# Contents

**Introduction**
- About this User Guide ........................................ 5
- What’s New? .................................................... 5
- Example Images .................................................. 6
- System Requirements ........................................... 6
- Installation ....................................................... 6
- Setting the NUKE_PATH Environment Variable ............ 8
- Moving the Plug-ins Directory ............................... 14
- Licensing Ocula ............................................... 14
- Other Foundry Products ....................................... 14

**Ocula Plug-ins**
- Solver ............................................................. 17
- DisparityGenerator .............................................. 32
- InteraxialShifter ............................................... 43
- VerticalAligner .................................................. 50
- ColourMatcher ................................................... 58
- NewView .......................................................... 66
- DepthToDisparity ............................................... 76
- DisparityToDepth ............................................... 83
- Correlate .......................................................... 91

**Appendix A**
- Release Notes .................................................. 99
- Ocula 2.0v1 ...................................................... 99
- Ocula 1.0v2 ...................................................... 103
- Ocula 1.0v1 ...................................................... 105

**Appendix B**
- End User License Agreement ................................. 107
INTRODUCTION

Welcome to this User Guide for Ocula 2.0 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 plug-ins 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 will tell you how to install and use the Ocula 2.0 plug-ins and tools. Each plug-in or tool is described in detail in later chapters. Licensing Ocula is covered in the separate Foundry FLEXlm Tools User Guide, which you can download from http://www.thefoundry.co.uk/licensing.

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 2.0 user guide on our web site at http://www.thefoundry.co.uk.

What’s New?

Have a look at the new features and improvements in “Appendix A”.

The Foundry Ocula 2.0
Example Images

Example images are provided for use with all of the plug-ins. You can download these images from our web site at [http://www.thefoundry.co.uk](http://www.thefoundry.co.uk) and try Ocula out on them. From the Ocula product page, click on the Tutorials & Example Images link.

System Requirements

Ocula 2.0 plugs into Nuke 5.1v3 (or higher) running on Windows, Mac OS X, or Linux. It does not work in Nuke 5.1v2, 5.1v1, 5.0, 4.8, or earlier versions.

Installation

Please note that installing Ocula 2.0 will effectively overwrite any previous versions of Ocula 2.0 that have been installed to the default location. However, installing Ocula 2.0 will NOT overwrite any versions of Ocula 1.0.

On Windows

Ocula is distributed as a software download from our web site at [http://www.thefoundry.co.uk/](http://www.thefoundry.co.uk/). To install Ocula on a computer running Windows follow these instructions:

1. Download one of the following files from our web site at www.thefoundry.co.uk:
   - For 32-bit Nuke 5.1:
     Ocula_2.0v1_Nuke_5.1-win-x86-release-32.exe
   - For 64-bit Nuke 5.1:
     Ocula_2.0v1_Nuke_5.1-win-x86-release-64.exe
   - For 32-bit Nuke 5.2:
     Ocula_2.0v1_Nuke_5.2-win-x86-release-32.exe
   - For 64-bit Nuke 5.2:
     Ocula_2.0v1_Nuke_5.2-win-x86-release-64.exe
2. Double-click on the exe file to launch the installer. Follow the on-screen instructions to install the plug-ins.
3. Proceed to “Setting the NUKE_PATH Environment Variable” on page 8.

On Mac
Ocula is distributed as a software download from our web site at http://www.thefoundry.co.uk/. To install Ocula 2.0 on a Mac, follow these instructions:
1. Download one of the following files from our web site at www.thefoundry.co.uk:
   - For 32-bit Nuke 5.1:
     Ocula_2.0v1_Nuke_5.1-mac-x86-release-32.dmg
   - For 32-bit Nuke 5.2:
     Ocula_2.0v1_Nuke_5.2-mac-x86-release-32.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 the plug-ins.
5. Proceed to “Setting the NUKE_PATH Environment Variable” on page 8.

On Linux
Ocula is distributed as a software download from our web site at http://www.thefoundry.co.uk/. To install Ocula 2.0 on a computer running Linux, follow these instructions:
1. Download one of the following files from our web site at www.thefoundry.co.uk:
   - For 32-bit Nuke 5.1:
     Ocula_2.0v1_Nuke_5.1-linux-x86-release-32.tgz
   - For 64-bit Nuke 5.1:
     Ocula_2.0v1_Nuke_5.1-linux-x86-release-64.tgz
   - For 32-bit Nuke 5.2:
     Ocula_2.0v1_Nuke_5.2-linux-x86-release-32.tgz
Setting the NUKE_PATH Environment Variable

For Nuke 5.1 or 5.2 to find the Ocula plug-ins, you need to set the NUKE_PATH environment variable to point to the directory the plug-ins were installed in.

For Nuke on Windows this is one of the following:

- C:\Program Files\Common Files\Nuke\5.1\plugins\Ocula\2.0
- C:\Program Files (x86)\Common Files\Nuke\5.1\plugins\Ocula\2.0
- C:\Program Files\Common Files\Nuke\5.2\plugins\Ocula\2.0
- C:\Program Files (x86)\Common Files\Nuke\5.2\plugins\Ocula\2.0

For 64-bit Nuke 5.2:
Ocula_2.0v1_Nuke_5.2-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 one of the following commands. This will create the 5.1/plugins/Ocula/2.0 or 5.2/plugins/Ocula/2.0 subdirectory (if it doesn’t already exist), and install the plug-ins in that directory.
   - tar xvzf Ocula_2.0v1_Nuke_5.1-linux-x86-release-32.tgz
   - tar xvzf Ocula_2.0v1_Nuke_5.1-linux-x86-release-64.tgz
   - tar xvzf Ocula_2.0v1_Nuke_5.2-linux-x86-release-32.tgz
   - tar xvzf Ocula_2.0v1_Nuke_5.2-linux-x86-release-64.tgz
   Which command you use depends on which file you downloaded from our web site.

4. Proceed to “Setting the NUKE_PATH Environment Variable” on page 8.
Setting the NUKE_PATH Environment Variable

**Note**  You may need to replace C:\ with another drive letter depending on the drive you are using.

For Linux:

- `/usr/local/Nuke/5.1/plugins/Ocula/2.0`
- `/usr/local/Nuke/5.2/plugins/Ocula/2.0`

For Mac OS X:

- `/Library/Application Support/Nuke/5.1/plugins/Ocula/2.0`
- `/Library/Application Support/Nuke/5.2/plugins/Ocula/2.0`

Note that the Ocula plug-ins for Nuke 5.1 are not compatible with the Ocula plug-ins for Nuke 5.2. This means you can have both sets of plug-ins installed, but you should NOT set NUKE_PATH to point to both installation directories. If you want to use Ocula 2.0 on both Nuke 5.1 and 5.2, you need to edit the environment variable every time you change from one version of Nuke to another.

To set the environment variable, do the following:

**On Windows**

1. Right-click on **My Computer** and select **Properties**.
2. Go to the **Advanced** tab.
3. Click the **Environment Variables** button.
   The Environment Variables dialog opens.
4. Do one of the following:
   - If NUKE_PATH is not already set (that is, it does not appear in the lists of environment variables), click the **New** button under either **User variables** or **System variables**. Which you select depends on whether you want to set the variable for the current user or all users. To set environment variables for all users, you need to have administrator privileges.
In the Variable name field, enter NUKE_PATH.

- If NUKE_PATH is already set, select it from the list and click the Edit button.

5. In the Variable value field, add one of the following directory paths:
   - `C:\Program Files\Common Files\Nuke\5.1\plugins\Ocula\2.0`
   - `C:\Program Files (x86)\Common Files\Nuke\plugins\5.1\Ocula\2.0`
   - `C:\Program Files\Common Files\Nuke\5.2\plugins\Ocula\2.0`
   - `C:\Program Files (x86)\Common Files\Nuke\5.2\plugins\Ocula\2.0`

Which path you use depends on whether you are using Nuke 5.1 or 5.2 and whether you are on 32- or 64-bit Windows.

If NUKE_PATH has already been set, you can use a semicolon (:) to add this path after any other paths. For example:

\SharedDisk\Nuke; C:\Program Files (x86)\Common Files\Nuke\5.2\plugins\Ocula\2.0

6. Click OK.


**Note**

When adding or editing user or system variables on Windows Vista, you need to log off and on again before your changes to environment variables take effect.

**Note**

You may need to replace `C:\` with another drive letter depending on the drive you are using.

---

### On Mac OS X

On Mac OS X, there is a special environment file - a .plist or property list file - which is read every time a user logs in. You may need to create the .plist file if it doesn’t already exist in `/Users/username/environment.plist` (where “~” indicates the user’s home directory, and “.” a hidden directory).

To see if the .plist file exists, do the following:
1. Go to `/~/.MacOSX` (where `~` indicates your home directory, and `.` a hidden directory). For example: `john/.MacOSX`. If you cannot see hidden directories in Finder, select Go > Go to Folder. In the dialog that opens, enter `/~/.MacOSX`.

2. Do one of the following:
   - If the environment.plist file doesn’t yet exist, proceed to “Creating an environment.plist File” on page 11.
   - If the environment.plist file already exists, proceed to “Editing an Existing environment.plist File” on page 12.

Creating an environment.plist File:
If the environment.plist file doesn’t already exist, do the following:

1. Open a Terminal window. By default, you should be in your home directory (your own directory in the Users folder). Enter `pwd` (present working directory) to verify this.
2. Enter `ls -al` to see a list of files in that directory. If `.MacOSX` is not in the list, enter `mkdir .MacOSX` to create the directory.
3. Launch TextEdit to create the .plist file.
4. Copy the following into the text document:
   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
   <plist version="1.0">
   <dict>
   <key>NUKE_PATH</key>
   <string>/Library/Application Support/Nuke/5.1/plugins/Ocula/2.0</string>
   </dict>
   </plist>
   ```

   If you want to use Ocula on Nuke 5.2, replace
   `/Library/Application Support/Nuke/5.1/plugins/Ocula/2.0` with
   `/Library/Application Support/Nuke/5.2/plugins/Ocula/2.0`. 
5. Select **Format > Make Plain Text**.

6. Save the file to your home directory with the name `environment.plist`. Make sure a .txt extension is not added to the end of the file name.

7. Quit TextEdit, and launch a Terminal window. Enter `pwd` to make sure you are in your home directory.

8. To move the `environment.plist` file from your home directory into the `.MacOSX` directory, enter:
   
   ```bash
   mv environment.plist .MacOSX
   ```

9. Log out and log in again.


**Editing an Existing environment.plist File**

If the `environment.plist` file already exists, do the following:

1. Open the `environment.plist` file in TextEdit.

2. Do one of the following:
   - If **NUKE_PATH** is not yet set, add the following lines between `<dict>` and `</dict>`:
     ```xml
     <key>NUKE_PATH</key><string>/Library/Application Support/Nuke/5.1/plugins/Ocula/2.0</string>
     ```
     If you want to use Ocula on Nuke 5.2, replace `/Library/Application Support/Nuke/5.1/plugins/Ocula/2.0` with `/Library/Application Support/Nuke/5.2/plugins/Ocula/2.0`.
   - If **NUKE_PATH** is set, add the following path between `<string>` and `</string>`:
     ```xml
     /Library/Application Support/Nuke/5.1/plugins/Ocula/2.0
     ```
     Use a colon (:) to separate this path from other paths. For example:
     ```xml
     <key>NUKE_PATH</key><string>/SharedDisk/Nuke:/Library/Application Support/Nuke/5.1/plugins/Ocula/2.0</string>
     ```
If you want to use Ocula on Nuke 5.2, replace
/Library/Application Support/Nuke/5.1/plugins/Ocula/2.0 with
/Library/Application Support/Nuke/5.2/plugins/Ocula/2.0.

3. Save your changes to the .plist file.
4. Log out and log in again.

On Linux

The procedure for setting an environment variable depends on what your default shell is.

1. To get the name of the shell you are using, launch a shell and enter:
   echo $SHELL

2. Depending on the output of the previous step, do one of the following:
   • If your shell is a csh or tcsh shell, add one of the following commands to the .cshrc or .tcshrc file in your home directory:
     setenv NUKE_PATH /usr/local/Nuke/5.1/plugins/Ocula/2.0
     setenv NUKE_PATH /usr/local/Nuke/5.2/plugins/Ocula/2.0
     Which command you use depends on whether you want to use Ocula on Nuke 5.1 or 5.2.
     If NUKE_PATH has already been set, you can use a colon (:) to add this path after any other paths. For example:
     setenv NUKE_PATH /SharedDisk/Nuke:/usr/local/Nuke/5.1/plugins/Ocula/2.0
   • If your shell is a bash or ksh shell, add one of the following commands to the .bashrc or .kshrc file in your home directory:
     export NUKE_PATH=/usr/local/Nuke/5.1/plugins/Ocula/2.0
     export NUKE_PATH=/usr/local/Nuke/5.2/plugins/Ocula/2.0
     Which command you use depends on whether you want to use Ocula on Nuke 5.1 or 5.2.
If NUKE_PATH has already been set, you can use a colon (:) to add this path after any other paths. For example:

```
export NUKE_PATH=/SharedDisk/Nuke:/usr/local/Nuke/5.1/plugins/Ocula/2.0
```

3. Save your changes.

**Moving the Plug-ins Directory**

You can put the Ocula plug-ins anywhere as long as you set the environment variable NUKE_PATH to point to it.

**Licensing Ocula**

Without a license key, the Ocula plug-ins will fail to run.

The license key is a sequence of numbers and letters, stored in a plain text file, that unlocks Ocula. License keys can be created for a particular computer enabling those plug-ins to run only on that computer. These are called node locked licenses. We also supply floating licenses that will unlock Ocula on any networked computer connected to a machine running the Foundry license server.

Tools to install license keys, manage floating licenses, and diagnose license problems can be downloaded from our web site, http://www.thefoundry.co.uk/licensing.

**Other Foundry Products**

The Foundry is a leading developer of visual effects software for film and video post production. Its products include Nuke, Furnace, Tinder, Tinderbox, and Keylight and run on a variety of compositing platforms including Discreet Flame from Autodesk, Avid DS and Apple’s Shake. For the full list of...
Nuke is an Academy Award® winning compositor. It has been used to create extraordinary images on scores of feature films, including *Australia, The Dark Knight, Quantum of Solace, The Curious Case of Benjamin Button, The Golden Compass, Iron Man, Transformers, King Kong, and Pirates of the Caribbean: At World’s End.*

Furnace is a collection 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.

Tinder and Tinderbox are collections of image processing effects including blurs, distortion effects, background generators, colour tools, wipes, matte tools, painterly effects, lens flares and much more.

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.

Visit The Foundry’s web site at [http://www.thefoundry.co.uk](http://www.thefoundry.co.uk) for further details.
INTRODUCTION
Other Foundry Products
**OCULA PLUG-INS**

**Solver**

**Introduction**

To produce their output, many Ocula plug-ins need *disparity fields*. A disparity field maps the location of a pixel in one view to the location of its corresponding pixel in the other view. To create a disparity field, you need two of the Ocula plug-ins:

1) First, use the O_Solver plug-in to define the geometrical relationship between the two views in the input images (that is, the camera relationship). To define this relationship, O_Solver detects a number of features in one view and locates the corresponding features in the other (Figure 1).

2) After the O_Solver node, add an O_DisparityGenerator

![Figure 1. O_Solver detects features in each view and tries to match them.](image)
node. This node creates the final disparity field.

In addition to O_DisparityGenerator, the O_VerticalAligner plug-in also requires an O_Solver node in its input. O_VerticalAligner uses the feature matches O_Solver generates between the two views to calculate a transform that aligns the views vertically.

This chapter gives you instructions on the use of O_Solver. For more information on O_DisparityGenerator and O_VerticalAligner, see “DisparityGenerator” on page 32 and “VerticalAligner” on page 50.

If you have a CG scene with camera information and z-depth map available, you can also create disparity fields using the O_DepthToDisparity node. For more information, see “DepthToDisparity” on page 76.

**Description**

As stated above, the O_Solver plug-in detects a number of features in each view and tries to match them to define the geometrical relationship between the two views.

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-analyse the camera relationship.

Note that the camera relationship is the same for any images filmed using a particular camera setup. When you have found
an image with features that O_Solver is able to match well, you can re-use the analysis of that image on other images shot with the same camera setup.

You can choose to analyse the camera relationship on every frame of the input clip or interpolate between manually set keyframes. Any frames set as keyframes show up on the Viewer timeline and can be visualised in the Curve Editor.

To get the best possible results, you can identify features to ignore in the analysis. This can be done either by deleting features displayed in a Viewer overlay, or by supplying a mask in the Ignore input.

If you have a pretracked camera that describes the camera setup used to shoot the images, you can also supply this in the Camera input. For more information, see “Inputs” below.

**Tip**  
When shooting bluescreen or greenscreen footage, you may be able to improve O_Solver’s feature matching by adding markers to the shot. If you want to do so, make sure the markers are as unreflective as possible, and stagger them over depth.

**Inputs**

O_Solver has three inputs:

- **Source** - 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.

- **Ignore** - 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.

- **Camera** - A pretracked Nuke stereo camera that describes the camera setup used to shoot the Source image. This
can be a camera you have imported to Nuke from a third-party camera tracking application, such as Boujou, PFTrack, or 3D Equalizer.

Quick Start

Connecting the O_Solver Node
To connect the O_Solver node, 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. 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.

3. Select Ocula > Ocula 2.0 > 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.

5. Once you are happy with the results of O_Solver, do one of the following:
   - select Ocula > Ocula 2.0 > O_DisparityGenerator to insert an O_DisparityGenerator node after O_Solver (Figure 2). This is necessary if you want to use O_InteraxialShifter,
O_ColourMatcher (when set to do block-based matching), O_NewView, or O_DisparityToDepth down the tree.

- select Ocula > Ocula 2.0 > 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_VerticalAligner does not need disparity vectors, so you do not need to use O_DisparityGenerator before this node.
Note that you can use the same O_Solver for both an O_DisparityGenerator and an O_VerticalAligner node. This way, you don’t have to calculate the camera relationship twice.

**Defining the Camera Relationship**

To define the geometrical relationship between the two views, do the following:

1. Connect the O_Solver node as described under “Connecting the O_Solver Node” on page 20.

2. Open the O_Solver controls. From the Views to Use menu or buttons, select which views you want to use for the left and right eye when calculating the camera relationship.

3. If you have a pretracked 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. Assuming the pretracked camera information is accurate, you do not need to do anything further.

   If you don’t have a pretracked camera for the Source shot, proceed to step 4.

4. Set Display to Feature Matches.

   The features and matches used to calculate the camera relationship are shown in a Viewer overlay. Note that the detected
features should be spread around the image. If this is not the case, adjust the Features controls.

Zoom in on some individual feature matches and switch between the two views to see if the matches are accurate. If they are not, delete poor matches by right-clicking on them and selecting delete.

If an entire area of the image is producing poor matches, drag to select multiple matches before clicking delete. You can also press Shift while dragging to select (or deselect) multiple areas. Alternatively, you can specify areas to ignore using a mask in either the Ignore input or the alpha of the Source image. In the O_Solver controls, set Ignore to the component you want to use as the mask.

Note that features generated on reflections often produce bad feature matches. However, you only need to delete such features if O_Solver is producing poor results.

5. If necessary, you can set O_Solver to interpolate between manually set keyframes rather than do the feature matching and analysis on every frame. To do so, set Analysis Type to Interpolate Keyframes. Using the Viewer timeline, scrub to a frame you want to analyse and use as a keyframe. Click Add Key. Repeat as necessary. To visualise the keyframed analysis in the Curve Editor, right-click on the Analysis Key field and
select **Curve editor**. Note, however, that you cannot edit the curve in the Curve Editor.

Interpolating between keyframes can be useful, for example, if you want the disparity to change smoothly across the sequence, or you know there is a zoom or a change in the camera setup on certain frames. Where the camera rig doesn’t change, you can set keyframes far apart.

6. To visualise the calculated camera relationship in a Viewer overlay, set **Display to Epipolar Lines**. Each epipolar line is formed by casting a ray out from a pixel in the adjacent stereo camera. For a perfectly parallel pair of cameras, the lines will all be horizontal, but in practice the lines are likely to be skewed. Use the epipolar lines to visualise how consistent the camera relationship is over time. This can help you select consistent frames in keyframe selection and interpolation.
7. If you are happy with the results, connect the O_Solver node to the **Source** input of either O_DisparityGenerator or O_VerticalAligner.

![Figure 7. With O_DisparityGenerator.](image1)

![Figure 8. With O_VerticalAligner.](image2)

If your input sequence doesn’t contain features that O_Solver is able to match well and you’re not happy with the results, try using O_Solver on another sequence shot with the same camera setup. Then, connect the O_Solver node to the **Solver** input of either O_DisparityGenerator or O_VerticalAligner.

![Figure 9. With O_DisparityGenerator.](image3)

![Figure 10. With O_VerticalAligner.](image4)

**Note**  
When you calculate a camera relationship for a different sequence and attach it to the **Solver** input of O_DisparityGenerator or O_VerticalAligner, the camera relationship will still be accessed at the current time frame. For example, if the **Solver** clip has frames...
1-10 and the **Source** clip has frames 1-100, the camera relationship from frame 10 of the **Solver** clip will be used for frames 90-100. You can resolve this by getting **O_Solver** to just give a single result that is always used. If this is not the case (that is, **O_Solver** has anything other than a single keyframe), the sequences may need to be retimed so that the calculated camera relationship is consistent with the sequence that is being analysed.

### Controls

**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.

The **Analysis** parameters control the way **O_Solver** calculates the camera relationship based on the feature matches it finds.

**Analysis Window** - The size of the temporal window used to calculate the camera relationship. The default value of 1 means the current frame and one frame on either side are used in the analysis. Increasing the value adds more adjacent frames into the analysis, and may result in more consistent results across frames. If a **Camera** input is used, this control has no effect.

**Analysis Type** - Switch between analysing every frame and interpolating between keyframes.

- **Analyse Every Frame** - The feature matching and analysis is done on every frame of the **Source** clip.
- **Interpolate Keyframes** - The feature matching and analysis is only done on manually set keyframes. Other frames are interpolated. This can be useful if you want the disparity to change smoothly across the sequence, or
you know there is a change in the camera setup on certain frames.

**Tip** Use keyframe interpolation to ensure smooth changes in the calculated camera relationship between views. One way to define the keyframes is to set keys at the start and end of the sequence, then add intermediate keys where the camera geometry changes. You can visualise how well the interpolated geometry matches the real images using O_VerticalAligner followed by an Anaglyph node. Set **Alignment Method** in O_VerticalAligner to **Camera Rotation** to interpolate the calculate camera relationship and check whether there is any vertical displacement between the aligned views in the anaglyph view. See page 55 for an example of how to use O_Solver, O_VerticalAligner and Anaglyph.

**Analysis Key** - The analysis keyframes. This field is for display only. To edit the keyframes, use **Add Key** and **Delete Key**.

**Add Key** - Set an analysis key at the current frame.

**Delete Key** - Delete an analysis key at the current frame.

**Delete All** - Delete all analysis keys.

**Interpolation Method** - Set the method used to interpolate the analysis between keyframes. Note that with both methods, any values before the first keyframe or after the last keyframe are clamped so that the value of the nearest keyframe is used.

- **Linear** - Use linear interpolation. This produces sharp changes at keyframes and straight lines between them.
- **Cubic** - Use cubic interpolation. This smoothes the curve.

**Display** - Change the display mode.
- **Nothing** - Only show the Source image.
• **Feature Matches** - Show the features and matches for the camera relationship calculation in a Viewer overlay. An example of what this mode does is shown in Figure 11.

![Figure 11. Showing the features and matches for the camera relationship calculation in a Viewer overlay.](image)

If you are not happy with the results of O_Solver, you can use this mode to see where it has found features and matches, and then adjust the controls accordingly. You can also delete poor matches by right-clicking on them and selecting *delete*.

If you have set **Analysis Type** to **Interpolate Keyframes**, the feature matches are only shown at the keyframes and the surrounding analysis window.

• **Epipolar Lines** – Visualise the camera relationship in a Viewer overlay. The lines should change consistently over
time. An example of what this mode does is shown in Figure 12.

![Image](image.png)

**Figure 12.** Epipolar lines help to visualise the camera relationship.

The **Features** parameters control how `O_Solver` searches for matching features in each view to calculate the relationship between the views.

**Ignore** – 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.

- **None** – Use the entire image area.
- **Source Alpha** – Use the alpha channel of the **Source** clip as an ignore mask.
- **Source Inverted Alpha** – Use the inverted alpha channel of the **Source** clip as an ignore mask.
- **Mask Luminance** – Use the luminance of the **Ignore** input as an ignore mask.
- **Mask Inverted Luminance** – Use the inverted luminance of the **Ignore** input as an ignore mask.
• **Mask Alpha** – Use the alpha channel of the *Ignore* input as an ignore mask.

• **Mask Inverted Alpha** – Use the inverted alpha channel of the *Ignore* input as an ignore mask.

**Number of Features** – Set the number of features to detect in each image.

**Feature Spread** – It is important that the detected features are spread around the entire image. If you set *Display* to *Feature Matches* and see that this is not the case, try increasing this value.

**Feature Separation** – It is important that the detected features do not cluster together. If you set *Display* to *Feature Matches* and see that this is the case, try increasing this value.

**Match Threshold** – Set the minimum similarity score for a pair of features to be considered a good match. You may want to lower this value when the default calculation is not likely to be able to match the corresponding features in the two views (for example, if there is a colour shift between the views).

**Max X Disparity** – Set the maximum distance to search for valid matches, as a fraction of the image width. If the corresponding features in your images are far apart horizontally and you get poor results with the default settings, you may want to increase this value.

**Max Y Disparity** – Set the maximum distance to search for valid matches, as a fraction of the image height. If the corresponding features in your images are far apart
vertically and you get poor results with the default settings, you may want to increase this value. Alternatively, if the vertical displacement between your images is small, try decreasing this value to prevent bad matches.

**Feature Size** - Set the width and height (in pixels) of the square area O_Solver tries to match around each feature. Note that for the feature detection, the image is temporarily converted into the PAL format (720x576) or close, depending on the pixel aspect ratio.

The **Clear Analysis** parameters let you delete feature matches and the calculated camera relationship on either the current frame or all frames that are not keyframes.

**Reset Current Frame** - Clear the feature matches and delete the calculated camera relationship for the current frame. This can be useful, for example, if you delete too many features and want to force the current frame to re-analyse.

**Reset Non-Keyframes** - Clear all feature matches that are not used in the keyframe analysis and delete the calculated camera relationship on all frames that are not keyframes. This can be useful if you want to cut down the size of the script after you have chosen the keyframes and have set Analysis Type to Interpolate Keyframes.

**Example**

See page 38 for an example of how to use O_Solver and O_DisparityGenerator to calculate a disparity field for a stereo image.
**DisparityGenerator**

**Description**

The O_DisparityGenerator is used to create disparity fields for stereo images. A *disparity field* maps the location of a pixel in one view 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.

Most Ocula plug-ins rely on disparity fields to produce their output. If you have more than one Ocula plug-in in your node tree with one or more of the same inputs, they might well require identical disparity field calculations. O_DisparityGenerator is a utility plug-in designed to save processing time by allowing you to create the disparity fields separately, so that the results can then be reused by other Ocula plug-ins.

To be able to produce a disparity field, O_DisparityGenerator usually requires an O_Solver node as one of its inputs. Before doing any processing, O_DisparityGenerator examines the data passed to it by the upstream O_Solver. If it detects that the matches between features are approximately horizontal, O_DisparityGenerator does not require any further data from the O_Solver. This means that when the disparity is in the horizontal direction only, any changes made to the O_Solver node (such as deleting feature matches) are unlikely to affect the generated disparity field.

If you already know that the disparity between views is only horizontal, for example if the images are vertically aligned, you can use O_DisparityGenerator without O_Solver by checking **Horizontal Shift Only** in the O_DisparityGenerator.
controls.

The final disparity vectors are stored in disparity channels, so you might not see any image data appear when you first calculate the disparity field. 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. An example of what a disparity channel might look like is shown on page 42.

In general, once you have generated a disparity field that describes the relation between the views of a particular clip well, it will be suitable for use in most of the Ocula plug-ins. We recommend that you insert a Write node after O_DisparityGenerator to render the original images and the disparity channels as an .exr file. 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 field will be loaded into Nuke together with the sequence and is readily available for the Ocula plug-ins. For information on how to generate a disparity field and render it as an .exr file, see page 34.

Tip  O_DisparityGenerator operates on luminance images. If you, for example, have a bluescreen image where the background and foreground luminance are not that different, you may be able to improve the results by keying out the background before using O_DisparityGenerator.

If you have a CG scene with camera information and z-depth map available, you can also create disparity fields using the O_DepthToDisparity node. For more information, see “DepthToDisparity” on page 76.
Ocula 2.0 The Foundry

DisparityGenerator

Inputs

O_DisparityGenerator has three inputs:

- **Source** - A stereo pair of images. If you’re not using the Solver input and have not checked **Horizontal Shift Only**, the images should be followed by an O_Solver node.

- **Solver** - If the Source sequence doesn’t contain features that O_Solver is able to match well, you can use O_Solver on another sequence shot with the same camera setup. If you do so, connect O_Solver to this input.

- **Fg** - An optional mask that delineates foreground regions from background. The mask ensures that disparities are constrained to remain within each delineated region. Note that masks should exist in both views, and O_DisparityGenerator expects alpha values of either 0 (for background) or 1 (for foreground).

Quick Start

To generate a disparity field for a stereo clip, 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 2.0 > 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). This plug-in calculates the geometrical relationship between the two views in the input clip.
For instructions on how to use O_Solver, see “Solver” on page 17.

4. Select Ocula > Ocula 2.0 > O_DisparityGenerator to insert an O_DisparityGenerator node after O_Solver. In most cases, the O_Solver node should be connected to the Source input of O_DisparityGenerator (Figure 13). However, if you want to calculate the disparity field for one clip by using the O_Solver node of another, connect the O_Solver node to the Solver input of O_DisparityGenerator (Figure 14).

5. Open the O_DisparityGenerator 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 field.

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

O_DisparityGenerator calculates the disparity field and stores it in the disparity channels.

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

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

8. If the calculated disparity field does not produce the results you’re after, use the O_DisparityGenerator controls to adjust the way the disparity field is calculated. The available controls are described below.

You can also use the fg input to separate foreground and background layers to handle severe occlusions.

**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 field. These views will be mapped for the left and right eye.

**Disparity Field Detail** - Adjust this to vary the resolution of the images used to calculate the disparity field. The larger the value, the greater the processing time, but the more detailed the vectors. The default value of 0.5 will generate a vector at every other pixel.
**Smoothness** – Set the smoothness weighting of the disparity calculation. Disparity fields usually have two important qualities: they should accurately match similar pixels in one image to another and they should be smooth rather than noisy. Often, it is necessary to trade one of these qualities off against the other. A high smoothness will miss lots of local detail, but is less likely to provide you with the odd spurious vector. A low smoothness will concentrate on detail matching, even if the resulting field is jagged. The default value of 0.5 should work well for most sequences. However, if your scene is planar or near planar, you can try increasing the value.

**Occlusions** – Choose the algorithm used to create the disparity field.
- **Normal** – In most cases, this option is faster and produces more accurate results than **Severe**.
- **Severe** – If there are large occluded areas in the Source image due to wide camera separation, this option may produce better results than **Normal**.

**Foreground Component** – Select the channel to use to delineate foreground regions from background. This ensures that disparities are constrained to remain within each delineated region.
- **None** – Use the entire image area.
- **Source Alpha** – Use the alpha channel of the Source clip.
- **Source Inverted Alpha** – Use the inverted alpha channel of the Source clip.
- **Mask Luminance** – Use the luminance of the Fg input.
- **Mask Inverted Luminance** – Use the inverted luminance of the Fg input.
- **Mask Alpha** – Use the alpha channel of the Fg input.
• **Mask Inverted Alpha** - Use the inverted alpha channel of the Fg input.

**Horizontal Shift Only** - If the disparity between your stereo views is horizontal only (corresponding pixels lie on the same scanline in both images), you can check this for faster processing. In this case, you do not need to use an O_Solver node before O_DisparityGenerator.

**Example**

In this example, we read in a stereo image, calculate its disparity field, and render the result as an .exr file that contains the left and the right view and the newly created disparity channels. The stereo image used here can be downloaded from our web site. For more information, please see “Example Images” on page 6.

**Step by Step**

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

2. Import graveyard_left.tif. In the Read node controls, locate the file field. Then, replace the word left in the file name with the variable %V. This way, Nuke reads in both the left and the right view using the same Read node.

3. Select the Read node and choose Ocula > Ocula 2.0 > O_Solver from the Toolbar. This inserts an O_Solver node after the stereo image.
4. Select **Ocula > Ocula 2.0 > O_DisparityGenerator** to insert an O_DisparityGenerator node after O_Solver.

![Image](image.png)

**Figure 16. The node tree with O_DisparityGenerator.**

5. Attach a Viewer to the O_DisparityGenerator node, and display one of the disparity channels. O_DisparityGenerator calculates the disparity field.

6. To better be able to evaluate the results, add a Multiply node after O_DisparityGenerator in a new branch of the node tree (hold down Shift and select **Color > Math > Multiply**). In the Multiply controls, set **channels** to **disparity** and **value** to 0.1. Select the Multiply node and press 2 to view its output. You
should now be better able to see boundaries of the objects in the image.

![Figure 17. The node tree with the Multiply node.](image)

7. Select **Image > Write** to insert a Write node between the O_DisparityGenerator and the Viewer. Your node tree should now look like the one in Figure 18.

![Figure 18. The node tree with the Write node.](image)

8. In the Write node controls, select **all** from the **channels** menu to include the disparity channels in the rendered file. Then, select **exr** as the **file type**.
9. In the file field, enter a file name and location for the new .exr file. Then, click the Render button to render the image as usual.

![Image of rendering settings]

Figure 19. Rendering the image with both views and the disparity channels.

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 display the disparity channels by selecting them from the channel set and channel pulldown menus in the upper left corner of the Viewer. When you do, you will probably see something colourful and seemingly
unreadable, much like the image in Figure 20. Don’t worry – that’s what the disparity channels are supposed to look like.

Figure 20. An example of a disparity channel.

You can now use the .exr file you created together with most of the other Ocula plug-ins without having to insert an O_Solver and an O_DisparityGenerator node before them.
InteraxialShifter

Description

The O_InteraxialShifter plug-in lets you change the *interaxial distance* of stereo images; that is, the distance between the two cameras. Using this plug-in, you can generate two new views at specified positions between the original images.

You may want to change interaxial distance during post production for a variety of reasons. For example, it may be necessary because interaxial distance needs to vary with the size of the screen used to view the stereoscopic images. The larger the screen, the further apart the cameras need to be. A film that looks great in the cinema may not work at all when seen on a TV screen. However, at shot time it is not always possible to predict what kind of screens will be used to view the images. In these cases, it is useful to correct their horizontal alignment later on.

Changing interaxial distance affects the perceived depth of the images when they are viewed with 3D glasses. It changes the depth between elements of the image so they appear closer together or further apart. This is illustrated in...
Figure 23 and Figure 24, where the grey rectangles represent elements depicted in a stereo image.

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

To generate the new view, the O_InteraxialShifter plug-in needs disparity vectors that relate the two views. You can use the O_DisparityGenerator plug-in to calculate these vectors. See “DisparityGenerator” on page 32 for how to do this.

Note This plug-in does not pass through any disparity channels fed into it. This is because after warping the input images the original disparity field is no longer valid. If you need disparity channels further down the tree, add another O_DisparityGenerator after O_InteraxialShifter.

Note 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. If you also check Use Ocula if available in the ReConverge controls, extra refinements are done when performing...
the convergence shift. For more information on the ReConverge node, refer to the Nuke User Guide.

Inputs

O_InteraxialShifter has two inputs:

- **Source** - A stereo pair of images. If disparity channels aren’t embedded in the images, you should add an O_Solver and an O_DisparityGenerator node after the image sequence. However, if you check **Horizontal Shift Only** in the O_DisparityGenerator controls, you can skip the O_Solver node.

- **Fg** - An optional mask that delineates foreground regions from background. The mask ensures that disparities are constrained to remain within each delineated region. Note that masks should exist in both views, and O_InteraxialShifter expects alpha values of either 0 (for background) or 1 (for foreground).

Quick Start

To change interaxial distance, do the following:

1. If there are disparity vectors in the data stream from an earlier O_DisparityGenerator node, or if disparity vectors exist in the image sequence itself, these are used when generating the two new views. If disparity vectors don’t yet exist in the script, however, you can use the O_Solver and O_DisparityGenerator plug-ins after your image sequence to calculate the disparity vectors. See “Solver” on page 17 and “DisparityGenerator” on page 32 for how to do this.

2. From the toolbar, select **Ocula > Ocula 2.0 > O_InteraxialShifter** to insert an O_InteraxialShifter node after either the O_DisparityGenerator node (if you added one in the previous step) or the image sequence whose interaxial distance you want to adjust.
3. Under **Views to Use**, select the views you want to use to create new views.

4. Use the **Interpolate Left Position** and **Interpolate 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.

5. Attach a Viewer to the O_InteraxialShifter node.

![Figure 25. A node tree with O_InteraxialShifter. O_Solver and O_DisparityGenerator are not needed in this tree if the necessary disparity channels are included in the Read node.](image)

6. If the results are not what you’re after, adjust the rest of the **O_InteraxialShifter** controls. All the controls are described below.

**Controls**

- **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 will be mapped for the left and right eye.

- **Interpolate Left Position** – Select a position between the views where you want to generate the left view. The position
is expressed as a fraction of the distance between the views.

**Interpolate Right Position** – Select a position between the views where you want to generate the right view. The position is expressed as a fraction of the distance between the views.

**Filtering** – Set the quality of filtering.
  - **Extreme** – Uses a sinc interpolation filter to give a sharper picture but takes a lot longer to render.
  - **Normal** – Uses bilinear interpolation which gives good results and is a lot quicker than extreme.

**Warp Mode** – Select a method to use to create the new views.
  - **Simple** – This is the quickest option, but may produce less than optimal results around image edges and objects that are visible in one view but not the other.
  - **Normal** – This is the standard option, with more optimal treatment of image edges and occlusions.
  - **Occlusions** – This is an advanced option that may improve the results.
  - **Sharp Occlusions** – This option is similar to **Occlusions**, but produces fewer artifacts where the disparity fields are generated from 3D CG sources.

**Foreground Component** – Select the channel to use as to delineate foreground regions from background. The mask ensures that disparities are constrained to remain within each delineated region.
  - **None** – Use the entire image area.
  - **Source Alpha** – Use the alpha channel of the **Source** clip.
• **Source Inverted Alpha** - Use the inverted alpha channel of the **Source** clip.
• **Mask Luminance** - Use the luminance of the **Fg** input.
• **Mask Inverted Luminance** - Use the inverted luminance of the **Fg** input.
• **Mask Alpha** - Use the alpha channel of the **Fg** input.
• **Mask Inverted Alpha** - Use the inverted alpha channel of the **Fg** input.

**Example**

This example shows you how to change the interaxial distance of stereo images. You can download the image used here from our web site. For more information, please see “Example Images” on page 6.

**Step by Step**

1. Start 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 bench.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. Using the Viewer controls, switch between the left and the right view. As you can see, the corresponding elements in these views are rather far apart.

![Figure 26. The original left view. Figure 27. The original right view.](image-url)
4. To reduce the interaxial distance, select Ocula > Ocula 2.0 > O_InteraxialShifter to insert an O_InteraxialShifter node between the image and the Viewer. In the O_InteraxialShifter controls, enter 0.3 as the value for Interpolate Left Position, and 0.7 as the value for Interpolate Right Position.

5. View the result and switch between the two views again. Compare the new views to the original views. You’ll notice that both views have shifted closer together.

Figure 28. The new left view.  Figure 29. The new right view.
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 problem is called keystoning. Unlike converging cameras, parallel cameras do not produce keystoning. This is illustrated in Figure 30 and Figure 31.

Converging cameras cause keystoning because the angle created affects the perspective in the two views. As a result, corresponding points in the two views are vertically misaligned. This is illustrated in Figure 32 and Figure 33.
Whether the vertical misalignment was caused by poorly positioned or converging cameras, it can lead into an unpleasant 3D stereo viewing experience. When a vertically misaligned stereo image is viewed with 3D glasses, the viewer’s mind 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.

The O_VerticalAligner plug-in lets you warp views vertically so that their corresponding features align, while keeping the horizontal position of each pixel the same. This can be done in seven different ways, depending on the alignment mode you choose. An example is shown in Figure 34 and Figure 35.

![Figure 34. Before O_VerticalAligner.](image)
![Figure 35. After O_VerticalAligner.](image)

When using any alignment mode except for Vertical Skew, the transformation in O_VerticalAligner is concatenated with other O_VerticalAligner nodes. This means that if you add a row of O_VerticalAligner nodes to a tree, their functions are combined. Because the image is only resampled once, there is no loss of image quality and processing time is decreased.

The O_VerticalAligner plug-in differs from most of the Ocula plug-ins in that it does not need disparity vectors. Therefore, you do not need to use the O_DisparsityGenerator.
with O_VerticalAligner. Instead, you can connect O_VerticalAligner directly to the output of O_Solver.

Note If disparity channels are fed into this plug-in, they are not passed through to the output. This is because after warping the input images the original disparity field is no longer valid. If you need disparity channels further down the tree, add an O_DisparityGenerator node after O_VerticalAligner. In the Vertical Skew mode, you also need to add an O_Solver node before O_DisparityGenerator if you haven’t checked Horizontal Shift Only in the O_DisparityGenerator controls.

Inputs

O_VerticalAligner has two inputs:

- **Source** - A stereo pair of images. If you’re not using the Solver input, the images should be followed by an O_Solver node.
- **Solver** - If the Source sequence doesn’t contain features that O_Solver is able to match well, you can use O_Solver on another sequence shot with the same camera setup. If you do so, connect O_Solver to this input.

Quick Start

To align a stereo pair of images vertically, do the following:

1. Select Ocula > Ocula 2.0 > O_Solver to insert an O_Solver node after your stereo clip.
2. Select Ocula > Ocula 2.0 > O_VerticalAligner to insert an O_VerticalAligner node after O_Solver. In most cases, the O_Solver node should be connected to the Source input of O_VerticalAligner (Figure 36). However, if you want to adjust the vertical alignment of one clip by using the O_Solver node of
another, connect the O_Solver node to the Solver input of O_VerticalAligner (Figure 37).

3. Connect a Viewer to the O_VerticalAligner node.
4. In the O_VerticalAligner controls, select the views you want to align from the Views to Align menu.
5. To better view the effect of O_VerticalAligner, insert an Anaglyph node between the O_VerticalAligner node and the Viewer.

Figure 36. Getting the camera relationship from the Source clip.
Figure 37. Getting the camera relationship from the Solver clip.
Figure 38. O_VerticalAligner node tree.
6. If you are not happy with the results, adjust the rest of the O_VerticalAligner controls and calculate the shift again.

**Controls**

**Views to Use** - From the views that exist in your project settings, select the two views you want to align. These views will be mapped for the left and right eye.

**Alignment Method** - Select the method you want to use to align the images.

- **Vertical Skew** - Align the features along the y axis using a skew. This does not move the features along the x axis.
- **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.
- **Rotation** - Align the features vertically by rotating the entire image around a point. The centre of the rotation is determined by the algorithm.
- **Scale** - Align the features vertically by scaling the image.
- **Simple Shift** - Align the features vertically by moving the entire image up or down.
- **Scale Rotate** - Align the features vertically by simultaneously scaling and rotating the entire image around a point. The centre of the rotation is determined by the algorithm.
- **Camera Rotation** - Align the features by first performing a 3D rotation of both cameras so that they have exactly the same orientation and a parallel viewing axis, and then reconverging the views to provide the original convergence. This method requires the camera geometry provided by an upstream O_Solver node. For best results, use the O_Solver Camera input to provide the information for the shooting cameras.
Align - Select how to move the views to align the images.
- **Both Views** - Move both views half way.
- **Left to Right** - Move the left view to line up with the right.
- **Right to Left** - Move the right view to line up with the left.

Filtering - Set the quality of filtering.
- **Extreme** - Uses a sinc interpolation filter to give a sharper picture but takes a lot longer to render.
- **Normal** - Uses bilinear interpolation which gives good results and is a lot quicker than extreme.

**Example**

In this example, we correct the vertical alignment of a stereo image. The image used here can be downloaded from our web site. For more information, please see “Example Images” on page 6.

**Step by Step**

1. Fire up Nuke. 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 the steep_hill.exr image. The image includes both the left and the right view.
3. Select **Ocula > Ocula 2.0 > O_Solver** to insert an O_Solver node after the stereo clip.
   Because O_VerticalAligner does not need disparity vectors, you do not need to insert an O_DisparityGenerator node after the O_Solver node.
4. Insert an O_VerticalAligner (**Ocula > Ocula 2.0 > O_VerticalAligner**) and an Anaglyph node (**Stereo > Views > Anaglyph**) after O_Solver.
5. Insert a Viewer after the Anaglyph node. Your node tree should now look like the tree in Figure 39.

![Figure 39. The node tree with O_VerticalAligner.](image)

6. `O_VerticalAligner` aligns the views vertically. To better see the effect, select the `O_VerticalAligner` node and press D repeatedly to disable and enable the node. With the node disabled, the views remain vertically misaligned, as shown in Figure 40.

![Figure 40. The zoomed in image with the `O_VerticalAligner` node disabled.](image)
However, when you enable the O_VerticalAligner node, the views align nicely. This is shown in Figure 41.

Figure 41. The zoomed in image with O_VerticalAligner enabled.
Description

The O_ColourMatcher plug-in lets you match the colours of one view with those of another. It has been specifically designed to deal with the subtle colour differences that are sometimes present between stereo views.

Colour discrepancies between views can be caused by several factors. For example, stereo footage may have been shot with cameras which were differently polarised, or there may have been slight differences between the physical characteristics of the two camera lenses or image sensors. If the colour differences are not corrected for, viewers of the footage may find it difficult to fuse objects in the scene and enjoy the viewing experience.

Correcting for colour differences manually in post production tends to be a time-consuming process and requires considerable skill. Using O_ColourMatcher, however, you can automate the colour grading required. O_ColourMatcher does the work for you by modifying the colour histogram of one view to match that of the other. You may find that the plug-in works better with subtle colour differences than cases of extreme colour discrepancy.

This plug-in differs from most of the Ocula plug-ins in that it does not always need disparity vectors. This is because in the Basic Global Matching and Complex Global Matching modes, O_ColourMatcher does not use any spatial information but tries to match the overall colour distribution of one view to that of the other. In these modes, you do not need to use O_Solver and O_DisparityGenerator with O_ColourMatcher. In the Block-based Matching mode,
however, O_ColourMatcher does need to be passed a disparity field.

**Inputs**

O_ColourMatcher has two inputs:

- **Source** – A stereo pair of images.
  
  If disparity channels aren’t embedded in the images and you are using the Block-based Matching mode, you should add an O_Solver and an O_DisparityGenerator node after the image sequence. However, if you check Horizontal Shift Only in the O_DisparityGenerator controls, you can skip the O_Solver node.

- **Mask** – An optional mask that determines where to take the colour 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 plug-in concentrates on matching the person.

  In the Basic Global Matching and Complex Global Matching modes, O_ColourMatcher calculates the transform on the masked area of the target view but applies it to the whole of the view it’s correcting. In the Block-based Matching mode, it calculates the transform on the masked area and applies it to the that area only.

**Quick Start**

To match the colours of one view with those of another:

1. Select Ocula > Ocula 2.0 > O_ColourMatcher to insert an O_ColourMatcher node after your stereo clip.
2. Connect a Viewer to the O_ColourMatcher node.

![Image of O_ColourMatcher node tree.](image)

Figure 42. O_ColourMatcher node tree.

3. In the O_ColourMatcher controls, select the views you want to match from the Views to Use menu.

4. From the Match menu, select if you want to adjust the colours of the left view to match them with those of the right, or vice versa.

5. If there are areas in the image that you want to ignore when calculating the colour transformation, supply a mask either in the Mask input or the alpha of the Source input. In the O_ColourMatcher controls, set Mask Component to the component you want to use as the mask.

6. If you are not happy with the results, set Mode to either Complex Global Matching or Block-based Matching and calculate the shift again.

Note that in the Block-based Matching mode, the plug-in requires disparity vectors in its input. If there are disparity vectors in the data stream from an earlier O_DisparityGenerator node, or if disparity vectors exist in the image sequence itself, these are used when matching the colours. If disparity vectors don’t yet exist in the script, however, you can use the O_Solver and O_DisparityGenerator plug-ins to calculate the disparity vectors. Select Ocula > Ocula 2.0 > O_Solver and Ocula > Ocula
2.0 > O DisparityGenerator to insert these nodes between the stereo clip and O_ColourMatcher.

**Controls**

- **Views to Use** - From the views that exist in your project settings, select the two views whose colours you want to match. These views will be mapped for the left and right eye.

- **Match** - Select how to match the colours.
  - **Left to Right** - Adjust the colours of the left view to match with those of the right.
  - **Right to Left** - Adjust the colours of the right view to match with those of the left.

- **Mode** - Use an alternative algorithm for the colour matching. You may want to change this setting if the default algorithm does not produce the results you are after.
  - **Basic Global Matching** - This mode takes the colour histogram of one entire view and modifies that to match the histogram of the other view.
  - **Complex Global Matching** - This mode is like Basic Global Matching but uses an alternative algorithm. Complex matching may work better than the basic matching when the colour difference between the two views is very subtle.
  - **Block-based Matching** - This mode first divides the two images into square blocks according to the Block Size control. Then, it matches the colour histograms between corresponding blocks in the two views. This can be useful if there are local colour differences between the views, such as highlights that are brighter in one view than the other. Note that this mode requires that there is a disparity field in the input data stream. If there isn’t, you
can create one using the O_Solver (see page 17) and O_DisparityGenerator (see page 32) plug-ins.

**Block Size** - This control is only available in the **Block-based Matching** mode. It defines the width and height (in pixels) of the square blocks that the images are divided into when calculating the colour match.

**Mask Component** - Select the channel to use as a mask when calculating the colour transformation.
- **None** - Use the entire image area.
- **Source Alpha** - Use the alpha channel of the **Source** clip as a mask.
- **Source Inverted Alpha** - Use the inverted alpha channel of the **Source** clip as a mask.
- **Mask Luminance** - Use the luminance of the **Mask** input as a mask.
- **Mask Inverted Luminance** - Use the inverted luminance of the **Mask** input as a mask.
- **Mask Alpha** - Use the alpha channel of the **Mask** input as a mask.
- **Mask Inverted Alpha** - Use the inverted alpha channel of the **Mask** input as a mask.

**Example**

In this example, we have a colour discrepancy between our stereo views. To fix this, we match the colours of the right view with those of the left. The stereo image used in the example can be downloaded from our web site. For more information, please see "Example Images" on page 6.
Step by Step

1. Start 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.exr. This image already includes both the left and the right view. Because O_ColourMatcher does not need disparity vectors in the Basic Global Matching mode (which is the default), you do not need to insert O_Solver and O_DisparityGenerator nodes into your script.

3. Attach a Viewer to the image. Using the Viewer controls, switch between the left and the right view. As you can see, there is a colour shift between the views. The colours of the right view seem blue and cold compared to the warmer colours of the left view. To achieve the sunny look of the left view, we are going to match the colours of the right view with those of the left.

![Figure 43. The original left view.](image1.png)  ![Figure 44. The original right view.](image2.png)
4. Select **Ocula > Ocula 2.0 > O_ColourMatcher** to insert an O_ColourMatcher node between the image and the Viewer.

![Figure 45. The node tree with O_ColourMatcher.](image)

5. In the O_ColourMatcher controls, select **Right to Left** from the **Match** menu.

6. View the result and switch between the two views again. Compare the new right view to the original right view by displaying the right view in the Viewer, selecting the O_ColourMatcher node, and pressing D a couple of times to disable and enable the node. You’ll notice that the colours of the right view now match those of the left.

![Figure 46. The left view.](image)  ![Figure 47. The colour corrected right view.](image)

7. If you’re not happy with the results, try one of the other modes in the ColourMatcher node’s **Mode** menu.
Note that in the **Block-based Matching** mode, `O_ColourMatcher` needs a disparity field in the input data stream. To create one, add `O_Solver` and `O_DisparityGenerator` nodes (see page 17 and page 32) between the `Read` node and `O_ColourMatcher`. 
Using the O_NewView plug-in, you can create a single view from a stereo pair of images. You can create this new view at any position between the original views. The new view replaces both of the existing views.

You can choose to construct the new view using one or both of the original views. Using just one view can be useful if you want to manipulate it with a gizmo, a plug-in, or a graphics editor, for example, and copy your changes into the other view. You can make your changes to one view, and use the O_NewView plug-in to generate the other view with these changes reproduced in the correct position.

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

To generate the new view, the O_NewView plug-in needs disparity vectors that relate the two views. You can use the O_Solver and O_DisparityGenerator plug-ins to calculate these vectors. See "Solver" on page 17 and "DisparityGenerator" on page 32 for how to do this.

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. Which you choose depends on which you think will produce better disparity vectors. The former method may be preferable, if
you are correcting an unwanted colour shift between views, for example. The latter method usually works best if your changes in one view produce an occlusion in the images, for example, when using a texture replacement plug-in or painting something on one view in another application.

Note that to reproduce changes you’ve made using Nuke’s Paint node, Bezier node, or any node or gizmo that has controls for x and y coordinates, it is easier to use the O_Correlate plug-in described on page 97.

Inputs

O_NewView has two inputs:

- **Source** – A stereo pair of images. If disparity channels aren’t embedded in the images, you should add an O_Solver and an O_DisparityGenerator node after the image sequence. However, if you check **Horizontal Shift Only** in the O_DisparityGenerator controls, you can skip the O_Solver node.

- **Fg** – An optional mask that delineates foreground regions from background. The mask ensures that disparities are constrained to remain within each delineated region. Note that masks should exist in both views, and O_NewView expects alpha values of either 0 (for background) or 1 (for foreground).

Quick Start

To create a new view:

1. If there are disparity vectors in the data stream from an earlier O_DisparityGenerator node, or if disparity vectors exist in the image sequence itself, these are used when generating the new view. If disparity vectors don’t yet exist in the script, however, you can use a combination of the O_Solver and O_DisparityGenerator plug-ins to calculate the disparity vec-
tors. Select Ocula > Ocula 2.0 > O_Solver and Ocula > Ocula 2.0 > O_DisparityGenerator to insert these in an appropriate place in your script.

2. Select Ocula > Ocula 2.0 > O_NewView to insert an O_NewView node after either the O_DisparityGenerator node or the stereo image sequence.

3. Using the Views to Use controls, select the views you want to map for the left and right eye.

4. From the Inputs pulldown menu, select which input(s) you want to use to create the new view:
   - Left – Only use the image mapped for the left eye to create the new view.
   - Right – Only use the image mapped for the right eye to create the new view.
   - Both – Use both images to create the new view.

5. Adjust the Interpolate Position slider to define where to build the new view. The values are expressed as a fraction of the distance between the two views.

6. Attach a Viewer to the O_NewView node.

---

Figure 48. A node tree with O_NewView. The O_Solver and O_DisparityGenerator nodes are not needed in this tree if the necessary disparity channels are included in the Read node.
7. If you are not happy with the results, try adjusting the rest of the O_NewView controls. They are described below.

**Controls**

**Views to Use** - From the views that exist in your project settings, select the two views you want to use to create the new view. These views will be mapped for the left and right eye.

**Inputs** - Select which inputs to use to generate the new view.
- **Left** - Only use the input mapped for the left eye to create the new view.
- **Right** - Only use the input mapped for the right eye to create the new view.
- **Both** - Use both inputs to create the new view.

**Interpolate Position** - Select the position between the existing views where you want to generate the new view. The position is expressed as a fraction of the distance between the views.

**Filtering** - Set the quality of filtering.
- **Extreme** - Uses a sinc interpolation filter to give a sharper picture but takes a lot longer to render.
- **Normal** - Uses bilinear interpolation which gives good results and is a lot quicker than extreme.

**Warp Mode** - Select a method to use to generate the new view.
- **Simple** - This is the quickest option, but may produce less than optimal results around image edges and objects that are visible in one view but not the other.
• **Normal** - This is the standard option, with more optimal treatment of image edges and occlusions.

• **Occlusions** - This is an advanced option that may improve the results.

• **Sharp Occlusions** - This option is similar to Occlusions, but produces fewer artifacts where the disparity fields are generated from 3D CG sources.

**Foreground Component** - Select the channel to use to delineate foreground regions from background. The mask ensures that disparities are constrained to remain within each delineated region.

• **None** - Use the entire image area.

• **Source Alpha** - Use the alpha channel of the **Source** clip.

• **Source Inverted Alpha** - Use the inverted alpha channel of the **Source** clip.

• **Mask Luminance** - Use the luminance of the **Fg** input.

• **Mask Inverted Luminance** - Use the inverted luminance of the **Fg** input.

• **Mask Alpha** - Use the alpha channel of the **Fg** input.

• **Mask Inverted Alpha** - Use the inverted alpha channel of the **Fg** input.

**Example**

In this example, we have a stereo image of a cathedral. In the left view, one of the cathedral windows has been removed using F_BlockTexture - a texture replacement tool included in The Foundry's Furnace plug-ins. Our aim is to reproduce this change in the right view using the **NewView** plug-in. We construct a new right view from the left view, with the changes in the correct position.
The stereo image used in the example can be downloaded from our web site. For more information, please see “Example Images” on page 6.

The necessary disparity channels have been embedded in the download image, so you don’t need to use the O_Solver and O_DisparityGenerator plug-ins in this example. However, you should note that the disparity channels were calculated using the original left and right views (with the window still in place in both views). This is because removing the window from one view but not the other produces an occlusion, and the O_NewView plug-in works better when there are no occlusions.

**Step by Step**

1. Start 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. Notice how in the left view, the cathedral has one window less than in the right view. This is highlighted in Figure 49 and Figure 50.

   ![Figure 49. The left view.](image1)
   ![Figure 50. The right view.](image2)

   We want to reproduce this change made to the left view in the right view using the O_NewView plug-in.

3. Select **Ocula > Ocula 2.0 > O_NewView** to insert an O_NewView node between the stereo image and the Viewer. In the
O_NewView controls, select Left from the Inputs menu to generate the new view from the left view. Enter 1 as the Interpolate Position to create the new view in the same position as the original right view.

As you can see from Figure 51, the window disappears, but the O_NewView plug-in creates some unwanted changes around the edges of the new view.

Figure 51. The new view generated with O_NewView.

To prevent this, we can take the area around the window from the new view, and composite that over the original right view.

4. Select the O_NewView node and Draw > Bezier from the Toolbar. This inserts a Bezier node after O_NewView.

5. In the Bezier controls, change the output to alpha, and premult to rgba. Attach the Viewer to cathedral1.exr, and view the right view.

6. With the Bezier controls open, Ctrl+Alt+click (Mac users Cmd+Alt+click) on the Viewer to draw a shape around the window that was removed from the left view. Then, Ctrl/ Cmd+drag each point on the shape outwards to blur the edges.
of the Bezier shape. Figure 52 shows a quickly drawn example of what your Bezier shape might now look like.

![Figure 52. Drawing a Bezier around the window.](image)

7. Attach the Viewer to the Bezier node, and zoom out. You should only see the area you defined with the Bezier node. This is what we are going to composite over the original right view.

8. To extract the original right view from cathedral1.exr, click on an empty space in the Node Graph and select Views > OneView. Connect the OneView node into cathedral1.exr.

9. In the OneView controls, select right from the view menu. If you now view the output of OneView, you should see the original right view with the window still in place.

10. To composite the output from the Bezier node on top of the original right view, click on an empty spot in the Node Graph and select Merge > Merge. Connect the A input of the resulting over node into the Bezier and the B input into the OneView.
node. Attach a Viewer to the over node. Your node tree should now look like the one in Figure 53.

![Figure 53. The node tree that creates the final right view.](image)

View the output of the over node. This is your final right view. Notice that the covered window has been copied from the left view into the correct position in the right view, but the rest of the right view hasn’t changed.

![Figure 54. The final right view.](image)

Now, we only need to combine the new right view with the original left view from cathedral1.exr.
11. Click on an empty spot in the Node Graph and select **Views > JoinViews**. Connect the **left** input from the JoinViews node into cathedral1.exr, and the **right** input into the over node. Attach a Viewer to the JoinViews node. Your node tree should look like the one in Figure 55.

![Figure 55. The final node tree.](image)

12. Using the Viewer controls, switch between the left and the right views. The changes made to the left view have been copied into the right view, in the correct position. This is shown in Figure 56 and Figure 57.

![Figure 56. The left view.](image)  ![Figure 57. The new right view.](image)
DepthToDisparity

Description
Many Ocula plug-ins rely on disparity fields to produce their output. Usually, disparity fields are created using a combination of the O_Solver (see page 17) and O_DisparityGenerator (see page 32) nodes.

However, if you have a CG scene with camera information and z-depth map available, you can also use the O_DepthToDisparity plug-in to generate the disparity field. 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.

Generating a Disparity Field
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 plug-in, the final disparity vectors are stored in disparity channels, so you might not see any image data appear when you first calculate the disparity field. 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. An example of what a disparity channel might look like is shown on page 42.

Once you have generated a disparity field that describes the relation between the views of a particular clip well, it will be...
suitable for use in most of the Ocula plug-ins. We recommend that you insert a Write node after O_DepthToDisparity to render the original images and the disparity channels as an .exr file. 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 field will be loaded into Nuke together with the sequence and is readily available for the Ocula plug-ins. For information on how to generate a disparity field using O_DepthToDisparity and render it as an .exr file, see page 77.

Inputs

O_DepthToDisparity has got two inputs:

- **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. 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 Shuf-ble 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.

- **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.

Quick Start

To generate a disparity field for a stereo clip, 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 2.0 > 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 field.

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

O_DepthToDisparity calculates the disparity field 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 pulldown menu. Choose exr as the file type. Render the clip.

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

![Figure 58. Rendering the output to combine the clip and the disparity channels for future use.](image)

**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 field. These views will be mapped for the left and right eye.

**Example**

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

**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 2.0 > 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 like the one in Figure 59.

![Figure 59. The Camera node should be connected to the Camera input of O_DepthToDisparity.](image)

6. Use the channel dropdown menus in the top left corner of the Viewer to display one of the disparity channels. O_DepthToDisparity calculates the disparity field and stores it in the disparity channels.

   Switch back to viewing the **rgba** channels.

7. To evaluate the quality of the disparity field, select **Ocula > Ocula 2.0 > O_NewView** to insert an O_NewView node between O_DepthToDisparity and the Viewer. By default, this node uses the generated disparity field to create a new view half way
between the left and the right view. If the disparity field is not accurate, O_NewView will produce poor results.

Figure 60. Using O_NewView to test the generated disparity field.

8. In the O_NewView controls, set Warp Mode to Sharp Occlusions. This gives the best results when the disparity fields are generated from CG depth (see Figure 61 and Figure 62).

Figure 61. Warp Mode set to Normal.  Figure 62. Warp Mode set to Sharp Occlusions.

As you can see, O_NewView has used the disparity field from O_DepthToDisparity to produce a reasonably accurate new view. However, there seems to be a thin “halo” around some object outlines. This is a result of anti-aliasing on object boundaries. Because anti-aliasing mixes the colours of the background and foreground, O_NewView doesn’t know which is
the correct colour to use in these regions, and the results may need further editing.

It is worth noting, however, that the results achieved with O_DepthToDisparity are usually better than those achieved with O_Solver and O_DisparityGenerator. The difference is particularly clear when there are several occluded areas in the images due to wide camera separation.
DisparityToDepth

Description

The O_DisparityToDepth plug-in produces a z-depth map for each view of a stereo clip based on the clip’s disparity field 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. You may need a z-depth map, for example, if you want to introduce fog and depth-of-field effects into a shot. In Nuke, the ZBlur node (Filter > ZBlur) requires a depth map in its input.

O_DisparityToDepth stores the final z-depth map in the depth channels, so you might not see any image data appear when you first calculate the depth map. To see the output inside Nuke, select the depth channels from the channel controls in the top left corner of the Viewer. An example of what a depth channel might look like is shown in Figure 63.

Figure 63. A depth map.
Inputs

O_DisparityToDepth has two inputs:

- **Source** - This is a stereo pair of images. There should be a disparity field in the disparity channels. For information on how to embed a disparity field in an image this way, see “DisparityGenerator” on page 32.

- **Camera** - A pretracked Nuke stereo camera that describes the camera setup used to shoot the Source images. This can be a camera you have imported to Nuke from a third-party camera tracking application, such as Boujou, PFTrack, or 3D Equalizer.

Quick Start

To generate a z-depth map for a stereo clip, do the following:

1. If there are disparity vectors in the data stream from an earlier O_DisparityGenerator node, or if disparity vectors exist in the image sequence itself, these are used when generating the z-depth map. If disparity vectors don’t yet exist in the script, however, you can use the O_Solver and O_DisparityGenerator plug-ins after your image sequence to calculate the disparity vectors. See “Solver” on page 17 and “DisparityGenerator” on page 11 for how to do this.

2. Select **Ocula > Ocula 2.0 > O_DisparityToDepth** to insert an O_DisparityToDepth node after either the O_DisparityGenerator node (if you added one in the previous step) or 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 or buttons, select which views you want to use for the left and right eye when creating the z-depth map.

5. Attach a Viewer to the O_DisparityToDepth node, and display one of the depth channels in the Viewer.
6. Select **Image > Write** to insert a Write node after **O_DisparityToDepth**. In the Write node controls, select **all** from the **channels** pulldown menu. Choose **exr** as the **file type**. Render the clip.

The newly created depth channels will be saved in the channels of your stereo clip. When you need to manipulate the same clip again later, the z-depth maps will be loaded into Nuke together with the clip.

![Figure 64. Rendering the output to combine the clip and the depth channels for future use.](image)

**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 will be mapped for the left and right eye.

**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. Start 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 SteepHill.exr and attach a Viewer to the image.
   The image already includes both the left and the right view as well as the necessary disparity channels.

3. Select Ocula > Ocula 2.0 > O_DisparityToDepth to insert an O_DisparityToDepth node after the Read node. Make sure the Read node is connected to the Disparity input of O_DisparityToDepth.

4. Connect the Camera node to the Camera input of O_DisparityToDepth. Your node tree should now look like the one in Figure 65.

Figure 65. The node tree with O_DisparityToDepth.
5. Use the channel controls in the top left corner of the Viewer to display one of the depth channels (Figure 66).

![Figure 66. The depth channel for the left view.]

6. As you can see, the depth channel looks mostly white. To better be able to evaluate the results, hold down Shift and select Color > Math > Multiply to add a Multiply node in a new branch after O_DisparityToDepth. In the Multiply controls, set channels to depth and value to 0.7.

Similarly, add a Gamma node (Color > Math > Gamma) after the Multiply node. In the Gamma controls, set channels to depth and value to 0.43.
Finally, select the Gamma node and press 2 to view its output.

Figure 67. Using the Multiply and Gamma nodes to evaluate the depth map.

Make sure the Viewer’s channel controls are set to display one of the depth channels. It should look a lot like Figure 68 and allow you to better see the boundaries of the objects in the image.

Figure 68. The left depth channel after the Multiply and Gamma nodes.
7. To test the accuracy of the depth channels in practice, select Filter > ZBlur to add a ZBlur node between O_DisparityToDepth and the Viewer. This node blurs the image according to the depth channels we’ve just created.

8. In the ZBlur controls, set channels to rgba, math to far=0, and focus plane (C) to 0.16.

9. View the output. To better see the effect of the ZBlur node, select it in the Node Graph and press D repeatedly to disable...
and enable it. As you can see, the areas further back in the scene receive more blur than the areas close to the camera.

Figure 70. The output of ZBlur.

10. If you are happy with the depth channels, you can save them in the channels of the input clip for later use. Select Image > Write to insert a Write node between O_DisparityToDepth and ZBlur. In the Write node controls, select all from the channels pulldown 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.
Correlate

Description

O_Correlate is a collection of tools to assist you when rotoscoping, creating paint effects, or doing other operations dependent on image locality. Unlike the Ocula plug-ins described elsewhere in this manual, O_Correlate cannot be found from the Ocula menu. It provides extensions directly to Nuke’s existing Correlate function.

In Nuke, the Correlate function appears on Paint nodes, Bezier nodes, and any nodes or gizmos that have controls for x and y coordinates. It automatically reproduces changes made to one view in another. If you use this function, the changes are reproduced by adding the disparity vectors in the disparity field to the current values. With Ocula installed, however, you can choose to use Ocula instead of Nuke to reproduce the changes. In this case, extra image processing and refinements are done to produce more accurate results. This can be important when working on live action footage rather than CG images.

Note that you can only use O_Correlate to reproduce changes made with the above mentioned nodes or gizmos. To reproduce any other changes (for example, changes made using other applications or plug-ins), you should use the O_NewView plug-in described on page 66.

For O_Correlate to work, you need disparity vectors that relate the two views. You can use the O_Solver and O_DisparityGenerator plug-ins to generate these vectors. See “Solver” on page 17 and “DisparityGenerator” on page 32 for how to do this.
Quick Start

Paint Effects
To create a paint stroke on one view and have it automatically generated for the other using Ocula, do the following:

1. If there are disparity vectors in the data stream from an earlier O_DisparityGenerator node, or if disparity vectors exist in the image sequence itself, these are used when calculating where a paint stroke in one view should appear in the other. If disparity vectors don’t yet exist in the script, however, you can use the O_Solver and O_DisparityGenerator plug-ins after your image sequence to calculate the disparity vectors.

2. Insert a Paint node after either the O_DisparityGenerator node (if you added one in the previous step) or the image sequence you want to paint on.

3. In the Paint node controls, select all from the view pulldown menu. Display the view you want to paint on in the Viewer.

4. With the Paint controls open, draw a paint stroke in the Viewer.

5. Select the paint stroke in the Paint node controls.

6. Click Correlate [view] from [view] with Ocula, for example Correlate right from left with Ocula.
This creates the corresponding paint stroke for the other view using Ocula rather than Nuke’s built-in Correlate function.

Because Ocula is used, extra refinements are made to produce more accurate results. After simply adding the disparity vectors in the disparity field to the current values, Ocula refines the results by taking a block of pixels around the correlated features and searching for that block in the surrounding region. When it finds a match, it then adjusts the results of the correlation accordingly. If necessary, you can define the size of this block (in pixels) using the blocksize parameter and the size of the search region using searchsize.

7. In the Viewer, switch between the two views to compare the original and the correlated paint strokes.

8. If you want to adjust the paint stroke further, you need to adjust both views independently. Adjustments you make to one view are not automatically generated for the other.

**Rotoscoping**

To create a roto shape on one view and have it automatically generated for the other using Ocula, do the following:
1. If there are disparity vectors in the data stream from an earlier O_DisparityGenerator node, or if disparity vectors exist in the image sequence itself, these are used when calculating where a roto shape in one view should appear in the other. If disparity vectors don’t yet exist in the script, however, you can use the O_Solver and O_DisparityGenerator plug-ins after your image sequence to calculate the disparity vectors.

2. Insert a Bezier node after either the O_DisparityGenerator node (if you added one in the previous step) or the image sequence you are manipulating.

3. With the Bezier controls open, create a roto shape for one view.

4. Go to the Bezier controls. From the View menu next to the shape control, select Correlate [view] from [view] with Ocula, for example Correlate left from right with Ocula. This adds a corresponding Bezier shape to the other view using Ocula rather than Nuke’s built-in Correlate function.

Because Ocula is used, extra refinements are made to produce more accurate results. After simply adding the disparity vectors in the disparity field to the current values, Ocula refines the results by taking a block of pixels around the correlated features and searching for that block in the surrounding region.
When it finds a match, it then adjusts the results of the correlation accordingly. If necessary, you can define the size of this block (in pixels) using the blocksize parameter and the size of the search region using searchsize.

5. If you want to adjust the Bezier shape further, you need to adjust both views independently. Adjustments you make to one view are not automatically generated for the other.

**Other Operations**

Whenever there are values in any x or y control in Nuke for one view, you can use Ocula to automatically generate the corresponding values for the other view. This is true for both nodes and gizmos.

For example, you can use a Tracker node to track something in one view, and then have the track’s x and y position generated for the other view using Ocula. Do the following:

1. If there are disparity vectors in the data stream from an earlier O_DisparityGenerator node, or if disparity vectors exist in the image sequence itself, these are used when calculating where the point indicated by the x and y values in one view can be found in the other. If disparity vectors don’t yet exist in the script, however, you can use the O_Solver and O_DisparityGenerator plug-ins after your image sequence to calculate the disparity vectors.

2. Insert a Tracker node after either the O_DisparityGenerator node (if you added one in the previous step) or the image sequence you are manipulating.

3. Attach a Viewer to the Tracker node, and track a feature in one view.

4. Open the Tracker controls. From the View menu next to the track 1 x and y controls, select Correlate [view] from [view] with Ocula, for example Correlate left from right with Ocula. This generates the corresponding track for the other view using Ocula rather than Nuke’s built-in Correlate function. Because
Ocula is used, extra refinements are made to produce more accurate results.

![Figure 73. Generating the corresponding x and y values for the other view.](image)

**Example**

In this example, we draw a roto shape on one view and have it automatically generated for the other using Ocula. The stereo image used in the example can be downloaded from our web site. For more information, please see “Example Images” on page 6.

**Step by Step**

1. Fire up Nuke. 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 cathedral2.exr. This image already includes both the left and the right view (as you can see if you switch between the views in the Viewer) and the necessary disparity channels.
3. Insert a Bezier node (Draw > Bezier) after the image.
4. Attach a Viewer to the Bezier node and display the left view.
5. Uncheck the red, green, and blue channels in the Bezier controls. With the Bezier node controls open, Ctrl+Alt+click (Mac users Cmd+Alt+click) on the Viewer to draw a roto shape over the image, like in Figure 74.

![Figure 74. A roto shape on the left view.](image1.jpg)

6. Now, display the right view in the Viewer. As you can see, the Bezier shape is not correctly positioned in this view. This is illustrated in Figure 75.

![Figure 75. The same roto shape on the right view.](image2.jpg)
7. In the Bezier controls, click the view button next to the shape control and select Correlate right from left with Ocula. Ocula's O_Correlate component generates a corresponding, correctly positioned roto shape for the right view. Because Ocula is used instead of the built-in Correlate function in Nuke, extra refinements are made to produce more accurate results.

Figure 76. The correctly positioned roto shape on the right view.
APPENDIX A

Release Notes

This section describes the requirements, new features, improvements, fixed bugs, and known bugs and workarounds for each release of Ocula.

Ocula 2.0v1

This is a major new release of Ocula with many new features, improvements, and bug fixes.

Release Date
8 October 2009

Requirements

- Nuke 5.1v3 or higher on
  - Windows XP SP2, XP64
  - Mac OS X 10.5 “Leopard” (32-bit and x86 only)
  - Linux CentOS 4.5 (32- and 64-bit)
- Foundry FLEXlm Tools (FFT 5.0v1 or later) for floating licenses.

New Features

Ocula 2.0 includes three new plug-ins:

- **O_Solver** – Some of the functionality from O_DisparityGenerator has been separated out into this plug-in, to allow a more flexible workflow. O_Solver determines the geometrical relationship between a stereo pair of views by detecting feature matches. If you have more than one sequence that were filmed with the same camera rig, it is only necessary to do this calculation on
one of them; the same O_Solver can then be reused for the other sequences. It also offers the following advantages over the old DisparityGenerator:

- An **Ignore** input so you can tell it which regions to ignore when detecting features.
- The ability to calculate the camera relationship over a temporal window, for greater robustness.
- The ability to calculate the camera relationship at intervals and interpolate smoothly between them for better temporal stability.
- A **Camera** input, allowing you to use pre-tracked cameras.
- Interactive editing of feature matches in the viewport.
- The features and camera relationship are stored as metadata, so they can be saved and reused further downstream.

- **O_DepthToDisparity** – a new plug-in to generate a disparity field from a stereo pair of depth maps plus a stereo camera set-up. This is intended for use with CG scenes.
- **O_DisparityToDepth** – a new plug-in to generate depth maps from a disparity field, given the stereo camera set-up.

There are also improvements to some of the existing plug-ins:

- **O_DisparityGenerator**:
  - New, improved disparity generation algorithm.
  - A **Solver** input to allow the camera solve from another sequence shot with the same rig to be reused.
• A foreground (Fg) input to delineate foreground regions from background. This helps to ensure that disparities are constrained to remain within each delineated region.

• **O_InteraxialShifter:**
  • A foreground (Fg) input to delineate foreground regions from background. This helps to ensure that disparities are constrained to remain within each delineated region.

• **O_NewView:**
  • A foreground (Fg) input to delineate foreground regions from background. This helps to ensure that disparities are constrained to remain within each delineated region.

• **O_VerticalAligner:**
  • A Solver input to allow feature matches to be reused.
  • Support for transform concatenation for multiple O_VerticalAligner nodes (except when Alignment Method is set to Vertical Skew).
  • Four new alignment methods: Scale, Simple Shift, Scale Rotate, and Camera Rotation.

• **O_ColourMatcher** (formerly O_ColourMatch):
  • A Mask input to allow you to specify a region of interest for the colour transform.
  • A new, block-based matching mode for dealing with local colour differences.

Note that O_InterocularShifter has been renamed to O_InteraxialShifter to remove ambiguity.
**Improvements**

Ocula 2.0 has been redesigned to allow a more flexible workflow. For details of the improvements to individual plug-ins, see the "New Features" section above. It also features a completely new disparity generation algorithm for greater accuracy and speed.

**Bug Fixes**

BUG ID 7387 - O_ColourMatch: super black material (negative) was wrapped / clipped back from 1.0.

**Known Bugs and Workarounds**

  Later releases of Nuke will pick up the Ocula plug-ins automatically. In the meantime, you will need to set your NUKE_PATH environment variable to (replace 5.x with 5.1 or 5.2 according to the version of Nuke you’re using):
  - On Windows: C:\Program Files\Common Files\Nuke\5.x\plugins\Ocula\2.0
  - On Mac OS X: /Library/Application Support/Nuke/5.x/plugins/Ocula/2.0
  - On Linux: /usr/local/Nuke/5.x/plugins/Ocula/2.0

- **BUG ID 8222 (Ocula) and BUG ID 8229 (Nuke)** - Ocula doesn’t obey requests to force the use of interactive licenses in render mode, i.e., when Nuke is run with "-xi". Instead, it will always request a render license in this mode. This is because Ocula 2.0 uses the Nuke NDK and there is currently no way for NDK plug-ins to tell the difference between this and the normal render mode. When a mechanism is provided in a later Nuke release, we will update the Ocula licensing to fix the problem.
Ocula 1.0v2

This is a maintenance release of Ocula.

Release Date
November 2008

Requirements
1. Nuke 5.1 on Windows, Mac OS X, or Linux.
2. Foundry FLEXlm Tools (FFT 4.0v1 or later) for floating licenses.

New Features
There are no new features in this release.

Improvements
There are no improvements in this release.

Bug Fixes
Fixed instability in plug-ins caused by OS incompatibility with FLEXlm 10.8 licensing module. Upgraded FLEXlm to 10.8.6 for improved Mac OS X 10.5 (Leopard) compatibility, and to 10.8.7 for improved 64-bit Linux compatibility.

Known Bugs and Workarounds
- BUG ID 5482 – Progress bar does not indicate what is being processed further up a tree when “Correlate using disparity” or “Correlate with Ocula” options are used. This will be fixed in a subsequent Nuke release.
- BUG ID 5904 – There is no progress bar when ”Correlate with Ocula” option is used. This will be fixed in a subsequent Nuke release.
- BUG ID 5979 – Running out of memory with complicated stereo scripts on 32-bit Windows. This will be fixed in a subsequent Nuke release.
• BUG ID 6075 - Slow processing when "Correlate with Ocula" option is used if source image is an EXR. This will be fixed in a subsequent Nuke release.

• BUG ID 6428 - ReConverge node: "Use Ocula if available" option causes a crash when reloading a script. This will be fixed in Nuke 5.1v3.
Ocula 1.0v1

This is the first release of Ocula 1.0 for Nuke.

Release Date
October 2008

Requirements
1. Nuke 5.1 on Windows, Mac OS X, or Linux.
2. Foundry FLEXlm Tools (FFT 4.0v1 or later) for floating licenses.

New Features
In this release, there are five plug-ins and a collection of tools that add extra functionality to existing Nuke features.

Improvements
This section will describe improvements to existing features in later versions.

Bug Fixes
This section will describe fixed bugs in later versions.

Known Bugs and Workarounds
- BUG ID 5482 – Progress bar does not indicate what is being processed further up a tree when "Correlate using disparity" or "Correlate with Ocula" options are used. This will be fixed in a subsequent Nuke release.
- BUG ID 5904 – There is no progress bar when "Correlate with Ocula" option is used. This will be fixed in a subsequent Nuke release.
- BUG ID 5979 – Running out of memory with complicated stereo scripts on 32-bit Windows. This will be fixed in a subsequent Nuke release.
• BUG ID 6075 - Slow processing when "Correlate with Ocula" option is used if source image is an EXR. This will be fixed in a subsequent Nuke release.

• BUG ID 6428 - ReConverge node: "Use Ocula if available" option causes a crash when reloading a script. This will be fixed in Nuke 5.1v3.
APPENDIX B

End User License Agreement

IMPORTANT: BY INSTALLING THIS SOFTWARE YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT DO NOT INSTALL, COPY OR USE THE SOFTWARE.

This END USER SOFTWARE LICENSE AGREEMENT (this “Agreement”) is made by and between The Foundry Visionmongers Ltd., a company registered in England and Wales, (“The Foundry”), and you, as either an individual or a single entity (“Licensee”).

In consideration of the mutual covenants contained herein and for other good and valuable consideration (the receipt and sufficiency of which is acknowledged by each party hereto) the parties agree as follows:

SECTION 1. GRANT OF LICENSE.

Subject to the limitations of Section 2, The Foundry hereby grants to Licensee a limited, non-transferable and non-exclusive license to install and use a machine readable, object code version of this software program (the “Software”) and the accompanying user guide and other documentation (collectively, the “Documentation”) solely for Licensee’s own internal business purposes (collectively, the “License”); provided, however, Licensee’s right to install and use the Software and the Documentation is limited to those rights expressly set out in this Agreement.

SECTION 2. RESTRICTIONS ON USE.

Licensee is authorized to use the Software in machine readable, object code form only, and Licensee shall not: (a) assign, sublicense, transfer, pledge, lease, rent, share or export the Software, the Documentation or Licensee’s rights hereunder; (b) alter or circumvent the copy protection mechanisms in the Software or reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software; (c) modify, adapt, translate or create derivative works based on the Software or Documentation; (d) use, or allow the use of, the Software or Documentation on any project other than a project produced by Licensee (an “Authorized Project”); (e) allow or permit anyone (other than Licensee and Licensee’s authorized employees to the extent they are working on an
Authorized Project) to use or have access to the Software or Documentation; (f) copy or install the Software or Documentation other than as expressly provided for herein; or (g) take any action, or fail to take action, that could adversely affect the trademarks, service marks, patents, trade secrets, copyrights or other intellectual property rights of The Foundry or any third party with intellectual property rights in the Software (each, a "Third Party Licensor"). Furthermore, for purposes of this Section 2, the term "Software" shall include any derivatives of the Software.

Licensee shall install and use only a single copy of the Software on one computer, unless the Software is installed in a "floating license" environment, in which case Licensee may install the Software on more than one computer; provided, however, Licensee shall not at any one time use more copies of the Software than the total number of valid Software licenses purchased by Licensee.

Furthermore, the Software can be licensed on an "interactive" or "non-interactive" basis. Licensee shall be authorized to use a non-interactive version of the Software for rendering purposes only (i.e., on a CPU, without a user, in a non-interactive capacity) and shall not use such Software on workstations or otherwise in a user-interactive capacity. Licensee shall be authorized to use an interactive version of the Software for both interactive and non-interactive rendering purposes, if available.

Finally, if the Software is an "Educational Version," Licensee may use it only for the purpose of training and instruction, and for no other purpose. Educational Versions of the Software may not be used for commercial, professional or for-profit purposes.

SECTION 3. BACK-UP COPY.

Notwithstanding Section 2, Licensee may store one copy of the Software and Documentation off-line and off-site in a secured location owned or leased by Licensee in order to provide a back-up in the event of destruction by fire, flood, acts of war, acts of nature, vandalism or other incident. In no event may Licensee use the back-up copy of the Software or Documentation to circumvent the usage or other limitations set forth in this Agreement.

SECTION 4. OWNERSHIP.

Licensee acknowledges that the Software and Documentation and all intellectual property rights relating thereto are and shall remain the sole property of The Foundry and the Third Party Licensors. Licensee shall not remove, or allow the removal of, any copyright or other proprietary rights notice included in and on the Software or Documentation or take any other action that could adversely affect the property rights of The Foundry or any Third Party Licensor. To the extent that Licensee is authorized to make copies of the Software or Documentation under this Agreement, Licensee shall
reproduce in and on all such copies any copyright and/or other proprietary rights notices provided in and on the materials supplied by The Foundry hereunder. Nothing in this Agreement shall be deemed to give Licensee any rights in the trademarks, service marks, patents, trade secrets, copyrights or other intellectual property rights of The Foundry or any Third Party Licensor, and Licensee shall be strictly prohibited from using the name, trademarks or service marks of The Foundry or any Third Party Licensor in Licensee's promotion or publicity without The Foundry's express written approval.

SECTION 5. LICENSE FEE.
Licensee understands that the benefits granted to Licensee hereunder are contingent upon Licensee's payment in full of the license fee payable in connection herewith (the "License Fee").

SECTION 6. TAXES AND DUTIES.
Licensee agrees to pay, and indemnify The Foundry from claims for, any local, state or national tax (exclusive of taxes based on net income), duty, tariff or other impost related to or arising from the transaction contemplated by this Agreement.

SECTION 7. LIMITED WARRANTY.
The Foundry warrants that, for a period of ninety (90) days after delivery of the Software: (a) the machine readable electronic files constituting the Software and Documentation shall be free from errors that may arise from the electronic file transfer from The Foundry and/or its authorized reseller to Licensee; and (b) to the best of The Foundry's knowledge, Licensee's use of the Software in accordance with the Documentation will not, in and of itself, infringe any third party's copyright, patent or other intellectual property rights. Except as warranted, the Software and Documentation is being provided "as is." THE FOREGOING LIMITED WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES OR CONDITIONS, EXPRESS OR IMPLIED, AND The Foundry DISCLAIMS ANY AND ALL IMPLIED WARRANTIES OR CONDITIONS, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, REGARDLESS OF WHETHER The Foundry KNOWS OR HAS REASON TO KNOW OF LICENSEE'S PARTICULAR NEEDS. The Foundry does not warrant that the Software or Documentation will meet Licensee's requirements or that Licensee's use of the Software will be uninterrupted or error free. No employee or agent of The Foundry is authorized to modify this limited warranty, nor to make additional warranties. No action for any breach of the above limited warranty may be commenced more than one (1) year after Licensee's initial receipt of the Software. To the extent any implied warran-
ties may not be disclaimed under applicable law, then ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO NINETY (90) DAYS AFTER DELIVERY OF THE SOFTWARE TO LICENSEE.

SECTION 8. LIMITED REMEDY.
The exclusive remedy available to the Licensee in the event of a breach of the foregoing limited warranty, TO THE EXCLUSION OF ALL OTHER REMEDIES, is for Licensee to destroy all copies of the Software, send The Foundry a written certification of such destruction and, upon The Foundry’s receipt of such certification, The Foundry will make a replacement copy of the Software available to Licensee.

SECTION 9. INDEMNIFICATION.
Licensee agrees to indemnify, hold harmless and defend The Foundry and The Foundry’s affiliates, officers, directors, shareholders, employees, authorized resellers, agents and other representatives (collectively, the “Released Parties”) from all claims, defense costs (including, but not limited to, attorneys’ fees), judgments, settlements and other expenses arising from or connected with the operation of Licensee’s business or Licensee’s possession or use of the Software or Documentation.

SECTION 10. LIMITED LIABILITY.
In no event shall the Released Parties’ cumulative liability to Licensee or any other party for any loss or damages resulting from any claims, demands or actions arising out of or relating to this Agreement (or the Software or Documentation contemplated herein) exceed the License Fee paid to The Foundry or its authorized reseller for use of the Software. Furthermore, IN NO EVENT SHALL THE RELEASED PARTIES BE LIABLE TO LICENSEE UNDER ANY THEORY FOR ANY INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, EXEMPLARY OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS OR LOSS OF PROFITS) OR THE COST OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, REGARDLESS OF WHETHER THE RELEASED PARTIES KNOW OR HAVE REASON TO KNOW OF THE POSSIBILITY OF SUCH DAMAGES AND REGARDLESS OF WHETHER ANY REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE. No action arising out of or related to this Agreement, regardless of form, may be brought by Licensee more than one (1) year after Licensee’s initial receipt of the Software; provided, however, to the extent such one (1) year limit may not be valid under applicable law, then such period shall limited to the shortest period allowed by law.
SECTION 11. TERM; TERMINATION.
This Agreement is effective upon Licensee’s acceptance of the terms hereof (by clicking on the "Accept" button) and Licensee’s payment of the License Fee, and the Agreement will remain in effect until termination. If Licensee breaches this Agreement, The Foundry may terminate the License granted hereunder by notice to Licensee. In the event the License is terminated, Licensee will either return to The Foundry all copies of the Software and Documentation in Licensee’s possession or, if The Foundry directs in writing, destroy all such copies. In the later case, if requested by The Foundry, Licensee shall provide The Foundry with a certificate signed by an officer of Licensee confirming that the foregoing destruction has been completed.

SECTION 12. CONFIDENTIALITY.
Licensee agrees that the Software and Documentation are proprietary and confidential information of The Foundry and that all such information and any communications relating thereto (collectively, "Confidential Information") are confidential and a fundamental and important trade secret of The Foundry. Licensee shall disclose Confidential Information only to Licensee’s employees who are working on an Authorized Project and have a "need-to-know" such Confidential Information, and shall advise any recipients of Confidential Information that it is to be used only as authorized in this Agreement. Licensee shall not disclose Confidential Information or otherwise make any Confidential Information available to any other of Licensee’s employees or to any third parties without the express written consent of The Foundry. Licensee agrees to segregate, to the extent it can be reasonably done, the Confidential Information from the confidential information and materials of others in order to prevent commingling. Licensee shall take reasonable security measures, which such measures shall be at least as great as the measures Licensee uses to keep Licensee’s own confidential information secure (but in any case using no less than a reasonable degree of care), to hold the Software, Documentation and any other Confidential Information in strict confidence and safe custody. The Foundry may request, in which case Licensee agrees to comply with, certain reasonable security measures as part of the use of the Software and Documentation. Licensee acknowledges that monetary damages may not be a sufficient remedy for unauthorized disclosure of Confidential Information, and that The Foundry shall be entitled, without waiving any other rights or remedies, to such injunctive or equitable relief as may be deemed proper by a court of competent jurisdiction.

SECTION 13. INSPECTION.
Licensee shall advise The Foundry on demand of all locations where the Software or Documentation is used or stored. Licensee shall permit The Foundry or its authorized agents to inspect all such locations during normal business hours and on reasonable advance notice.
SECTION 14. NONSOLICITATION.
Licensee agrees not to solicit for employment or retention, and not to employ or retain, any of The Foundry's current or future employees who were or are involved in the development and/or creation of the Software.

SECTION 15. U.S. GOVERNMENT LICENSE RIGHTS.
The Software, Documentation and/or data delivered hereunder are subject to the terms of this Agreement and in no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication or disclosure by the U.S. Government is subject to the applicable restrictions of: (i) FAR §52.227-14 ALTS I, II and III (June 1987); (ii) FAR §52.227-19 (June 1987); (iii) FAR §12.211 and 12.212; and/or (iv) DFARS §227.7202-1(a) and DFARS §227.7202-3.

The Software is the subject of the following notices:
* Copyright (c) 2009 The Foundry Visionmongers, Ltd. All Rights Reserved.
* Unpublished—rights reserved under the Copyright Laws of the United Kingdom.

SECTION 16. SURVIVAL.
Sections 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17 and 18 shall survive any termination or expiration of this Agreement.

SECTION 17. IMPORT/EXPORT CONTROLS.
To the extent that any Software made available hereunder is subject to restrictions upon export and/or reexport from the United States, Licensee agrees to comply with, and not act or fail to act in any way that would violate, the applicable international, national, state, regional and local laws and regulations, including, without limitation, the United States Foreign Corrupt Practices Act, the Export Administration Act and the Export Administration Regulations, as amended or otherwise modified from time to time, and neither The Foundry nor Licensee shall be required under this Agreement to act or fail to act in any way which it believes in good faith will violate any such laws or regulations.
SECTION 18. MISCELLANEOUS.

This Agreement is the exclusive agreement between the parties concerning the subject matter hereof and supersedes any and all prior oral or written agreements, negotiations, or other dealings between the parties concerning such subject. This Agreement may be modified only by a written instrument signed by both parties. If any action is brought by either party to this Agreement against the other party regarding the subject matter hereof, the prevailing party shall be entitled to recover, in addition to any other relief granted, reasonable attorneys’ fees and expenses of litigation. Should any term of this Agreement be declared void or unenforceable by any court of competent jurisdiction, such declaration shall have no effect on the remaining terms of this Agreement. The failure of either party to enforce any rights granted hereunder or to take action against the other party in the event of any breach hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights or subsequent actions in the event of future breaches. This Agreement shall be governed by, and construed in accordance with English Law.

Copyright (c) 2009 The Foundry Visionmongers Ltd. All Rights Reserved. Do not duplicate.
APPENDIX B

End User License Agreement
INDEX

A-Z

A
alignment
  horizontal 43
  vertical 50

C
cameras
  converging 50
  parallel 50
colour discrepancies 58
  colours
    matching between views 58
    converging cameras 50
    creating
      a new view 66
      two new views 43

D
disparity channel 33
disparity fields 17, 32
disparity vectors 32

E
End User License Agreement 107

F
Foundry, The 14
  web site 15

H
horizontal alignment
  correcting 43

I
installation 6
  on Linux 7
  on Mac 7
  on Windows 6
interaxial distance 43

K
keystoning 50

L
Licensing Ocula 14

O
O_ColourMatcher 58
O_Correlate 91, 97
O_DepthToDisparity 32, 76
O_DisparityGenerator 32
O_DisparityToDepth 83
O_InteraxialShifter 43
O_NewView 66
O_Solver 17
O_VerticalAligner 50
occlusions 44, 66

P
paint effects 91
parallel cameras 50

R
release notes 99
rotscoping 91

S
system requirements 6

T
The Foundry 14
  web site 15
The Foundry products 14

V
vertical alignment
  correcting 51
views
  creating new 66

W
web site 15

The Foundry Ocula 2.0