are areas in the image that you want to ignore when generating the disparity map, supply a mask either in the **Mask** input or the alpha of the **Source** input. In the O_DisparityGenerator controls, set **Mask** to the component you want to use as the mask. Note that masks should exist in both views, and O_DisparityGenerator treats the alpha values of 1 as foreground and blurs to the 0 value using nearby disparity to recreate object boundaries, rather than image data. When you create a mask using Roto or RotoPaint, you can use the **feather** control to extend the calculation. For example, the disparity map may have a sharper transition at depth edges with a binary mask, but applying feather on the mask can help smooth the resulting image.

7. Attach a Viewer to the O_DisparityGenerator node, and display one of the disparity channels in the Viewer by selecting it from the **channels** dropdown above the Viewer. O_DisparityGenerator calculates the disparity map and stores it in the disparity channels.

**TIP:** Reading depth in disparity maps can be tricky in RGB. There are a number of ways to make the depth easier to read, but the simplest is to:

- Set the Viewer **channels** control to **disparityL** and the R layer within the channel.
- Move the pointer around the image to locate the highest positive or negative red value using the Viewer info bar.
- Set the **gain** field to +/-1 divided by the red value. For example, **1/20** for positive values or **-1/20** for negative values.
• Adjust the **gain** and **gamma** controls to reveal edges and areas of contrast. Darker areas of the image are closer to the camera and lighter areas farther away. If you use a positive gain, the light and dark colors are flipped.

---

8. If the calculated disparity map does not produce the results you’re after (and you have already checked the quality of the solve as described in **Reviewing and Editing the Results**), use the O_DisparityGenerator controls to adjust the way the disparity map is calculated. You can either adjust the controls manually or use the **Preset** dropdown to automatically make adjustments for you:
   - **Custom** - automatically selected when you adjust the controls manually,
   - **Normal** - the default values for all controls.
   - **Strong** - reduces match **Stabilisation** between images and concentrates on the **Strength** of image matching.
   - **Aggressive** - increases the **Strength** to reconstruct images as close as possible to the source, but reduces **Stabilisation** in favor of accuracy.
   - **Smooth** - reduces match **Strength** between images and concentrates on **Stabilisation**.
   - **Aligned** - enables the **Alignment** control, which requires an O_Solver upstream. See **Solver** and **Generating Disparity Using Alignment** for more information.
   - **Fast** - disables the **Stabilisation** control, speeding up processing time.

The available controls are described in **O_DisparityGenerator Controls**.

9. You can also use the **Parallax Histogram** display in O_DisparityViewer to review the disparity range. For more details, see **DisparityViewer**.
TIP: To check the quality of the generated disparity map, you can add a DisparityReviewGizmo from the Ocula > Ocula 4.0 menu. This gizmo allows you to view the disparity in each view using the output and background dropdowns to control what is displayed in the Viewer. See Quality Control Tools for more information.

You can then use a RotoPaint (Draw > RotoPaint) node to edit the generated disparity channels. For example, if a specific region in the image is producing incorrect disparity vectors and you know that those vectors should match the vectors in the surrounding areas, you can use the Clone tool to clone out the problematic area.

You can also use O_MultiSample to fill or replace problematic areas. See MultiSample for more information.

Generating Disparity Using Alignment

Using O_DisparityGenerator in conjunction with O_Solver alignment data allows you to constrain the resulting disparity vectors to match global plate alignment. You might want to do this if your plates don't contain much detail, such as bluescreen images with markers in the background or plates with a lot of featureless areas like sky.

Using the O_Solver alignment data can reduce changes in vertical disparity with depth that are required for Local alignment in O_VericalAligner and cause a vertical shift in O_NewView, where disparity doesn't pick up the local vertical shift required to match the images.

To generate a disparity map using O_Solver alignment data, do the following:

1. Start Nuke and press S on the Node Graph to open the project settings. Go to the Views tab and click the Set up views for stereo button.
2. From the Toolbar, select Image > Read to load your stereo clip into Nuke. If you don't have both views in the same file, select Views > JoinViews to combine them, or use a variable in the Read node's file field to replace the name of the view (use the variable %V to replace an entire view name, such as left or right, and %v to replace an initial letter, such as l or r). For more information, refer to the Nuke User Guide.
3. Select Ocula > Ocula 4.0 > O_Solver to insert an O_Solver node after either the stereo clip or the JoinViews node.

NOTE: Solve data passed downstream to O_VericalAligner is updated to match the aligned plates, except when Vertical Skew or Local Alignment is enabled. See VerticalAligner for more information.
O_Solver calculates the geometrical relationship between the two views in the input clip and requires at least one keyframe. For more instructions on how to use O_Solver, see Solver.

4. Select Ocula > Ocula 4.0 > O_DisparityGenerator to insert an O_DisparityGenerator node after the Read node.

5. Set the Alignment control to a value greater than 1, so that disparity requires solve data upstream.

6. Continue from step 4 in Generating Disparity Maps to generate disparity vectors, including alignment data from the solve.

Writing Disparity into a Clip

When you're happy with the disparity map generated, you can save time down the line by writing the disparity into a new clip combining the source and disparity channels.

1. Select the O_DisparityGenerator node in the Node Graph.

2. Select Image > Write (or press W on the keyboard) to insert a Write node after O_DisparityGenerator.

3. In the Write node controls, select all from the channels dropdown and set file type to exr.

4. Enter a name for the clip in the file field (for example, my_clip.####.exr), and click Render.

The newly created disparity channels are saved in the channels of your stereo clip. When you need to manipulate the same clip again later, the disparity vectors are loaded into Nuke together with the clip. The disparity channels are represented on nodes in the Node Graph by the dark green chip on the right of the node.

Rendering the output to combine the clip and the disparity channels for future use.
**O_DisparityGenerator Controls**

**Use GPU**

Open the O_DisparityGenerator controls. O_DisparityGenerator renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_DisparityGenerator defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke's **Preferences** and selecting an alternative card from the **GPU Device** dropdown.

![NOTE: Selecting a different GPU requires you to restart Nuke before the change takes effect.]

**Views to Use**

From the views that exist in your project settings, select the two views you want to use to create the disparity map. These views are mapped for the left and right eye.

**Preset**

Use the **Preset** dropdown to automatically make adjustments to the disparity results by changing the appropriate refinement controls:

<table>
<thead>
<tr>
<th>Preset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Custom</strong></td>
<td>Automatically selected when you adjust the controls manually,</td>
</tr>
<tr>
<td><strong>Normal</strong></td>
<td>The default value for all controls.</td>
</tr>
<tr>
<td><strong>Strong</strong></td>
<td>Increases match <strong>Strength</strong> between images at the expense of <strong>Stabilisation</strong>.</td>
</tr>
<tr>
<td></td>
<td>You can use <strong>Strong</strong> when picture building is poor using O_NewView, O_ColourMatcher, and O_FocusMatcher.</td>
</tr>
<tr>
<td><strong>Aggressive</strong></td>
<td>Increases the <strong>Strength</strong> used to reconstruct images as close as possible to the source, but reduces <strong>Stabilisation</strong> in favor of accuracy.</td>
</tr>
<tr>
<td></td>
<td>This option gives the best result in O_NewView to reproduce the</td>
</tr>
</tbody>
</table>
appearance of one view from another and is useful for color or focus matching.

NOTE: Aggressive calculation can produce poor stability, which makes it unsuitable for copying fixes from one view to the other.

### Smooth

Reduces match **Strength** between images and concentrates on **Stabilisation**.

This option produces smoother, temporally stable vectors. Use **Smooth** to generate clean depth maps and to prevent flicker when copying a fix from one view to the other.

NOTE: Smooth vectors are cleaner and stable from one frame to the next, but the image may not reconstruct well with O_NewView.

### Aligned

This option enables the **Alignment** control, which requires an O_Solver upstream.

### Fast

This option disables the **Stabilisation** control, reducing processing time.

### Mask

An optional mask that specifies areas to exclude from the disparity calculation. You can use this input to prevent distortions at occlusions or to calculate disparity for a background layer by ignoring all foreground elements.

Note that masks should exist in both views, and O_DisparityGenerator treats the alpha values of 1 as foreground and blurs to the 0 value using nearby disparity to recreate object boundaries, rather than image data. When you create a mask using Roto or RotoPaint, you can use the **feather** control to extend the calculation. For example, the disparity map may have a sharper transition at depth edges with a binary mask, but applying feather on the mask can help smooth the resulting image.

<table>
<thead>
<tr>
<th>None</th>
<th>Use the entire image area.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Alpha</strong></td>
<td>Use the alpha channel of the <strong>Source</strong> clip as an ignore mask.</td>
</tr>
<tr>
<td><strong>Source Inverted Alpha</strong></td>
<td>Use the inverted alpha channel of the <strong>Source</strong> clip as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Luminance</strong></td>
<td>Use the luminance of the <em>Mask</em> input as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Inverted Luminance</strong></td>
<td>Use the inverted luminance of the <em>Mask</em> input as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Alpha</strong></td>
<td>Use the alpha channel of the <em>Mask</em> input as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Inverted Alpha</strong></td>
<td>Use the inverted alpha channel of the <em>Mask</em> input as an ignore mask.</td>
</tr>
</tbody>
</table>

**Vector Detail**

Adjusts the density of the calculated disparity vectors. Higher detail picks up finer disparity changes, but takes longer to calculate.

**Strength**

Sets the strength applied when matching pixels between the left and right views. Higher values allow you to accurately match similar pixels in one image to another, concentrating on detail matching even if the resulting disparity map is jagged. Lower values may miss local detail, but are less likely to provide you with the odd spurious vector, producing smoother results.

Often, it is necessary to trade one of these qualities off against the other. You may want to increase **Strength** to force the views to match where fine details are missed, or decrease it to smooth out the disparity map.

![Low Strength](image1) ![High Strength](image2)

Low **Strength** values smooth the disparity map, but can miss local detail. High **Strength** values add detail, but can cause jagged disparity.

**Consistency**

Sets how accurately the same points in the left and right views are mapped to each other. Increase the value to encourage the left and right disparity vectors to match.
Smoothness

Controls how image edges are used as clues for sharp transitions in depth in the scene. The higher the value, the smoother the transition between depths at edges in the image.

Low **Smoothness** can produce better depth definition at edges. High **Smoothness** produces smoother disparity maps, but can lose depth detail at edges.

Alignment

Sets how much to constrain the disparities to match the vertical alignment defined by an upstream O_Solver node. Increasing the value forces the disparities to be aligned and requires at least one keyframe match.

**NOTE:** The default **Alignment** value of 0 calculates the disparity using unconstrained motion estimation, and therefore, does not require O_Solver data.

Using O_DisparityGenerator in conjunction with O_Solver alignment data allows you to constrain the resulting disparity vectors to match global plate alignment. You might want to do this if your plates don't contain much detail, such as bluescreen images with markers in the background or plates with a lot of featureless areas like sky.

**NOTE:** Using the O_Solver alignment data can reduce changes in vertical disparity with depth that are required for **Local alignment** in O_VerticalAligner and cause a vertical shift in O_NewView, where disparity doesn't pick up the local vertical shift required to match the images.

Stabilisation

Sets how heavily vectors are forced to be temporally consistent by generating disparity over multiple frames, rather than a single stereo frame. Increasing the value produces smoother vectors, which are
more stable over time, but at the expense of increased processing time.

**TIP:** Decreasing the value can produce more accurate results when rebuilding views using O_NewView. See NewView for more information.

Default **Stabilisation** can lose detail at edges in the image. High **Stabilisation** produces smoother vectors, which are more stable over time.

**O_DisparityGenerator Example**

In this example, we read in a stereo image, use O_DisparityGenerator to calculate its disparity map, review the results using DisparityReviewGizmo, and render the result as a single .exr file that contains the left and the right view and the newly created disparity channels. Later, whenever you use the same image, the disparity map is loaded into Nuke together with the image. This makes the disparity map readily available for the other Ocula nodes, many of which need it to produce their output.

The stereo image used in this example can be downloaded from our website at http://www.thefoundry.co.uk/support/user-guides#ocula.

**Step by Step**

1. Start Nuke and press S on the Node Graph to open the project settings. Go to the Views tab and click the Set up views for stereo button.
2. Select Image > Read to import Dance_Group.exr. The .exr format allows both views to exist in a single file, so Nuke reads in both the left and the right view using the same Read node.
3. Select Ocula > Ocula 4.0 > O_DisparityGenerator to insert an O_DisparityGenerator node.
4. Display one of the disparity channels by selecting it from the channel set and channel menus in the upper-left corner of the Viewer.

O_DisparityGenerator calculates the disparity map. You will probably see something colorful and seemingly unreadable, much like the image below. Don't worry - that's what the disparity channels are supposed to look like.

Reading depth in disparity maps can be tricky in RGB. There are a number of ways to make the depth easier to read, but the simplest is to adjust the Viewer controls to display the results:

- Set the Viewer channels control to disparityL and the R layer within the channel.
- Move the pointer around the image to locate the highest negative red value using the Viewer info bar.
- Set the Viewer gain numeric field to \(-1/\text{<red value>}\), such as \(-1/40\).
- Adjust the Viewer gain and gamma controls to reveal edges and areas of contrast. Darker areas of the image are closer to the camera and lighter areas farther away.

Ocula also ships with a disparity checking gizmo, called DisparityReviewGizmo, to help you check that your disparity calculations are correct before you pass the data down the Node Graph.

5. Select Ocula > Ocula 4.0 > DisparityReviewGizmo to insert the gizmo after the O_DisparityGenerator node.

NOTE: To use the gizmo's full toolset, you'll also need an occlusion map. See OcclusionDetector for more information on how to generate occlusion maps.
6. The gizmo’s default settings display your stereo source image, desaturated so that the overlay generated by the gizmo is easily visible. Disparity on the X and Y axes are displayed in red and green, respectively, and occlusions are displayed in blue (if you've generated an occlusion map).

![Disparity overlay image](image)

In our example, we haven't added occlusion data (blue) and there isn't any appreciable Y axis displacement (green). The red overlay is generally picking out the changes in depth pretty well.

7. Select Image > Write to insert a Write node between the O_DisparityGenerator and the Viewer.

8. In the Write node controls, select all from the channels menu to include the disparity channels in the rendered file.

9. In the file field, enter a file name and location for the new .exr file. Make sure file type is set to exr. Then, click the Render button to render the image as usual.

10. Import the .exr file you created. Using the Viewer controls, check that it contains the left and the right view and the disparity channels.

   You can now use the .exr file you created together with many of the other Ocula nodes without having to insert an O_DisparityGenerator node before them.
OcclusionDetector

Description

OcclusionDetector generates a mask for the occluded regions in each view. Occluded regions are pixels that are visible in one view but not the other.

An occlusion mask identifies areas that cannot be rebuilt as some of the pixels in one view are not visible in the other. You can use it to quality check the result of O_DisparityGenerator and to identify image regions that are likely to fail when using the Ocula nodes that rely on rebuilding one view from the other. If the pixel information isn't there in one view, it cannot be generated for the other. After you have identified the areas which are not suitable for picture building, you can choose how to handle these in order to get the best possible result.

You may want to generate an occlusion mask for each view when using the following nodes:

<table>
<thead>
<tr>
<th>Node</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>FocusMatcher</td>
<td>This node requires an occlusion mask upstream to produce its output.</td>
</tr>
<tr>
<td>ColourMatcher</td>
<td>This node requires an occlusion mask upstream to produce its output.</td>
</tr>
<tr>
<td>DisparityGenerator</td>
<td>This node does NOT require an occlusion mask, but you can use one downstream to quality check the generated disparity map.</td>
</tr>
<tr>
<td>NewView</td>
<td>This node requires an occlusion mask upstream to produce its output.</td>
</tr>
<tr>
<td>InteraxialShifter</td>
<td>This node requires an occlusion mask upstream to produce its output.</td>
</tr>
<tr>
<td>Retimer</td>
<td>This node does NOT require an occlusion mask to produce its output, but you can use one upstream to preview where they may struggle to generate a new view.</td>
</tr>
</tbody>
</table>
The final occlusion masks for each view are stored in the `mask_occlusion` channel. You can view them in Nuke by setting the alpha channel menu to `mask_occlusion.alpha` and pressing M on the keyboard with the Viewer selected. This superimposes the occlusion mask for the current view as a red overlay on top of the image's RGB channels as shown below.

![An occlusion mask displayed on top of the color channels.](image)

After you have generated an occlusion mask, you can use a Write node to render the mask into the channels of your stereo `.exr` file along with the color and disparity channels. When you use the same image sequence at a different time, the occlusion mask is loaded into Nuke along with the sequence.

The O_OcclusionDetector requires upstream disparity channels to produce its output. For an in-depth explanation of how to create disparity channels, refer to the DisparityGenerator chapter.

## Inputs

O_OcclusionDetector has one input:

| Source | A stereo pair of images. If disparity channels are not embedded in the images, you need to add an O_DisparityGenerator node after the image sequence. |

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.
Creating and Editing Occlusion Masks

Creating an Occlusion Mask

**NOTE:** Disparity vectors are required to generate an occlusion mask.

To generate an occlusion mask, do the following:

1. If disparity vectors do not yet exist in the script, insert an O_DisparityGenerator node after your image sequence to calculate the disparity vectors. See DisparityGenerator for more information.

   **NOTE:** Disparity vectors may already exist in the image sequence, in which case you do not need to insert an O_DisparityGenerator node to generate an occlusion mask.

2. From the toolbar, select **Ocula > Ocula 4.0 > O_OcclusionDetector** to add an O_OcclusionDetector node. Insert the O_OcclusionDetector node either after the O_DisparityGenerator node (if you added one in the previous step), or the stereo image sequence. Your node tree should now look something like this:

   ![Node Tree Image]

3. Open the O_OcclusionDetector controls. O_OcclusionDetector renders using the Local GPU specified, if available, rather than the CPU. The output between the GPU and CPU is identical, but using the GPU can significantly improve processing performance. See O_OcclusionDetector Controls for more details.

4. In the O_OcclusionDetector controls, you can see all the views that exist in your project settings under **Views to Use**. Select the two views you want to use to calculate the occlusion mask. The two views you select are mapped for the left and right eye. O_OcclusionDetector calculates the occlusion mask and stores it in the **mask_occlusion.alpha** channel.

5. In the Viewer controls, set the alpha channel menu to **mask_occlusion.alpha** as shown below.

   ![Viewer Controls Image]
Set this menu to `mask_occlusion.alpha`. This sets the occlusion mask that O_OcclusionDetector generated as the channel displayed in the alpha channel.

6. Next, you can either select Matte overlay from the RGB dropdown or press M on the keyboard with the Viewer selected, to superimpose that channel as a red overlay on top of the image’s RGB channels.

```
<table>
<thead>
<tr>
<th>Viewer1</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgba</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>mask_occlusion.alpha</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Mat</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Set RBG to Matte overlay

The red overlay indicates occluded regions where picture building operations are likely to fail.

7. Adjust the Gradient Threshold, Consistency Threshold and Dilate Occlusions controls as required to generate the best possible result. See O_OcclusionDetector Controls for information about the controls.

8. When you’re happy with the results, press M again to return to the RGB display.

**Manually Editing an Occlusion Mask**

If there are occluded regions in the mask, which have not been marked correctly, you can manually edit the mask by doing the following:

1. Press P on the Node Graph to add a RotoPaint node after O_OcclusionDetector.
2. To add regions to the occlusion mask, set `output` to `mask_occlusion` in the RotoPaint controls, and use the paint tools to mark additional regions.

3. To remove or correct existing occluded regions, set `output` to `mask_occlusion` in the RotoPaint controls, and set the paint brush color to black to manually paint out or correct the occluded areas.

**Rendering Out and Loading Occlusion Masks**

To render and save the color channels, disparity channels and occlusion mask into the channels of a stereo `.exr` file, insert a Write node after the O_OcclusionDetector node. When you load the image sequence at a different time, the occlusion mask is loaded with it.

```
NOTE: If you correct occlusions using RotoPaint, insert a Write node after the RotoPaint node to include your corrections in the occlusion channel.
```

**Editing Rendered Occlusion Masks**

You can edit the rendered occlusion mask that you have loaded for use with O_FocusMatcher or O_ColourMatcher, using two different methods:

- Insert a RotoPaint node to adjust the occlusions manually with paint.
- Replace the occlusions by inserting a new O_OcclusionDetector. This overwrites the original occlusion mask. However, you can disable or delete the new O_OcclusionDetector node to revert back to the original mask.

**O_OcclusionDetector Controls**

**Use GPU**

Open the O_OcclusionDetector controls. O_OcclusionDetector renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_OcclusionDetector defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke's Preferences and selecting an alternative card from the GPU Device dropdown.

```
NOTE: Selecting a different GPU requires you to restart Nuke before the change takes effect.
```
Views to Use

From the views that exist in your project settings, select the two views you want to use to generate an occlusion mask. These views are mapped for the left and right eye.

Gradient Threshold

The gradient measures the change in depth from horizontal disparity. You can use the **Gradient Threshold** control to define where occlusions occur at depth changes. The lower the gradient threshold, the greater the number of occluded regions.

Consistency Threshold

The **Consistency Threshold** control allows you to set occlusions where the left and right disparities are not consistent. A low **Consistency Threshold** value, detects more inconsistencies.
Dilate Occlusions

You can use the **Dilate Occlusions** control to expand the occluded regions by a specified number of pixels. The maximum you can dilate the occlusions by is 20 pixels.

Occlusion Dilate = 1.  Occlusion Dilate = 15.

**O_OcclusionDetector Example**

See [O_FocusMatcher Example](#) for an example of how to use O_OcclusionDetector with O_FocusMatcher.
NewView

Description

You can use the O_NewView node to reconstruct a view – either left or right – using the pixels from the other view. For example, you can choose to reconstruct the left view using the pixels from the right view. This can be useful if you want to manipulate one view (with a gizmo, node, or graphics editor for example) and replicate the changes into the other view.

NOTE: The O_NewView node requires disparity vectors that relate the two views. If they don’t already exist, you can use the O_DisparityGenerator node to calculate these vectors. See DisparityGenerator for how to do this.

If there are no occlusions (features visible in one view but not the other), O_NewView generally produces a good result. When there are occlusions, the result may require further editing but can often save you time over not using the node at all.

If you use O_NewView to reproduce changes made to one view in the other view, you may want to create the disparity vectors using either the modified view and its corresponding view, or the original views with no changes applied. It’s recommended to choose the views that produce the best disparity vectors. For example, the former method may be preferable if you are correcting an unwanted color shift between views. The latter method may be preferable if your changes in one view produce an occlusion or a change in texture appearance, which makes the process of finding correspondences between the modified images harder.

When you are using Ocula to update one view to match another, it is advised to quality check the updated view using the DisparityReviewGizmo. See DisparityReviewGizmo for more information.

NOTE: To reproduce changes you have made using Nuke’s Roto node, RotoPaint node, or any node or gizmo that has controls for x and y coordinates, see the Stereoscopic Projects chapter in the Nuke User Guide.

Inputs

O_NewView has the following inputs:
CleanPlate

A clean background plate used to fill areas of occlusion when Correction is set to Use CleanPlate.

NOTE: If no image is connected, a Channels missing at CleanPlate input. Please connect RGB input to CleanPlate error is displayed.

Source

A stereo pair of images. If disparity channels are not embedded in the images, you need to add an O_DisparityGenerator node after the image sequence. O_NewView also requires an occlusion mask, which you can generate using an O_OcclusionDetector node.

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Creating a New View

NOTE: O_NewView requires disparity vectors and an occlusion mask to operate correctly.

To create a new view, complete the following steps:

1. If disparity vectors don’t yet exist in the script, you can insert an O_DisparityGenerator node to calculate the disparity vectors.
2. Select Ocula > Ocula 4.0 > O_OcclusionDetector to insert an O_OcclusionDetector node. Insert the O_OcclusionDetector node after either the O_DisparityGenerator node (if you added one in the previous step) or the stereo image sequence.
3. Select Ocula > Ocula 4.0 > O_NewView to insert an O_NewView node after the O_OcclusionDetector node.
4. Attach a Viewer to the O_NewView node. Your node tree should now look something like this:

5. In the O_NewView controls, select the two views you want to use under View to Use. The two views you select are mapped for the left and right eye.
6. From the View to build dropdown menu, select either Left from Right or Right from Left, depending on which view you want to rebuild.

7. Adjust the required controls to get the best possible result. See O_NewView Controls for more information.

**O_NewView Controls**

**Use GPU**

Open the O_NewView controls. O_NewView renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_NewView defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke’s Preferences and selecting an alternative card from the GPU Device dropdown.

![NOTE: Selecting a different GPU requires you to restart Nuke before the change takes effect.]

**Views to Use**

From the views that exist in your project settings, select the two views you want to use to generate the new view. These views are mapped for the left and right eye.

**View to Build**

Select which inputs to use to generate the new view.

<table>
<thead>
<tr>
<th>Left from Right</th>
<th>Use the pixels from the right view to build a new left view.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right from Left</td>
<td>Use the pixels from the left view to build a new right view.</td>
</tr>
</tbody>
</table>

**Pass through other view**

Select the Pass through other view checkbox to output both views; the new view and the original source view. If this checkbox is disabled, only the new view is output.
Occlusions

Output occlusions to alpha

Select the **Output occlusions to alpha** checkbox to output the occlusions to the alpha channel. Occlusions occur when some pixels are not visible in the source view, and therefore cannot be used to create the new view. You can use the alpha channel as an overlay to determine where the occlusion correction is applied.

Correction

You can use the **Occlusions – Correction** control to determine how occluded regions are dealt with.

| **Use original** | Select this option to retain the original view in the occluded regions. For example, if you are using the original left view to build a new right view, the original right view is retained in the occluded regions to help build the new right view. |
| **Expand foreground** | Select this option to fill occluded regions by expanding the surrounding area from the original source view. For example, if you are using the original left view to build a new right view, the foreground of the original left view is expanded to help build the new right view. |
| **Use CleanPlate** | Select this option to fill occluded areas using a clean background, which is connected using the **CleanPlate** input. |
| **None** | Select this option to avoid filling the occluded regions. |

**NOTE:** Expanding the foreground may offset the edges.

**NOTE:** If no image is connected, a **Channels missing at CleanPlate input. Please connect RGB input to CleanPlate** error is displayed.
Edges

Output edges to alpha

You can use the **Output edges to alpha** checkbox to output the edges to the alpha channel. Use the alpha channel as an overlay to determine where the edge correction is applied.

Correction

You can use the **Edges – Correction** control to determine how image edges at depth boundaries are handled.

<table>
<thead>
<tr>
<th>Match original</th>
<th>Select this option to match the appearance of the original view at the edges. For example, if you are using the original left view to build a new a right view, the edges are matched from the original right view to help build the new right view.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match foreground</td>
<td>Select this option to match the edges from the original source view that was used to build the new view. For example, if you are using the original left view to build a new right view, the edges are matched from the original left view to help build the new right view.</td>
</tr>
<tr>
<td>None</td>
<td>Select this option to avoid applying edge correction.</td>
</tr>
</tbody>
</table>

Adjust Edges

You can use the **Adjust Edges** control to set the extent of the region where the edge correction is applied. To blend the correction into the background, use a positive value. To restrict the correction to the edges, use a negative value.

O_NewView Example

In this example, we have a stereo image of a cathedral. In the left view, one of the cathedral windows is missing. Our aim is to reproduce the missing window from the right view into the left view using O_NewView. To do this, we can use O_NewView to produce a completely new left view, using the pixels from the right view.

The stereo image used in the example can be downloaded from our website. For more information, see Example Images.
The necessary disparity channels have been embedded in the download image, so you don’t need to insert an O_Solver node and an O_DisparityGenerator node in this example.

**Step by Step**

1. Launch Nuke and open the project settings (press S on the Node Graph). Go to the Views tab and click the Set up views for stereo button.
2. Import cathedral1.exr and attach a Viewer to the image. Switch between the left and the right view using the Viewer controls (or the ; and ' hotkeys). Notice that the cathedral is missing a window in the left view, as shown below.

![The left view.](image1.png) ![The right view.](image2.png)

We want to build a new left view to replicate the additional window, using the O_NewView node.
3. Select Ocula > Ocula 4.0 > O_OcclusionDetector to insert an O_OcclusionDetector node after the stereo image.
4. Select Ocula > Ocula 4.0 > O_NewView to insert an O_NewView node after the O_OcclusionDetector node.

5. In the O_NewView controls, select Left from Right from the View to Build menu to generate the new left view using the right view as a source. The image below shows the new left view. As you can see, it now includes the window that was previously missing.
6. Select the **Pass through other view** checkbox. This means both the new view and the original source view are output.

7. Using the Viewer controls, switch between the left and the right views. The window that was previously missing from the left view, is now present in both views.
ColourMatcher

Description

The O_ColourMatcher node enables you match the colors of one view with those of another. It has been specifically designed to deal with the subtle color differences that are sometimes present between stereo views.

Color discrepancies between views can be caused by several factors. For example, stereo footage may have been shot with cameras that had different polarization, or there may have been slight differences between the physical characteristics of the two camera lenses or image sensors. If the color differences are not corrected, viewers may experience difficulty in fusing objects and as a result may not enjoy the viewing experience.

Correcting color differences manually in post-production can be a time-consuming process and requires considerable skill. O_ColourMatcher enables you to automate the color grading required.

O_ColourMatcher has two different modes you can use to perform a color match; Basic mode and Local Matching mode. Both modes require an O_DisparityGenerator node and an O_OcclusionDetector node upstream of the O_ColourMatcher node.

Basic Mode

The Basic color matching mode takes the color distribution of one entire view and modifies it to match the distribution of the other view.

Local Matching Mode

The Local Matching mode first divides the two images into square blocks according to the Block Size control. Then, it matches the color distribution from the view that want to modify to a reconstructed version of the same view, which has been constructed using the pixels of the source view. When an
occluded pixel is detected by an upstream occlusion mask, O_ColourMatcher finds the closest unoccluded pixel and then uses this to make the color match for the occluded pixel.

**Local Matching** mode can be useful if there are local color differences between the views, such as highlights that are brighter in one view than the other.

### Inputs

O_ColourMatcher has the following inputs:

<table>
<thead>
<tr>
<th>Source</th>
<th>A stereo pair of images. If disparity channels and occlusion masks are not embedded in the images and you are using the <strong>Local Matching</strong> mode, you need to add an O_Solver, an O_DisparityGenerator, and an O_OscillationDetector node after the image sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask</td>
<td>An optional mask that determines where to take the color distribution from. For example, if you have a clip showing a person in front of a green screen, you might want to use a mask to exclude the green area so the node concentrates on matching the person. In the <strong>Basic</strong> mode, O_ColourMatcher calculates the transform on the masked area of the source view but applies it to the whole of the view it's correcting. In the <strong>Local Matching</strong> mode, it calculates the transform on the masked area and applies it to that area only.</td>
</tr>
</tbody>
</table>

To see a table listing the nodes or channels each Ocula node requires in its inputs, turn to **Appendix A: Node Dependencies**.

### Performing a Color Match

O_ColourMatcher has two different modes you can use to perform a color match: **Basic** mode and **Local Matching** mode. Both modes require an O_DisparityGenerator and an O_OscillationDetector node upstream of the O_ColourMatcher node.

To perform a color match, complete the following steps:

1. If they don't exist already, insert an O_DisparityGenerator node and an O_OscillationDetector node after your stereo clip. See **DisparityGenerator** and **OcclusionDetector**

2. Select **Ocula > Ocula 4.0 > O_ColourMatcher** to insert an O_ColourMatcher node after the O_DisparityGenerator and O_OscillationDetector nodes.
3. Connect a Viewer to the O_ColourMatcher node. Your node tree should now look something like this:

![O_ColourMatcher node tree.](image)

4. In the O_ColourMatcher controls, select the two views you want to use for the color match under **View to Use**. The two views you select are mapped for the left and right eye.

5. From the **Match** menu, select either **Left to Right** or **Right to Left** depending on the direction in which you want to perform the color match.

6. Select the required mode from the **Mode** dropdown in the O_ColourMatcher controls. See **O_ColourMatcher Controls** for more information about the different modes.

7. Adjust the O_ColourMatcher controls to get the best possible result. See **O_ColourMatcher Controls** for more information about the controls.

### Editing Color Match Results

If you can see areas where the color match is wrong, make sure they are included in the upstream occlusion mask. You can edit the occlusion mask in two ways:

- Adjust O_OcclusionDetector controls.
- Use a RotoPaint node before O_ColourMatcher and manually edit the mask by using the paint tool to add occluded regions into the **mask_occlusion** channel.

### O_ColourMatcher Controls

#### Use GPU

Open the O_ColourMatcher controls. O_ColourMatcher renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_ColourMatcher defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke’s **Preferences** and selecting an alternative card from the **GPU Device** dropdown.

- **NOTE:** Selecting a different GPU requires you to restart Nuke before the change takes effect.
Views to Use

From the views that exist in your project settings, select the two views you want to use to generate an occlusion mask. These views are mapped for the left and right eye.

Match

The Match control allows you to specify the direction in which to perform the color match.

| Left to Right | Adjust the colors of the left view to match with those of the right. |
| Right to Left | Adjust the colors of the right view to match with those of the left. |

Mode

You can use two different modes to perform a color match:

| Basic | This mode takes the color distribution of one entire view and modifies that to match the distribution of the other view. |
| Local Matching | This mode first divides the two images into square blocks according to the Block Size control. Then, it matches the color distribution from the view that wants to modify to a reconstructed version of the same view, which has been constructed using the pixels of the source view. This can be useful if there are local color differences between the views, such as highlights that are brighter in one view than the other. |

NOTE: Both modes require an O_DispparityGenerator node and an O_OcclusionDetector node upstream of the O_ColourMatcher node.

Mask

If there are areas in the image that you want to ignore when calculating the color transformation, you can use the Mask control to supply a mask. In the O_ColourMatcher controls, set Mask to the component you want to use as the mask. The following Mask settings are available:
<table>
<thead>
<tr>
<th>None</th>
<th>Use the entire image area.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Alpha</td>
<td>Use the alpha channel of the Source clip as a mask.</td>
</tr>
<tr>
<td>Source Inverted Alpha</td>
<td>Use the inverted alpha channel of the Source clip as a mask.</td>
</tr>
<tr>
<td>Mask Luminance</td>
<td>Use the luminance of the Mask input as a mask.</td>
</tr>
<tr>
<td>Mask Inverted Luminance</td>
<td>Use the inverted luminance of the Mask input as a mask.</td>
</tr>
<tr>
<td>Mask Alpha</td>
<td>Use the alpha channel of the Mask input as a mask.</td>
</tr>
<tr>
<td>Mask Inverted Alpha</td>
<td>Use the inverted alpha channel of the Mask input as a mask.</td>
</tr>
</tbody>
</table>

NOTE: You can also use the alpha channel of the Source input to supply a mask.

View Menu Button

The View menu button allows you to select different settings for the left and right view. It is displayed to the right of a control that it can be applied to. See Block Size further in this section for an example of how to use it.

Local Matching

NOTE: These settings are only available with the Local Matching mode selected.

Preview colour correction

Enable this control to preview the areas of color correction applied to the original image as a difference overlay.

Block Size

This control defines the width and height (in pixels) of the square blocks that the images are divided into when calculating the color match. You can set different block sizes for each view by doing the following:
1. Click the View menu button to the right of the block size.
2. Select Split off left.
This displays two fields for **Box Size**, one for the left view and one for the right view.

3. To revert back to using the same size for both views, click the **View menu** button again and select **Unsplit left**.

**Scale**

Set the image scale for local color matching. You can increase the **Scale** to broaden the color update and preserve the image structure, helping to prevent image shift and wobble. Decrease the **Scale** to pick up highlights and detailed color changes.

**Limit**

Sets a limit on local color matching against the average correction in a region. If you notice excessive color changes in areas of highlight, try reducing the **Limit**.

**Noise**

The **Noise** control allows you to set how much noise to retain from the original image. If the color matching smooths the input noise, increase the **Noise** value. If the value is too high, color differences at very fine details are retained.

**Occlusions**

- **NOTE**: These settings are only available with the **Local Matching** mode selected.

**Output corrected area to alpha**

Select the **Output corrected area to alpha** checkbox to output the corrected area to the alpha channel. The corrected area can consist of the occlusion mask and the disparity edge mask set using the **Adjust Edges** control, depending on what the **Correction** control is set to (Occlusions, Occlusions and Edges, or None).
Correction

Defines which areas receive the color correction when local color matching is not valid.

<table>
<thead>
<tr>
<th>Occlusions</th>
<th>Fill occluded pixels only, where color is missing from the other view.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occlusions and Edges</td>
<td>Fill occluded pixels, where color is missing from the other view, and compensate for disparity changes at edges where matching and/or reconstruction can fail.</td>
</tr>
<tr>
<td>None</td>
<td>Apply no occlusion or edge correction.</td>
</tr>
</tbody>
</table>

Adjust Edges

The Adjust Edges control allows you to set the threshold for treating image edges as occlusions to reduce haloing and edge flicker. The higher the value, the more image edges are considered occlusions even if they are not marked as such in the upstream occlusion mask.

Colour Tolerance

The Colour Tolerance control allows you to set the amount of blurring across edges in the color match at occluded regions. Decrease this to restrict the color correction in occluded regions to similar colors. Increase the value to blur the color correction.

Support Size

Use the Support Size control to set the size of the region (in pixels) of unoccluded pixels used to calculate the color correction at an occluded pixel. O_ColourMatcher first finds the closest unoccluded pixel and then expands that distance by this number of pixels to determine the amount of unoccluded pixels to use.

Stabilise Occlusions

Enabling this control can reduce flicker in occluded areas by using data from multiple frames.

**O_ColourMatcher Example**

In this example, there is a subtle color discrepancy between our stereo views and reflections that are present in one view but not the other. To fix this, we need to match the colors of the right view with
those of the left using the **Local Matching** mode. The stereo image used in the example can be downloaded from our website. For more information, please see **Example Images**.

### Step by Step

1. Start Nuke and open the project settings by pressing **S** on the Node Graph. Select the **Views** tab and click the **Set up views for stereo** button.

2. Import the **dance_group_disp.exr** footage. This image already includes both the left and the right view, and the necessary disparity channels.

3. Attach a **Viewer** to the image. Using the Viewer controls (or the ; and ' hotkeys), switch between the left and the right view. As you can see, there is a subtle color difference between the views.

   ![The original left view.](image1) ![The original right view.](image2)

   We are going to match the colors of the left view with those of the right.

4. Select **Ocula > Ocula 4.0 > O_OcclusionDetector** to add an O_OcclusionDetector node after the stereo images.

5. Insert an **O_ColourMatcher** node after O_OcclusionDetector by selecting select **Ocula > Ocula 4.0 > O_ColourMatcher**.

   ![The node tree with O_ColourMatcher.](image3)

6. In the O_ColourMatcher controls, you can select either **Left to Right** or **Right to Left** from the **Match** dropdown, depending on the direction in which you want to perform the color match. The **Match** menu is already set to **Left to Right**, which is what we want.

7. Set **Mode** to **Local Matching**.

   In this mode, O_ColourMatcher first divides the two images into square blocks according to the **Block Size** control. Then, it matches the color distributions between corresponding blocks in the two views.
As the **Occlusion** options are enabled in the **Local Matching** mode, O_ColourMatcher can produce better results in the occluded areas defined by the upstream occlusion mask. In these areas, O_ColourMatcher cannot correct the color in one view by using the color from the other. Instead, it looks for similar colors in the nearby unoccluded areas that it has already been able to match and uses the closest color it finds.

8. View the result and switch between the two views again. Compare the new left view to the original left view by displaying the left view in the Viewer, selecting the O_ColourMatcher node, and pressing D a couple of times to disable and enable the node. Notice that the colors of the left view now match those of the right, but there are some artifacts in the middle of the image.

![The color corrected left view.](image)

9. To fix this, we are going to add more areas to the occlusion mask. Press P on the Node Graph to add a RotoPaint node after O_OcclusionDetector.

![The node tree with RotoPaint.](image)

10. In the Viewer controls, set the alpha channel menu to `mask_occlusion.alpha` as shown below.
This sets the occlusion mask that O_OcclusionDetector generated as the channel that is displayed in the alpha channel.

Next, press M on the keyboard with the Viewer selected to superimpose that channel as a red overlay on top of the image’s RGB channels.

The occlusion mask in a Viewer overlay.

11. Open the RotoPaint controls and set output to mask_occlusion. Activate the Brush tool in the Viewer toolbar and paint over any areas that were producing poor results. If it helps, press D on the O_ColourMatcher node to disable it and stop it updating after each paint stroke. Pressing D again re-enables the node.
Adding more areas to the occlusion mask.

12. Press M on the Viewer to hide the occlusion mask overlay.

13. Enable and disable O_ColourMatcher to compare the original and the color corrected view. The results should now be more accurate. If you still see some problematic areas, you can add them to the occlusion mask too or adjust the Local Matching and Occlusion controls in the O_ColourMatcher Properties.

The final left view.  The original right view.
FocusMatcher

Description

O_FocusMatcher is designed to correct subtle focus differences that are sometimes present between the left and right views of a stereo image. It does this by matching the focus distribution of one view to the other, based on the disparity vectors upstream. For details on how to calculate disparity vectors, see DisparityGenerator.

The focus matching can be done using two different modes; the Match Edges mode and the Reconstruct Edges mode.

- **Match Edges Mode**
  
  The Match Edges mode matches the appearance of the edges from one view to the other. If you want to preserve the original image structure, it is recommended to use this mode. Also, if the blurring in your input images is subtle, the Match Edges mode may produce the best result.

- **Reconstruct Edges Mode**
  
  The Reconstruct Edges mode rebuilds the edges in one view from scratch, using the pixels from the other. If you want to rebuild focus exactly, or if the blurring in your image is heavy or varying, it is recommended that you use this mode.

  **NOTE:** Both modes require an O_DisparityGenerator node and an O_OcclusionDetector node upstream of the O_FocusMatcher node.

  **NOTE:** The result of this mode depends on the accuracy of the existing disparity vectors.
Inputs

O_FocusMatcher has the following inputs:

<table>
<thead>
<tr>
<th>Source</th>
<th>A stereo pair of images. If disparity channels and occlusion masks are not embedded in the images, you need to insert an O_DisparityGenerator and an O_OcclusionDetector node after the image sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask</td>
<td>An optional mask that determines where to perform the focus matching calculation.</td>
</tr>
</tbody>
</table>

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Performing a Focus Match

O_FocusMatcher has two different modes you can use to perform a color match: Match Edges mode and Reconstruct Edges mode. Both modes require an O_DisparityGenerator node and an O_OcclusionDetector node upstream of the O_FocusMatcher node.

To perform a focus match, complete the following steps:

1. If they don't exist already, insert an O_DisparityGenerator node and an O_OcclusionDetector node after your stereo clip. See DisparityGenerator and OcclusionDetector for more information.
2. From the toolbar, select Ocula > Ocula 4.0 > O_FocusMatcher to insert an O_FocusMatcher node after the O_OcclusionDetector node. Your node tree should now look something like this:

   ![Node Tree Diagram]

3. Select the required mode from the Mode dropdown in the O_FocusMatcher controls. You can use the Match Edges mode or the Reconstruct Edges mode. See O_FocusMatcher Controls for more details about the different modes.
4. Select the two views you want to use for the color match under View to Use. The two views you select are mapped for the left and right eye.
5. From the **Match** menu, select either **Left to Right** or **Right to Left** depending on the direction in which you want to perform the focus match.

6. Adjust the **Local Matching** and **Occlusion** controls to get the best possible result. See **O_FocusMatcher Controls** for more information about the controls.

**O_FocusMatcher Controls**

**Use GPU**

Open the O_FocusMatcher controls. O_FocusMatcher renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_FocusMatcher defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke's **Preferences** and selecting an alternative card from the **GPU Device** dropdown.

⚠️ **NOTE:** Selecting a different GPU requires you to restart Nuke before the change takes effect.

**Views to Use**

From the views that exist in your project settings, select the two views you want to use to generate an occlusion mask. These views are mapped for the left and right eye.

**Match**

The Match control allows you specify the direction in which to perform the focus match in.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Left to Right</strong></td>
<td>Deblur or rebuild the left view to match the right.</td>
</tr>
<tr>
<td><strong>Right to Left</strong></td>
<td>Deblur or rebuild the right view to match the left.</td>
</tr>
</tbody>
</table>

**Mode**

You can use two different modes to perform a focus match:
Match Edges

The **Match Edges** mode matches the appearance of the edges from one view to the other. If you want to preserve the original image structure, it is recommended to use this mode. If the blurring in your input images is subtle, the **Match Edges** mode may produce the best possible result.

Reconstruct Edges

The **Reconstruct Edges** mode rebuilds the edges in one view from scratch using the pixels from the other. If you want to rebuild focus exactly, or if the blurring in your image is heavy or varying, it is recommended that you use this mode. The result of this mode depends on the accuracy of the existing disparity vectors.

Mask

If there are areas in the image that you want to ignore when calculating the focus calculation, you can use the **Mask** control to supply a mask. In the O_FocusMatcher controls, set **Mask** to the component you want to use as the mask. The following **Mask** settings are available:

<table>
<thead>
<tr>
<th>None</th>
<th>Use the entire image area.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Alpha</td>
<td>Use the alpha channel of the <strong>Source</strong> clip as a mask.</td>
</tr>
<tr>
<td>Source Inverted Alpha</td>
<td>Use the inverted alpha channel of the <strong>Source</strong> clip as a mask.</td>
</tr>
<tr>
<td>Mask Luminance</td>
<td>Use the luminance of the <strong>Mask</strong> input as a mask.</td>
</tr>
<tr>
<td>Mask Inverted Luminance</td>
<td>Use the inverted luminance of the <strong>Mask</strong> input as a mask.</td>
</tr>
<tr>
<td>Mask Alpha</td>
<td>Use the alpha channel of the <strong>Mask</strong> input as a mask.</td>
</tr>
<tr>
<td>Mask Inverted Alpha</td>
<td>Use the inverted alpha channel of the <strong>Mask</strong> input as a mask.</td>
</tr>
</tbody>
</table>

**NOTE:** You can also use the alpha channel of the **Source** input to supply a mask.
Local Matching

Edge Scale
This allows you to scale the edges where focus matching is performed. To restrict matching to sharp edges, use a small scale value. To match wider edges in the image, increase the scale value.

Strength
You can use the Strength control to set the amount of focus correction to apply. Set this to 0 for no correction, or 1 for complete correction.

Noise

NOTE: This option is only available when the Reconstruct Edges mode is selected.

The Noise control allows you to preserve the noise of the original image when using the Reconstruct Edges mode. To ignore noise, set this control to a low value. This matches the focus of fine details and can reconstruct noise from the other view. To retain as much of the original noise as possible, use higher values. This ensures that noise is not coherent between views, but may not match the focus at fine details in the image.

Occlusions

Output occlusions to alpha
Select the Output occlusions to alpha checkbox to output the corrected area to the alpha channel. The corrected area can consist of the occlusion mask and the disparity edge mask set using the Adjust Edges control, depending on what the Correction control is set to (Occlusions, Occlusions and Edges, or None).

Correction
Defines which areas receive the focus correction when local focus matching is not valid.

<p>| Occlusions | Fill occluded pixels only, where color is missing from the other view. |</p>
<table>
<thead>
<tr>
<th>Occlusions and Edges</th>
<th>Fill occluded pixels, where color is missing from the other view, and compensate for disparity changes at edges where matching can fail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Apply no occlusion or edge correction.</td>
</tr>
</tbody>
</table>

Adjust Edges

The **Adjust Edges** control allows you to set the threshold for treating image edges as occlusions to reduce haloing and edge flicker. The higher the value, the more image edges are considered occlusions, even if they aren't marked as such in the upstream occlusion mask.

Colour Tolerance

The **Colour Tolerance** control allows you to set the amount of blurring across edges in the focus match at occluded regions. Decrease this to restrict the color correction in occluded regions to similar colors. Increase the value to blur the focus correction.

Support Size

Use the **Support Size** control to set the size of the region (in pixels) of unoccluded pixels used to calculate the focus correction at a corrected pixel.

Stabilise occlusions

Enabling this control can reduce flicker in occluded areas by using data from multiple frames.

**O_FocusMatcher Example**

In this example, we correct the focus distribution in the right view of a stereo image by rebuilding it using the pixels from the left view.

You can download the stereo image used here from our website - please see **Example Images**.

**Step by Step**

1. Launch Nuke. Open the project settings (press S on the Node Graph), select the **Views** tab, and click the **Set up views for stereo** button.
2. Import **lobby.exr**. This image already includes both the left and the right view as well as the necessary disparity channels.

3. Attach a Viewer to the image and zoom in. Switch between the left and the right view using the Viewer controls (or the ; and ' hotkeys). As you can see, the focus distribution of the right view doesn't match that of the left.

![The left view.](image1.jpg) ![The right view.](image2.jpg)

O_FocusMatcher can fix this by rebuilding the right view using pixels from the left. In order to do so, it needs an upstream occlusion mask that identifies occluded pixels in each view. You can generate an occlusion mask using the O_OcclusionDetector node.

4. From the toolbar, select **Ocula > Ocula 4.0 > O_OcclusionDetector** to insert an O_OcclusionDetector node between the image and the Viewer.

![O_OcclusionDetector node](image3.png)

O_OcclusionDetector calculates a mask for the occluded pixels in each view and stores it in the **mask_occlusion** channel. You can adjust the mask using the O_OcclusionDetector controls, but for this example, we're going to go with the default settings.

5. Next, add an O_FocusMatcher node after O_OcclusionDetector.

![O_FocusMatcher node](image4.png)

6. By default, O_FocusMatcher is set to rebuild the left view to match the focus of the right. We need it to do the opposite, so set the **Match** menu to **Right to Left**.

O_FocusMatcher now rebuilds the right view using pixels from the left. If the upstream disparity map is accurate and there are no occlusions (pixels visible in one view but not the other), this generally produces good results. We have already generated an occlusion mask in step 4, so we can use it to check which areas are occluded.
7. To see the occlusion mask for the right view, select the right view from the Viewer controls and set the alpha channel menu to `mask_occlusion.alpha`. Then, press M on the Viewer.

![Viewer controls](image)

The occlusion mask is shown in a red overlay on top of the color channels. Any pixels highlighted in red are only visible in the right view but not the left.

![Occlusion mask in red overlay](image)

Because these pixels don’t exist in the left view, they cannot be used to rebuild the right view. In other words, O_FocusMatcher is likely to produce poor results in these occluded areas.

8. To get the best result, adjust the **Local Matching** and **Occlusion** controls. If required, you can manually edit the occlusion mask using a RotoPaint node (see Creating and Editing Occlusion Masks).

9. You now have your final result, so compare the rebuilt right view to both the original right view and the left view in the Viewer. You should see that the focus distribution of the right view better matches that of the left.
Solver

Introduction

The O_Solver node defines the geometric relationship between the two views in the input images (that is, the camera relationship or solve). This is necessary when aligning footage with O_VerticalAligner. It is also required to calculate aligned disparity vectors, when using the alignment control in DisparityGenerator. O_Solver data is not necessary for color and focus matching using O_ColourMatcher and O_FocusMatcher.

To define the camera relationship, O_Solver detects a number of features in one view and locates the corresponding features in the other (see the image below). The feature matches and analysis data are not available until you have set at least one analysis key on O_Solver. Any frames set as analysis keys show up on the Viewer timeline and can be visualized in the Curve Editor and Dope Sheet.

O_Solver detects features in each view and tries to match them.

O_Solver calculates alignment data at the keyed analysis frames. Alignment at other frames is created by interpolating between the results at the analysis frames. This ensures that the alignment data delivered to O_DisparityGenerator and O_VerticalAligner varies smoothly across the sequence.

TIP: If you have an interactive license (ocula_i and nuke_i), you can run O_Solver from the terminal to automatically set up analysis frames. Running from the terminal also removes the need to manually set up an Ocula node tree. See Solving Using Python for more information.

The output of the O_Solver node consists of:
• the unaltered input images, and
• the results of the feature detection and analysis, which are passed down the node tree as hidden metadata.

Because the results of the analysis are available downstream, you can use multiple Ocula nodes in the tree without having to re-analyze the camera relationship. However, if a node generates or modifies views, the current metadata becomes invalid and is removed from the tree from that point forward.

To get the best possible results, you can identify features to ignore in the analysis. This can be done by supplying a mask in the **Ignore** input.

You can also add your own feature matches to the automatically detected ones. O_Solver considers any feature matches you’ve added yourself superior to the ones it detects automatically and pays them more attention. This can also influence which of the automatically detected features are included in the final solve. To force the feature matches to be recalculated based on the manual feature matches, use the **Re-analyse Frame** button.

If you have a pre-tracked camera that describes the camera setup used to shoot the images, you can also supply this in the **Camera** input. If you connect the Camera node before adding a keyframe, the automatically-detected feature matches are validated against the input camera. Alternatively, you can add the Camera node after the analysis and use the **Re-analyse Frame** button to recalculate matches based on the input camera. For more information, see **Inputs** below.

| TIP: You can improve the alignment data calculated by O_Solver by adding user matches. This can be used to correct O_VerticalAligner in tricky shots, where there are few automatic matches, such as on bluescreen or greenscreen footage. See **Solver** for more information. |

### Inputs

O_Solver has the following inputs:

<table>
<thead>
<tr>
<th><strong>Camera</strong></th>
<th>A pre-tracked Nuke stereo camera that describes the camera setup used to shoot the <strong>Source</strong> image. This can be a camera you have tracked with the CameraTracker node or imported to Nuke from a third-party camera tracking application. This input is optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TIP:</strong> In Nuke, a stereo camera can be either:</td>
<td></td>
</tr>
</tbody>
</table>
Solving the Camera Relationship

1. Launch Nuke and press S on the Node Graph to open the project settings. Go to the Views tab and click the Set up views for stereo button.

2. From the Toolbar, select Image > Read to load your stereo clip into Nuke. This can either be the clip you want to work on, or another clip shot with the same camera setup.

   If you don't have both views in the same file, select Views > JoinViews to combine them, or use a variable in the Read node's file field to replace the name of the view (use the variable %V to replace an entire view name, such as left or right, and %v to replace an initial letter, such as l or r). For more information, refer to the Nuke User Guide.

<table>
<thead>
<tr>
<th>Ignore</th>
<th>A mask that specifies areas to ignore during the feature detection and analysis. This can be useful if an area in the Source image is producing incorrectly matched features. This input is optional.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>A stereo pair of images. These can either be the images you want to work on, or another pair of images shot with the same camera setup.</td>
</tr>
</tbody>
</table>

To see what data each Ocula node requires in its inputs, see Appendix A: Node Dependencies.
3. Select **Ocula > Ocula 4.0 > O_Solver** to insert an O_Solver node after either the stereo clip or the JoinViews node (if you inserted one in the previous step).

4. Connect a Viewer to the O_Solver node.

![Node tree with O_Solver](image)

The node tree with O_Solver.

5. Open the O_Solver controls. Under **Views to Use**, you can see all the views that exist in your project settings. Select the two views you want to use for the left and right eye when calculating the camera relationship.

   The two views you selected are mapped for the left and right eye.

6. If you have a pre-tracked Nuke stereo camera that describes the camera setup used to shoot the Source image, connect that to O_Solver’s **Camera** input. O_Solver uses the camera information to calculate the relationship between the two views.

7. Set keyframes on your sequence for O_Solver to analyze:
   - **Key Frame** - if the camera rig doesn’t change, you can set analysis keys only on one or two frames (for example, on the first and last frames). If you do set a few analysis keys, you can also check **Single Solve From All Keys** in the O_Solver controls. This tells O_Solver to calculate a single solve using all keyframes, which can improve the results.
   - **Key Sequence** - click to analyze the whole sequence automatically. This adds analysis keys where a change in camera alignment is detected.
   - **Key Nominated** - click to analyze the frames specified in the **Render** dialog. You can specify frames using Nuke’s regular frame expressions. For example, if you enter "1-5 8 10 15 22-25", only those 12 frames are keyed.

   If you know there is a zoom or change in the camera setup on certain frames, you need to add more keyframes in between. Leave **Single Solve From All Keys** unchecked to use a separate solve for each analysis key, and place keyframes where the camera alignment changes.

   **NOTE:** You can remove analysis keys one at a time by scrubbing the playhead to the required frame and clicking **Delete Key** or remove all the analysis keys from the sequence by clicking **Delete All**.

   O_Solver analyzes the frames you added and, if it finds more than one analysis key, it interpolates the results between them. Interpolating between analysis keys ensures that the calculated camera relationship varies smoothly across the sequence.
To visualize the analysis in the Curve Editor or Dope Sheet, right-click on the **Analysis Key** field and select **Curve editor** or **Dope sheet**. Note, however, that you cannot edit the curve in either.

8. Proceed to **Reviewing and Editing the Results**.

**NOTE:** Once O_Solver has detected feature matches, they are fixed and do not update in response to changes in the node tree. You can edit them manually, however, or click **Re-analyse Frame** to force O_Solver to recalculate the current frame.

### Solving Using Python

If you have an interactive license (ocula_i and nuke_i), you can run O_Solver using Python. Running from the command line, with the `-i` argument for interactive licenses, allows you to automate the setup of O_Solver for different shots without having to open the UI and manually analyze frames.

**WARNING:** If there is an interruption between the license server and Ocula, rendering aborts with an exit code of 1. You can use the `--cont` command line argument to force Ocula to continue rendering on failure, producing black license failure frames rather than aborting the whole render.

Using Python, you can:

- Scan the entire sequence automatically, adding keyframes when a change in camera is detected using `analyseSequence`.

```python
# Create an O_Solver to analyze the whole sequence
def autoKeyOculaTree(filename, first_frame, last_frame):
    # set up views
    nuke.root()("setlr").execute()

    # create the read and set up for the frame range
    reader = nuke.createNode("Read", inpanel=False)
    reader.knob("file").setValue(filename)
    reader.knob("first").setValue(first_frame)
    reader.knob("last").setValue(last_frame)

    # set up the O_Solver node and create a key
    solver = nuke.createNode('O_Solver4_0', inpanel=False)
    solver.setInput(0, reader)
    solver['analyseSequence'].execute()
```
# set the file path and frame range
autoKeyOculaTree('/myFilePath/myFootage.####.srx', 1, 206)

# save the script once complete
nuke.scriptSaveAs('/Users/OculaExpert/shot001.nk', True)

• Script the analysis of a specific set of frames through python.

    
    frameList = [25, 73, 123]
    frameRanges = nuke.FrameRanges(frameList)
    nuke.execute(solver, frameRanges)

O_Solver also has a **Python** tab in the Properties panel, allowing you to call Python functions automatically when various events happen in Nuke. See **Help > Documentation > Python Developers Guide** for more information.

### Reviewing and Editing the Results

1. Set **Display** to **Keyframe Matches**, if it's not displaying them already, and make sure you are viewing a keyframe.

   The features and matches used to calculate the camera relationship are shown in a Viewer overlay. The views set up in the **Project Settings** dictate the color of the features in the overlay. If you used **Set up views for stereo** to create the views, red indicates a feature in the left view and green a feature in the right view.

2. You can specify areas of the footage to ignore using a mask in either the **Ignore** input or the alpha of the **Source** image. In the O_Solver controls, set **Mask** to the component you want to use as the mask.

   **NOTE:** Features generated on reflections often produce bad feature matches, but you can add user matches around reflective areas if the auto-matches are poor. See **Adding User Matches** for more information.

3. To preview how well the detected features describe the alignment of the stereo camera, set **Display** to **Preview Alignment**.

   **Preview Alignment** shows the aligned matches at keyframes, but also calculates matches at non-keyframes. This allows you to review how well the interpolated solve works and whether additional keyframes are required.
Display set to Preview Alignment.

4. Ideally, most lines in the overlay should be horizontal. Any matches that aren't horizontal and have a vertical error greater than Error Threshold are pre-selected and displayed in yellow. These are considered poor matches. If you scrub through the timeline and find frames with a lot of yellow matches, add more keyframes for O_Solver to analyze or add user matches manually. See Adding User Matches for more information.

5. Increase the Match Offset value to artificially increase the disparity, so you can better see how horizontal the feature matches are. Again, if you scrub through the timeline and find frames with a lot of yellow matches, add more keyframes for O_Solver to analyze or add user matches manually. See Adding User Matches for more information.

6. Next, decrease Match Offset to examine different points in the image. Accurate feature matches should sit on top of each other when you converge on them. If you can see a vertical offset between any feature matches, add more keyframes for O_Solver to analyze or add user matches manually. See Adding User Matches for more information.
Adding User Matches

User matches assist O_Solver when calculating the camera relationship. The solve considers these manually added matches superior to the ones detected automatically by Ocula, and pays them more attention when calculating the final results.

You can add more feature matches manually if the automatic feature detection didn't produce enough matches in some parts of the image. In cases like this, it's a good idea to add at least four user matches (one in each corner of the image), but the more (accurate) matches you have, the better.

To add a feature match:
1. Click the Add User Match button above the Viewer to enable add mode.
2. Locate a feature in either view that is easily recognizable in both views (for example, edges or areas of high contrast), and then click in the Viewer to place the user match. The Add User Match button stays enabled so you can continue adding user matches.

   TIP: You can also add user matches by holding Ctrl/Cmd+Alt and clicking in the Viewer.

   A cross is placed in the Viewer, representing the user match in that view, and then O_Solver automatically adds a corresponding match in the other view.
3. You can fine-tune matches by dragging a user match in one view to its corresponding position in the other view. By default, the two views are overlaid using a difference merge, in a 256px texture at x2 magnification.
You can change the size of the overlay and magnification using the **texture size** and **magnification** dropdowns above the Viewer.

**TIP:** Holding **Ctrl/Cmd** displays the left view and **Ctrl/Cmd+Shift** the right view, allowing you to ping-pong between views. Holding **Shift** displays a left/right mix, using an over merge, to help you locate bad matches.

4. If you’re not happy with the results, you can try using **O_Solver** on another sequence shot with the same camera setup.

**WARNING:** If you use another sequence to calculate the solve, check **Single Solve From All Keys** or you’ll be taking the interpolated solve from one sequence and applying it to the frames on the other.

5. Then, connect the **O_Solver** node to the **Solver** input of **O_DisparityGenerator**, if you intend to use the camera **Alignment** to generate disparity, or **O_V VerticalAligner**.

6. Once you are happy with the results of **O_Solver**, proceed to Feeding the Results to Other Ocula Nodes.

**Feeding the Results to Other Ocula Nodes**

You can use the same **O_Solver** output throughout your script, so you don’t have to calculate the camera relationship several times.

Do one of the following:

- Select **Ocula > Ocula 4.0 > O_DisparityGenerator** to insert an **O_DisparityGenerator** node after **O_Solver**. This is necessary if you want to use **O_DisparityGenerator**’s **Alignment** control to constrain
the resulting disparity vectors to match global plate alignment. You might want to do this if your plates don't contain much detail, such as bluescreen images with markers in the background.

O_Solver followed by O_DisparityGenerator.

NOTE: If you don't intend to use O_DisparityGenerator's **Alignment** control, you don't need an O_Solver node. See DisparityGenerator for more information.

• Select **Ocula > Ocula 4.0 > O_VerticalAligner** to insert an O_VerticalAligner node after O_Solver. This node can be used to correct the vertical alignment of either O_Solver's input clip or another clip shot with the same camera setup.

O_Solver followed by O_VerticalAligner.

NOTE: If you intend to use O_VerticalAligner's **Local alignment** control, you also need an O_DisparityGenerator node. See DisparityGenerator for more information.

To learn more about O_DisparityGenerator and O_VerticalAligner, review the chapters on DisparityGenerator and VerticalAligner.

### O_Solver Controls

#### O_Solver Tab

**Views to Use**

From the views that exist in your project settings, select the two views you want to use to calculate the features and the camera relationship. These views will be mapped for the left and right eye.
Analysis

Mask

If an area in the **Source** clip is producing poor feature matches, you can use this control to select areas of the image to ignore during the feature detection and analysis.

![NOTE: Masks should exist in both views, and O_DisparityGenerator expects alpha values of either 0 (for regions to use) or 1 (for regions to ignore).]

<table>
<thead>
<tr>
<th>None</th>
<th>Use the entire image area.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Alpha</strong></td>
<td>Use the alpha channel of the <strong>Source</strong> clip as an ignore mask.</td>
</tr>
<tr>
<td><strong>Source Inverted Alpha</strong></td>
<td>Use the inverted alpha channel of the <strong>Source</strong> clip as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Luminance</strong></td>
<td>Use the luminance of the <strong>Ignore</strong> input as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Inverted Luminance</strong></td>
<td>Use the inverted luminance of the <strong>Ignore</strong> input as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Alpha</strong></td>
<td>Use the alpha channel of the <strong>Ignore</strong> input as an ignore mask.</td>
</tr>
<tr>
<td><strong>Mask Inverted Alpha</strong></td>
<td>Use the inverted alpha channel of the <strong>Ignore</strong> input as an ignore mask.</td>
</tr>
</tbody>
</table>

Analysis Key

This shows the analysis keys that have been created. When you add an analysis key, O_Solver calculates feature matching and analysis and then sets an analysis key. The solves for all other frames are created by interpolating between the results on the analysis keys on either side. This field is for display only. To edit the keyframes, use **Key Frame** and **Delete Key**.

![NOTE: Keyframe interpolation helps to ensure smooth changes in the calculated camera relationship between views. We recommend using **Key Sequence** to analyze the entire sequence, and then adding additional analysis keys where the offsets between matches occur. You can visualize how well the interpolated geometry matches the images by setting **Display** to **Preview Alignment** in the O_Solver controls. If you see a lot of yellow matches (matches that have a vertical error greater than the **Error Threshold**), you may need to add more keyframes.]

**TIP:** Alternatively, you can quality check the interpolated geometry by using O_VerticalAligner followed by an Anaglyph node. Enable **Global > Preset > Full** to interpolate the calculated camera relationship and check whether there is any vertical displacement between the aligned views in the anaglyph view. See O_VerticalAligner Example for an example of how to use O_Solver, O_VerticalAligner, and Anaglyph.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delete Key</strong></td>
<td>Delete an analysis key at the current frame.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Delete All</strong></td>
<td>Delete all analysis keys.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Key Frame</strong></td>
<td>Set an analysis key at the current frame.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Key Sequence</strong></td>
<td>Analyze the whole sequence automatically and set analysis keys when a change in camera alignment is detected.</td>
<td></td>
<td><strong>TIP:</strong> Key Sequence detects changes in camera alignment automatically, but can be time-consuming on longer sequences.</td>
</tr>
<tr>
<td><strong>Key Nominated</strong></td>
<td>Set analysis keys at the frames specified in the Render dialog. You can specify frames using Nuke's regular frame expressions. For example, if you enter &quot;1-5 8 10 15 22-25&quot;, only those 12 frames are keyed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Single Solve From All Keys</strong></td>
<td>When enabled, O_Solver calculates a single solve using all the keyframes you have set. Use this for rigs that don't change over time to get more accurate results than when using a single keyframe or when the O_Solver analysis is performed on one clip and then re-used for another clip. Do not use this if there is jitter in the alignment or there is a change in separation, convergence, or zoom. Instead, use a separate solve for each keyframe and place keys where the alignment changes.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Display

**Display Dropdown**

Change the display mode:
<table>
<thead>
<tr>
<th>Nothing</th>
<th>Only show the <strong>Source</strong> image.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyframe Matches</strong></td>
<td>Show the features and matches for the camera relationship calculation in a Viewer overlay. Feature matches are only calculated for the keyframes.</td>
</tr>
</tbody>
</table>

Features and **Keyframe Matches** for the camera relationship calculation in a Viewer overlay.

You can use this mode to see where O_Solver has found features and matches, and evaluate how accurate they are. You can also edit the feature matches manually. To delete a poor match, right-click on it and select **delete selected**. To add matches manually, see Adding User Matches.

You can also activate this mode by selecting **display matches** from the Viewer's right-click menu.

| **Preview Alignment** | Preview how well the calculated feature matches describe the alignment of the stereo camera. This shows the aligned matches at keyframes, but also calculates matches at non-keyframes, allowing you to review how well the interpolated solve works and whether additional keyframes are required. If the lines between feature matches are horizontal, they describe the alignment of the camera rig well. If any lines are skewed (and displayed in yellow), you may want to delete the feature matches in question. If necessary, you can also add manual feature matches to replace them and preview the effect of the manual matches in the overlay. |
Visualizing the alignment of the calculated feature matches.

You can also activate this mode by selecting `preview alignment` from the Viewer's right-click menu.

**Match Offset**

The offset (in pixels) applied to the aligned feature matches. You can:

- increase this value to artificially increase the disparity, so it's easier to see how horizontal the feature matches are.
- decrease this value to set the disparity of particular matches to zero and examine the vertical offset at each feature. The matches should sit on top of each other.

⚠️ **NOTE:** If you find a lot of yellow matches, you can add user matches manually. See `O_Solver Controls` for more information.

The `Match Offset` control is only available when `Display` is set to `Preview Alignment`. You can also adjust it by selecting `decrease offset` or `increase offset` from the Viewer's right-click menu.

**Error Threshold**

The threshold on the vertical alignment error in pixels. When `Display` is set to `Preview Alignment`, any matches with a vertical error greater than the threshold are highlighted in the Viewer. This allows you to easily delete poor matches with large errors when previewing alignment at keyframes - adjust the `Error Threshold` to highlight poor matches and press `Backspace` to remove them.
Current Frame

**Re-analyse Frame**
Clear the automatic feature matches from the current frame and recalculate them. This can be useful if there have been changes in the node tree upstream from O_Solver, you have deleted too many automatic feature matches, or you want to calculate the automatic matches based on any user matches you have created.

**Delete Auto Matches**
Delete all automatically generated matches added to the current frame.

**Delete User Matches**
Delete all user matches you have manually added to the current frame.

Python Tab

These controls are for Python callbacks and can be used to have Python functions automatically called when various events happen in Nuke.

<table>
<thead>
<tr>
<th><strong>before render</strong></th>
<th>These functions run prior to starting rendering in <code>execute()</code>. If they throw an exception, the render aborts.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>before each frame</strong></td>
<td>These functions run prior to starting rendering of each individual frame. If they throw an exception, the render aborts.</td>
</tr>
<tr>
<td><strong>after each frame</strong></td>
<td>These functions run after each frame is finished rendering. They are not called if the render aborts. If they throw an exception, the render aborts.</td>
</tr>
<tr>
<td><strong>after render</strong></td>
<td>These functions run after rendering of all frames is finished. If they throw an error, the render aborts.</td>
</tr>
<tr>
<td><strong>render progress</strong></td>
<td>These functions run during rendering to determine progress or failure.</td>
</tr>
</tbody>
</table>

O_Solver Example

In this example, we read in a stereo image, use O_Solver to calculate the camera relationship, review the results, and add a user match to improve the solve data.
Step by Step

1. Start Nuke and press S on the Node Graph to open the Project Settings. Go to the Views tab and click the Set up views for stereo button.

2. Select Image > Read to import Dance_Group.exr. The .exr format allows both views to exist in a single file, so Nuke reads in both the left and the right view using the same Read node.

3. Insert an O_Solver node after the stereo image by choosing Ocula > Ocula 4.0 > O_Solver from the Toolbar.

   The purpose of this node is to define the geometrical relationship between the two views in the input image (that is, the camera relationship or solve). The solve information can then be fed to an O_DisparityGenerator, if you want to use the alignment information to create a disparity map, or to an O_VerticalAligner to correct vertical offset between views.

   For now, we'll concentrate on creating an accurate solve using O_Solver.

4. Attach a Viewer to the O_Solver node.

5. In the O_Solver controls, click Key Frame to set a keyframe for O_Solver to analyze.

   Our example here consists of just one frame, but if you were using a sequence, click Key Sequence to automatically analyze the sequence and add keyframes when the camera setup changes. If the setup doesn't change, you can get away with using just one or two keyframes. Remember that keyframes should be placed on frames that are easy to match between views - ideally, with enough picture detail, but no motion blur, occluding fog, or dust.

   Every time you add a keyframe, O_Solver analyzes the footage and calculates the solve.

6. O_Solver displays Keyframe Matches by default, but you can change modes using the right-click menu in the Viewer or the Display dropdown on the Viewer toolbar.

   NOTE: Matches are only displayed when the playhead is on a keyframe, marked with a blue chip.

   The calculated feature matches are displayed in a Viewer overlay, and you can switch between views to compare them.
Display set to Keyframe Matches.

7. Now, set Display to Preview Alignment. This allows you to check the quality of your solve by previewing the alignment of the calculated feature matches in the Viewer. Ideally, the lines should all be horizontal. If they have a vertical error greater than the Error Threshold, they are considered poor matches and displayed in yellow.

8. To help guide the solve, we can add feature matches manually. It's a good idea to do this if you have a particularly tricky part of a shot where the automatic feature matching has produced a few yellow matches.

   There's an area in the top-right of the frame containing bad matches, so zoom in on it.

9. Locate a feature in either view that is easily recognizable in both views (for example, edges or areas of high contrast).

10. Hold Ctrl/Cmd+Alt and then click in the Viewer to place the user match.

    TIP: You can also click the Add User Match button above the Viewer to enable add mode.

    A cross is placed in the Viewer, representing the user match in that view, and then O_Solver automatically adds a corresponding match in the other view.

11. You can align views by dragging a user match in one view to its corresponding position in the other view. By default, the two views are overlaid using a difference merge, in a 256px texture at x2 magnification.

    You can change the size of the overlay and magnification using the texture size (in pixels) and magnification dropdowns above the Viewer.

    TIP: Holding Ctrl/Cmd displays the left view and Ctrl/Cmd+Shift the right view, allowing you to ping-pong between views. Holding Shift displays a left/right mix, using an over merge, to help you locate bad matches.

    You can add as many manual feature matches as you like, so if you see any other areas that might benefit from them, feel free to add more.
O_Solver considers any feature matches you've added yourself superior to the ones it detects automatically and pays them more attention. This can also influence the automatic matches displayed in the Preview Alignment mode. Any automatic matches that don't agree with the alignment defined by your user matches are highlighted in yellow and should be deleted.

![Poor auto-match alignments are highlighted yellow in the Viewer.](image1)

Adding user matches allows you to delete poor matches by pressing Backspace.

12. Next, decrease the Match Offset value gradually until some matches have no horizontal offset. This sets the disparity of those matches to zero, which means accurate feature matches should sit on top of each other. You may need to zoom in to see if they do.

The matches in the top-left of the Viewer display vertical offset between the feature matches, and are highlighted in yellow. Place a user match on the corner of the building to correct them.

13. Fine-tune matches by dragging the user match in one view to its corresponding position in the other view. By default, the two views are overlaid using a difference merge, in a 256px texture at x2 magnification.

You can change the size of the overlay and magnification using the texture size (in pixels) and magnification dropdowns above the Viewer.

![Adding user matches allows you to delete poor matches by pressing Backspace.](image2)

TIP: Holding Shift when lining up the user match displays a left/right mix of the views, using an over merge, holding Ctrl/Cmd shows the left view, and Ctrl/Cmd+Shift shows the right view in the overlay.
14. The yellow matches can be deleted by pressing Backspace to improve the solve. We recommend adding at least five user matches, one toward each corner of the image to correct the overall alignment, and one at the center on the subject to focus alignment where it is important.
VerticalAligner

Description

If the cameras used to shoot stereoscopic images are poorly positioned or converge (point inwards), some features in the resulting two views may be vertically misaligned. In the case of converging cameras, the misalignment may be due to keystoning. Unlike converging cameras, parallel cameras do not produce keystoning.

Keystoning is when an image is distorted because the angle created by converging cameras affects the perspective in the two views. As a result, corresponding points in the two views are vertically misaligned.

Whether the vertical misalignment was caused by poorly positioned or converging cameras, it can result in an unpleasant 3D stereo viewing experience. When a vertically misaligned stereo image is viewed with 3D glasses, the viewer's brain attempts to line up the corresponding points in the images, often causing eye strain and headaches. To avoid this, stereo images should only contain horizontal disparity, not vertical.
O_VerticalAligner allows you to warp views vertically so that their corresponding features align horizontally. The **Vertical Skew** and **Local Alignment** options allow you to warp the views, while keeping the horizontal position of each pixel the same so that there is no change in convergence.

Before O_VerticalAligner. Notice that the curved line at the bottom of the image and the controls on the left are misaligned. After O_VerticalAligner. Notice that the curved line at the bottom of the image and the controls on the left have now been aligned.

There are several modes: **Global Alignment**, **Local Alignment**, **Fix Scale** and **Fix Offset**. If the none of the method checkboxes are selected in the Local, Fix Scale, or Fix Offset sections of the O_VerticalAligner controls, the **Global Alignment** mode is on by default.

**Global Alignment Mode**

In the Global Alignment mode, O_VerticalAligner performs a global transform to align the views. You can choose between several alignment types. All methods concatenate. This means that if you select several alignment types from the Global section in the O_VerticalAligner controls, their functions are combined. See O_VerticalAligner Controls for information about each alignment type.

If you have a pre-tracked Nuke stereo camera that describes the camera setup used to shoot the **Source** images, you can attach it to the O_Solver node and use O_VerticalAligner in the **Global Alignment** mode to analyze the sequence and output a vertically-aligned camera pair. This works with all global methods except **Vertical Skew** (which can’t be represented by a camera transform). For more information, see Using O_VerticalAligner.

**Local Alignment Mode**

In the **Local Alignment** mode, O_VerticalAligner rebuilds the image per-pixel to account for any local distortions in the mirror or lens, and changes in alignment with depth using O_Solver data.

The **Local Alignment** mode always requires a disparity map upstream. You can create one using an O_DisparityGenerator node upstream of the O_VerticalAligner node.
NOTE: You can create disparity once and it is aligned to match the aligned plate. There is no need to recalculate disparity.

Fix Scale Mode

The Fix Scale method allows you to zoom the plate if the original footage was not over-scanned, and the alignment pulls black into the format.

You can scale the image to prevent pulling pixels from outside the input image. To minimize the scale change, align Both Views.

WARNING: The scale has to be applied to both images, even when aligning Left to Right or Right to Left.

Fix Offset Mode

The Fix Offset method allows you to correct any convergence change that has happened on the subject to preserve the original subject parallax and hence depth.

You can shift the image to preserve the parallax at the fix-point.

NOTE: The Fix Offset mode requires upstream disparity vectors. If they do not already exist in the image sequence, insert an O_DisparityGenerator node to calculate them.

Inputs

O_VerticalAligner has the following inputs:

| Source                      | A stereo pair of images. Global Alignment mode is on by default. In all modes the images should be followed by an O_Solver node. If you are using the Local Alignment, or Fix Offset mode, you also need an O_DisparityGenerator node (if disparity vectors do not already exist) upstream of O_VerticalAligner. |

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.
Using O_VerticalAligner

To vertically align a pair of stereo images, you can several modes: **Global Alignment**, **Local Alignment**, **Fix Scale**, and **Fix Offset**. You can choose to use a mixture of these modes, for example, you can perform a global alignment and a local alignment which are concatenated into a single filter operation.

You can apply a global image transform to align the feature matches generated by an upstream O_Solver node, or you can rebuild the view(s) to remove vertical disparity calculated by an upstream O_DisparityGenerator using the **Local Alignment** mode.

NOTE: The **Global Alignment** mode is on by default.

To use the O_VerticalAligner, complete the following steps:

1. Select **Ocula > Ocula 4.0 > O_Solver** to insert an O_Solver node after your stereo clip. For more information on how to use O_Solver, see **Solver**.
2. Select **Ocula > Ocula 4.0 > O_VerticalAligner** to insert an O_VerticalAligner node after either O_Solver.
3. Connect a Viewer to the O_VerticalAligner node. Your node tree should now look something like this:

![Node Tree Screenshot]

4. Open the O_VerticalAligner controls. Under **Views to Use**, you can see all the views that exist in your project settings. Select the two views you want to use for the left and right eye when correcting the alignment. The two views you select are mapped for the left and right eye.
5. From the **Align** menu, select how to move the images to align the views: **Both Views**, **Left to Right**, or **Right to Left**. See **O_VerticalAligner Controls** for information about the **Align** options.
6. Select the **filter** to handle the vertical alignment transform, or use the default **Lanczos6**, which is a good sharpening and scaling down filter. For more information on the available filters, see **Filter** under **O_VerticalAligner Controls** or **Choosing a Filtering Algorithm** in the Nuke User Guide.
7. Enable **Output STMap** to include the **Global** or **Local** correction as a uv coordinate map along with alignment information.
8. From the **Global** section, select types of alignment you want to use. You can select preset options including **Transform**, **Match Camera**, **Keystone Only**, and **Full**. You can also select **Custom** and manually select the types of alignment you want to include. See **O_VerticalAligner Controls** for more information about the types of alignment.
9. If you want to perform a local alignment, insert an O_DisparityGenerator node after O_Solver, and select the **Local Alignment** checkbox in the **Local** section of the O_VericalAligner controls. You can now adjust the **Local** controls. See **O_VericalAligner Controls** for more details.

10. To view the effect of O_VericalAligner more accurately, you can:

   - Insert an Anaglyph node between the O_VericalAligner node and the Viewer,

   ![Diagram](image)

   OR

   - Add a StereoReviewGizmo to view alignment. See **StereoReviewGizmo** for more information.

11. Adjust the required settings to get the best possible result. See **O_VericalAligner Controls** for information about the settings.

### Analysing and Using Output Data

In all global methods except **Vertical Skew**, you can use the **Analyse Sequence** control to create output data and use the data for the following:

- Vertically align a pre-tracked Nuke stereo camera. This allows you to continue using pre-tracked cameras after your footage has been vertically aligned. Note that you can only create a vertically aligned stereo camera when a pre-tracked camera is connected to the **Camera** input of O_Solver.

- Create a Nuke CornerPin2D node that produces the same result as O_VericalAligner.

The output data is also stored on the **Output** tab of the node controls, where you can see the transform represented as a four-corner pin and a transform matrix per view.

To analyze and use the output data, complete the following steps:

1. After performing a vertical alignment, click **Analyse Sequence** under the **Analysis** section in the O_VericalAligner controls.
NOTE: You cannot use Analyse Sequence with the Local Alignment checkbox selected.

2. When prompted, enter a frame range to analyze. O_VerticalAligner analyzes the sequence.

![Frame range input dialog](image)

3. You can now use the output data in the following ways:
   - To output a vertically aligned camera pair, click either Create Camera or Create Rig. Create Camera produces a single Camera node with split controls to hold the left and right view parameters. Create Rig produces two Camera nodes and a JoinViews node that combines them.
   - To create a Nuke CornerPin2D node that represents the result of O_VerticalAligner, click Create Corner Pin. A CornerPin2D node that produces the same result as O_VerticalAligner appears in the Node Graph.

Scripting Analysis and CornerPin Creation

Given a standard Node Graph containing your footage, an O_Solver, and an O_VerticalAligner, Ocula allows you to automate the setup of a CornerPin for a range of frames without having to manually create the required analysis frames.

The following Python script examines feature matches in the entire sequence, analyzes the vertical alignment for a given frame range for both views, and then creates a CornerPin node containing data for the four pinned points in the Viewer.

TIP: If you don’t want the analysis to continue on failure, remove the 1 after the frame range in the second Python call.

```python
# Run the analysis pass on the Solver
nuke.toNode('O_Solver1')['analyseSequence'].execute()

# Run the analysis pass on the VerticalAligner between frames 1001 and 1085
nuke.execute("O_VerticalAligner1", 1001, 1085, 1, ['left','right'])

# Create the cornerPin
nuke.toNode('O_VerticalAligner1')['createPin'].execute()
```
O_VerticalAligner also has a **Python** tab in the Properties panel, allowing you to call Python functions automatically when various events happen in Nuke. See *Help > Documentation > Python Developers Guide* for more information.

## O_VerticalAligner Controls

### O_VerticalAligner Tab

#### Views to Use

From the views that exist in your project settings, select the two views you want to use to generate an occlusion mask. These views are mapped for the left and right eye.

#### Align

Select how to move the views to align the images.

<table>
<thead>
<tr>
<th><strong>Both Views</strong></th>
<th>Move both the left and right views so that they are aligned.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Left to Right</strong></td>
<td>Move the left view only to line up with the right.</td>
</tr>
<tr>
<td><strong>Right to Left</strong></td>
<td>Move the right view only to line up with the left.</td>
</tr>
</tbody>
</table>

#### Filter

Select the filtering algorithm you want to use when remapping pixels from their original positions to new positions. This helps avoid problems with image quality, particularly in high contrast areas of the frame (where highly aliased, or jaggy, edges may appear if pixels are not filtered and retain their original values).

<table>
<thead>
<tr>
<th><strong>Impulse</strong></th>
<th>This option means no filtering is done. Each output pixel equals an input pixel.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cubic</strong></td>
<td>Remapped pixels receive some smoothing.</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>Remapped pixels receive some smoothing and minor sharpening.</td>
</tr>
</tbody>
</table>
**Output STMap**

When enabled, this allows you to output an STMap along with an aligned image and disparity vectors.

**Global**

Select the types of alignment you want to use to vertically align the images. You can select preset options, or you can select **Custom** and manually define the types of alignment you want to include. See the table below for the types of alignment each preset option includes.

<table>
<thead>
<tr>
<th>Preset Option</th>
<th>Description</th>
<th>Camera Correction</th>
<th>Focal Length</th>
<th>Vertical Shift</th>
<th>2D Rotation</th>
<th>Perspective Warp</th>
<th>Vertical Skew</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform</td>
<td>Use this option to perform a 2D correction</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Note that this option is not ideal for vertical alignment; instead **Cubic** or **Parzen** are recommended for this.
without any change in pixel aspect or skew.

<table>
<thead>
<tr>
<th><strong>Match Camera</strong></th>
<th>Use this option to correct using a match move camera connected to the upstream O_Solver.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Keystone Only</strong></th>
<th>Use this option to correct vertical alignment without changing the parallax.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Full</strong></th>
<th>Use <strong>Full</strong> to include all the alignment options except <strong>Vertical skew</strong>.</th>
</tr>
</thead>
</table>

All alignment methods concatenate. This means that if you select several alignment types, their functions are combined. You can also analyze the data to create corner pin and camera information in all methods except **Vertical Skew**. See the table below for a description of what each alignment type does. The alignment types are applied in the following order.

<table>
<thead>
<tr>
<th><strong>Camera correction</strong></th>
<th>Correct the vertical alignment for a match-move camera connected to an upstream O_Solver. If there is no camera connected, this uses the internal camera calculated by O_Solver. The correction is refined by other alignment options that are selected.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Focal length</strong></th>
<th>Align the features by calculating a 2D scale to correct focal length differences.</th>
</tr>
</thead>
</table>
### Vertical shift
Align the features vertically by moving the entire image up or down. Calculate a 2D vertical shift to correct a global offset.

### 2D rotation
Align the features vertically by rotating the entire image around a point. The center of the rotation is determined by the algorithm. This helps correct in-plane camera roll.

### Perspective warp
Do a four-corner warp on the images to align them on the y axis. This may move the features slightly along the x axis. **Perspective warp** can help correct camera tilt as well as roll.

**NOTE:** A perspective change can alter pixel aspect ratio and introduce pixel skew.

### Vertical skew
Align the features along the y axis using a skew. This does not move the features along the x axis. **Vertical skew** allows you to correct keystoning without changing horizontal disparity. This varies the vertical shift across the image without changing the horizontal position of pixels. However, this may also introduce pixel skew.

**NOTE:** If you select or deselect an alignment type after selecting a preset option, the preset Type dropdown, automatically updates to **Custom**.

### Local Options
There are two modes you can use for vertical alignment. The **Global Alignment** method is on by default. Select the **Local Alignment** checkbox to perform a local alignment in addition to the global alignment.

<table>
<thead>
<tr>
<th>Global Alignment</th>
<th>This applies a global image transform to align the feature matches generated by an upstream O_Solver node.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Alignment</td>
<td>This rebuilds the view(s) to remove vertical disparity calculated by an upstream O_DisparityGenerator. Use this mode to create a per-pixel correction if there are any local distortions in the mirror or lens and changes in alignment with depth.</td>
</tr>
</tbody>
</table>
Pre-blur

The **Pre-blur** control allows you to set the size of the blur applied to the disparity before performing a local alignment. To smooth out the correction across depth boundaries, increase the blur size.

Correction

The **Correction** control allows you to set the amount of local correction to apply between the global transform at 0 and the full correction at 1. This would be useful, for example, if you want to tone down the local distortion that is applied.

Fix Scale Options

Zoom to Prevent Black in the Frame

Select the **Zoom to prevent black in frame** checkbox to scale the image in order to prevent pulling pixels from outside the input image. To minimise the scale, change the **Align** dropdown to **Both Views**.

> **WARNING:** The scale has to be applied to both images, even when aligning **Left to Right** or **Right to Left**.

Calculate Scale

The **Calculate scale** control allows you to calculate the scale at the current frame. If the alignment options change, the scale needs to be recalculated. You can lock the scale correction to prevent any changes by selecting the **Lock scale** checkbox.

Scale

The **Scale** control allows you to set the global scale that is applied to prevent black in the frame. You can set a key to interpolate the scale calculated at different frames.

> **WARNING:** Animating the **Scale** control creates a dynamic zoom on the shot, which may not be the stereographer's original intent. In this case, it is recommended to use a static zoom from a single frame, to preserve the original intent of the shot.
Fix Offset Options

Preserve Subject Parallax

Select the **Preserve subject parallax** checkbox to shift the image in a way that preserves the parallax at the specified **Fix Point**. This requires the input to have disparity vectors. If disparity vectors do not already exist, you need to add an O_DisparityGenerator node upstream of O_VerticalAligner.

Fix Point

You can move the **Fix Point** to sample the input disparity and update the applied offset to preserve the parallax. View the output of O_VerticalAligner to set the **Fix Point** when the **Output** is set to **Image**. You can lock this to prevent any changes by selecting the **Lock offset** checkbox.

To change the **Fix Point**, drag the **fixPoint** widget from the bottom-left corner of the Viewer and drop it in the new position. Disparity is then recalculated according to the new **Fix Point**. You can also use the user matches determined by O_Solver to set the **Fix Point**.

Offset

You can use the **Offset** control to set the correction in pixels, that is applied to prevent parallax changes at the **Fix Point**. To interpolate the offset calculated at different frames, set a key.

**NOTE:** Note that you can re-converge the views after you have aligned the plates to preserve depth.

Analyze Sequence

Analyze the sequence to create a corner pin or an aligned camera output. Use **Analyze Sequence** to create the output data in all global methods except **Vertical Skew** (the default). Then, you can apply the data to the **Create Corner Pin**, **Create Camera**, or **Create Rig** controls.

**NOTE:** You cannot use **Analyze Sequence** with the **Local alignment** checkbox selected.

| Create Corner Pin | Click this to create a corner pin representing the result of O_VerticalAligner after you have clicked **Analyze Sequence**. This works in all global methods except **Vertical Skew**. |
Create Camera

If you have a pre-tracked Nuke stereo camera connected to the Camera input of the O_Solver up the tree and you click **Analyze Sequence**, you can then click **Create Camera** to create a vertically aligned camera from the analysis. This gives you a single Camera node with split controls to hold the left and right view parameters. This works in all global methods except **Vertical Skew**.

Create Rig

If you have a pre-tracked Nuke stereo camera connected to the Camera input of the O_Solver up the tree and you click **Analyze Sequence**, you can then click **Create Rig** to create a vertically aligned camera rig from the analysis. This gives you two Camera nodes and a JoinViews node that combines them. This works in all global methods except **Vertical Skew**.

Output Tab

**Four Corner Pin**

This represents the 2D corner pin that can be applied to the input image to create the same result as O_VerticalAligner (in all global methods except **Vertical Skew**). This allows you to do the analysis in Nuke, but take the matrix to a third-party application – such as Baselight – and align the image or camera there.

**Transform Matrix**

This provides the concatenated 2D transform for the vertical alignment. The matrix is calculated when you click **Analyze Sequence** on the O_VerticalAligner tab. There is one matrix for each view in the source.

Python Tab

These controls are for Python callbacks and can be used to have Python functions automatically called when various events happen in Nuke.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>before render</strong></td>
<td>These functions run prior to starting rendering in execute(). If they throw an exception, the render aborts.</td>
</tr>
<tr>
<td><strong>before each frame</strong></td>
<td>These functions run prior to starting rendering of each individual frame. If they throw an exception, the render aborts.</td>
</tr>
<tr>
<td><strong>after each frame</strong></td>
<td>These functions run after each frame is finished rendering. They are not</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>after render</td>
<td>These functions run after rendering of all frames is finished. If they throw an error, the render aborts.</td>
</tr>
<tr>
<td>render progress</td>
<td>These functions run during rendering to determine progress or failure.</td>
</tr>
</tbody>
</table>

## O_VerticalAligner Example

In this example, we correct the vertical alignment of a stereo image using the **Global Alignment** mode. The image used here can be downloaded from our website. For more information, please see [Example Images](#).

### Step by Step

1. Launch Nuke. Open the project settings (press S on the Node Graph), select the **Views** tab, and click the **Set up views for stereo** button.
2. Import the `steep_hill.exr` image and connect it to a Viewer. The image includes both the left and the right view.
3. Select **Ocula > Ocula 4.0 > O_Solver** to insert an O_Solver node after the stereo clip. See [Solver](#) for more information.
4. Click **Key Frame** to set at least one keyframe.
5. Insert an O_VerticalAligner (**Ocula > Ocula 4.0 > O_VerticalAligner**), followed by an Anaglyph node (**Views > Stereo > Anaglyph**) after O_Solver.

**NOTE:** The Anaglyph node is for illustrative purposes in this example. You could just as easily use the DisparityReviewGizmo to view the alignment.

6. By default, O_VerticalAligner is in **Global** alignment mode with the **Local alignment** checkbox disabled. In this mode, O_VerticalAligner applies a global image transform to align the feature matches generated by the upstream O_Solver node. In **Global** alignment mode, O_VerticalAligner does not need disparity vectors upstream.
7. To see the effect more accurately, select the O_VerticalAligner node and press D repeatedly to disable and enable the node. With the node disabled, the views remain vertically misaligned. However, when you enable the O_VerticalAligner node, the views align nicely.
The zoomed in image with the O_VerticalAligner node disabled.

The zoomed in image with O_VericalAligner enabled.
VectorGenerator

Description

O_VectorGenerator generates motion vector fields for each view in a stereo image. Motion vectors map the location of a pixel on one frame to the location of the corresponding pixel in a neighboring frame. It has the same dimensions as the image, but contains an (x,y) offset per pixel. These offsets show how to warp a neighboring image onto the current image.

Clearly, as most of the images in a sequence have two neighbors, each can have two vector fields. These are called the **forward** motion vectors where they represent the warp of the next frame on to current frame, and **backward** motion vectors where they represent the warp of the previous frame on to current frame.

Disparity vectors map pixels between views, whereas motion vectors map them between frames.

O_VectorGenerator stores the motion vectors in the backward and forward motion channels. To view these in Nuke, select **motion**, **forward**, or **backward** from the channel set menu in the top left corner of the Viewer.
Forward motion vectors.  Backward motion vectors.

If you want to use pre-calculated motion vectors rather than generate vector fields each time you need them, you can use a Write node to render them into the channels of your stereo .exr file along with the color and disparity channels. Later, whenever you use the same image sequence, the motion vectors are loaded into Nuke together with the sequence.

Ocula’s Retimer node relies on motion vectors to produce its output, but you may also want to use O_VectorGenerator for other purposes (for example, for generating motion blur).

**Inputs**

O_VectorGenerator has the following inputs:

<table>
<thead>
<tr>
<th>Source</th>
<th>A stereo pair of images.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mask</strong></td>
<td>An optional mask that specifies areas to exclude from the motion calculation. You can use this input to prevent distortions at occlusions or to calculate motion for a background layer by ignoring all foreground elements.</td>
</tr>
</tbody>
</table>

Note that masks should exist in both views, and O_VectorGenerator treats the alpha values of 1 as foreground and blurs to the 0 value using nearby vectors to recreate object boundaries, rather than image data. When you create a mask using Roto or RotoPaint, you can use the feather control to extend the calculation. For example, the vector map may have a sharper transition at edges with a binary mask, but applying feather on the mask can help smooth the resulting image.
To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Generating Motion Vectors

To generate motion vectors for a stereo pair of images, do the following:

1. Select **Ocula > Ocula 4.0 > O_VectorGenerator** to add an O_VectorGenerator node after either a stereo clip or a JoinViews node.
2. Make sure you are viewing the output from O_VectorGenerator.
3. In the O_VectorGenerator controls, you can see all the views that exist in your project settings under **Views to Use**. Select the two views you want to use to calculate the motion vectors. The two views you selected are mapped for the left and right eye.
4. If there are areas in the image that you want to ignore when generating the motion vector field, supply a mask either in the **Mask** input or the alpha of the **Source** input. In the O_VectorGenerator controls, set **Mask** to the component you want to use as the mask.
   
   If there are areas in the image that you want to ignore when generating vectors, supply a mask either in the **Mask** input or the alpha of the **Source** input. In the O_VectorGenerator controls, set **Mask** to the component you want to use as the mask.
   
   The white areas of the image have their vectors calculated as normal, whereas black areas take their vectors from nearby areas. When you create a mask using Roto or RotoPaint, you can use the **feather** control to tune the calculation.
5. Set the Viewer’s channel set menu to **motion, forward, or backward**. O_VectorGenerator calculates the motion vectors and displays them in the Viewer.
6. If necessary, adjust Vector Detail, Strength, Consistency, and Smoothness and view their effect on the motion vector field. For more information on these parameters, see O_VectorGenerator Controls.

Writing Motion Vectors into a Clip

When you're happy with the motion vectors generated, you can save time down the line by writing the vectors into a new clip combining the source and motion channels.

1. Select the O_VectorGenerator node in the Node Graph.
2. Select Image > Write (or press W on the keyboard) to insert a Write node after O_VectorGenerator.
3. In the Write node controls, select all from the channels dropdown and set file type to exr.
4. Enter a name for the clip in the file field (for example, my_clip.####.exr), and click Render.

The newly created motion channels are saved in the channels of your stereo clip. When you need to manipulate the same clip again later, the motion vectors are loaded into Nuke together with the clip.

Rendering the output to combine the clip and the motion channels for future use.
O_VectorGenerator Controls

Use GPU

Open the O_VectorGenerator controls. O_VectorGenerator renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_VectorGenerator defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke's Preferences and selecting an alternative card from the GPU Device dropdown.

NOTE: Selecting a different GPU requires you to restart Nuke before the change takes effect.

Views to Use

From the views that exist in your project settings, select the two views you want to use to calculate the motion vectors. These views are mapped for the left and right eye.

Mask

An optional mask that specifies areas to exclude from the motion calculation. You can use this input to prevent distortions at occlusions or to calculate motion for a background layer by ignoring all foreground elements.

Note that masks should exist in both views, and O_VectorGenerator treats the alpha values of 1 as foreground and blurs to the 0 value using nearby vectors to recreate object boundaries, rather than image data. When you create a mask using Roto or RotoPaint, you can use the feather control to extend the calculation. For example, the vector map may have a sharper transition at edges with a binary mask, but applying feather on the mask can help smooth the resulting image.

<table>
<thead>
<tr>
<th>None</th>
<th>Use the entire image area.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Alpha</td>
<td>Use the alpha channel of the Source clip as an ignore mask.</td>
</tr>
<tr>
<td>Source Inverted Alpha</td>
<td>Use the inverted alpha channel of the Source clip as an ignore mask.</td>
</tr>
<tr>
<td>Mask Luminance</td>
<td>Use the luminance of the Mask input as an ignore mask.</td>
</tr>
<tr>
<td>Mask Inverted Luminance</td>
<td>Use the inverted luminance of the Mask input as an ignore mask.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Mask Alpha</td>
<td>Use the alpha channel of the Mask input as an ignore mask.</td>
</tr>
<tr>
<td>Mask Inverted Alpha</td>
<td>Use the inverted alpha channel of the Mask input as an ignore mask.</td>
</tr>
</tbody>
</table>

**Vector Detail**

Adjusts the detail of the calculated motion vectors. Higher detail picks up finer movement, but takes longer to calculate.

**Strength**

Sets the strength in matching pixels between frames. Higher values allow you to accurately match similar pixels in one image to another, concentrating on detail matching even if the resulting motion field is jagged. Lower values may miss local detail, but are less likely to provide you with the odd spurious vector, producing smoother results. Often, it is necessary to trade one of these qualities off against the other. You may want to increase this value to force the images to match, for example, where fine details are missed, or decrease it to smooth out the motion vectors.

**Consistency**

Sets how heavily the forward and backward vectors are forced to match. Increase the Consistency to make the forward and backward vectors more similar to each other, but this may cause the vectors to match the image less.

**Smoothness**

Applies extra smoothing to the motion vector field as a post-process, after image matching. The higher the value, the smoother the result.

**O_VectorGenerator Example**

See O_Retimer Example for an example of how to use O_VectorGenerator and O_Retimer to calculate a motion vector field for a stereo image and use it to retime the sequence.
Retimer

Description

O_Retimer is designed to retime footage so that it plays back faster or in slow-motion. O_Retimer uses upstream motion vectors generated by an O_VectorGenerator node to retime the footage. These motion vectors describe how each pixel moves from frame to frame (see VectorGenerator). With accurate motion vectors, it is possible to generate an output image at any point in sequence timeline by interpolating along the direction of the motion.

By default, O_Retimer is set to perform a half-speed slow down. This is achieved by generating new frames at quarter and three-quarter positions (.25 and .75) between the original frames at 0 and 1. In this way, none of the original frames are used in the retimed sequence, as shown in the following diagram:

Timing Methods

You can retime footage using two different methods; the Speed method and the Source Frame method. By default, the Timing control is set to the Speed method.

The Speed method allows you to set a new speed at which to play the footage back. A speed value below 1 slows the clip down; and a speed value above 1, speeds it up. The default value is 0.5, which creates a half-speed retime.
NOTE: When using the Speed method, we recommend adding a FrameRange node before the O_Retimer to control the input frame range. This allows you to visualize the retime using the Curve Editor more easily.

Alternatively, you can perform a retime using the Source Frame method. This retimes the footage in terms of specifying source frames at different outputs on the timeline. For example, you can specify frame 1 in the output clip to read frame 1 of the source clip, and specify frame 100 in the output clip to read frame 50 of the source clip. This is the equivalent of doing a half-speed retime.

Inputs

O_Retimer has the following inputs:

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion</td>
<td>If motion vectors are supplied here, O_Retimer uses them and does not require motion in the Source input. This can be useful if, for example, your input sequence is very noisy, as too much noise interferes with the motion estimation. In that case, you should supply a smoothed version of the sequence and an O_VectorGenerator node here.</td>
</tr>
<tr>
<td>Source</td>
<td>A stereo pair of images. If motion vectors are not embedded in the images, you need to insert an O_VectorGenerator node after the image sequence to calculate them.</td>
</tr>
</tbody>
</table>

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Using O_Retimer

NOTE: O_Retimer requires upstream motion vectors to operate correctly.

Retiming Stereo Footage Using Speed

To perform a linear retime using the Speed method, do the following:

1. If motion vectors don't yet exist in the script, you can use the O_VectorGenerator node to calculate them. See VectorGenerator for information on how to do this.

2. Select Ocula > Ocula 4.0 > O_Retimer to insert O_Retimer either after the O_VectorGenerator node if you added one in the previous step, or after the stereo image sequence.
3. In the O_Retimer controls, select the two views you want to use for retiming under View to Use. The two views you select are mapped for the left and right eye.

4. Connect a Viewer to the O_Retimer node. Your node tree should now look something like this:

5. Select a new speed value by either entering it the Speed box or dragging the Speed slider to the required value.

6. Play through the clip to calculate the retiming for all the frames.

   **NOTE:** Constant retimes produce a number of frames equal to last frame - first frame / speed. For example, a 10 frame clip at half speed would produce 10-1/0.5 = 18 frames.

---

**Retiming Stereo Footage Using Source Frame**

   **NOTE:** You need to set a minimum of two keyframes to retime footage using the Source Frame method.

To perform a linear retime using the Source Frame method, do the following:

1. Repeat steps 1 to 4 in the Retiming Stereo Footage Using Speed section.

2. Select Source Frame from the Timing dropdown.

3. Move the playhead to the first frame you want to change the output for on the Viewer timeline.

4. Set the source frame number that you want to appear at the selected output position by either entering it in the Frame box, or dragging the Frame slider to the required frame.

5. Set a keyframe for this by selecting the animation menu next to the Frame parameter, and clicking Set key. A keyframe is indicated by a blue marker.

6. Move the playhead on the Viewer timeline to the next frame you want to change the output for.
7. Set the source frame number that you want to appear at the selected output position by either entering it in the **Frame** box, or dragging the **Frame** slider to the required frame. A keyframe is set automatically, identified with a blue marker.

8. Play through the clip to calculate the retiming for all the frames.

**NOTE:** Constant retimes produce a number of frames equal to last frame - first frame / speed. For example, a 10 frame clip at half speed would produce 10-1/0.5 = 18 frames.

### Varying the Retime Speed

To vary the speed in your sequence, you can use either of the **Timing** methods. As with the **Source Frames** method, you add keyframes at different points on the Viewer timeline and specify varying speeds. This means, the speed changes at every keyframe, varying the speed throughout the clip.

**NOTE:** When you are retiming footage using the **Speed** method, ensure that you set the keyframes on the input frames by using a FrameRange node, as setting keyframes on the input frames alters the number of output frames.

### Varying Retimes Using Speed

To vary the speed throughout the footage using the **Speed** method, do the following:

1. Repeat steps 1 to 4 in the **Retiming Stereo Footage Using Speed** section.

2. Add a FrameRange (Time > FrameRange) node between the O_VectorGenerator and O_Retimer.

3. Set the timeline range to **Input** using the dropdown under the Viewer.

4. In the FrameRange node's Properties, specify the first and last frames you want to retime. For example, 1021 - 1040. This sets the input frames and allows you to visualize the retime using the Curve Editor more easily.
5. Set a keyframe on the first frame specified by the FrameRange node and then set the required keyframes to retime the footage as required.

6. In the O_Retimer Properties, right-click the keyframed Speed control and select Curve editor. The Curve Editor tab displays the keyframes you added to retime the footage.

7. If you move points on the curve, you can see the out point on the Viewer timeline updates to show you how many frames of output are created from the input frames.

**TIP:** You can add points to a curve by holding Ctrl/Cmd+Alt and clicking on the curve. This also adds a corresponding keyframe to the Viewer timeline.

The left-hand curve creates 1042 output frames, whereas the shallower retime curve on the right produces fewer output frames - just 1035 frames. However, the input frames displayed on the x axis of the Curve Editor remain constant between 1021 and 1040.

Varying Retimes Using Source Frames

To vary the speed throughout the footage using the Source Frames method, do the following:

1. Repeat steps 1 to 4 in the Retiming Stereo Footage Using Speed section.
2. Select Source Frame from the Timing dropdown.
3. Move the playhead to the first frame you want to change the output for on the Viewer timeline.
4. Set the source frame number that you want to appear at the selected output position by either entering it in the Frame box, or dragging the Frame slider to the required frame.
5. Set a keyframe for this by selecting the animation menu next to the Frame parameter, and clicking Set key. A keyframe is indicated by a blue marker.
6. Move the playhead on the Viewer timeline to the next frame you want to change the output for.

7. Set the source frame number that you want to appear at the selected output position by either entering it in the Frame box, or dragging the Frame slider to the required frame. A keyframe is set automatically, identified with a blue marker.

8. Add as many keyframes as necessary to produce the required retime. For example, if you had a clip consisting of 15 frames, you could:
   • Set frame 1 to output frame 1, and frame 5 to output frame 2.5, resulting in a half-speed retime.
   • Set frame 10 to output frame 15, resulting in a third speed up.

This creates a retime curve, which you can see if you right-click the Frame control and select Curve editor or Dope sheet from the animation menu. The graph would look something like this:

![Retime Curve Graph](image)

The y axis shows the source frames, and the x axis shows the output frames. By using the Curve Editor or Dope Sheet to adjust this curve, you can create an arbitrarily changing speed for the sequence.

**TIP:** You can add points to a curve by holding **Ctrl/Cmd+Alt** and clicking on the curve. This also adds a corresponding keyframe to the Viewer timeline.
O_Retimer Controls

Use GPU

Open the O_Retimer controls. O_Retimer renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_Retimer defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke's Preferences and selecting an alternative card from the GPU Device dropdown.

NOTE: Selecting a different GPU requires you to restart Nuke before the change takes effect.

Views to Use

From the views that exist in your project settings, select the two views you want to use to perform the retime. These views are mapped for the left and right eye.

Timing

Use the Timing control to set the retiming method.

<table>
<thead>
<tr>
<th>Speed</th>
<th>Select this method if you want to define the retiming in terms of speed of playback and total duration: double-speed halves the duration of the clip and half-speed doubles the duration of the clip.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Frame</td>
<td>Select this method if you want to define the retiming in terms of specifying source frames at different output points in the Viewer timeline. You need to set a minimum of two keyframes to use the Source Frame method.</td>
</tr>
</tbody>
</table>

NOTE: When you are retiming footage using the Speed method, ensure that you set the keyframes on the input frames, as setting keyframes on the input frames alters the number of output frames.
Speed

The Speed control is only available when the Timing method is set to Speed. Use this to alter the playback speed. Values below 1 slow down the clip. Values above 1 speed up movement. For example, to slow down the clip by a factor of two (half speed), set this value to 0.5. Quarter speed would be 0.25.

Frame

The Frame control is only available when the Timing method is set to Source Frame. Use this to specify the source frame at the currently selected output frame on the Viewer timeline. For example, to slow down a 50 frame clip by half, set the Source Frame to 1 at frame 1 and the Source Frame to 50 at frame 100. The default expression results in a half-speed retime.

Edges

Output edges to alpha

You can use the Output edges to alpha checkbox to output the edges to the alpha channel. Use the alpha channel as an overlay to determine where the edge correction is applied.

Adjust Edges

Use the Adjust Edges slider to change the size of the edge mask. To dilate the mask, use a positive value. You can use a negative value to erode the mask.

Feather

You can use the Feather slider to specify how much feathering to apply to the edges. Increasing the Feather value softens the edges; decreasing the value, sharpens the edges.

O_Retimer Example

In this example, we generate motion vectors using O_VectorGenerator and feed them to O_Retimer in order to vary the speed throughout the clip.

You can download the image used here from our website. For more information, please see Example Images.
Step by Step

Generating Motion Vectors

1. Launch Nuke and open the project settings by pressing S on the Node Graph. Go to the Views tab and click the Set up views for stereo button.
2. Import dance_group.##.exr. This image already includes both the left and the right views.
3. Add a Viewer to the image.
4. Select Ocula > Ocula 4.0 > O_VectorGenerator to insert an O_VectorGenerator node after the image sequence.

The purpose of O_VectorGenerator is to calculate the motion vectors required later for retiming the sequence.

5. In the Viewer, select motion from the channel dropdown. The calculated forward and backward motion vectors are displayed in the Viewer.

6. You can render out the motion vectors with the image by selecting Image > Write to insert a Write node after O_VectorGenerator. In the Write node controls, select all from the channels dropdown menu. Choose exr as the file type. Select a location for the clip in the file field and enter dance_group_motion.##.exr as the name. Click Render. The newly created motion vectors are saved in the channels of the clip.

7. Continue to the next section to retime the sequence using these motion vectors.
Retiming the Sequence

1. Import the `dance_group_motion.##.exr` clip you rendered in the previous step and connect a Viewer to it.

![Image of Viewer with dance group motion clip]

2. Play through the clip in the Viewer to get a sense of the motion.
3. Insert an O_Retimer node after the clip.

![Image of O_Retimer node connected to clip]

By default, this node is set to perform a half speed slow down. However, in this example, we want to speed up the sequence.

4. Instead of changing the clip's playback speed in terms of overall duration, we are going to define the retiming by specifying source frames at selected output frames in the Viewer timeline. In order to do this, set Timing to Source Frame in the O_Retimer controls.

Notice that the Speed control is not available and the Frame control is activated. The default Frame setting is 1, which sets the first frame on the timeline to frame 1 of the input image.

![NOTE: You need to set a minimum of two keyframes to use the Source Frame method.]

5. Click the animation menu next to Frame and select Set key in order to set a keyframe at the first frame on the timeline.

![Image of Set key animation menu]

A keyframe is indicated by a blue chip.

6. Move the playhead to frame 8 and set the Frame value to 4. A keyframe is set automatically. O_Retimer sets frame 8 on the timeline to frame 4 of the source image, effectively reducing the playback speed by half leading up to frame 8.

![Image of O_Retimer node with keyframes]
7. Move the playhead to frame 15 and set the **Frame** value to 15. O_Retimer sets frame 15 on the timeline to the last frame of the input clip, effectively returning the playback speed to normal at frame 15.

8. Press Play in the Viewer to process and review the results. After the process is complete, you are able to see the result of the retime. You may have to reduce the **fps** control to see the result clearly.

9. Return to the beginning of the clip and play through each frame with the Next Frame arrow in the Viewer. You should notice that the **Frame** value changes over time – slowly up to frame 8 and more quickly towards the final frame.
InteraxialShifter

Description

The O_InteraxialShifter node allows you to adjust the interaxial distance of stereo images. Interaxial distance is the distance between the left and right cameras. Using this O_InteraxialShifter node, you can generate two new views at specified positions between the left and right images.

Changing interaxial distance is the equivalent of moving the cameras closer together or further apart. The greater the interaxial distance, the greater the depth perception. This is illustrated below, where the gray rectangles represent elements depicted in a stereo image.

When the 3D image was shot with the cameras far apart, objects on the screen seem further apart from each other; the foreground objects look closer to you, and the background objects look further away.

When the 3D image was shot with cameras closer together, objects on the screen seem close to each other; the foreground objects don't look much closer to you than the background objects.

You may want to change interaxial distance during post-production for a variety of reasons. For example, it can be useful when trying to match the depths between scenes in order to make transitions more comfortable for the viewer, or simply because the desired depth of a shot has been reconsidered as the final film evolves. It might also help in the process known as depth grading, where the depth of field is adjusted in order to ensure the stereo effect can be comfortably viewed on the intended screen size. The apparent depth of the scene depends upon a combination of the screen size and the distance from the screen to the viewer.
To generate new views with a different interaxial distance, the O_InteraxialShifter node requires upstream disparity vectors that relate the two views. You can use the O_DisparityGenerator node to calculate these vectors. See DisparityGenerator for how to do this.

NOTE: This node does not pass through any disparity channels fed into it. This is because, after warping the input images, the original disparity map is no longer valid. If you need disparity channels further down the tree, add another O_DisparityGenerator node after O_InteraxialShifter.

TIP: Changing interaxial distance is different to changing convergence (the inward rotation of the cameras). You can change convergence using Nuke's ReConverge node. This way, you can have any selected point in the image appear at screen depth when viewed with 3D glasses.

Inputs

O_InteraxialShifter has the following inputs:

| Source               | A stereo pair of images. If disparity channels and occlusion masks are not embedded in the images, you need to insert an O_DisparityGenerator and an O_OcclusionDetector node after the image sequence. |

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Using O_InteraxialShifter

NOTE: O_InteraxialShifter requires disparity vectors and an occlusion mask to operate correctly.

To change the interaxial distance, do the following:

1. If disparity vectors don't yet exist in the script, insert an O_DisparityGenerator node after your image sequence to calculate the disparity vectors. See DisparityGenerator for how to do this.
2. O_InteraxialShifter requires an occlusion mask. Insert an O_OcclusionDetector node after either the O_DisparityGenerator node (if you added one in the previous step) or the stereo image sequence.
3. From the toolbar, select Ocula > Ocula 4.0 > O_InteraxialShifter to insert an O_InteraxialShifter node after the O_OcclusionDetector node.

4. In the O_InteraxialShifter controls, select the two views you want to use for creating the new views under View to Use. The two views you select are mapped for the left and right eye.

5. Use the Left Position and Right Position sliders to indicate where you want to build the new left and right views. The values are expressed as a fraction of the distance between the two views. For example, if the Left Position is set to 0.25, it is at a quarter of the total distance between the cameras.

6. Attach a Viewer to the O_InteraxialShifter node. Your node tree should now look something like this:

   ![Node Tree Example]

7. Adjust the Edges settings to get the best possible result. See O_InteraxialShifter Controls for more information.

---

**O_InteraxialShifter Controls**

**Use GPU**

Open the O_InteraxialShifter controls. O_InteraxialShifter renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_InteraxialShifter defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke's Preferences and selecting an alternative card from the GPU Device dropdown.

blockquote

> **NOTE:** Selecting a different GPU requires you to restart Nuke before the change takes effect.

**Views to Use**

From the views that exist in your project settings, select the two views you want to use to create the new views. These views are mapped for the left and right eye.
Left Position

Select a position between the views where you want to generate the new left view. The position is expressed as a fraction of the distance between the views.

Right Position

Select a position between the views where you want to generate the new right view. The position is expressed as a fraction of the distance between the views.

Edges

Output edges to alpha

You can use the **Output edges to alpha** checkbox to output the edges to the alpha channel. Use the alpha channel as an overlay to determine where the edge correction is applied.

Adjust Edges

Use the **Adjust Edges** slider to change the size of the edge mask. To dilate the mask, use a positive value. You can use a negative value to erode the mask.

Feather

You can use the **Feather** slider to specify how much feathering to apply to the edges. Increasing the **Feather** value, softens the edges; decreasing the value, sharpens the edges.
DepthToDisparity

Description

Many Ocula nodes rely on disparity maps to produce their output. Usually, disparity maps are created using a combination of the O_Solver (see Solver) and O_DisparityGenerator (see DisparityGenerator) nodes.

However, if you have a CG scene with stereo camera information and a z-depth map available, you can also use the O_DepthToDisparity node to generate the disparity map. Provided that the camera information and z-depth map are correct, this is both faster and more accurate than using the O_Solver and O_DisparityGenerator nodes.

Using one of the camera transforms and the corresponding depth map, O_DepthToDisparity does a back projection from one view to find the position of each image point in 3D space. It then projects this point with the other camera transform to find the position of the point in the other view. The difference between the two positions gives the disparity in one direction.

As with the O_DisparityGenerator node, the final disparity vectors are stored in disparity channels, so you might not see any image data appear when you first calculate the disparity map. To see the output inside Nuke, select the disparity channels from the channel set and channel controls in the top left corner of the Viewer. Examples of what a disparity map might look like using the RGB and R channels are shown below. As you can see, the RGB layers on the left are harder to read than the single R channel.

Once you have generated a disparity map that describes the relation between the views of a particular clip well, it is suitable for use in most of the Ocula nodes. We recommend that you insert a Write node after O_DepthToDisparity to render the original images and the disparity channels as a stereo .exr file. This format allows for the storage of an image with multiple views and channel sets embedded in it. Whenever you use the same image sequence, the disparity map is loaded into Nuke together with the sequence and is readily available for the Ocula nodes. For information on how to
generate a disparity map using O_DepthToDisparity and render it as an .exr file, see Generating a Disparity Map from Depth.

NOTE: To use O_DepthToDisparity, you need the positions of the stereo camera rig for the two views.

## Inputs

O_DepthToDisparity has the following inputs:

| Camera | A Nuke stereo Camera node. This is the camera the scene was rendered with. In most cases, you would import this into Nuke from a third-party 3D application. For information on how to do this, see the Nuke User Guide. In Nuke, a stereo camera can be either:
|        | • a single Camera node in which some or all of the controls are split, or
|        | • two Camera nodes (one for each view) followed by a JoinViews node (Views > JoinViews). The JoinViews node combines the two cameras into a single output. |

| Depth  | This is a stereo pair of images (usually, a scene you've rendered from a 3D application). In the depth channel, there should be a z-depth map for each view. |

NOTE: If this input is an .exr file, the z-depth map may already be embedded in the clip.
If you’re using another file format and have saved the depth map as a separate image, you can use a Nuke Shuffle node (Channel > Shuffle) to get the z-depth map in the depth channel of the Depth image. For information on how to do this, see the Nuke User Guide.

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Generating a Disparity Map from Depth

To generate a disparity map from a depth channel, do the following:

1. Start Nuke and press S on the Node Graph to open the Project Settings. Go to the Views tab and click the Set up views for stereo button.

2. From the Toolbar, select Image > Read to load your stereo clip (usually, a rendered 3D scene) into Nuke. If you don’t have both views in the same file, select Views > JoinViews to combine them, or use a variable in the Read node’s file field to replace the name of the view (use the variable %V to replace an entire view name, such as left or right, and %v to replace an initial letter, such as l or r). For more information, refer to the Nuke User Guide.

Make sure the stereo clip contains a z-depth map for each view in the depth channels. If this is not the case and you have saved the depth maps as separate images, you can use a Nuke Shuffle node (Channel > Shuffle) to shuffle them into the depth channels.

3. Select Ocula > Ocula 4.0 > O_DepthToDisparity to insert an O_DepthToDisparity node after either the stereo clip or the JoinViews node (if you inserted one in the previous step).

4. Connect the camera that the scene was rendered with to the Camera input of O_DepthToDisparity. It is important the camera information is correct for the scene.

5. Open the O_DepthToDisparity controls. From the Views to Use menu or buttons, select which views you want to use for the left and right eye when creating the disparity map.

6. Attach a Viewer to the O_DepthToDisparity node, and display one of the disparity channels in the Viewer.

O_DepthToDisparity calculates the disparity map and stores it in the disparity channels.

7. Select Image > Write to insert a Write node after O_DepthToDisparity. In the Write node controls, select all from the channels dropdown menu. Choose exr as the file type. Render the clip.

The newly created disparity channels are saved in the channels of your stereo clip. When you need to manipulate the same clip again later, the disparity vectors are loaded into Nuke together with the clip.
Rendering the output to combine the clip and the disparity channels for future use.

O_DepthToDisparity Controls

Views to Use

From the views that exist in your Project Settings, select the two views you want to use to create the disparity map. These views are mapped for the left and right eye.

O_DepthToDisparity Example

This example shows you how to calculate a disparity map for a stereo pair of images using O_DepthToDisparity.

You can download the script used here from our website. For more information, please see Example Images.

Step by Step

1. Start Nuke and select File > Open to import the gherkin.nk script.
2. This script has the left and the right view set up in the Project Settings. In the Node Graph, there is a stereo camera node.
3. Select Image > Read to import gherkin.exr and attach a Viewer to the image.
   The image is a render of a 3D scene. It already includes the left and the right view, and a depth channel for both views.
4. Select Ocula > Ocula 4.0 > O_DepthToDisparity to insert an O_DepthToDisparity node between the Read node and the Viewer. Make sure the Read node is connected to the Depth input of O_DepthToDisparity.
5. Connect the Camera node to the **Camera** input of O_DepthToDisparity. This node is the camera the 3D scene was rendered with. Your node tree should now look something like the one shown below.

![Node Tree Diagram]

6. Use the channel menus in the top left corner of the Viewer to display one of the disparity channels.

   O_DepthToDisparity calculates the disparity map and stores it in the disparity channels.

7. Temporarily decrease the **gain** value in the Viewer to see detail in the disparity channels and then switch back to viewing the **rgba** channels.

8. To evaluate the quality of the disparity map, select **Ocula > Ocula 4.0 > DisparityReviewGizmo**. The default settings show that the disparity map is pretty good, with clearly defined depth edges picked out in red. See **DisparityReviewGizmo** for more information.

![Disparity Review Gizmo Screenshot]

It is worth noting that the results achieved with O_DepthToDisparity are usually better than those achieved with O_Solver and O_DisparityGenerator for rendered 3D scenes, though this is not the case with non-CG footage.
DisparityToDepth

Description

The O_DisparityToDepth node produces a z-depth map for each view of a stereo clip, based on the clip's disparity map and stereo camera setup.

A z-depth map is an image that uses the brightness of each pixel to specify the distance between the 3D scene point and the virtual camera used to capture the scene. For example, you may use a z-depth map if you want to introduce fog and depth-of-field effects into a shot. In Nuke, the ZDefocus node (Filter > ZDefocus) requires a depth map in its input.

O_DisparityToDepth stores the final z-depth map in the depth channel. Select depth from the channel dropdown above the Viewer to display the z-depth map. You can then select the red only channel (R) and adjust the gain and gamma sliders above the Viewer to display the depth map more clearly.

It's worth mentioning that depth is the distance along the Z axis for the camera, not the distance along the ray from the camera center to the 3D surface point. This matches the depth output of the ScanlineRender node in Nuke. In the figure below, each pixel forms a ray, and AB measures the physical distance from the camera to the 3D point, whereas AC measures the distance along the Z axis for the camera.
Inputs

O_DisparityToDepth has the following inputs:

| Camera | A pre-tracked Nuke stereo camera that describes the camera setup that is used to shoot the **Source** images. This can be a camera you have tracked with the CameraTracker node or imported to Nuke from a third-party camera tracking application. In Nuke, a stereo camera can be either:
|        | • a single Camera node in which some or all of the controls are split, or
|        | • two Camera nodes (one for each view) followed by a JoinViews node (**Views > JoinViews**). The JoinViews node combines the two cameras into a single output.
| Disparity | A stereo pair of images. O_DisparityToDepth requires upstream disparity
vectors. If they do not already exist, insert an O_DisparityGenerator node after the image sequence to calculate them. See DisparityGenerator for more information.

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Using O_DisparityToDepth

NOTE: O_DisparityToDepth requires upstream disparity vectors to operate correctly.

To generate a z-depth map for a stereo clip, do the following:
1. If disparity vectors do not yet exist in the script, you need to insert an O_DisparityGenerator node after your image sequence to calculate the disparity vectors. See DisparityGenerator for how to do this.

TIP: O_DisparityGenerator's Alignment control can reduce noise on the vertical component of disparity and produce smoother variations in depth, but requires solve data upstream. See Solver for more information on calculating solve data.

2. Select Ocula > Ocula 4.0 > O_DisparityToDepth to insert an O_DisparityToDepth node either after the O_DisparityGenerator node if you added one in the previous step, or after the image sequence.
3. Connect a pretracked Nuke stereo camera to the Camera input of O_DisparityToDepth.
4. Open the O_DisparityToDepth controls. From the Views to Use menu, select which views you want to use for the left and right eye when creating the z-depth map.
5. Connect a Viewer to the O_DisparityToDepth node. Your node tree should now look something like this:
6. Select **depth** from the channel dropdown above the Viewer.
7. Select the red channel (R) from the **RGB** dropdown.
8. Adjust the **gain** and **gamma** controls to display the z-depth map more clearly.
9. To render out the stereo clip with the depth map stored in the depth channel, select **Image > Write** to insert a Write node after **O_DisparityToDepth**. In the Write node controls, select **all** from the **channels** dropdown menu. Choose **exr** as the **file type**. Render the clip. When you need to use the same clip again later, the z-depth map is loaded into Nuke along with the clip. To view it, select **depth** from the channel dropdown.

**Using O_DisparityToDepth with DeepMerge**

You can use choose to use DeepMerge with O_DisparityToDepth to insert objects at specific depth positions. For example, if you have an image of dancers, you may want to add in smoke around them.

![NOTE: The original image or image sequence you are using must contain an alpha channel for DeepMerge to operate correctly.](image)

To use O_DisparityToDepth with DeepMerge, you can do the following:

1. Ensure your footage has an alpha channel and repeat steps 1 to 5 from the previous section to calculate the depth from disparity.
2. Insert a **DeepFromImage** node after the **O_DisparityToDepth** node by selecting **Deep > DeepFromImage**.
3. Insert a **DeepMerge** node after the **DeepFromImage** node, and change the input from the **DeepMerge** node so that the **DeepFromImage** node is connected using the **B** input.
4. Now, insert a separate **DeepRead** node by selecting **Deep > DeepRead** and navigate to the deep footage that you want to insert into your original image. In this case, we are inserting smoke around dancers in a stereo scene.
5. Attach the **DeepRead** node to the **A** input of **DeepMerge**. Your node tree should now look something like this:
6. The smoke does not appear in the image. This must mean that the depth values of the image and the deep smoke are vastly different; one is too far in front of the other to be visible.

7. You can visualise the different depth positions by using a point cloud. To do this, insert a DeepToPoints node after the DeepMerge node.

8. Attach the camera input of the DeepToPoints node to the existing camera. This shows you a 3D view with a camera detailing the xyz coordinates of the footage in space.

As you can see, the smoke footage is much too far back and also too large.

9. To change the position and scale of the smoke footage, insert a DeepTransform node after the smoke footage DeepRead node and before the DeepMerge node. Your node tree should now look something like this:

10. Open the DeepTransform controls. To move the position of the smoke footage forward, you need to increase the z value in the translate control. To decrease the depth of the smoke footage, you need to increase the zscale value. This is a divisor, so numbers above 1 decrease the depth.
The DeepToPoints view of the original stereo image from the side.

The DeepToPoints view of the smoke footage and the original stereo image from the side.

The same DeepToPoints view of the original stereo image from the front.

The same DeepToPoints view of the smoke footage and the original stereo image from the front.

11. Press **D** over the DeepToPoints node to disable it and display the result. Press **Tab** to revert back to 2D.
O_DisparityToDepth Controls

Views to Use

From the views that exist in your project settings, select the two views you want to use to generate the z-depth map. These views are mapped for the left and right eye.

O_DisparityToDepth Example

In this example, we first generate a depth map for a stereo pair of images using O_DisparityToDepth. Then, we blur the image according to the depth map.

Step by Step

1. Launch Nuke and select **File > Open** to import the **depth.nk** script. This script has the left and the right view set up in the Project Settings. In the Node Graph, there is a stereo Camera node.

2. Select **Image > Read** to import **Dance_Group.exr** into Nuke.

3. To calculate disparity vectors, insert an **O_DisparityGenerator** node after the image. See **DisparityGenerator** for more information.

4. Select **Ocula > Ocula 4.0 > O_DisparityToDepth** to insert an **O_DisparityToDepth** node after the **O_DisparityGenerator** node. Ensure the **O_DisparityGenerator** node is connected to the **Disparity** input of **O_DisparityToDepth**.
5. Connect the Camera node to the **Camera** input of **O_DisparityToDepth**. Your node tree should now look something like this:

![Node Tree Diagram](image)

6. Select **depth** from the **channel** dropdown above the Viewer to display the z-depth map.
7. Select the red channel (R) from the **RGB** dropdown, and adjust the **gain** and **gamma** sliders above the Viewer to see the depth map more clearly.

![Stereoscopic Images](image)

| The left view of a stereo image. | The red channel of the depth map (with adjusted gain and gamma). |

8. To use **O_DisparityToDepth** with **ZDefocus**, continue to the next section.

### An Example of Using **O_DisparityToDepth** with **ZDefocus**

The **ZDefocus** node blurs the image background according to the depth channels we've just created in the previous section.

1. Continue with the node tree you created in the previous section.
2. If you inserted a **DisparityReviewGizmo** node, either disable or delete it.
3. Select **Filter > ZDefocus** to add a **ZDefocus** node between **O_DisparityToDepth** and the Viewer. Your node tree should now look something like this:
4. Open the ZDefocus controls. A focal_point widget is displayed in the Viewer. Click and drag the focal_point widget to the area on the image you want to be in focus. In this case, we want the dancer at the front to be the focus point.

5. Then set the ZDefocus controls as follows:
   - channels to rgba
   - math to far=0
   - size to 63

6. View the output. To see the effect of the ZDefocus node more accurately, select the node in the node tree and press D once to disable and again to enable it.

7. You can save the depth in the channels of the input clip for later use. To do this, select Image > Write to insert a Write node between O_DisparityToDepth and ZDefocus. In the Write node controls, select all from the channels dropdown menu. Use the file control to give the file a location and a new name. Choose exr as the file type and click Render to render the image.
MultiSample

Description

The O_MultiSample node allows you to use a Sample input to sample a selected area for one of the following uses:

• Fill the whole image with a smooth interpolation of the data from the sample area. Any channels can be used including depth, color, disparity, and so on. For example, if you like the disparity vectors in a certain area, you can expand them to fill the rest of the image by selecting the area you like in the Sample input and choosing disparity from the Channels dropdown.

• Fill the area defined in the Sample input with the surrounding channel data. This allows you to fill holes or replace unwanted areas. Any channels can be used including depth, color, disparity, and so on. For example, if you had an irregular area in a depth map, you can select the irregular area and input it as a sampled area. You can then invert the sampled area to fill it with the surrounding depth data.

• Fill a different area specified in the Mask input, with the channel data from the area specified in the Sample input. This allows you to correct areas that have not been rebuilt correctly, or replace any irregular areas. Any channels can be used including depth, color, disparity, and so on. For example, if you had an area that has not been rebuilt correctly when using O_NewView, you can correct it using O_MultiSample. See Using O_MultiSample with the Sample and Mask Inputs for more information.

Inputs

O_MultiSample has the following inputs:
Sample | An input that defines an area to sample channel data from. This can be used for filling the whole image with a smooth interpolation of the data from the sample area, inverting the sampled area and therefore filling the it with the surrounding channel data, or using the sample channel data to fill another area specified in the optional Mask input.

Source | An image or a stereo pair of images.

Mask | An optional mask to define an area that you want to fill or replace using the channel data from the area defined by Sample input.

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Using O_MultiSample

You can use O_MultiSample with only the Sample input to fill in holes or replace data, or you can use O_MultiSample with the Mask and Sample input, to apply the sample data to a different area specified in the Mask input.

Using O_MultiSample with the Sample Input

You can use O_MultiSample to replace an area of channel data that is not accurate. For example, if you have generated a disparity map and there is a section that you are not happy with, you can use O_MultiSample to fill the specified shape with the surrounding disparity vector data from the whole image. This means that the disparity vectors in the selected area are replaced.

To use O_MultiSample to remove an area of disparity, you can do the following:
1. If disparity vectors do not already exist in the image sequence, insert an O_DisparityGenerator node after your image sequence to calculate the disparity vectors. See DisparityGenerator for more information.
2. Select Ocula > Ocula 4.0 > O_MultiSample to insert an O_MultiSample node, either after the O_DisparityGenerator node if you added one in the previous step, or after the image sequence.
3. Create a Roto node and connect it to the Sample input of the O_MultiSample node.
4. Connect a Viewer to the O_MultiSample node. Your node tree should now look something like this:
5. Open the Roto controls. Draw a bezier around the area of disparity you want to remove.

6. Select disparity from the channel dropdown. Select Luminance from the RGB dropdown.
7. Adjust the gain and gamma controls to display the disparity map more clearly.
8. Open the O_DisparityGenerator controls. You can set the Strength control to 2, and the Smoothness control to 3.
9. Open the O_MultiSample controls and set the Channels control to disparity.
10. Set the Sample control to Sample Inverted Alpha. This removes the disparity from the selected area by filling it using the surrounding disparity vector data.
Using O_MultiSample with the Sample and Mask Inputs

You can use O_MultiSample to correct a specific area with data from the defined sample area. For example, if you use O_NewView to create a new left view and an area of it doesn't build correctly, you can use O_MultiSample to correct it.

In this case, you first create a new view:

1. O_NewView requires disparity vectors to operate correctly. If disparity vectors do not exist already, insert an O_DisparityGenerator to calculate them either after the image sequence, or – if you added one – the JoinViews node.
2. O_NewView also requires an O_OcclusionDetector node. Insert this after the O_DisparityGenerator node.
3. Select Ocula > Ocula 4.0 > O_NewView to insert an O_NewView node and attach a Viewer to the O_NewView node. Your node tree should now look something like this:

![Node Tree Diagram]

4. In the O_NewView controls, select **Left from Right** from the **View to Build** dropdown to rebuild the left view using the pixels from the right.

![Example Images]
The rebuilt left view from the right. The original right view.
Notice the shoulder has not been reconstructed correctly.

In this case, the left view has not been reconstructed correctly as a result of an irregular area in the disparity. This can be corrected by using O_MultiSample:

5. Insert an O_MultiSample node before the O_NewView node.
6. Create a Roto node and attach it to the Sample input of O_MultiSample. Draw a bezier around an area of disparity that you want to sample. In most cases, sampling the disparity from an area that is close produces the best results (see screenshot below). In this case, we need to sample the disparity from a close area at the same depth / or surface to get the best result.

7. Create a RotoPaint node and attach it to the Mask input of O_MultiSample. Your node tree should now look something like this:

8. From the RotoPaint node, use a paint brush to paint a stroke over the area of disparity that you want to correct. To view the stroke on the image, connect the bg input of the RotoPaint node to
the O_DisparityGenerator node and then attach the RotoPaint to Viewer. Choose to display Viewer input 2. Your node tree should now look something like this:

In this case, when you have the node tree set up like this and you display Viewer input 2, this is what you see:

9. Choose to display Viewer input 1.

10. Open the O_MultiSample controls, set the Channels control to disparity.

11. Set the Sample control to Sample Alpha, and the Mask control to Mask Alpha. This now fills the area specified area in the Mask input, with the data sampled from the specified area in the Sample input. In the image, the painted area is now rebuilt correctly. To compare the result to the original, press D several times with the O_MultiSample node selected, to disable and re-enable it.
The image with the O_MultiSample node disabled, showing the incorrectly built shoulder.

The image with the O_MultiSample node enabled, showing the correctly rebuilt view.

O_MultiSample Controls

Use GPU

Open the O_MultiSample controls. O_MultiSample renders using the Local GPU specified, if available, rather than the CPU. The GPU may significantly improve processing performance.

If there is no suitable GPU, or the required NVIDIA CUDA drivers are unavailable, O_MultiSample defaults to using the CPU. You can select a different GPU Device, if available, by opening Nuke’s Preferences and selecting an alternative card from the GPU Device dropdown.

NOTE: Selecting a different GPU requires you to restart Nuke before the change takes effect.

Channels

You can use the Channels control to select the channels you want to sample from the Source input. For example, you can select disparity to sample the disparity vector data only in the selected area. The sample data can then be applied to the whole image, or the selected area can be filled (replaced) using the surrounding channel data. When you select a channel set, the individual channels within that set are displayed to the right as checkboxes. You can toggle these on and off to view selected channels.

When you select an additional channel from the second channel dropdown, the checkbox beside it is automatically enabled. You can then toggle this on and off to use the additional channel.
Sample

The **Sample** control defines the area to sample. You can connect a RotoPaint or Roto node to define an area, and connect it to the **Sample** input, or you can use the source alpha channel.

![NOTE: Areas with non-zero values in the Sample input are sampled. Pixels with higher values are given more weight when they are expanded.](image)

Select one of the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>None</strong></td>
<td>Use the entire image area.</td>
</tr>
<tr>
<td><strong>Source Alpha</strong></td>
<td>Use the alpha channel of the <strong>Source</strong> input to define the area of the <strong>Source</strong> input that is sampled and expanded. The selected area is embedded in the image sequence.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="The left view." /> <img src="image" alt="A matte overlay of the Source alpha. White pixels define where the Source input is sampled." /></td>
</tr>
<tr>
<td><strong>Source Inverted Alpha</strong></td>
<td>Use the inverted alpha channel of the <strong>Source</strong> input to define the area of the <strong>Source</strong> input that is sampled and expanded. The selected area is embedded in the image sequence.</td>
</tr>
</tbody>
</table>
The left view.

A inverted matte overlay of the inverted Source alpha. White pixels determine where the Source input is sampled.

| Sample Luminance | Use the luminance of the Sample input to define the area of the Source input that is sampled and expanded. |
| Sample Inverted Luminance | Use the inverted luminance of the Sample input to define the area of the Source input that is sampled and expanded. |
| Sample Alpha | Use the alpha channel of the Sample input to define the area of the Source input that is sampled and expanded. |
| Sample Inverted Alpha | Use the inverted alpha channel of the Sample input to define the area of the Source input that is sampled and expanded. |

Mask

The Mask control defines the area in which to expand the sample data. You can connect a RotoPaint or Roto node to define an area, and connect it to the Mask input, or you can use the source alpha channel.

![Mask](image)

NOTE: The Mask input is used to perform a keymix between the Source input and the expanded result.

Select one of the following options:

| None | The Mask input is not used, and therefore the sample data is expanded to fill the whole image. |
| Source Alpha | Use the alpha channel of the Source input to define the area that is |
replaced by the expanded data. The selected area is embedded in the image sequence.

<table>
<thead>
<tr>
<th>Mask</th>
<th>Source Inverted Alpha</th>
<th>Use the inverted alpha channel of the Source input to define the area that is replaced by the expanded data.</th>
<th>The left view.</th>
<th>A matte overlay of the Source alpha. White pixels define the area to be replaced by the expanded data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask Luminance</td>
<td>Use the luminance of the Mask input to define the area that is replaced by the expanded data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mask Inverted Luminance</td>
<td>Use the inverted luminance of the Mask input to define the area that is replaced by the expanded data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mask Alpha</td>
<td>Use the alpha channel of the Mask input to define the area that is replaced by the expanded data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mask Inverted Alpha</td>
<td>Use the inverted alpha channel of the Mask input to define the area that is replaced by the expanded data.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
O_MultiSample Example

In this example, we first generate a disparity map for a stereo pair of images using O_DISparityGenerator. Then, we use O_MultiSample to remove an area of the disparity and fill it using the surrounding disparity vector data.

Step by Step

1. Launch Nuke. Open the Project Settings (press S on the Node Graph), select the Views tab, and click the Set up views for stereo button.
2. Select Image > Read to import Dance_Group.exr.
3. To calculate disparity vectors, insert an O_DisparityGenerator node after the image sequence. See DisparityGenerator for more information.
4. Select Ocula > Ocula 4.0 > O_MultiSample to insert an O_MultiSample node after the O_DisparityGenerator node.
5. Insert a Roto node and connect it to the Sample input of the O_MultiSample node.
6. Attach a Viewer to the O_MultiSample node. Your node tree should now look something like this:

7. Select disparity from the channel dropdown above the Viewer to display the disparity map.
8. Select the red channel (R) from the RGB dropdown, and set the gain control to -1/45.
9. Adjust the gamma slider above the Viewer to display the disparity map more clearly.
10. Open the Roto controls and select Bezier to the left of the Viewer. Draw a bezier around the dancer on the right.
11. Open the O_MultiSample controls. Select disparity from the Channels control.
12. Set the Sample control to Sample Inverted Alpha. The selected area of disparity is now filled using the surrounding disparity vector data, replacing the disparity in the selected area.
The original left view of the disparity map with a bezier marking the selected area. The left of view of the disparity with the selected area removed.
Quality Control Tools

Description

There are several tools in Ocula that you can use to check the quality of stereo footage and disparity in an image sequence. These tools include O_DisparityViewer, DisparityReviewGizmo and StereoReviewGizmo.

DisparityViewer

Description

The O_DisparityViewer node allows you to visualize the disparity vectors in your node tree, display a histogram detailing positive and negative parallax, or overlay the Viewer with parallax violations.

NOTE: All three O_DisparityViewer modes are baked into your render if the node is enabled when you write your sequence out. This allows you to render the output alongside disparity for review.

You can add O_DisparityViewer after any node in the Node Graph. As long as there is a disparity channel at that point in the tree, O_DisparityViewer produces a Viewer overlay with arrows showing the disparity vectors at regular intervals, a histogram, or parallax violations depending on the Display control.

O_DisparityViewer allows you to view diagnostic tools to help you determine where your stereo footage needs work:

• **DisparityViewer** - the disparity vectors at any given point in your node tree for a selected view or both views.

• **Displaying Parallax Histograms** - a histogram showing the parallax, in pixels, on the x axis, the number of image pixels on the y axis, and the negative and positive parallax violation areas in red and green, respectively.

• **Displaying Parallax Violation Overlays** - highlights the areas of negative and positive parallax violation - that is, areas outside the limits specified in the Negative Limit and Positive Limit controls.
Displaying Disparity Vectors

To visualize the disparity vectors at any given point of your node tree, do the following:

1. Select a node at any point in the node tree where there is a disparity channel.
   - If you don’t have a disparity channel in the data stream, you can add one using an O_DisparityGenerator node. See DisparityGenerator for more information.

2. Choose Ocula > Ocula 4.0 > O_DisparityViewer from the Toolbar.
   - This inserts an O_DisparityViewer node in your node tree.

3. Under Views to Use, select the views you want to use to visualize the disparity vectors. These views are mapped for the left and right eye.

4. Using the Display menu, select DisparityVectors.

5. If you want to display the vectors for both views rather than just the current view, check Show Both Directions.

6. Zoom in to better see the disparity vectors in the overlay.

7. If the Viewer seems too cluttered or the arrows overlap, increase the Vector Spacing value.

8. If necessary, use the disparityR and disparityL parameters to change the color of the arrows. You may want to do this, for example, if the default color is very close to the colors in your input image, or if you want to compare disparity methods and have more than one O_DisparityViewer overlay displayed at once. See O_DisparityViewer Controls for more information.

**NOTE:** Disparity vectors are baked into your render if the node is enabled when you write your sequence out.

Displaying Parallax Histograms

To view a parallax histogram for any given point of your node tree, do the following:

1. Select a node at any point in the node tree where there is a disparity channel.
If you don't have a disparity channel in the data stream, you can add one using an O_DisparityGenerator node. See DisparityGenerator for more information.

2. Choose Ocula > Ocula 4.0 > O_DisparityViewer from the Toolbar.
   This inserts an O_DisparityViewer node in your node tree.

3. Under Views to Use, select the views you want to use to visualize as a histogram. These views are mapped for the left and right eye.

4. Using the Display dropdown menu, select Parallax Histogram.

The Viewer displays a histogram showing Parallax (in pixels) on the x axis, the number of image pixels on the y axis, and the negative and positive parallax violation areas in red and green, respectively.

The screen is placed at zero on the x axis, so negative parallax refers to parts of the image that are in front of the screen and positive parallax to the parts that are behind the screen.

5. Use the Histogram controls to define your histogram as necessary. See O_DisparityViewer Controls for more information.

NOTE: Histograms are baked into your render if the node is enabled when you write your sequence out.

Displaying Parallax Violation Overlays

To view a parallax violation overlay for any given point of your node tree, do the following:

1. Select a node at any point in the node tree where there is a disparity channel.
   If you don't have a disparity channel in the data stream, you can add one using an O_DisparityGenerator node. See DisparityGenerator for more information.

2. Choose Ocula > Ocula 4.0 > O_DisparityViewer from the Toolbar.
This inserts an O.DisparityViewer node in your node tree.

3. Under **Views to Use**, select the views you want to use to visualize parallax violation. These views are mapped for the left and right eye.

4. Using the **Display** dropdown menu, select **Parallax Violation**.

Parallax violation overlay.

The parallax violation overlay appears in the Viewer, highlighting the areas of negative and positive parallax violation - that is, areas outside the limits specified in the **Negative Limit** and **Positive Limit** controls.

5. Use the **Parallax** controls to define your overlay parameters as necessary. See O.DisparityViewer Controls for more information.

![Note](image.png)

**NOTE:** Parallax violation overlays are baked into your render if the node is enabled when you write your sequence out.

**Inputs**

O.DisparityViewer has the following inputs:

| Source | This is any node in the node tree with a disparity map in the disparity channels. |

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.
O-DispositionViewer Controls

Views to Use

From the views that exist in your project settings, select the two views you want to use when visualising the disparity vectors. These views are mapped for the left and right eye.

Display

Select the display mode from the dropdown menu:

<table>
<thead>
<tr>
<th>Display Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disparity Vectors</td>
<td>Overlays the disparity vectors at any given point in your node tree for a selected view or both views.</td>
</tr>
<tr>
<td>Parallax Histogram</td>
<td>Displays a histogram showing the parallax, in pixels, on the x axis, the number of image pixels on the y axis, and the negative and positive parallax violation areas in red and green, respectively.</td>
</tr>
<tr>
<td>Parallax Violation</td>
<td>Highlights the areas of negative and positive parallax violation - that is, areas outside the limits specified in the Negative Limit and Positive Limit controls.</td>
</tr>
</tbody>
</table>

Vectors

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disparityR</td>
<td>Color of the arrows used for displaying left-to-right disparity. You may want to change this, for example, if the color of the arrows is very close to the colors in your input image, or if you want to compare the vectors from multiple O-DispositionViewers in the same Viewer.</td>
</tr>
<tr>
<td>disparityL</td>
<td>Color of the arrows used for displaying right-to-left disparity. You may want to change this, for example, if the color of the arrows is very close to the colors in your input image, or if you want to compare the vectors from multiple O-DispositionViewers in the same Viewer.</td>
</tr>
<tr>
<td>Show Both Directions</td>
<td>Check this to show the disparity vectors for both views rather than just the current view.</td>
</tr>
</tbody>
</table>
## Vector Spacing

How often a disparity vector is drawn. If necessary, you can increase this value to make the display less cluttered. You may want to do so, for example, if the disparities are large and you don't want neighboring vectors to overlap one another.

### Histogram

<table>
<thead>
<tr>
<th><strong>Histogram Range</strong></th>
<th>Use this menu to select the histogram range:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Automatic</strong> - the range is scaled to fit the range of disparity.</td>
<td></td>
</tr>
<tr>
<td>- <strong>User Defined</strong> - the range is defined using the Histogram Min and Max controls as a percentage of screen width.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Histogram Min</strong></th>
<th>Controls the lower limits of the histogram as a percentage of screen width, that is, the left-most points on the x axis.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTE: This control is only active when Histogram Range is set to UserDefined.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Histogram Max</strong></th>
<th>Controls the upper limits of the histogram as a percentage of screen width, that is, the right-most points on the x axis.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTE: This control is only active when Histogram Range is set to UserDefined.</td>
<td></td>
</tr>
</tbody>
</table>

## Parallax

<table>
<thead>
<tr>
<th><strong>Negative Limit</strong></th>
<th>Sets the amount of negative parallax allowed as a percentage of screen width. Areas outside this negative limit are marked by the overlay in the color specified in the Negative Violation control.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Pixels</strong></th>
<th>Displays the number of pixels allowed by negative parallax.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Positive Limit</strong></th>
<th>Sets the amount of positive parallax allowed as a percentage of screen width. Areas outside this positive limit are marked by the overlay in the color specified in the Positive Violation control.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Pixels</td>
<td>Displays the number of pixels allowed by positive parallax.</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Negative Violation</strong></td>
<td>Sets the overlay color for pixels outside the specified <strong>Negative Limit</strong> value.</td>
</tr>
<tr>
<td><strong>Positive Violation</strong></td>
<td>Sets the overlay color for pixels outside the specified <strong>Positive Limit</strong> value.</td>
</tr>
</tbody>
</table>

**O_DisparityViewer Example**

In this example, we use O_DisparityViewer to evaluate the disparity map produced by O_DisparityGenerator. For information on where to get the sample footage, please see Example Images.

**Step by Step**

1. Start Nuke and press `S` on the Node Graph to open the Project Settings. Go to the Views tab and click the **Set up views for stereo** button.
2. Select **Image > Read** and browse to where you saved the tutorial files. Go to the **O_DisparityGenerator** directory, select **Dance_Group.exr**, and click **Open**.
   
   A Read node is added to the Node Graph.

   📌 TIP: If you've already run through the **O_DisparityGenerator Example** to create a disparity map, you can open the image you rendered earlier and skip to step 4.

3. Select **Ocula > Ocula 4.0 > O_DisparityGenerator** from the Toolbar.
   
   This inserts an O_DisparityGenerator node.

4. Select the O_DisparityGenerator node (or the Read node pointing to your own pre-rendered image) and choose **Ocula > Ocula 4.0 > O_DisparityViewer** from the Toolbar.
   
   This inserts an O_DisparityViewer node and forces Ocula to calculate the disparity channel.

5. Connect a Viewer to the O_DisparityViewer node so we can see what's happening. Your node tree should look similar to the image shown.

6. Using the default O_DisparityViewer settings, the Viewer is a little crowded. To make the display less cluttered, set **Vector Spacing** to 80.
7. To display vectors for both views, check **Show Both Directions** in the O_DisparityViewer controls. The vectors for the left-to-right disparity are shown in red, and the vectors for the right-to-left disparity in green.

![Image of vectors for both views]

The main purpose of this tutorial is to show you how to use the **Parallax Histogram** and **Parallax Violation** display modes.

8. In the O_DisparityViewer controls, click on the **Display** dropdown menu and select **Parallax Histogram**.

   The Viewer displays a histogram showing parallax (in pixels) on the x axis, the number of image pixels on the y axis, and the negative and positive parallax violation areas in red and green, respectively.

   **NOTE:** The screen is placed at zero on the x axis, so negative parallax refers to parts of the image that are in front of the screen and positive parallax to the parts that are behind the screen.

   ![Histogram with red and green areas indicating parallax violations]

   **HistogramRange** defaults to **UserDefined**, so the graph produced may not contain all the pixels in the image.

9. Click the **Histogram Range** menu and select **Automatic**.

   The histogram is re-rendered to automatically fit the range of disparities in the image.
As you can see in this example, a small proportion of the image exceeds the positive violation threshold (green) and none of the image exceeds the negative threshold (red).

10. In the O_DisparityViewer controls, click on the Display menu and select ParallaxViolation. The parallax violation shown in the histogram is overlaid on the Viewer highlighting, in this case, the areas of positive parallax violation.

11. If you adjust the Positive Limit slider down to 1.4 and up to 3, you can see the changing extent of the positive violation limit. In the right-hand image, the violation has disappeared completely.

12. You can adjust the Negative Limit in the same way, though in this case you’ll need to input a figure greater than 0 to see any violation.
DisparityReviewGizmo

Description

The DisparityReviewGizmo node allows you to check the quality of the disparity vectors. The DisparityReviewGizmo node is inserted before the Viewer in the node tree. This displays the image with a desaturated background by default, and highlights any depth changes, alignment changes, and occluded regions.

⚠️ **NOTE:** To detect the occluded regions, you need to insert an O_OcclusionDetector node upstream in the node tree.

Inputs

O_DisparityReviewGizmo has the following inputs:
<table>
<thead>
<tr>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>A stereo pair of images. DisparityReviewGizmo requires upstream disparity vectors. If they do not already exist, insert an O_DisparityGenerator node after the image sequence to calculate them. See DisparityGenerator</td>
</tr>
</tbody>
</table>

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.

Using DisparityReviewGizmo

⚠️ **NOTE:** DisparityReviewGizmo requires upstream disparity vectors to operate correctly.

1. If disparity vectors do not yet exist in the script, you need to insert an O_DisparityGenerator node after your image sequence to calculate the disparity vectors. See DisparityGenerator for how to do this.

2. Select **Ocula > Ocula 4.0 > DisparityReviewGizmo** to insert an DisparityReviewGizmo node either after the O_DisparityGenerator node if you added one in the previous step, or after the image sequence.

3. Connect a Viewer to the O_DisparityReviewGizmo node. Your node tree should now look something like this:

4. In the DisparityReviewGizmo controls, output is set to stereo by default. You can also change this to disparityR (right), disparityL (left), or side-by-side. See the Controls section for more information.

5. You can choose different backgrounds to view including image, desaturated and disparity. By default, the background control is set to desaturated. This displays the image with a desaturated background and highlights the following:
   - Depth changes are highlighted in red.
   - Alignment changes are highlighted in green.
   - Occluded pixels are highlighted in blue. To detect the occluded regions, you need to insert an O_OcclusionDetector node upstream in the node tree.
The original left view of a stereo image.

The left view with DisparityReviewGizmo node highlighting any depth changes, alignment changes, and occluded regions.

6. Adjust the DisparityReviewGizmo controls to get the required overlay. See the following Controls section for more information.

**DisparityReviewGizmo Controls**

**Output**

This is set to **stereo** by default. When you are using Ocula to update one view to match another, it is advised to check the quality of the view that has been updated. Set the **output** to one of the following:

<table>
<thead>
<tr>
<th><strong>stereo</strong></th>
<th>Select <strong>stereo</strong> to check the quality of both views (left and right) at the same time.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>disparityR (right)</strong></td>
<td>Select this to check the quality of the right view and <strong>disparityR</strong> only.</td>
</tr>
</tbody>
</table>

**disparityR** is a vector that starts in the left view, which points to the matching pixels in the right view, and pulls them back into the left view.

samples these pixels.

So the **R** in **disparityR** means that it samples from the right view and the
vectors originate in the left view. They define how to match the right view and rebuild the left just from the right pixels.

**disparityL (left)**

Select this to check the quality of left view and **disparityL** only. **disparityL** is a vector that starts in the left view, which points to the matching pixels in the right view, samples these pixels, and pulls them back into the left view.

So the L in **disparityL** means that it samples from the left view and the vectors originate in the right view. They define how to match the right view and rebuild the left just from the right pixels.

**side-by-side**

Select **side-by-side** to output both views beside each other.

---

**Background**

Use the **background** control to set the image displayed in the background, which the highlights are then displayed on top of. This is set to **desaturated** by default.

**image**

Select **image** to display the original image in the background.
<table>
<thead>
<tr>
<th>desaturated</th>
<th>Select <strong>desaturated</strong> to display a grayscale version of the original image. This makes the highlights more visible.</th>
</tr>
</thead>
<tbody>
<tr>
<td>disparity</td>
<td>Select <strong>disparity</strong> to view the disparity vectors for the selected view. This is useful for when you are tuning or editing upstream disparity as it helps you understand how the vectors change as well as allowing you to quality check the output. Set the disparity <strong>Limits</strong> controls to grade the disparity vectors.</td>
</tr>
</tbody>
</table>

**Intensity**

Use the **intensity** control to set the intensity of the background image. To make the quality check highlights more visible, reduce the **intensity** control.

**xDisparity**

You can use the **xDisparity Intensity** control to set the amount of horizontal disparity changes (depth changes) displayed. These changes are highlighted in red. This is useful when you are tuning O_DispparityGenerator to ensure depth changes are correct, or when checking a disparity precomp to determine if there are any irregularities that need to be corrected.

**yDisparity**

Use the **yDisparity Intensity** control to set the amount of vertical disparity changes (alignment changes) displayed. These changes are highlighted in green. This is useful to check the quality of vertical disparity when performing a vertical alignment, or to check vertical alignment along with depth to see if there is a misalignment in the camera rig.
Occlusion

NOTE: To detect the occluded regions, you need to insert an O_OcclusionDetector node upstream in the node tree.

You can use the occlusion control to set the amount of occluded regions displayed. In DisparityReviewGizmo, these changes are highlighted in blue. Occluded regions are pixels that are visible in one view and not the other. This is useful to quality check occluded regions when rebuilding or updating one view from another.

Limits

NOTE: These controls are only available when the background control is set to disparity.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xMinimum</td>
<td>Set the minimum parallax used to grade horizontal disparity.</td>
</tr>
<tr>
<td>xMaximum</td>
<td>Set the maximum parallax used to grade horizontal disparity.</td>
</tr>
<tr>
<td>yMinimum</td>
<td>Set the minimum pixel offset used to grade the vertical disparity.</td>
</tr>
<tr>
<td>yMaximum</td>
<td>Set the maximum pixel offset used to grade the vertical disparity.</td>
</tr>
</tbody>
</table>

Set Limits

Click Set Limits to automatically set the minimum and maximum parallax and vertical offset, by sampling the input disparity at the current frame. You can keyframe the limit settings at different frames in the sequence.

Reset Limits

Click this to reset the disparity limits to the default values.
StereoReviewGizmo

Description

You can use the StereoReviewGizmo node to perform a quick quality check on stereo footage. StereoReviewGizmo requires upstream disparity vectors. If disparity vectors do not already exist in the image sequence, the StereoReviewGizmo node calculates disparity.

The StereoReviewGizmo can display a difference view to show the differences in the stereo sequence more clearly. You can set the point of interest using the Parallax controls and you can set a sample subject to focus on by dragging and dropping the sample widget in the Viewer.

The original left view of a stereo image. The difference view created by the StereoReviewGizmo node.

You can use StereoReviewGizmo to quickly toggle between the difference view, the stereo view, and the left and right views to do quick comparisons. Perform a short playback in any view to check temporal stability, or animate the depth rack to display the difference between the specified near and far settings.

Inputs

O_StereoReviewGizmo has the following inputs:

| Source | A stereo pair of images. StereoReviewGizmo uses disparity vectors. However, if there are no disparity vectors available, StereoReviewGizmo calculates the disparity. |

To see a table listing the nodes or channels each Ocula node requires in its inputs, see Appendix A: Node Dependencies.
Using StereoReviewGizmo

1. Select Ocula > Ocula 4.0 > StereoReviewGizmo to insert an StereoReviewGizmo node either after the O_DisparityGenerator node if you added one, or after the image sequence.
2. Connect a Viewer to the O_StereoReviewGizmo node. Your node tree should now look something like this:

3. Adjust the Parallax controls to re-converge on points of interest. You can either set the Parallax controls manually, or you can click Auto Near/Far to automatically set the parallax range. The output control is automatically updated to far and a difference view is displayed in the Viewer, allowing you to adjust the settings until the point is aligned. This makes it easier to line up the left and right view.
4. You can then use Toggle Left/Right to review the footage at this point with the left and right output.
5. Use Start Playback to play a 10 frame sequence for the current output. This allows you to review the point of interest defined by the Parallax controls, for any temporal issues. This control now toggles to STOP Playback. Click this to restore the timeline.
6. By default, subjectPt is enabled in the StereoReviewGizmo controls. With this enabled, you can drag and drop the subjectPt widget in the Viewer to select a point of interest. To disable subjectPt, select the hide checkbox to the right of the subjectPt controls.
7. Adjust the Parallax settings and toggle between different views using the output controls to examine the differences.
8. Select Start Depth Rack to animate the through depth between the specified near and far settings. See the following Controls section for more information.

StereoReviewGizmo Controls

Output

Use the output control to select the Viewer output. Select one of the following:

| stereo  | The stereo output shows the input re-converged at the depth defined by slider. |
| **left** | The **left** output shows the re-converged left view only. |
| **right** | The **right** output shows the re-converged right view only. |
| **slider** | Select this to show a difference view, re-converged by the **Parallax** controls. Adjust the controls until the image becomes aligned on the point of interest. |
| **near** | Select this to show a difference view, re-converged by the **Parallax** controls. Adjust the controls until the image becomes aligned on the point of interest. |
| **subject** | Select this to show a difference view, re-converged by the **Parallax** controls. Adjust the controls until the image becomes aligned on the point of interest. |
| **far** | Select this to show a difference view, re-converged by the **Parallax** controls. Adjust the controls until the image becomes aligned on the point of interest. |

### Align Input Checkbox

Select the **Align input for colour match review** checkbox to align the input using disparity to review color differences only. You can toggle this option on and off to review the color match if the input has not been aligned, and to detect where there is misalignment in the shot.

### View Slider

Use **View Slider** to change the output to show the difference view, set by the **Parallax – slider** control.

### Toggle Left/Right

Click **Toggle Left/Right** multiple times to toggle between the left and right views. You can use this to review the footage at a particular point of interest, after re-converging using the **Parallax – slider** control.

### Start Playback

Use **Start Playback** to play a 10 frame sequence of the current output. You can use this to review temporal stability. After you click the **Start Playback** control, it toggles to **STOP Playback**; press this
to restore the timeline.

NOTE: You can use Ctrl/Cmd + click on the Viewer timeline to change the playback range.

Start Depth Rack

You can click Start Depth Rack to activate a FrameHold and animate through depth between the defined near and far settings, showing the image difference. You can use this to check for alignment errors and color differences across depth. After you have press the Start Depth Rack control, it toggles to STOP Depth Rack; press this to restore the timeline and disable the FrameHold.

NOTE: You can manually scrub through the depth changes by dragging the Parallax – slider control.

Parallax Controls

Use the slider, near, subject and far controls to zero disparity and re-converge at a point of interest. When you adjust one of these controls, the output is automatically updated to reflect it. For example, when you adjust the near control, the output control is updated to near and a difference view is displayed in the Viewer.

The difference view showing the automatically specified near and far controls.

The difference view with the Parallax controls greatly increased.

Adjust the Parallax controls until the image is aligned at the point of interest. You can then use the Toggle Left/Right control to switch between the input views at this point to review the input footage. You can also use the Start Playback control to review the input at this depth.
Setup

Auto Near/Far

You can use the **Auto Near/Far** control to automatically set the **near** and **far** parallax controls by sampling the input disparity at the current frame. You can optionally key this at different frames in the shot.

subjectPt

By default, **subjectPt** is enabled. To disable it, select the hide checkbox to the right of the **subjectPt** controls. Use this control to set the subject position at the current frame. Adjust the position to sample the input disparity and automatically set the **subject** parallax. You can adjust the position by either entering the x and y coordinates of the point, or by dragging and dropping the **subjectPt** widget in the Viewer.

NOTE: The subject point must be defined in the left view.

Bake Samples

Click **Bake Samples** to automatically sample the subject disparity at the animated sample position. You can use this control when you are importing a track as the sample point in the left view. Use **Bake Samples** to sample the subject disparity and parallax for a specific frame range.

Export Controls

Create Subject Transform

Use this control to export a Transform node to align left and right views, using sampled subject disparity. First enable the **sample subject** checkbox to sample the subject disparity. You can then use **Create Subject Transform** to create a separate transform node that aligns the left and right views to quality check the subject.

You can create a transform for a tracked point by setting the track on the sample point and using **Bake Samples**. The track has to be defined for the left view.
Views

Use the Views dropdown to set how the transform is applied, to line up the subject in the left and right view,
# Appendix A: Node Dependencies

This appendix lists the data each Ocula node requires in its inputs (in addition to a stereo pair of images).

<table>
<thead>
<tr>
<th>Node and Settings</th>
<th>Solve Data</th>
<th>Disparity Map</th>
<th>Occlusion Mask</th>
<th>Vector Generator</th>
<th>Depth Channel</th>
<th>Stereo Camera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>O_Solver</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>O_DispparityGenerator</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Alignment = 0 (the default setting)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Alignment &gt; 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>O_OcclusionDetector</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>O_ColourMatcher</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>O_FocusMatcher</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>O_VeriticalAligner</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table above outlines the required input data or channels for each Ocula node. The presence of a dot indicates that the corresponding data or channel is required for that node.
<table>
<thead>
<tr>
<th>Node and Settings</th>
<th>Required Input Data or Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solve Data</td>
</tr>
<tr>
<td>Local mode</td>
<td></td>
</tr>
<tr>
<td><strong>O_NewView</strong></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
</tr>
<tr>
<td><strong>O_InteraxialShifter</strong></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
</tr>
<tr>
<td><strong>O_VectorGenerator</strong></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
</tr>
<tr>
<td><strong>O_Retimer</strong></td>
<td></td>
</tr>
<tr>
<td>No Motion input</td>
<td></td>
</tr>
<tr>
<td><strong>O_DepthToDisparity</strong></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
</tr>
<tr>
<td><strong>O_DisparityToDepth</strong></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
</tr>
<tr>
<td><strong>O_DisparityViewer</strong></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
</tr>
<tr>
<td><strong>O_MultiSample</strong></td>
<td></td>
</tr>
<tr>
<td>All modes</td>
<td></td>
</tr>
</tbody>
</table>
# Appendix B: External Software

This appendix lists third party libraries used in Ocula, along with their licenses.

<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
<th>License</th>
</tr>
</thead>
</table>
| Boost   | Source code function / template library | Boost Software License - Version 1.0 - August 17th, 2003  
Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the “Software”) to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:  
The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.  
THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. |
<p>| Expat   | XML parser | Copyright © 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper |</p>
<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Copyright © 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.</td>
</tr>
<tr>
<td>FreeType</td>
<td>Font support</td>
<td>Portions of this software are copyright © 2008 The FreeType Project (<a href="http://www.freetype.org">www.freetype.org</a>). All rights reserved.</td>
</tr>
<tr>
<td>FTGL</td>
<td>OpenGL support</td>
<td>FTGL - OpenGL font library Copyright © 2001-2004 Henry Maddocks <a href="mailto:ftgl@opengl.geek.nz">ftgl@opengl.geek.nz</a> Copyright © 2008 Sam Hocevar <a href="mailto:sam@zoy.org">sam@zoy.org</a> Copyright © 2008 Sean Morrison <a href="mailto:learner@brlcad.org">learner@brlcad.org</a> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software,</td>
</tr>
</tbody>
</table>
and to permit persons to whom the Software is furnished to do so, subject to the following conditions

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>VXL</td>
<td>Computer vision</td>
<td>Copyright © 2000-2003 TargetJr Consortium</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GE Corporate Research and Development (GE CRD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Research Circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Niskayuna, NY 12309</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All Rights Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reproduction rights limited as described below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notice and this permission notice appear in all copies of the software and related documentation, (ii) the name TargetJr Consortium (represented by GE CRD), may not be used in any advertising or publicity relating to the software without the specific, prior written permission of GE CRD, and (iii) any modifications are clearly marked and summarized in a change history log.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>THE SOFTWARE IS PROVIDED “AS IS” AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING</td>
</tr>
<tr>
<td>Library</td>
<td>Description</td>
<td>License</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE TARGETJR CONSORTIUM BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR ON ANY THEORY OF LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.</td>
<td></td>
</tr>
</tbody>
</table>