8. Working with stereoscopic 3D

A starter guide to stereo 3D production

No one denies that black-and-white photography is elegant. In the hands of an expert photographer, it can communicate ideas and emotions incredibly well. But today it has been largely superseded by color photography. Despite the artistic potential of black-and-white, people have a natural yearning for images with more realism, more dimensions. When color cameras and TVs became affordable, not many people preferred to stay with black-and-white even if it had much artistic potential left untapped and cost a bit less than color.

Color is one essential dimension of our vision system; depth is another. Three-dimensional viewing is essential to how humans make sense of the world around them. Yet, except for a few failed commercial attempts and those charmingly retro ViewMaster photo viewers, depth has been lacking from our everyday images. The main reason is simply that the technology hasn't been there – delivering a good 3D image all the way to consumers has been extremely difficult. When done wrong, 3D can give you headaches and nausea, which hasn't exactly reduced the challenges of delivering 3D.

In recent years, the pieces have finally come together thanks to digital technology. There are now several good and commonly available technologies for viewing 3D images, both in movie theatres and on private screens. Some companies use polarized glasses, others use active shutter glasses. These technological differences are all on the projection side, and they don't directly affect how the content is produced. In both movie and TV production, practically everyone now agrees that **dual-stream stereoscopic 3D** is the unavoidable next big step in how we create and consume moving images.

A few words on the terminology… The word *stereo* is familiar to most from the audio world. In 3D video, it has essentially the same meaning as in audio. Stereo means that our content is produced in a way that matches the human sensory model. For sound, that means separate signals for each ears ("stereophonic"); for video, it means separate streams for each eye ("stereoscopic"). Two eyes, hence *dual-stream*. You'll encounter that word fairly often in this document.

In PixelConduit, *dual-stream* more precisely means that there are two discrete video streams for the left and right eyes. The alternative is to somehow "pack" the two images into a single video stream, for example by placing them side-by-side within a regular video frame. This is commonly done today, but it unavoidably reduces the image resolution to half. Dual-stream is the true 3D format, and PixelConduit supports it everywhere. (This is comparable to audio – who wouldn't prefer a real stereo music player to a system where the two signals are combined into a mono signal?)

Working with dual-stream 3D in PixelConduit is easy thanks to the *node-based* user interface. Conduit represents video images as connections

between nodes, which can either be sources (for example a camera or a video file) or they can process or display the image in some form. “Upgrading” from ordinary mono video to stereo 3D simply means that we now have two connections flowing between the nodes. The streams can be separated and rejoined at any point in the processing flow, so it’s easy to modify just one eye’s image if needed. It’s equally easy to process the two streams at once and even promote mono video to stereo, because PixelConduit Complete includes effect tools specifically designed for this purpose.

This guide will cover the following topics:

- Importing stereo 3D images and splitting packed 3D video into dual streams
- Viewing 3D images using the Stereo 3D Preview tool
- Adjusting the stereo effect using the Stereo 3D Adjust tool
- Lining up stereo images and fixing alignment problems using the Stereo 3D Align Images tool
- Applying color correction and other effects using the Stereo 3D Conduit Effect tool
- Creating stereo 3D layers from regular video
- Rendering and exporting 3D, including batch automation.

Finally, I’ll briefly brush on advanced topics including how to use cue lists to automate live effect setups; how to use custom effect nodes; and how to use JavaScript to render stereo 3D graphics.

1. Getting the streams flowing: Importing (and perhaps splitting) video

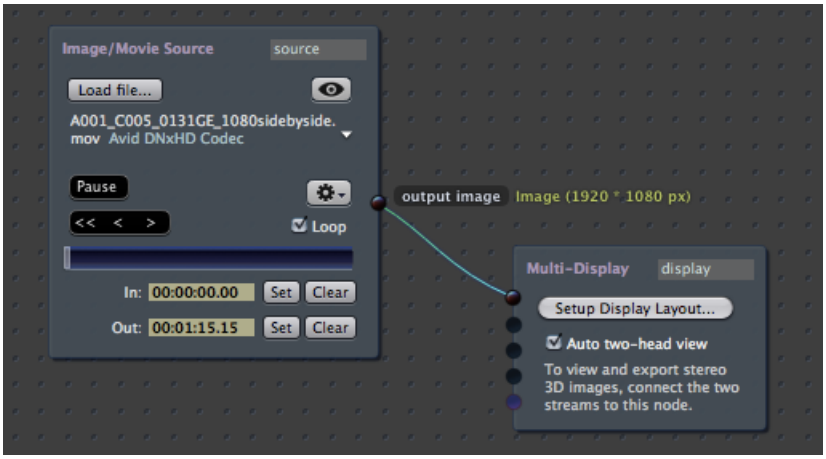
Unless you’re generating graphics from scratch, there’s usually some kind of source material from which to start working. In Conduit 3D, these source materials can be almost anything:

- Regular video files (sometimes called ‘mono’ video)
- Dual-stream 3D video (i.e. separate movie files or image sequences)
- Packed 3D video (e.g. video containing the two eyes’ images placed side-by-side)
- Live video from a camera
- Live 3D video captured from two separate cameras
- Live dual-stream 3D SDI-HD video captured via a 3D capture unit by BlackMagic Design
- Or even images loaded from the Internet (using the Web Data Source node widget)

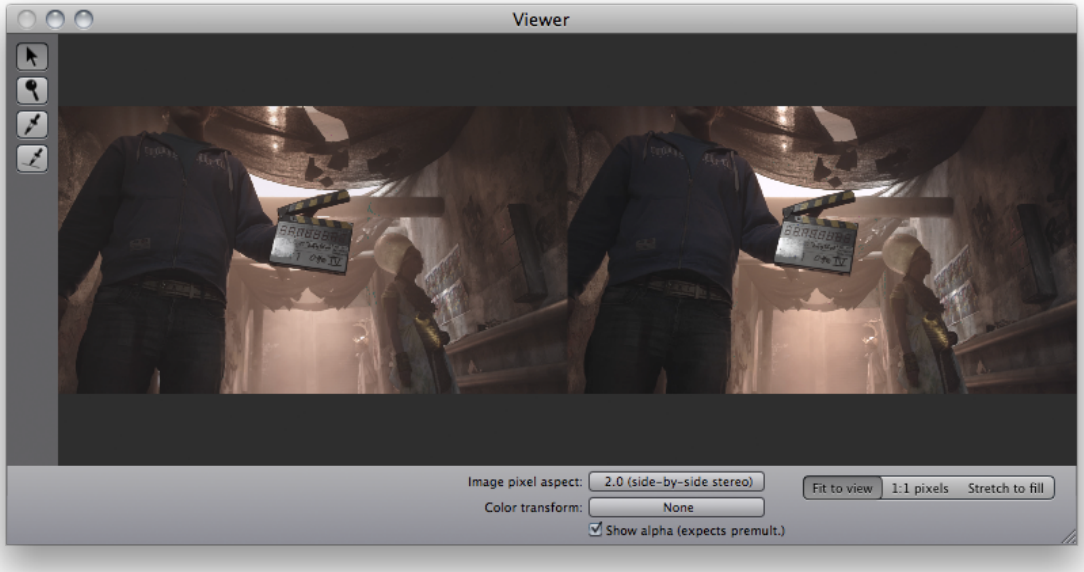
Both mono and stereo video is loaded in the same way, using the *Image/Movie Source* node widget. These are created in the Project window.

Throughout this guide, the project is expected to be in **Timeline** mode. This is appropriate for rendering out effects. The other mode, Free run, is meant for projects like live shows that don't have a set duration. You can find out more about these topics in the first chapter of this book.

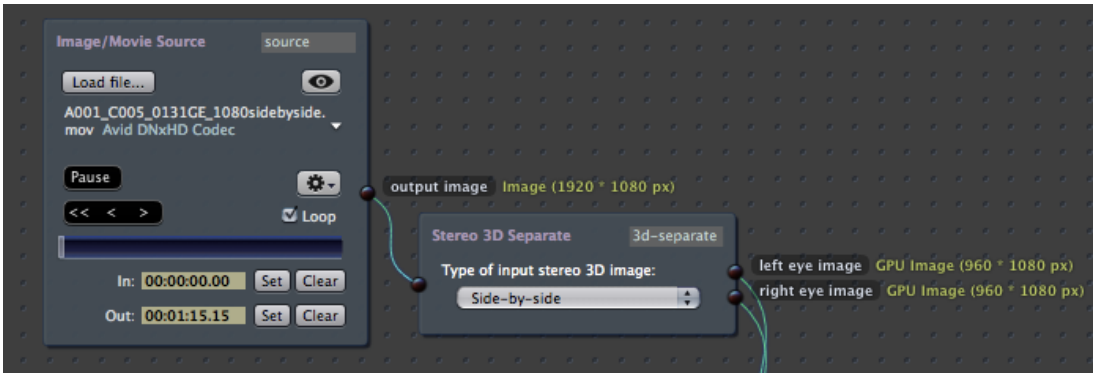
For regular mono video (i.e. plain old 2D video) or packed 3D video, we need a single Image/Movie Source. To view it, you can open its Preview window (the "eye" button), or connect it to the Multi-Display node widget:



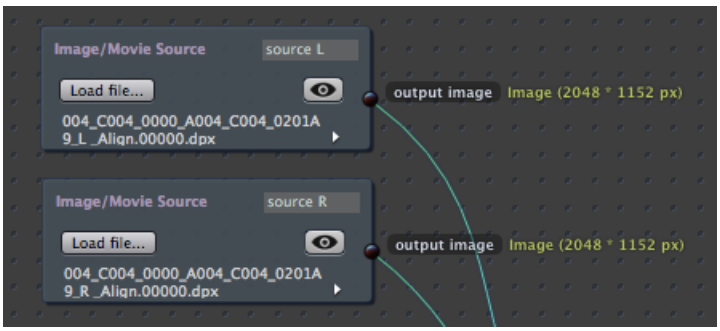
The clip loaded here is an example of packed 3D video: it's a QuickTime movie containing side-by-side 3D. When viewed on a regular display, it appears squashed to half. In Conduit there's an easy way to view the images in the right proportions. In the Viewer window, click on "Image pixel aspect" and choose the option labelled "2.0 side-by-side stereo":



To process this side-by-side stereo image, it needs to be separated into dual streams. The *Stereo 3D Separate* node widget is the right tool for this job:



To load dual stream 3D from two separate files, we need two instances of Image/Movie Source:



PixelConduit can read and write most pro video formats, including QuickTime files with 10/16 bits per channel Y'CbCr color, as well as DPX/Cineon/TIFF image sequences with high color depths. Preserving color fidelity and high precision thanks to floating point rendering has been a high priority for Conduit development. Hopefully you'll find that PixelConduit can integrate into whatever kind of production pipeline you have in place.

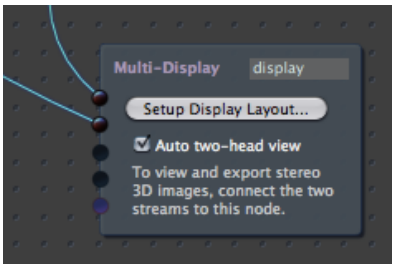
A live camera setup could be constructed like the dual stream input above, just using camera sources instead. PixelConduit supports all video capture devices that have a QuickTime driver. Depending on the manufacturer's QuickTime driver implementation, it may be possible to install multiple cards of the same model in one Mac Pro, allowing dual capture. (Contact the capture card vendor about this feature. If it's not supported, you could always use two cards from different manufacturers to avoid "driver clash".)

However, to capture live 3D from professional devices, you don't have to use two input cards. Conduit 3D has direct support for capture units created by BlackMagic Design (*www.blackmagic-design.com*). Their lineup of devices includes the **DeckLink Extreme 3D**, a dual-stream capture card for Mac Pros with 2K support and HD-SDI interfaces; as well as the new **UltraStudio 3D**, a portable solution that takes advantage of the superfast Thunderbolt interface found in latest MacBook Pro and iMac computers.

So, if you have a BlackMagic unit (whether it's 3D or not), you should always use the *Live Source (BlackMagic)* node widget in Conduit rather than the QuickTime capture equivalent. Conduit's BlackMagic capture support interacts directly with the low-level driver created by BlackMagic Design,

thus guaranteeing best performance and compatibility with cutting-edge features like 3D dual-stream capture.

Now we have dual streams. How to view them? For a quick preview, we can just connect the two streams to the Multi-Display that's part of the project:



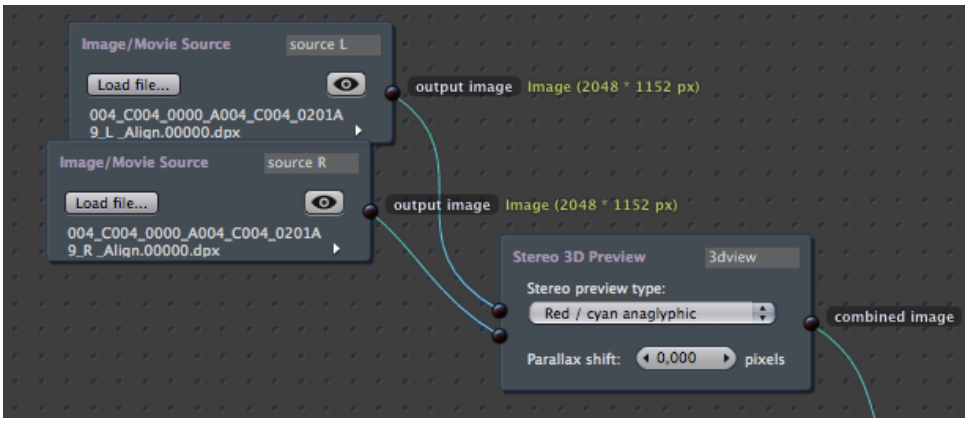
As you can see, the Multi-Display has a setting called "Auto two-head view". With this setting enabled, the Viewer is intelligently switched to a side-by-side mode when two streams are connected.

This is a convenient way to check that your dual-stream images look as intended, but it doesn't give an idea of the depth effect. For that, we need to make a real 3D view somehow. The next section will discuss this topic.

2. Viewing 3D

The simplest way to view dual-stream 3D with genuine depth vision is to use **anaglyphic color glasses**. Yes, that means those '50s style cardboard glasses with different colored lenses for each eye. They're not great for long viewing sessions, but definitely good enough to get an idea of what the depth effect looks like. And the eyeglasses are of course very cheap. They come in red/cyan and green/magenta varieties; Conduit 3D supports both.

The tool to use here is the *Stereo 3D Preview* node widget:



There is a drop-down menu where you can choose the type of preview. Options include:

- Red / cyan anaglyphic
- Green / magenta anaglyphic
- Side-by-side
- Top/bottom

- Interlaced

When you want to move beyond anaglyphic, **packed 3D** is the most easily deployed option. As the above list shows, you can use the Preview node widget also for packed 3D previews.

Practically all 3D TVs and displays support some kind of packed 3D. It's usually side-by-side, but interlaced is also found on some models.

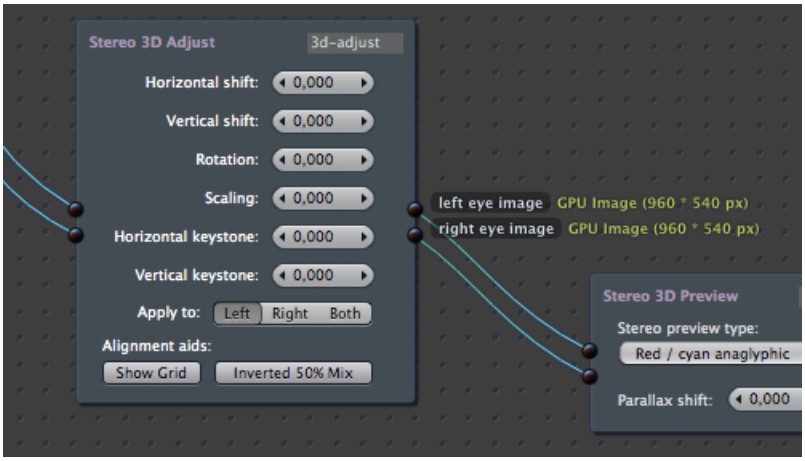
To view on a 3D TV, connect the TV to your Mac using HDMI or some other digital output. (There are various video output ports on Macs, but DVI and DisplayPort can be easily converted to HDMI, so that's usually the best option.) Don't use display mirroring, but instead set the TV as a second display so that you can use Conduit on your main display while viewing on the TV.

Then enable full-screen mode in Conduit (it can be found at the top of the Project window). Just select the appropriate packing mode in the Stereo 3D Preview node widget, connect it to the Multi-Display output, and you're all set to watch video in 3D through Conduit. (If your display device requires interlaced input, make sure to match the project's render resolution to the size of the display output so that the interlaced lines are precisely aligned. The render resolution is found in Conduit's Project window under the "Project Settings" tab.)

3. Going deep or staying shallow: Adjusting the 3D effect

Now that we have 3D input and viewing set up, it's time to look into essential stereo 3D adjustments. To make the depth effect work, we often need to tweak the relationship between the two images. By shifting the images horizontally, it's possible to move the *screen plane* forwards or backwards – in other words, the part of the image where the two eyes' views don't diverge at all. To the viewer of a 3D image, this feels like the plane of the screen. (Changing this plane is also called parallax shifting.) Any objects with depth will be either behind or in front of the screen plane, so moving the screen plane is an important tool for making choices about the depth effect.

The Stereo 3D Adjust node widget can be used to perform dual-stream adjustments:

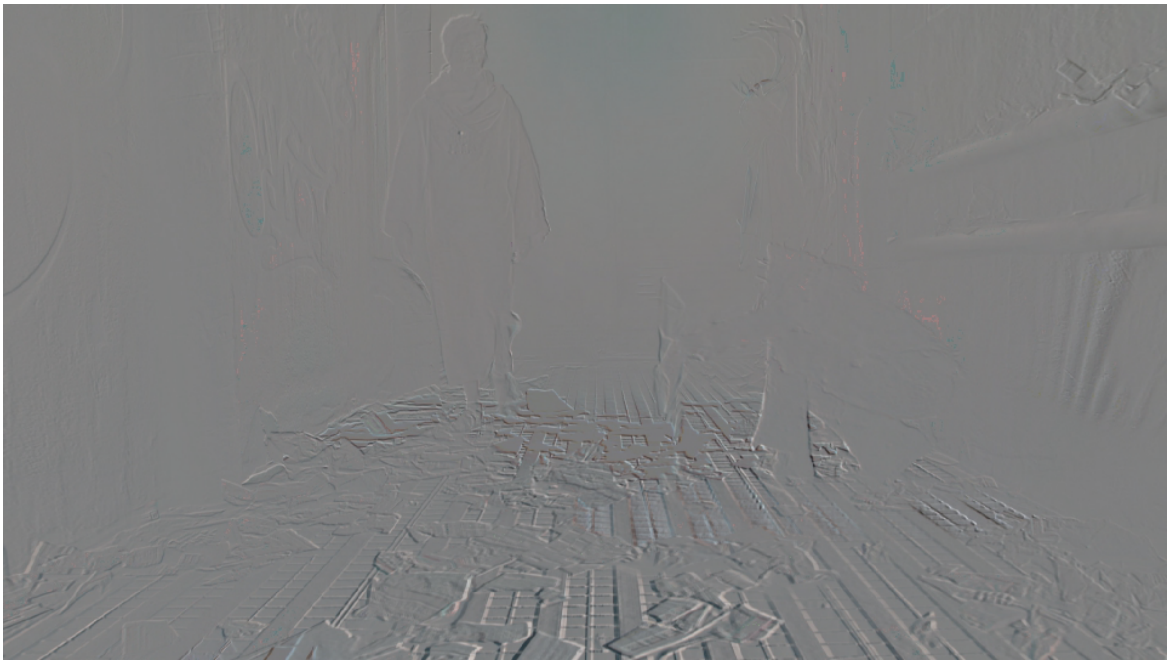


Another useful adjustment is horizontal keystone, where the images are transformed so that the eyes are tilted slightly towards each other (or away from each other). This can be used to modify the depth focus more widely than just shifting the screen plane.

Like most anything in 3D vision, if you adjust too much, the image will become difficult to view and the depth effect may stop working. However, where these limits lie is not really set in stone. Rather it depends on several things: the viewer's personal tolerance is a factor, but also the context of the viewing. 3D images always require some mental adaptation from the viewer, and that doesn't change in the blink of an eye. For example, if a particular shot is going to be edited into a sequence with very flat shots that are completely behind the screen, then you probably should try to avoid making the shot such that there's suddenly lots of action in front of the screen because that would make the viewer uncomfortable as she has to "readjust her brain".

To make alignment easier, the Stereo 3D Adjust node has two viewing aid modes. They are the buttons labelled "Show Grid" and "Inverted 50% Mix". The former is easily explained – it just shows a grid that you can use to place elements with more precision.

Inverted 50% Mix is a visual aid that has a bit of a learning curve, but is really useful once you know how to interpret it. Functionally it inverts the second image, then mixes the two images together. If the images are the same, the resulting image would be 50% gray. But with stereo 3D images, there will be slight differences. This viewing mode brings out those differences in way that makes it quite easy to see which parts of the 3D image are behind the screen and which parts are in front. Here's an example:



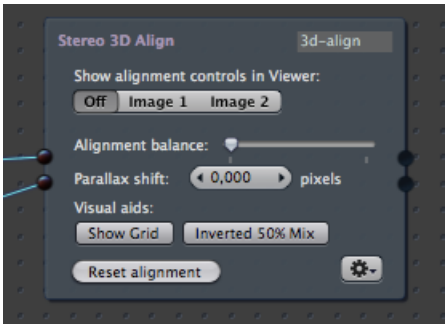
This is a well-shot 3D image that has been aligned in post. We can see how inverted 50% mix gives the image a relief-like quality. The parts of the image that are in front of the screen are "embossed" one way, and those that are behind the screen are embossed the other way.

The lines on the floor, extending from the front of screen into the image offer a good example of how depth actually works in stereo 3D images: we can easily see how the disparity between the lines gets wider at the front where it's closer to the camera and thus the difference between the view seen by the two eyes is larger.

4. Cross-eyed and painless – fixing alignment issues

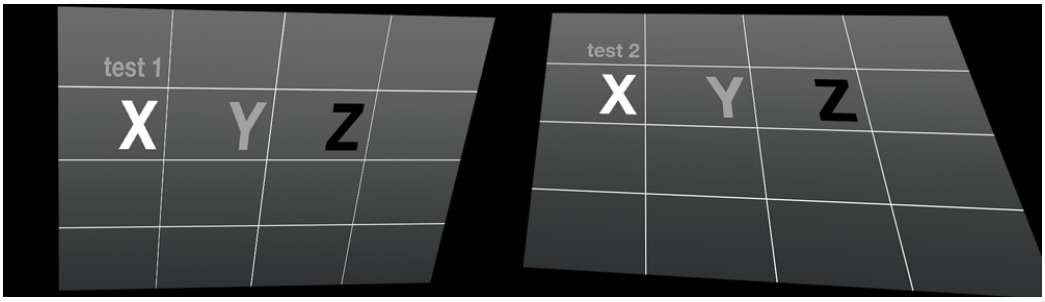
The unfortunate reality is that stereo 3D images very often aren't perfectly shot. It's difficult to align cameras precisely, and the resulting images may be "off" by so much that the depth effect doesn't work at all, or at least doesn't edit well with other content.

To deal with these alignment problems, Conduit 3D includes a tool called *Stereo 3D Align*:



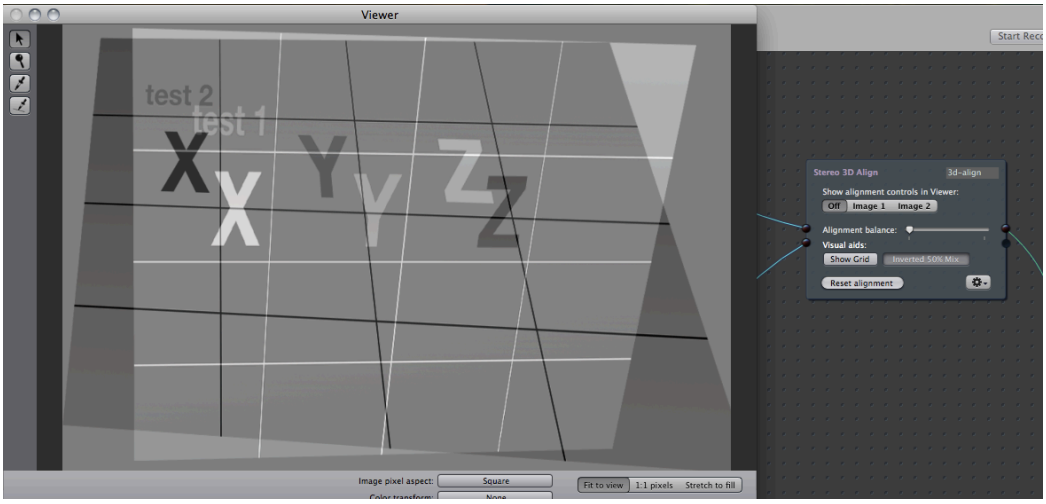
This tool does perspective-corrected alignment. The basic idea is that you pick four points in both images, and the alignment tool then warps the images so that those points are aligned.

Let's try it with an artificial "nightmare scenario", just to get an idea of how much warping can be fixed. Here's a test image (ugly but informative):



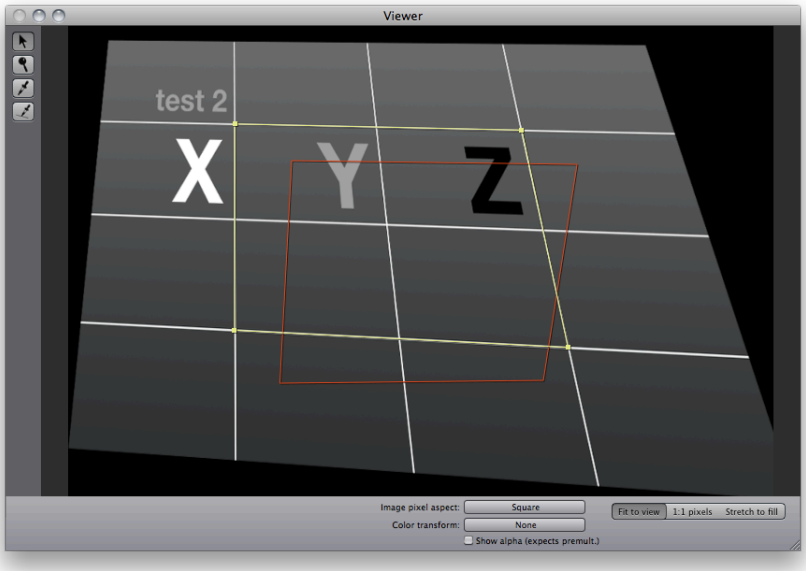
Except for the numbers 1 and 2, this is simply the same image that has been distorted in two different ways.

When viewed using Inverted 50% mix, it's obviously a mess with no alignment at all:



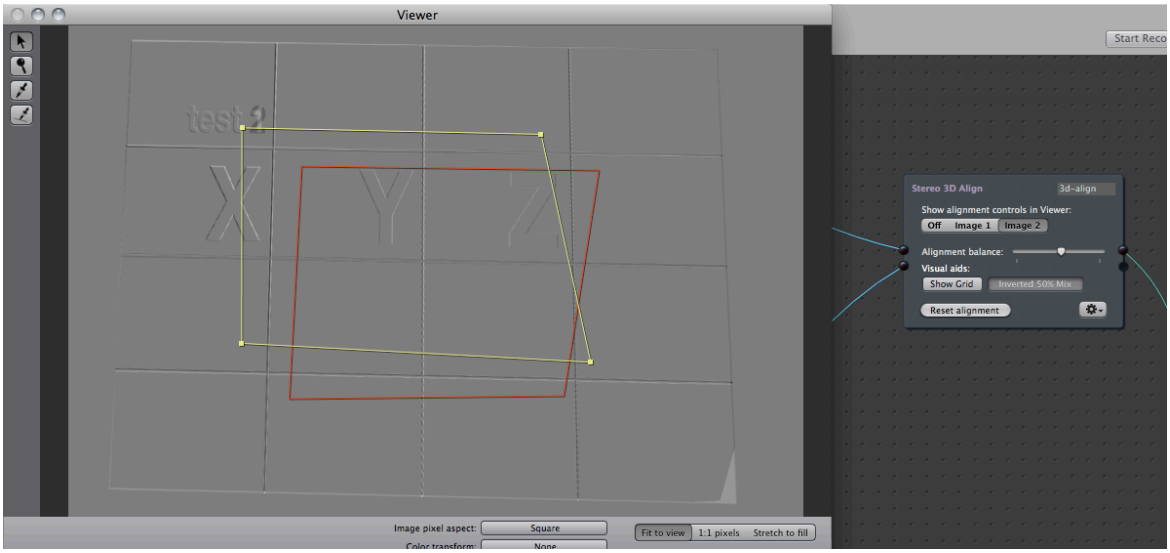
We must start by deciding the points that should be aligned in the result. In a real-world image, these aligned points should be on the screen plane, facing the cameras as directly as possible. In this case, we'll choose the four middle corners of the grid inside the test image (i.e. the corner at the top-right of the letter "X", and the three other points in the square of those grid line intersections).

First the left-eye image is aligned by clicking on the "Image 1" button to show its controls in the Viewer window. The four points are initially at the middle of the screen, so we just drag them precisely on top of those corner points. The same operation is then done for "Image 2". This is what it looks like in the Viewer:



The corner points being edited are highlighted in yellow. (It's a bit difficult to see with this greyscale test image, but you can see it better in the next screenshot.) In addition to the yellow rectangle, there's a red rectangle which shows the corner points that were picked for the first image, i.e. where the same points lie in the left-eye image.

Now the points have been specified, so let's click "Off" (to hide the draggable controls in the Viewer) and view the image again using 50% inverted mix:

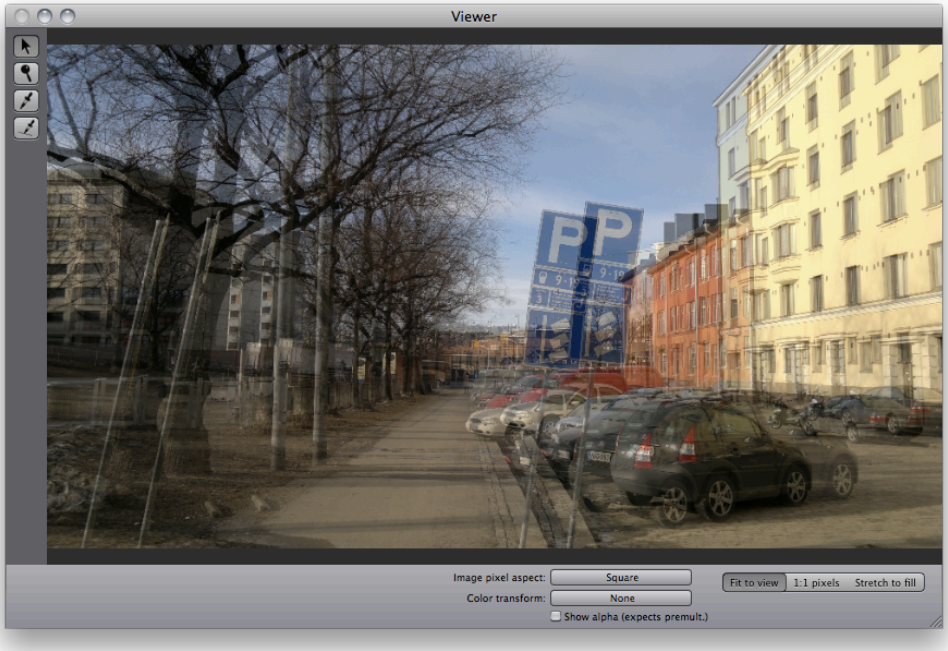


As you can see, the lines are almost perfectly aligned. By tweaking the points while zoomed in the Viewer, we could make the alignment even more precise and then those grid lines would vanish completely into gray.

Notice how the resulting image has the grid lines almost straight vertical, instead of at an angle like in both of the original images? This is due to the "Alignment balance" setting, which you can see in the above screenshot has been adjusted to near the middle of the slider. This setting is important because it determines how much the images are warped towards each other: when the balance slider is fully at left, it means that the right-eye image will be warped to match the left-eye image. By dragging the slider, we can find an alignment that is a suitable compromise between the two original positions.

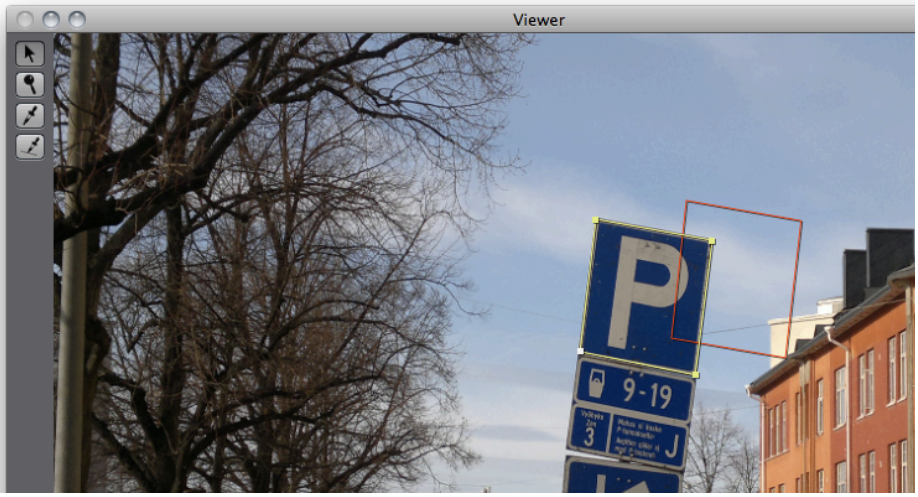
Notice also how, in the bottom-right-hand corner, there's an error that we can't fix by alignment: the other image's corner has been cut. If you look at the original right eye image, the plane is indeed slightly cut. What this means is that the aligned result has an error, an artifact visible in only one eye. We could hide it by zooming in the image, or perhaps by masking out the corner of the image using something like a soft mask shape. That kind of effects can be done with Conduit's 3D-aware effect features… But that's a topic for a later section.

To get a better feel for this alignment tool, let's try it on some real images. The following pictures of a street were shot with a handheld still camera with no regard for the 3D effect, so they are completely out of alignment. Here are the pictures blended on top of each other:



(This blending is done with the regular *Conduit Effect* node widget. Simply connect the two streams to the Conduit Effect. It creates a default blending effect like the above, using an Over node. You can then edit the effect further by clicking on "Edit" to open the Conduit Editor, an effect design tool with lots of effect nodes and many possibilities – you can read more about it in the Conduit wiki manual.)

With the two images so out of alignment, is there any hope to get a working depth effect here…? For this example, I tried to align the corners of the "P" parking sign:



Here you can see how the yellow points have been dragged on top of the corners of the P sign, while the red outline shows the same shape in the other image.

It's a good idea to choose points that are perpendicular to the cameras. However they don't have to actually be points that are on the depth level that

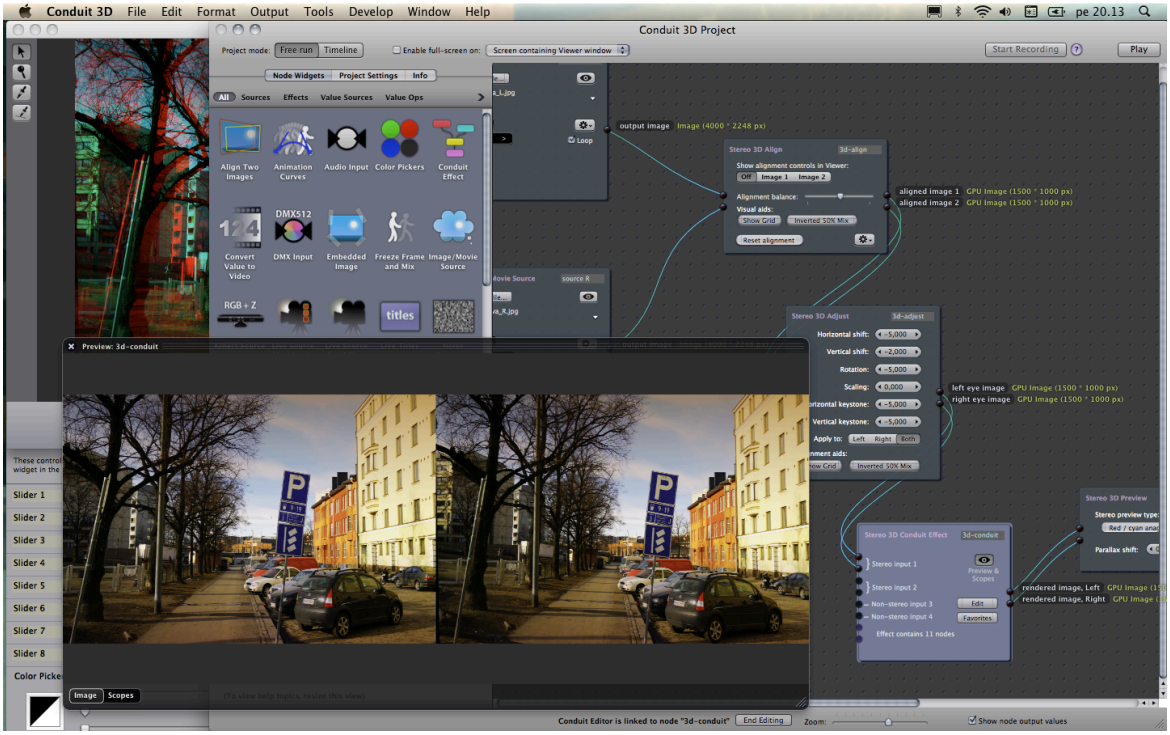
you'd want to be aligned eventually, i.e. the screen plane. In fact, the Stereo 3D Align tool makes it easy for you to change the screen plane: there is a "Parallax shift" setting which is applied right together with the alignment for best results. First find suitable points, then modify the shift so that the points lie on the depth level you want.

5. Color, gloss, shadows: Using effects in 3D

In the previous section, we aligned the "P" sign in the image of the street, and tweaked the adjustment to get the screen plane roughly in place. Next, we'll improve the look of the image by applying some color correction.

The original image is quite pale and lacks depth. The 3D effect would be stronger if the image had more punch at the middle, at the farthest distance. To achieve this, we'll apply a vertical gradient that subtly darkens the top and bottom parts of the image. Also, the yellow-orange colors will be given more energy with a Hue/Saturation/Value adjustment.

This screenshot shows the result of these simple color adjustments:



In the Project view, a new node widget called *Stereo 3D Conduit Effect* has been added.

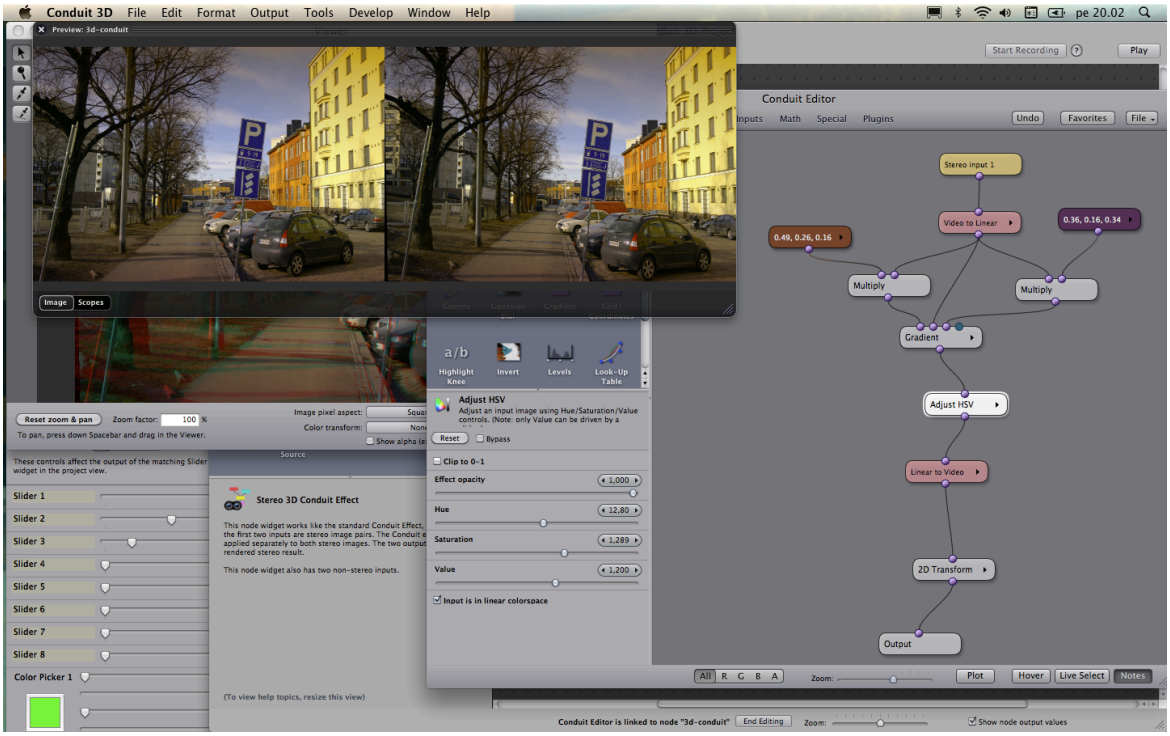
The "Conduit effect" is a customizable image mixer tool that can be used for layering, color corrections, drawing shapes, and more. Inside this node widget, there's an effect setup that can be created and modified from simple building blocks, image processing nodes. Click on "Edit" to open the *Conduit Editor*, the user interface for editing these setups.

In the above screenshot, the dark floater window with the two side-by-side images is the preview of the Conduit effect. (You can open previews by

clicking on the eye icon.) Preview windows also have a vectorscope view that allows for more detailed inspection of the image's luminance and color levels – this is most useful for effects that require precise color manipulation, in particular keying.

The adjustment and preview node widgets we've used until now have only had two inputs, left and right eye images. The Stereo 3D Conduit Effect has many more. The special thing about this effect widget is that it accepts both stereo and mono image inputs, and allows you to treat them equally. There are two stereo inputs and two mono inputs (labelled "non-stereo input" 3 and 4 in the user interface). There are also two more inputs that accept *value* types instead of images. These inputs can be used to pass control values like sliders and color pickers into the effect. (This makes it easy to use Conduit's sliders for controlling multiple effects, and you can also drive effects with something like a MIDI controller or JavaScript with no extra hassle.)

The following screenshot shows how the Conduit Editor looks for this effect:



Effect setups in the Conduit Editor are read from top to bottom. Just like in the Conduit Project window, each node here represents some kind of input or processing operation. There's an important difference, however: the nodes in the Conduit Editor are graphics operations that can be compiled to run very efficiently on the graphics card (GPU). Thanks to this *node fusion* feature, you can easily combine lots of these operations within the Conduit Editor. In contrast, the node widgets found in the Conduit project view represent specific larger units, e.g. video files, live cameras, display outputs…

Here's the most essential thing to understand about the Conduit Editor: whatever images you've plugged into the Conduit Effect node widget become available inside the effect setup as **Input** nodes. The effect doesn't care what images you feed it, or where its rendered output ends up – that's something you decide in the Project window. Within the Conduit Editor, that outside world is only represented by the Input and Output nodes (as well as Slider

and Color Picker nodes which represent the numeric and color values you can also plug into this Conduit Effect).

As was mentioned above, the Stereo 3D Conduit Effect has four inputs – two stereo and two mono. This means you can use four Input nodes inside the effect. (In this example, I'm only using the first input, but to access the other inputs you'd create additional Input nodes by dragging from the node box at the top-left-hand corner of the Conduit Editor window.)

There is also a regular *Conduit Effect* node widget in Conduit 3D. The special thing about this Stereo 3D Conduit Effect is of course that it renders in stereo – the effect is applied to both eyes' images automatically. If you import mono images, they are treated like stereo that's just flat in the screen plane, so combining stereo and mono images is very easy. Additionally, some nodes support true 3D rendering, so you can warp images in 3D space. (See the next section, *Creating 3D Layers*, for more information.)

Let's pick apart the effect setup shown in the above screenshot. At the top, the yellow node is the input image. Immediately below it's converted to **linear light colorspace**. This is one of Conduit's great advantages. Briefly, converting to linear removes the gamma correction that is applied to digital images to make them easier to process in traditional systems without floating point precision. When your image is converted to linear, the pixel values correspond closely to the original light values picked up by the camera sensor. This way, we can create effects that have a more photographic appearance because we're working with true light values and how they combine together.

The next node is a **Gradient**. More precisely, the input image is first multiplied by two different colors – a brown and a purple, as you can see by the color of the nodes. This multiplication is simply a color tint effect (think of it as shining a light of a specific color – if we shine a yellow light on a white paper, it's equivalent to multiplying white by yellow). Then, a gradient is rendered using the original image and the two multiplied versions as inputs. This gradient is vertical. The purple-tinted image is used to tint the sky, and the brown-tinted image gives the earth a deeper color. This simple gradient effectively works much like a photographic filter. (Again, it's worth mentioning that this effect would look substantially different and more "digital" if the image were not in linear light.)

Next, we have an **Adjust HSV** node. This is used to color the image warmer and more saturated. You can see how the yellow/orange walls now pop out much more compared to the original on page 9.

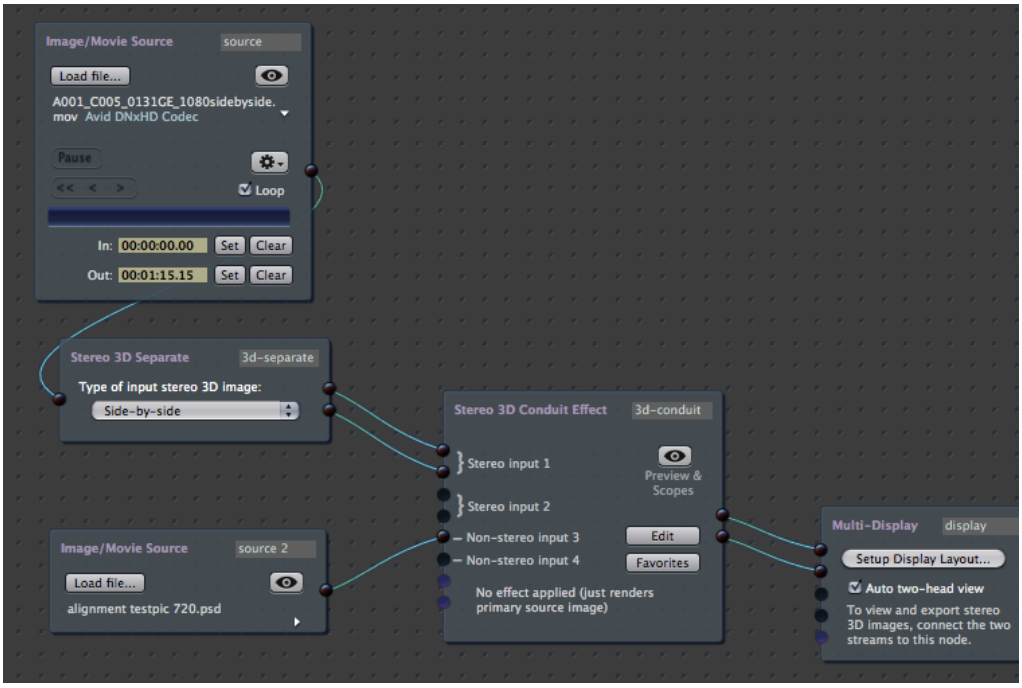
Finally, the image is restored to video colorspace using a **Linear to Video** node. There's also a 2D Transform node that I used to scale the image up so that the borders of the alignment would not be visible.

6. A good use of space – creating 3D layers

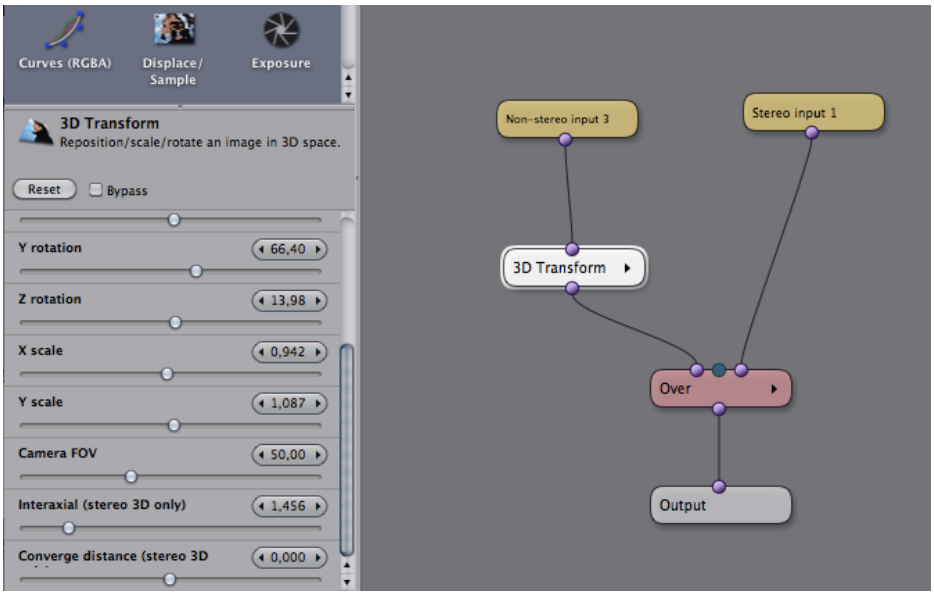
The previous section showed color correction being applied to a stereo image. That didn't modify the depth of the image; the disparity between the left and right images remained the same regardless of how much the color is tweaked. But it's also possible to render 3D layers using the Conduit Editor, and these will have depth in them.

In the next example, I'll show how to use the *3D Transform* node for this, and also show a quick example of using a Shapes node to mask the rendered layer so that it can be more smoothly composited with other stereo elements.

This project consists of a stereo source video and a mono source image. They are connected to the respective inputs on a Stereo 3D Conduit Effect:



Inside the Conduit effect, we have the following:

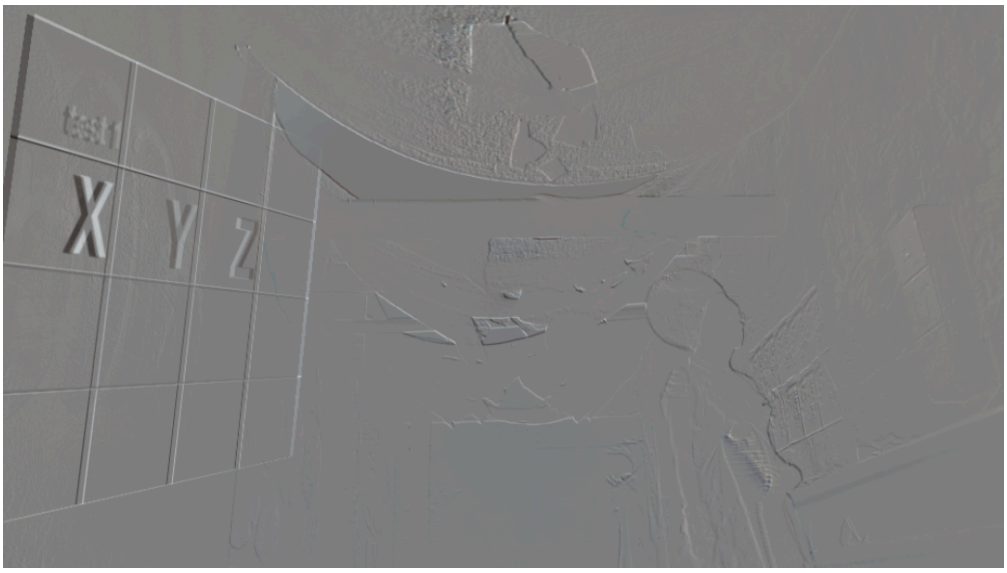


This setup is very simple: the mono image is rotated and positioned in space using the 3D Transform node. Then it's composited on top of the other input image using an Over node.

Here, the background image is a beautiful corridor shot in 3D, while the mono image being transformed is a simple test grid (in fact the same one used previously when discussing alignment). The result looks like this:

The Over node was set to use the Screen compositing mode, so the test image is semi-translucent on top of the background.

Looking at the stereo images side-by-side, it's not obvious that the test layer has in fact been rendered in stereo 3D. To verify that there is in fact depth there, we can use the 50% Inverted Mix feature previously discussed. It's part of the Stereo 3D Adjust node. This function renders the left and right images on top of each other so that the differences between them are accentuated. Here's what it looks like:



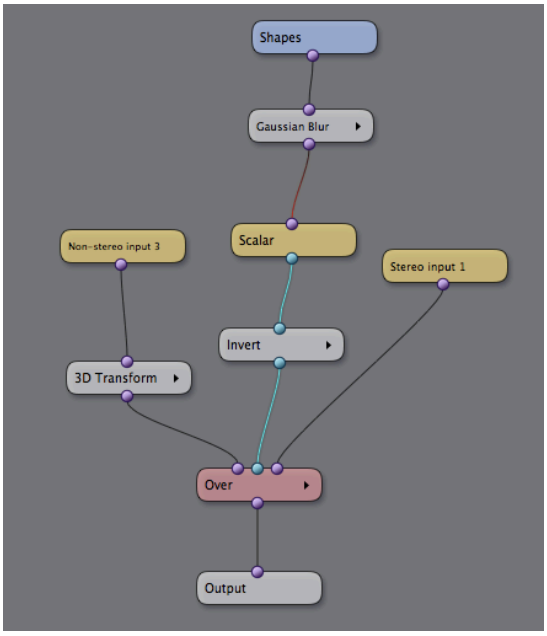
With this preview, it's more obvious that there is depth in the test layer (i.e. the grid with "XYZ" labels).

In fact, we can now see that there's too much depth there compared to the background image: the letter X has way too much disparity compared to anything else in the image. To control the depth effect of the rendered 3D layer, we could return to the Conduit Editor and tweak two parameters of the Transform 3D node that affect the stereo camera setup. These parameters are called **Interaxial** and **Converge distance**.

Interaxial is the distance between the two cameras. If it's zero, no depth is rendered. Converge distance determines how much the cameras are pointed towards each other. If it's zero, the cameras are fully perpendicular. You can use this parameter to determine where the screen plane is located – but be careful with it, because too much tilting will result in a cross-eyed effect. These settings should be based on the size of the objects in view and the range of depth that you want to have. There are many books available on this topic, but you can also learn by experimentation to see what works – just make sure to take regular breaks from 3D viewing to avoid a headache from playing around with unnatural depth settings!

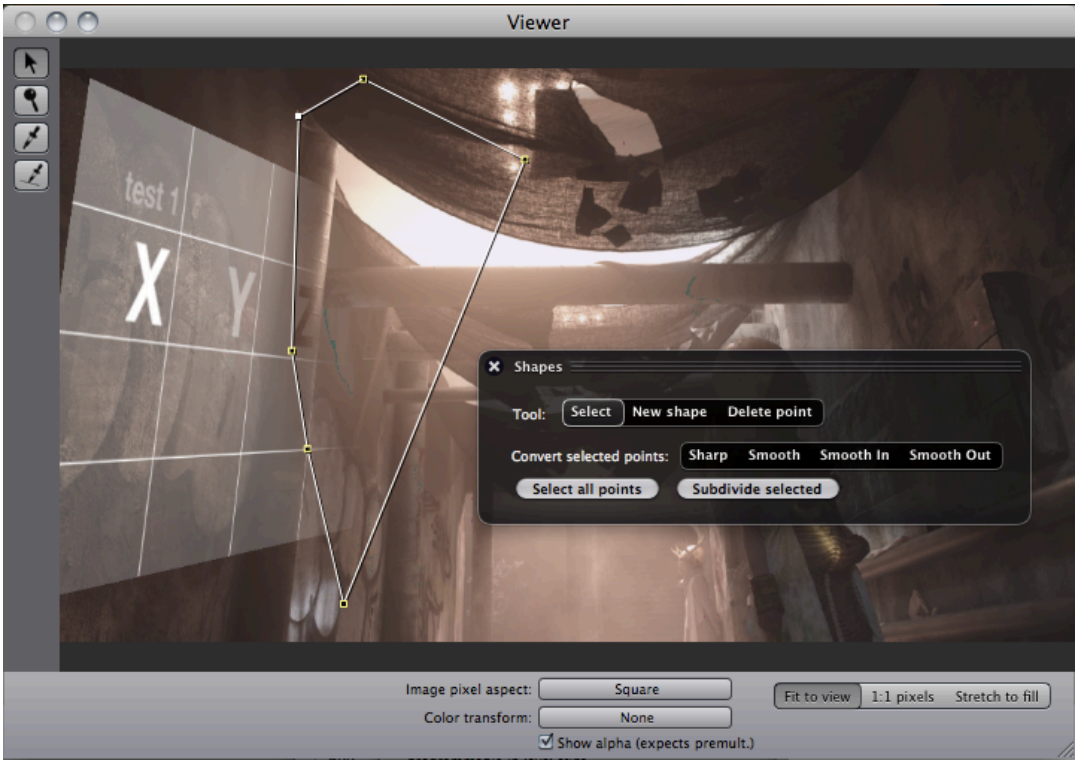
Next, let's apply a mask to the 3D layer. There is a node called *Shapes* that can be used to draw, well, shapes.

I drew a blob to mask out the right-hand side of the layer, then blurred it and used it as the mask to the 3D layer. To use it as a mask for the Over node, it needed to be converted to "scalar" (which simply means a single-channel image, i.e. a grayscale image – it stands to reason that a mask shouldn't have color):



In the Viewer, the result looks like the following screenshot.

Note that when the Shapes node is selected, its controls appear directly over the rendered image. This allows the shape to be edited in the proper context of the output, with the blur and masking applied at full quality – even when the shape points are dragged:



7. Exporting 3D and using batch automation

Once the project is done, you'll usually want to render it out to a file. In section 1, two different types of 3D video files were discussed: packed (single stream) and dual-stream. You can easily export either or both from Conduit 3D.

To render out dual streams, just connect the left and right eye streams to the Multi-Display node widget (as shown in section 1). Then, choose **Export Dual-Stream Movies** from the File menu. This will open the Export setup window, where you can pick what formats to export.

Once you've chosen the formats, you'll be prompted whether to create separate folders for the two streams. This is usually a good idea if you're exporting image sequences. (This way, each image sequence goes in its own folder and is marked by an *_L* or *_R* suffix to signify which eye it is.)

Alternatively, you can render out a video that contains the two eyes' images packed into a single stream. To create this kind of packed video, use the Stereo 3D Combine node that was discussed in section 2. Choose the packing type (e.g. side-by-side or interlaced) and connect the Stereo 3D Combine to the Multi-Display's first output. Then choose **Export** from the File menu.

You can also do automated batch exports in Conduit 3D. This is accomplished using actions that can modify many things within the project. They can be automatically created from a folder of files, or manually edited to perform specific actions. All the controls are located in the **Render Automation** window, in the Tools menu.

For more information on using batch actions with 3D, please see the section *Render Automation* in this book.

8. What's next? Advanced topics

There's a lot more in Conduit 3D. This guide can't hope to cover it all, but to give you an idea of what to look for, here are some topics and further links to explore:

Cue lists

Most parts of Conduit 3D can be automated. Using a feature called **Stage Tools**, you can build cues that perform combinations of commands such as loading video files, changing effect setups or animating sliders. These cues can be compiled into a cue list, which makes it very easy to perform series of actions in a live setting. This can be very useful e.g. during a shoot when previewing different 3D setups and effects.

For more information on cue lists, see *Live control and performance* in this book.

Custom effect nodes

The Conduit Editor is an easy way to create effect setups. But did you know you can package effect setups into supernodes that behave like single nodes? This is a practical way of customizing Conduit for your own workflows.

Once you've packaged an effect into a supernode, it can be compiled into a plugin and distributed to other users.

For more information, see *Custom rendering* section in this book.

Rendering stereo 3D graphics in JavaScript

Conduit can be scripted using JavaScript. The "language of the web" has always been easy to learn, but now it's also powerful enough to render realtime graphics! Conduit uses a JavaScript compiler that makes scripts run much faster than traditional interpreters.

There are two important JavaScript programming interfaces included in Conduit:

- Canvas for 2D graphics.
  This interface is also part of the upcoming HTML5 standard. It's simple and easy to pick up, and you can find lots of resources about it on the web.

- Surface for accelerated 3D graphics.
  This interface is custom-designed for Conduit. You can use it to access most features of Conduit's accelerated rendering engine. It's possible to mix Surface-based JavaScript rendering with effect setups designed visually in the Conduit Editor.

To render Canvas graphics within a Conduit 3D project, use the *Script*

Canvas 2D Effect node widget. To render Surface-based accelerated graphics, use the *Scripted Effect* node widget.

With scripting, the sky is the limit! There is an example script called *Frame Delay* included in the default installation. As the name suggests, it delays input video by a specified number of frames. It demonstrates some useful techniques for working with the Surface interface. (To use it, create a Scripted Effect node widget, click on "Edit Script", and choose Frame Delay from the "Favorites" button menu.)

For stereo 3D rendering, you can use two renderer node widgets together. Here's how:

Set the same script code for both. Inside the script program code, detect whether the left or right eye is being rendered (e.g. by checking the name of the node, or from an UI widget created by the script, or whatever mechanism suits you).

Modify the rendering accordingly. Even simple things can be very efficient in stereo. For example, within a Canvas 2D effect, you might have a background that you want to recede into the depth. Simply render it slightly offset to the right in the left-eye image, and offset the other way in the right-eye image. Now you have a stereo 3D image from 2D graphics.