FURNACE

User Guide for Furnace 4.0 for Shake

14th May 2008

The Foundry Plug-in Visual Effects

User Guide for Furnace 4.0 for Shake

Software engineering Ben Kent, Phil Parsonage, Simon Robinson, Lucy Hallpike, Bruno Nicoletti, Ralph McEntagart, Andrew Whitmore and Dan Alderman.

Algorithms Dr. Bill Collis, Dr. Anil Kokaram of Trinity College Dublin and Prof. Paul White of the University of Southampton.

Product testing Jack Binks and Jonathan Barson.

Writing and layout design Andrew Whitmore, Lucy Hallpike, Bruno Nicoletti, Simon Robinson, Phil Parsonage, Jack Binks, Pierre Wooldridge and Jonathan Barson using LaTeX

Proof reading Jack Binks and Jonathan Barson.

Contents

Introduction

Welcome to this User Guide for Furnace on Shake. Furnace is a rich collection of image processing tools to help compositors tackle common problems when working on films. We have spent many years working closely with post production houses in London developing tools that will save you time.

About this User Guide

This User Guide will tell you how to install and use the Furnace plug-ins. This guide assumes you are familiar with Shake and the machine it is running on.

What's New?

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

Example Images

Example images are provided for use with most of the plug-ins. You can download these images from our web site at www.thefoundry.co.uk and try Furnace out on them. (From the main page, go to Products > for Shake > Furnace, then click on the Examples link on the right hand side.)

Notation

In this User Guide we will refer to machines running Furnace and Shake as clients and machines that are running the Foundry FLEXlm Tools as servers.

Before Installing Furnace on Mac OS X

Please note that installing Furnace 4.0 on Mac OS X will effectively overwrite any previous versions of Furnace 4.0 that have been installed to the default Shake location. Scripts will, however, be compatible between Furnace 4.0 versions, and all other Furnace builds with different major and minor version numbers will remain intact.

To install multiple versions of Furnace please read the section on Customised Installation on page 20, or the section on Installing Furnace in the introduction of the Readme.pdf file that accompanies your Mac OS X installer.

Installing Furnace

Furnace is available as a download from our web site, www.thefoundry.co.uk. The downloads are in compressed tar format (tgz) for Linux, and dmg format for Mac OS X machines. Furnace should be installed on the client machines. The plug-ins are licensed with FLEXlm. We supply a suite of tools to manage and monitor floating licenses running on a server across a network of machines. These tools are called Foundry FLEXlm Tools (FFT) and can be downloaded free of charge from our web site. The Foundry FLEXlm Tools should be installed on the server.

Note **Commands in these instructions may be shown wrapped over more than one line, with subsequent lines being indented to indicate the continuation. Regardless, these should be typed on a single line. So**

```
a command that wraps around from one line
  to the next line
```

should be typed like this:

```
a command that wraps around from one line to the next line
```

Furnace on Linux

Follow these instructions if you wish to install Furnace on a Linux machine running Shake. Bear in mind that you are likely to need admin privileges, so you should probably log in to the terminal as root in order to do this.

1. Download the file from our web site (www.thefoundry.co.uk)
2. Move the download file to /usr/nreal

   ```
   mv Furnace4.0v4_shake4.10-linux-x86-release-32.tgz
     /usr/nreal
   ```

3. Change directory to /usr/nreal

   ```
   cd /usr/nreal
   ```

4. Extract the files from the archive using the command:

   ```
   tar xzvf Furnace4.0v4_shake4.10-linux-x86-release-32.tgz
   ```

5. A directory structure is created.

   ```
   /usr/nreal/Furnace4.0v4_shake4.10-linux-x86-release-32/
     include
   ```

   ```
   /usr/nreal/Furnace4.0v4_shake4.10-linux-x86-release-32/
     icons
   ```

```
/usr/nreal/Furnace4.0v4_shake4.10-linux-x86-release-32/
   docs
```

6. You need to tell Shake where to look for the plug-ins. There are several ways of doing this and you should refer to your Shake documentation. However, to use the Nothing Real environment variable method you should add the following environment variables to your .cshrc file using a text editor (vi). (If you're running bash rather than tcsh, you'll need to edit your .bashrc file and use the "export" command in place of "setenv" below.) These environment variables enable Shake to locate the plug-ins and icons.

   ```
   setenv NR_INCLUDE_PATH /usr/nreal/
      Furnace4.0v4_shake4.10-linux-x86-release-32/
      include
   ```

   ```
   setenv NR_ICON_PATH /usr/nreal/
      Furnace4.0v4_shake4.10-linux-x86-release-32/
      icons
   ```

   If you already have these variables set you can add to them by putting a colon between the paths, for example:

   ```
   setenv NR_INCLUDE_PATH /usr/nreal/
      Furnace4.0v4_shake4.10-linux-x86-release-32/
      include:/usr/nreal/
      Tinder2.0v1shake4.10-linux-x86-release-32/
      include
   ```

7. Proceed to "Licensing Furnace" on page 21.

Furnace on Mac OS X

Follow these instructions if you wish to install Furnace on a Mac OS X machine running Shake.

Default Installation

This will install our plug-ins into the Shake directory structure so that the plug-ins are picked up automatically. To have the flexibility of choosing the version of Furnace when running Shake use the "Customised Installation" method on the following page.

1. Download the file from our web site (www.thefoundry.co.uk).
2. Double click on the downloaded dmg file.

   ```
   Furnace4.0v4_shake4.10-mac-ppc-release-32.dmg
   ```

3. Double click on the pkg file that is created.

   ```
   Furnace4.0v4_shake4.10-mac-ppc-release-32.pkg
   ```

4. Follow the onscreen instructions to install Furnace directly into the Shake 4.10 directory.
5. Proceed to "Licensing Furnace" on the next page.

Customised Installation

This will install our plug-ins to a directory of your choice. This gives you the flexibility of choosing different versions of Furnace on starting Shake, but requires the use of environment variables to pick up the plug-ins. To get the plug-ins to work you will have to tell Shake where to find them using the NR_INCLUDE_PATH and NR_ICON_PATH environment variables.

1. Download the file from our web site (www.thefoundry.co.uk)
2. Create a directory into which the plug-ins will be installed. We recommend the following:

```
mkdir -p /Users/Shared/
  Furnace4.0v4_shake4.10-mac-ppc-release-32
```

3. Double click on the downloaded dmg file.

```
Furnace4.0v4_shake4.10-mac-ppc-release-32.dmg
```

4. Double click on the pkg file that is created.

```
Furnace4.0v4_shake4.10-mac-ppc-release-32.pkg
```

5. Follow the onscreen instructions. When you get to Select Destination click on Choose (not Continue) and browse to the directory you wish to install the plug-ins. For example, select /Users/Shared/Furnace4.0v4_shake4.10-mac-ppc-release-32.
6. Continue with the installation.
7. Quit.
8. Start a terminal and set the environment variables as follows:

```
setenv NR_INCLUDE_PATH /Users/Shared/
   Furnace4.0v4_shake4.10-mac-ppc-release-32/
   shake.app/Contents/PlugIns

setenv NR_ICON_PATH /Users/Shared/
   Furnace4.0v4_shake4.10-mac-ppc-release-32/
   shake.app/Contents/Resources/icons
```

9. Proceed to "Licensing Furnace" below.

Licensing Furnace

Furnace uses FLEXlm encryption in the license keys. For information on licensing Furnace, setting up a floating license server, adding new license keys and managing license usage across a network you should read the Foundry FLEXlm Tools (FFT) User Guide which can be downloaded from our web site.

Documentation

Furnace comes with two sets of documentation, this pdf and a set of html files which connect to the Help button in Shake. This button will work immediately if you have installed to the default location, however, if you installed elsewhere the software must know where to look for the documentation. If you've set the NR_INCLUDE_PATH environment variable, and the docs are installed alongside the plug-ins (as they are by default), the HTML documentation will be found automatically. If your plug-ins and documentation are stored separately, set the FOUNDRY_DOC_DIR environment variable to point the directory where the documentation is stored. If the documentation files can't be found at startup, you'll get a warning message printed to the console, telling you where the plugin has looked and failed to find the HTML. Multiple paths may be supplied, separated by a colon.

About Furnace Plug-ins

All Furnace plug-ins integrate seamlessly into Shake. They are applied to your clips as you would any other plug-in and they can all be animated using the standard animation tools. You can load Furnace plug-ins from the Furnace tab page as shown in Figure 1.1.

Figure 1.1 Furnace nodes

Nodes and Macros

A green dot in the corner of a plug-in's icon means that the plug-in contains at least one file-in node, and will prompt you to select an input clip when it is selected. Several plug-ins which require a window of frames from the input clip appear as both a normal node and one with a green dot, for example F_FrameRepair (Figure 1.2).

Figure 1.2 FrameRepair icons.

The windows of frames are there to allow us to work around the limitations of Shake's API, which does not allow temporal frame access during renders. The green dot nodes can simplify the work flow in some situations, as they will temporally offset the input frames and connect them to the appropriate inputs for you.

After you have created one of the Furnace green dot nodes, you can use the source group controls to change the input clip. Plug-ins that can have additional input clips will have one or more extra file-in groups, such as one for a matte. These extra groups will be bypassed by default when the node is created. To use them, specify a clip file in the group's parameters and switch off the bypass.

Network Rendering

A number of plug-ins in Furnace require cached data from previous frames to generate a new frame. Shake scripts containing these plug-ins will render correctly if executed on one machine, however, if the scripts are distributed for rendering across multiple machines on a network, then these cached frames will not be available to construct the the new frame and the render will fail. These network rendering restrictions apply to the following plug-ins or parameter settings.

- F_DeFlicker1.
- F_BlockTexture with **temporalConsistency** switched on.
- F_Shadow with **temporalSmoothing** switched on.
- F_Splicer with **temporalConsistency** switched on.
- F_SmartPlate

- F_SmartZoom
- F_MotionMatte with **rolling** switched on.

A blue dot in the top right corner of a plug-in's icon (as in Figure) means that the plug-in will not render correctly when distributed across multiple machines.

Figure 1.3 BlockTexture icon with blue dot.

Progress Bars

Some plug-ins now display a pop-up to indicate their progress (Shake 4.00 and later only). These can be disabled by setting the environment variable FOUNDRY_PROGRESS_BAR to 0.

Motion Vector Inputs

Many Furnace plug-ins, such as F_Kronos, rely on Local Motion Estimation, which is a per-pixel analysis of the motion between pairs of frames. Several of these plug-ins, including F_Kronos, now have optional motion vector inputs, to allow you to reuse the results from previous analyses of the motion. This is designed to save processing time, as local motion estimation is computationally intensive, so it makes sense to avoid doing these calculations more than once if possible. The new plug-in F_VectorGenerator allows you to do local motion estimation in isolation, so that the results can then be passed to other Furnace effects. The plug-in F_VectorConverter is also worth mentioning here. This allows you to convert between Furnace's vector format and other formats, to enable you to use external vectors in Furnace, or alternatively to export Furnace's vectors to other applications.

On-Screen Tools

Some plug-ins have their own on-screen tools for region selection or to facilitate the changing of various parameters. These inherit their colours from Shake's own on-screen tools, so you can change the colour of them in the same way as you would for the overlays in Shake.

All on-screen tools are transparent when not active. Parts of the widget that can be moved become solid when the mouse is

moved near to them. When selected, they are extended out to the edges of the viewing area.

The rectangular tools used to select a region - for sampling or analysis, for example - sometimes apply to a single frame only. In this case, the on-screen tool will have continuous lines on this frame, and dotted lines on all other frames. When a region is selected, the frame associated with it will be automatically updated to the current frame.

Running Older Versions of Furnace

From Furnace 4.0v1, the Furnace plug-in names have the version number appended to them. This means you can now run older versions of Furnace alongside the latest one.

Other Foundry Products

The Foundry is a leading developer of plug-in visual effects for film and video post production. Its products include Furnace, Tinder, Tinderbox, Keylight, Forge and Anvil and run on a variety of compositing platforms including Discreet Flame, Discreet Flint, Discreet Fire, Discreet Inferno and Discreet Smoke, Shake, Avid|DS, After Effects, Combustion, and a variety of OFX compliant hosts. For the full list of products and supported platforms see our web site www.thefoundry.co.uk

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

Anvil is a collection of colour correction and colour manipulation tools originally developed by First Art.

Keylight is an award winning 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.

Forge is a command line application that processes digital film scans and automatically detects and corrects flecks of dirt, dust and hair.

Visit The Foundry's web site at www.thefoundry.co.uk for further details.

Overview

Great - so you've just downloaded, installed, and licensed up your copy of Furnace, but where do you start? In this section we'll have a quick look at the various plug-ins, broken down by what we would envision their application to be. Please bear in mind, many of the tools offer much greater flexibility than we can cover in this manual, so have a good play around with them, and if you find a cool new use, or some tips or tricks you want to share, then let us know in our support forum (http://support.thefoundry.co.uk)!

Please select an overview from the list of sections below. When you see an individual plug-in that you'd like to find out a little more about, click on it and you will be taken to the plug-in reference section for that tool. Grain Management | Retiming | Dust Busting and Restoration | Clean Up, Touch Up and Removing In-Scene Objects | Tracking, Stabilisation and Alignment | Arbitrary Image Keying, Segmentation and Analysis | Roto Assist Tools | Grading | Clean Plate Generation.

Grain Management

Amongst the most commonly used, and well known, of the Furnace plug-ins are the grain management tools. These three plug-ins allow you to quickly remove and replicate grain patterns from both modern day fine-grained film stock and old, degraded, archive material. So why (and when) would you want to manage grain? In a whole host of different places; for example:

- When comping (or even editing) two shots together it is very obvious to the viewer when the grain characteristics vary; take, for example, comping a moving Computer Generated (CG) element against a still frame: the background still has static, unmoving grain, and the CG element will have no grain at all. In this circumstance you'd replicate the grain from the still frame and generate further frames so that the grain appears to be in motion. This would then be applied back against both the original still and the CG element, helping create the illusion that it was all shot in camera.
- When colour grading you'll find that as you begin to push the processing, the grain will be the first part of the image that clips and otherwise introduces undesirable artefacts. By removing it beforehand, then replicating it on the finished shot you'll save yourself a great deal of trouble.

DeGrain is a fast (spatial) grain suppression tool, useful for quickly cutting levels of grain on modern film stocks.

DeNoise is a temporal grain and noise removal tool, good for everything from modern stock through to archival footage and video noise.

ReGrain allows you to reapply grain to a source clip. A number of preset stocks are included, or you can sample a plate of your choice and have ReGrain create a statistically identical moving grain sample.

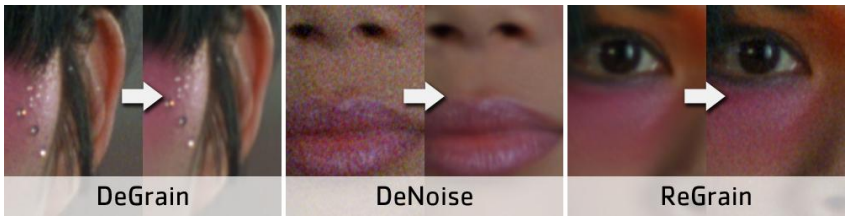


Figure 2.1 Grain management tools.

Retiming

Retiming is the process of altering the speed of a clip, to get either slow or fast motion. Historically this was a ugly process which introduced juddery or artefact-ridden output. By making a best-guess about where objects move within a scene (motion estimation) we are able to create frames between existing ones. Once we know how these objects move we are also able to add motion blur to these objects in motion as though it was shot in-camera.

Kronos is Furnace's retimer. It gives you full control over the speed of playback of your clip and allows you to add motion blur.

MotionBlur allows you to add true-to-life motion blur to objects within a scene, without the added complexity of the clip retiming controls.

Figure 2.2 Retiming tools.

Dustbusting and Restoration

Outside of the Grain Management mentioned earlier there are a number of other issues which plague both modern day film processing and archival restoration. Furnace contains a number of tools aimed at cutting down the amount of time you spend painstakingly hand painting out such issues (and even fix problems which are conventionally impossible to correct).

DirtRemoval repairs the objects that, when motion compensated, only appear on a single frame, and spatially appear to be dirt. The result is an unsupervised dustbusting powerhouse.

DeFlicker1 and DeFlicker2 are two different algorithms for dealing with film flicker from a variety of sources, be it poor lighting ballast right through to bad cranking on a 1920s film strip. Although there are no hard-and-fast rules - try using DeFlicker with flicker that does not originate from the original scene, e.g. ageing film, dust and chemical exposure. Give DeFlicker2 a go with in-scene flicker, poorly synchronised light rigs, stray light etc. If one doesn't quite give you the results you are after, try the other!

FrameRepair generates replacement frames when original frames are either fully or partially missing. Great for archive material, damaged tapes or missing 3d renders.

ScratchRepair does just what you'd expect from the name! Pop the widget where the scratch appears, and the defect will disappear whilst the film grain and remaining image detail is retained.

ChannelRepair rebuilds all or part of a single colour channel, using the information from the other colour channels.

ColourAlign automatically realigns the RGB colour channels on footage.

DeBlur Out of focus footage? Too much motion blur? Far more than a simple sharpen filter, DeBlur intelligently reverses the effects of blurring. You might like to try this in conjunction with DeGrain if too much grain is introduced in the result.

Steadiness provides automated 4-corner stabilisation of a sequence to help suppress any shakes or wobbles.

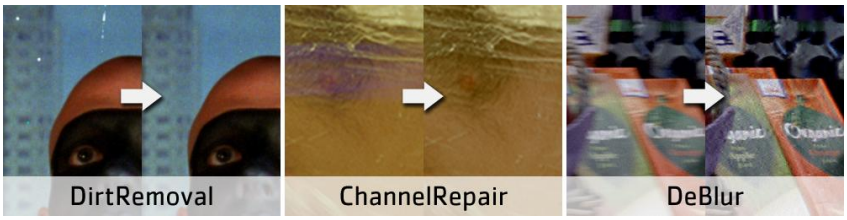


Figure 2.3 Restoration tools.

Clean Up, Touch Up and Removing In-Scene Objects

RigRemoval Got an object moving in relation to the background (or vice versa), that you want to get rid of? RigRemoval scans forward and backwards within a sequence to find an area where the background in question was unobstructed by the object, and then copies that back into place. Great for unwanted traffic, people, and more.

WireRemoval That perennial paint favourite; cloning out, frame-by-frame, the wires used for death defying stunts. Not anymore! WireRemoval can automatically track the movement of the wire and repair it using a variety of different repair types.

ShadowRemoval Suppress an excessive shadow using colour information from the shadow itself.

MotionSmooth Boiling matte, stop-motion or filtered footage? Try using MotionSmooth to blend motion compensated information together and help suppress such artefacts.

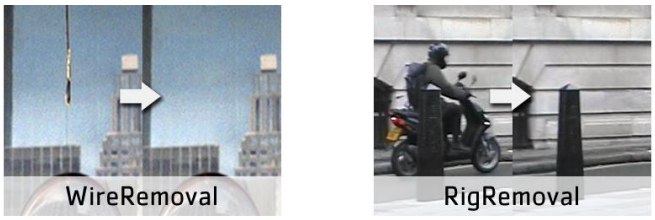


Figure 2.4 Clean-up tools.

Tracking, Stabilisation and Alignment

So you've got anything from a wobbly shot to a repair you want to stick over the defect? These tools are the answer.

Tracker An alternative to the Shake tracker, this robust tracker requires far less user tweaking and provides a concise workflow for correcting any errors seen.

Steadiness Suppress wobbles in handheld footage or lock the sequence position against a certain frame (great for when a small shake hits an otherwise locked off shot), Steadiness requires no tracking markers to be specified.

Align Lock one shot of a scene against another. Handy for anything from doubling up the crowd size by comping together two shots, to locking your freshly generated clean plate to the original.

MotionMatch Planar tracking embedded right in your host application. Facilitates screen replacements - without having to resort to an external application.

Figure 2.5 F_Tracker.

Arbitrary Image Keying, Segmentation and Analysis

So you want to separate an object from it's background, but you didn't blue-screen it? These tools should get you some of the way there.

ColourMatte By drawing a couple of relatively rough rotos around your object (one inside the object, and one outside) this will pull out tricky-to-matte edges such as fur, hair and feathers by using the relative colours of foreground against background.

MotionMatte will attempt to pull a rough matte from an object based on its motion. This can then be refined using the roto assist tools outlined in the next section. Given the intensive amount of processing that MotionMatte requires, we would recommend that trying this on a few representative frames of footage first. If it gives you good results on this small sub-set â then run the sequence through.

Depth extracts the relative depths of objects within a source sequence using their motion. Want to reduce depth of field? This should give you the matte to base your zblur on.

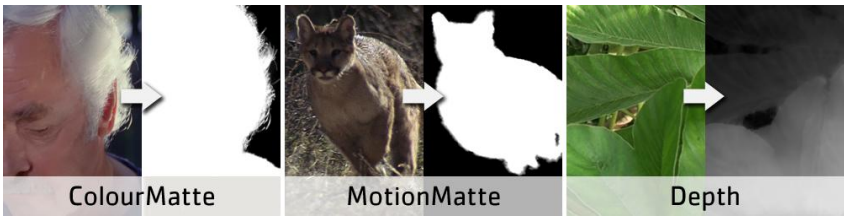


Figure 2.6 Keying, segmentation and analysis tools.

Roto Assist Tools

Don't have time for all that rotoing? These tools should help speed you up. All the rotoshape tools allow you to both load and save shape data, so you can easily pass the roto between the plug-ins.

RotoShape An alternative to the Shake rotoshape node without the the major shortcoming - i.e. the artefacted softness. As an extra plus it can load and save Autodesk FFFIS raw gmask files.

MatteToRoto The name says it all! If you've got a matte (e.g. generated from MotionMatte) and you want to tweak it, apply this plug-in and it'll convert it to a standard Shake rotoshape.

EdgeMatte automatically snaps a roughly drawn roto to the edges detected within an image.

RotoTracker tracks a rotoshape through a sequence, so you don't have to keyframe it every single frame.

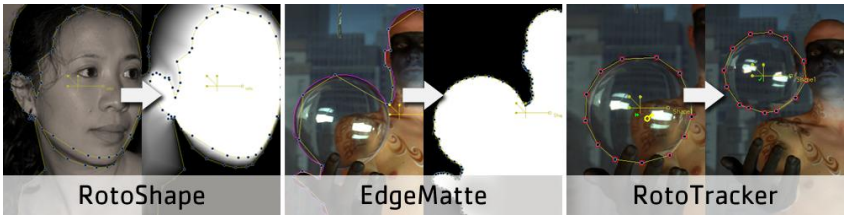


Figure 2.7 Roto tools.

Grading

Help speed up the grading process using these image-enhancing tools.

MatchGrade Got two shots from different times of day that need to be comped or edited together? This plug-in analyses the colour histogram of a reference image and automatically applies the result to your source sequence. This allows sequences that were shot with subtly different lighting to have the same 'look'.

Contrast Make that shot match the DoP's memory of how it looked on the day. This plug-in performs a spatially-dependant contrast adjustment that makes for a punchier, more vibrant image.

Figure 2.8 F_MatchGrade.

Texture Tools

Furnace also contains a number of tools designed for fixing problem areas and generating plausible image textures from small regions. These are useful for both image augmentation and 3D texture artists.

BlockTexture generates large-scale textures from a source image, such as distant crowds.

PixelTexture generates small-scale textures from a source image, which can be aligned with a rough colour image to match a particular formation. Good for flowers, pebbles and so forth.

Tile creates tileable textures from a source image. One for the 3D guys, but also good for tiling brickwork, planking, etc.

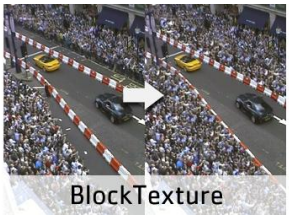


Figure 2.9 Crowd replication with F_BlockTexture.

Clean Plate Generation

Aside from the texture tools, there are a number of other plug-ins dedicated to helping you generate, and manipulate, still frames.

SmartFill is a hybrid texture tool, which in-paints a region specified by you - generating a realistic looking fill from other areas of the picture.

SmartPlate stitches together frames in a sequence to make one high-resolution plate.

Splicer joins two images across an arbitrary path by finding the most likely splice. Great on highly textured matte paintings.

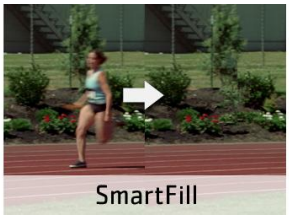


Figure 2.10 Clean plate generation with F_SmartFill.

Align

This chapter looks at how to register (line up) two separate but similar shots with F_Align. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_Align uses Global Motion Estimation (GME) to calculate a four corner pin so that each frame in one shot will be aligned with the corresponding frame in a second reference shot. For more of an overview of Global Motion Effects, and a description of the common way of working many of these effects have, please see the Global Motion Effects chapter on page 259.

F_Align is an Analysing Global Motion Effect, which can pre-analyse a sequence for global motion and store the calculated four corner pin as key framed parameters. This analysis is done in the user interface of the plug-in and any keyframes are used to align the clips in subsequent renders.

However, F_Align, unlike the other Analysing Global Motion Effects, can still do something useful during render without an analysis. If F_Align does not find a key framed pin on the frame being rendered, it will calculate the inter-shot alignment on the fly. The advantage of this is that you don't have to pre-analyse the complete sequence. The disadvantage is that you don't have direct access to the calculated corner pins and any re-rendering will be significantly more expensive, as the 'on the fly' calculations will have been lost and F_Align will have to analyse again.

If at any stage you modify the effect in such a way to invalidate the keyframed analysis (for example changing the accuracy parameter), a warning will be posted and the effect will analyse on-the-fly during render, ignoring the keyed analysis.

The analysis region is used to control which section of the reference frame is being matched to each source frame. Typically, leaving the region at its default is good enough; however, a heavy mis-match in foreground detail may make it necessary to change the region to a section that is shared between shots.

Quick Start

This section gives a very brief outline of how to use the plug-in.

Analysing On The Fly

1. Find two shots that are of the same scene, but have slightly different camera motion and foreground objects.
2. Create two file in nodes for these shots.
3. Create an F_Align node and attached the two shots.
4. The first shot will have been immediately repositioned so that it aligns to the second shot without any need for analysis.
   (a) You will see a banner in the overlay saying 'Note: the keyframed analysis is currently invalid. Analysing during render.'
   (b) depending on the exact difference between the two shots you may need to enable the 'scale' and/or the 'perspective' toggles to get a decent alignment,
   (c) you may also need to reposition the analysis box depending on the differences in foreground detail. However generally, leaving it at the default works well for most shots.
5. To see how closely the two clips have aligned, create an ISubA shake node (from the 'Layer' page of nodes) and wire into it the output of F_Align and the **second** input to F_Align. This will show you where the two clips differ.
6. That's it. You can now play or render the result out.

Pre-Analysing

1. Use the same two shots you used in 'Analysing On The Fly', and wire them up the same way.
2. Click on the 'analyse' push button.
   (a) F_Align will now start analysing each frame in the shot, figuring out the smoothing pin and writing it as keyframes to the corner pin parameters, cornerLL, cornerLR, cornerUL and cornerUR.
   (b) F_Align will update the time-line at each frame and you will see the aligned image render in the viewer.
   (c) If you want to interrupt analysis, either hit the ESC key or clip with the left mouse button. The pins it has keyed until that point will be retained.
3. Play or scrub through the aligned frames. The rendering will be faster as F_Align will no longer need to analyse on the fly.

(a) However, if you scrub to a frame where a corner pin has not been keyed, F_Align will re-analyse that frame on the fly.

4. That's it! You can now play or render the result out.

Inputs

F_Align has two input clips, the first **Source** input is moved so that each frame matches the second **Reference** input frame.

Parameters

Align shares all the Analysing Global Motion Effect parameters which are described in the Global Motion Effects chapter on page 259. It has no extra parameters of its own.

Examples

The images for the following examples can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Aligning Belle

In this example we make use of the two clips 'belleCentre' and 'belleLeft' and align one to the other.

1. Create two file in nodes, one for belleLeft.####.tif and one for belleRight.####.tif,

Figure 3.1 The belleLeft and belleRight clips.

2. Create an F_Align node and connect belleLeft to the first input and belleRight to the second input,

(a) belleLeft will immediately be aligned with belleCentre, you will see the image jump immediately to the left and slightly up

3. Disable the **rotation** toggle.
   (a) due to the nature of this shot (the back drop undulates slightly and the slight amount of perspective difference) a better alignment is acheived if we disable rotations
4. Create an ISubA node and wire the output of F_Align and BelleCentre into it.
   (a) this will show you how well the two clips align.

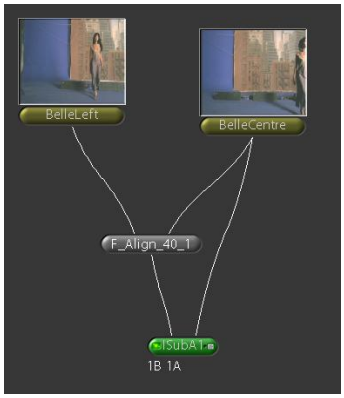


Figure 3.2 The tree aligning the two shots of Belle.

5. You now have aligned belleLeft with bellCentre. Render her out, or scrub through the shot to see the results.

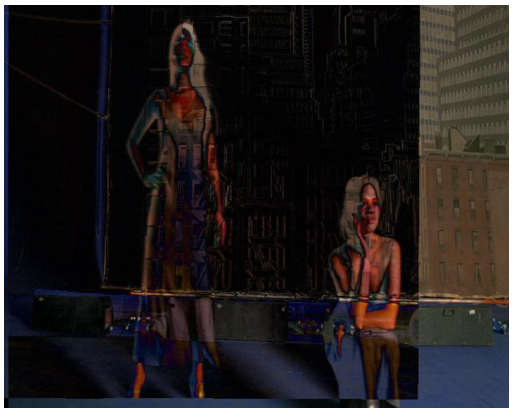


Figure 3.3 The difference between the aligned 'belleLeft' and the 'belleRight' clips.

Cloning Belle

In this example, we take the two belle clips from the last example and 'clone' her so that she appears twice in the same shot. We do that by keying the aligned BelleLeft shot over the BelleCentre shot.

1. Create the tree that was used in the example 'Aligning Belle', and analyse the whole 100 frames by clicking on **analyse**.

2. Add a SwitchMatte node and wire the output of the F_Align node into the first input. Set the SwitchMatte's clipMode param to **foreground**.

3. Create a RotoShape node and wire that into the second input of the SwitchMatte node.

4. Create an Over node and wire the output of the SwitchMatte node into first input and the original belleCentre clip into the second input.

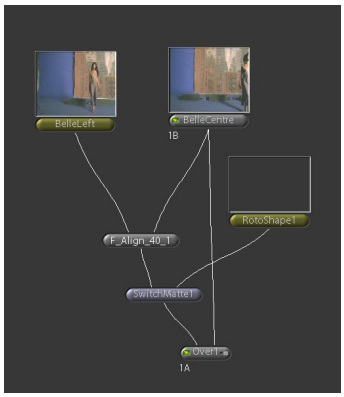


Figure 3.4 The Shake tree to clone Belle.

5. We now need to create a roto shape to put into the aligned clip so that it keys over the original. To start that,

   (a) view the output of F_Align and edit the Rotoshape then scrub to frame 1,

   (b) enable keyframing in the viewer,

   (c) draw a roto shape around Belle so that it just encloses her right hand side and extends all the way to the left of the shot,

   (d) make the right edge of the roto shape soft by dragging the edge points away slightly,

   (e) now view the output of the Over node, whilst still editing the rotoshape, this should show you the aligned BelleLeft shot keyed over the BelleCentre.

Figure 3.5 Rotoing BelleLeft onto BelleRight.

6. Now scrub through the timeline and add keys to the rotoshape every so often so that BelleLeft is cleanly composited over BelleRight.
7. Ta-da! You have now made a single shot where Belle and her clone appear together. You will notice a few problems as you go that are not completely addressable without a bit of extra work.
   (a) there is a slight difference between the colour of the two shots, which needs a bit of colour correcting to overcome (F_MatchGrade will help here),
   (b) BelleCentre's shadow is being cut off by the keyed BelleLeft. If limiting the shot to the area of the backdrop, this is not a problem. If not, you can improve this a bit by tweaking the rotoshape you are compositing through, however to make it properly clean will involve some degree of painting of the matte to get a cleaner edge than the roto shape,
   (c) the aligned plate has black being keyed in at the bottom, this could be got around with a bit of better rotoshaping or a bit of painting.

Figure 3.6 Belle and her Clone

BlockTexture

Introduction

Furnace includes a number of texture generation plug-ins. These are used to create resolution independent images that have the same look and feel as a small sample region from a source image. F_BlockTexture is one such plug-in and should be used for replicating fairly large scale textures. By that we mean textures that contain shapes rather than more uniform textures that would be suited to F_PixelTexture.

F_BlockTexture works by taking blocks of data from the original image and rearranging them in a random fashion in such a way that the joins between the blocks are invisible and look plausible. The size of the blocks depends on the scale of the texture being replicated and the overlap of the blocks controls how well the joins between blocks are hidden. This plug-in tends to generate the best results on more natural looking textures, for example, leaves blowing in the wind or even crowds of people.

F_BlockTexture also has the ability to generate temporally consistent frames of textures but only over small motion ranges. Note that network rendering is not possible on a script containing F_BlockTexture with **temporalConsistency** switched on.

Figure 4.1 Original image showing the river Thames in London.

Figure 4.2 BlockTexture used to generate moving water texture.

Quick Start

The plug-in has three texture inputs, a source input and a matte input. The texture inputs should be connected to the image or images whose texture you wish to sample. The source input will have the generated texture applied to it. If no texture inputs are connected, the texture is both sampled from and applied to the source. The matte is an optional input that

allows you to specify a region in the source image that you wish to fill with the new texture. This region can be blended with the original image to produce a seamless fill.

Figure 4.3 The original image is shown on the left. This image has recognisable shapes and is more suited to F_BlockTexture. F_BlockTexture has been used to generate the image in the middle which shows good shape retention. For comparison, F_PixelTexture has been used on the right.

If you have more than one region in the matte, they must be connected together by a thin line. The example below shows the matte used to add crowds to two new sections of a football stadium. Note that the regions need to be joined or only one region will be filled.

Figure 4.4 Incorrect.

Figure 4.5 Correct.

After connecting the texture inputs use the on screen **bounds** selection tool to define the area that will be sampled. A larger source region will mean less noticeable repetitiveness in the new texture.

Alternatively, if you have supplied a matte input of the region you wish to fill, selecting **avoidMatte** will copy texture from any regions not specified by the matte. Make sure **fillScreen** is switched off though.

If a matte input is connected, texture will now be copied into the source image but only where there is full alpha in the matte. If there is no matte, switching on **fillScreen** will generate a complete frame of new texture. Adjust **blockSize** to

find a suitable sized building block to generate the new texture. **Overlap** determines the amount adjacent blocks are overlapped. A greater overlap gives a better chance of hiding the edge join. If the edges are still visible increasing **blendSize** will blur the join, but too much blurring will result in a noticeable soft edge to the blocks. Increasing **samples** will increase the quality of the fit between blocks, but at the expense of processing time. The whole process is seeded by a random number so changing the seed will give you alternative texture fills, some of which may be more pleasing than others.

If you are using a matte input to specify the region to be filled, **pathWidth** specifies the quality of the join between the texture region and the source image. Similarly **edgeBlend** specifies the extent to which this join is blurred. If there is luminance change across the region being filled then switching on **luminanceMatch** will attempt to align the luminance between the new region and original image. **LuminanceBlur** controls the extent of this matching process.

Finally, if you are attempting to generate temporal textures switch on **temporalConsistency** before rendering an output sequence. This will ensure the new texture is generated in a consistent manner on successive frames.

Crowds

F_BlockTexture can also be used for crowd replication. In Figure 4.6, the original image is a locked off shot of a small crowd filling only part of a stadium. During the sequence the crowd stand up and wave flags. By drawing a matte to specifiy where we want to put new crowd, F_BlockTexture can be used to split up the crowd into plausible jigsaw pieces and reassemble them in the locations defined by the matte. This plug-in works temporally, so that the new crowd also stand up but at different times to the original.

Note **You should note that with temporalConsistency switched on the sample texture will be taken from the sample region on the start frame only. Any changes to the size or position of the sample region over the sequence will be ignored by the algorithm. The algorithm also assumes the matte is unchanging.**

Figure 4.7 on the next page and Figure 4.8 on the following page shown another example of crowd replication. You can try this shot for yourself. See the Examples in section4.

Figure 4.6 Crowd Replication using F_BlockTexture. Images courtesy of The Moving Picture Company from the film Mike Bassett: England Manager 2001 Film Council, Hallmark Entertainment and Entertainment Film Distributors

Figure 4.7 Before.

Figure 4.8 After.

Inputs

F_BlockTexture has five inputs. The first input (source) defines the size of the texture that will be generated. If it is the only input the texture will also be sampled from it. The second, third and fourth inputs (texture), if connected, are sampled for texture.

Tip **If you have just one image from which to sample texture, connect this to texture1 then rotate it and connect to texture2 then scale and connect to texture3. By providing additional inputs you will reduce repetition in the generated texture.**

The fifth input (matte) defines the area that will be filled with the generated texture. This is then composited over the source input.

Parameters

The parameters for this plug-in are described below.

fill - defines the area to fill with the generated texture

- **Full Frame** - fill the whole of the destination image, which will be the same size as the source image.
- **Source Alpha** - fill the region defined by the alpha of the source.
- **Source Inverted Alpha** - fill the region defined by the inverted alpha of the source.
- **Matte Luminance** - fill the region defined by the luminance of the matte.
- **Matte Alpha** - fill the region defined by the alpha of the matte.
- **Matte Inverted Luminance** - fill the region defined by the inverted luminance of the matte.
- **Matte Inverted Alpha** - fill the region defined by the inverted alpha of the matte.

blockSize - The size of the building blocks used to make the new texture.

overlap - The amount the building blocks are overlap. A larger overlap is more likely to generate a better edge join.

blendSize - The amount of blur at the edges between building blocks. A small amount is good to help hide the edges on some images but too much will result in an obvious softness.

samples - The number of different blocks that are tried before choosing the best match. The higher the number the better the match but the longer the render time.

seed - The random number used to start the texture generation process. Select another seed to see a different texture pattern.

pathWidth - Sets the width of the join between the source image and the new texture. The larger the path the better the quality of join.

edgeBlend - The amount of blur between the source image and the new texture. A small amount is good to help hide the edges but too much will result in an obvious softness.

avoidMatte - Switch this on to sample texture from the region that is not specified by the matte as opposed to the region specified by sampleRegion.

temporalConsistency - Switch this on to force temporal consistency in the new texture between successive frames. Note that the sample region and the matte that the texture will be pasted into, will be taken from the first frame. Any changes to

the sample region or matte on subsequent frame will be ignored by the algorithm if temporalConsistency is switched on.

Warning! **Shake scripts containing F_BlockTexture with temporalConsistency switched on cannot be distributed across multiple machines on a network render farm. See "Network Rendering" on page 22.**

luminanceMatch - Switch this on to match the luminance of the new texture with the luminance of the source region in order to help hide the join.

luminanceBlur - Controls how much effect **luminanceMatch** has on the new texture.

bounds - Defines the sample area that will be used to generate the texture.

boundsMin - Sets the bottom left of the bounds.

boundsMax - Sets the top right of the bounds.

Examples

All the images for the following examples can be downloaded from our web site. For more information, please see the Example Images section on page 17. There are also some sample textures that work well with F_BlockTexture in the Textures directory.

Hedge

In this example, we'll take the flickery clip of the hedge and try to extend its height. There are, of course, lots of ways of doing this, but F_BlockTexture should save you some time.

Figure 4.9 Hedge.

Figure 4.10 Reconstruction.

Step by Step

1. FileIn the hedge.####.tif
2. Connect the hedge to the first (source) input of F_BlockTexture.

3. Connect a QuickShape into the last (matte) input of F_BlockTexture and draw a matte around the top of the hedge. This will define the area we wish to replace with generated texture.

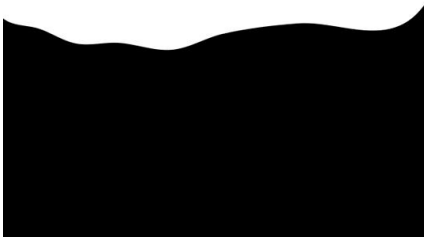


Figure 4.11 QuickShape.

4. Position the box sample area. The image in this box will be sampled and used to generate new texture.

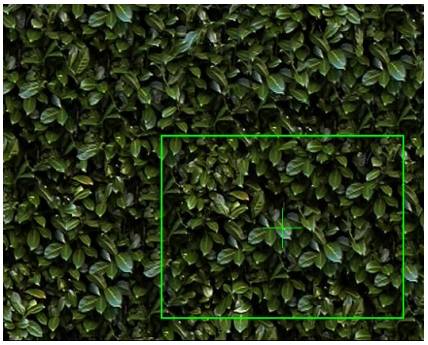


Figure 4.12 Sample Box.

5. Make sure temporalConsistency is switched on and render.

Crowd Replication

In this example, we'll add in some crowds to this shot of a Formula 1 Parade on Regents Street in London. The result is not perfect but a good start. To improve it you could add some waving flags to break up the join with the barricade and possibly some camera flashes to cover some of the repititions.

Figure 4.13 Before.

Figure 4.14 After.

Step by Step

1. Start Shake and FileIn F1Parade.####.tif, F1ParadeMatte1.tif and F1ParadeMatte2.tif.
2. Build the tree shown in Figure 4.15.

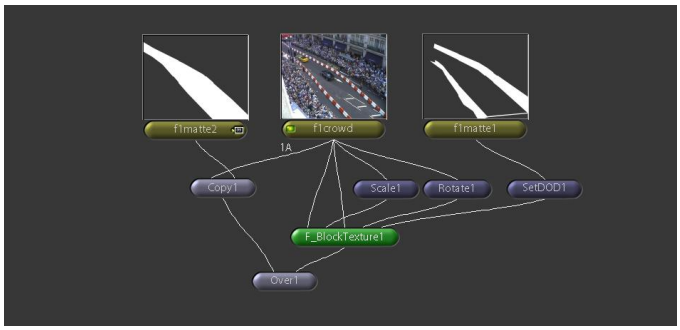


Figure 4.15 F_BlockTexture tree.

3. The crowds are fed into the first and third inputs. For the second input scale the crowd (xScale = 0.9) and feed this in. For the fourth input rotate the crowd (angle = -10). The fifth input defines the area that will be filled with generated crowd texture.
4. In F_BlockTexture set the **bounds** parameters as follows: **boundsMin**=71.65,212.95, **boundMax**=141.80,316.47. And to improve the edges of the block tiles increase the blendSize to 1.
5. Since the edge of the crowd overlaps the track you need to fix this by recompositing the original image using f1matte2.tif.
6. It's worth trying different positions for the sourceRegion. You could even chain together multiple F_BlockTexture nodes with different sample areas.
7. That's it.

ChannelRepair

This chapter looks at repairing single colour channel damage using Furnace's plug-in F_ChannelRepair. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

Occasionally damage can occur in only one or two colour channels in an image, for example, a scratch may be only two channels deep or chemical damage may only affect the top layer of the film. In these cases F_ChannelRepair can be used to repair the damaged channel(s) by using information from the other colour channels.

Figure 5.1 Blue channel damage.

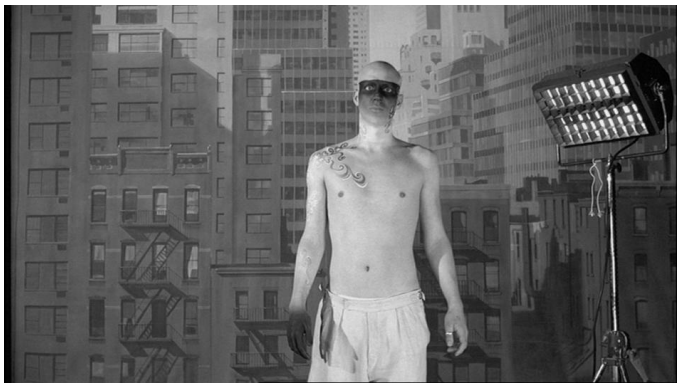


Figure 5.2 Undamaged green channel.

The first spatial repair algorithm, **Blend**, tries to match the statistics of the reference channel to the undamaged area of the damaged channel. It then simply blends the matched pixels of the reference channel to fix the damage. The second algorithm, **ReMatch**, is more complex and attempts to adjust the local statistics of the proposed **Blend** repair to those in the

undamaged area. It is entirely dependent on the footage as to which algorithm gives the best result.

It is possible to tune the **ReMatch** algorithm to improve the repair quality. F_ChannelRepair uses a block based approach to model the damage in the image. The clean and damaged channels are split into multiple blocks of size **blockSize**. We then try and modify the local statistics of the damaged block to make it equal to the local statistics of the clean block. In practice, if there is motion in the sequence the pixels in the two blocks will not be the same, which will result in incorrect estimation of the local statistics. To minimise this effect, when the image is divided into blocks they are overlapped by a small region. To this region we apply the statistics calculated for both adjacent blocks. If the results from these two blocks are not consistent we assume that the parameters have been calculated incorrectly due to motion and we discard them both, replacing them with the global statistics for the whole damaged region. The criterion for discarding parameters is set by **motionThreshold**. So, if you think the damage is only in parts of the image that are static, it is quite safe to increase this value. If there is little information in the block, for example in a plain area, it is difficult to obtain reliable estimates for the statistics. We check for this by discarding the parameters for any blocks with a variance below **weightThreshold**. Having obtained a reliable set of parameters for each block these are then smoothed using a combing technique. The number of smoothing iterations is set by **smoothingIterations**.

Quick Start

Connect the damaged frame or sequence to the first (**source**) input of F_ChannelRepair. Create a matte of the damaged region and connect to the second (**matte**) input. Set **repairChannel** to the damaged channel and **useChannel** to the channel from which you wish to take information to make the repair. Set the **matchMethod** to **Blend**.

Adjust **weightThreshold**, **blockSize**, **motionThreshold** and **smoothingIterations** as described in the parameter section to tune the output if necessary.

Inputs

F_ChannelRepair has two inputs. The first input (**source**) contains the damaged frame or sequence. The second input (**matte**) should contain a frame or sequence that specifies where the damage occurs. Use **the matteComponent** parameter to define which areas are repaired, by default the

Figure 5.3 Repaired image.

Luminance of the **matte** is used.

Parameters

The parameters for this plug-in are described below.

repair - defines the area to repair.

- **Full Frame** - fill the whole of the source image.
- **Source Alpha** - repair the region defined by the alpha of the source.
- **Source Inverted Alpha** - repair the region defined by the inverted alpha of the source.
- **Matte Luminance** - repair the region defined by the luminance of the fillMatte.
- **Matte Alpha** - repair the region defined by the alpha of the fillMatte.
- **Matte Inverted Luminance** - repair the region defined by the inverted luminance of the fillMatte.
- **Matte Inverted Alpha** - repair the region defined by the inverted alpha of the fillMatte.

useChannel - selects the undamaged channel of the image from which to take information to make the repair.

repairChannel - Selects the damaged channel of the image to be repaired.

matchMethod - There are two possible techniques to take information from one channel and apply it to another.

- **Blend** - Feathers the duplicated pixels from **useChannel** into **repairChannel**.

- **ReMatch** - Attempts to make the local statistics between **useChannel** and **repairChannel** identical.

The following four parameters apply only to the **ReMatch** method.

weightThreshold - Sets the threshold value for plain areas above which blocks are discarded due to the lack of reliable data. If your results are poor, it's worth decreasing this value.

blockSize - Sets the size of the blocks that are analysed. Smaller blocks will allow more complex damage to be modeled but will produce less accurate and robust results, and will take longer to render.

motionThreshold - Sets the threshold value above which localised damage is abandoned because of motion in the frame. If your results are poor and you can see that there is little motion in the sequence, you should increase this value. Setting the motionThresh to 1 will turn off localized damage repair for all moving objects and apply a global algorithm to reduce the damage.

smoothingIterations - Sets the number of times the damage parameters are smoothed between blocks.

ColourAlign

This chapter looks at colour channel registration (alignment) using Furnace's plug-in F_ColourAlign. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

Occasionally due to damage to the film or errors in the scanning process the red, green and blue channels can become misaligned. F_ColourAlign will automatically repair this damage by aligning the channels with a reference channel of your choice.

This type of colour damage is often particularly obvious when restoring old three-strip Technicolour® type footage.

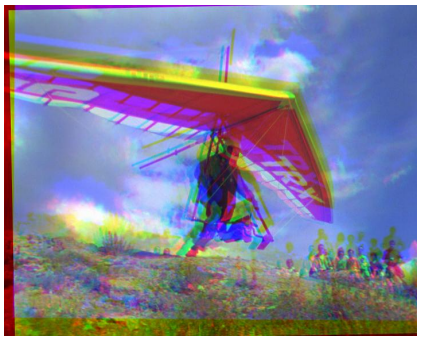


Figure 6.1 Mis-aligned.

Figure 6.2 Aligned.

Quick Start

Connect the damaged frame or sequence to F_ColourAlign. Select the reference channel that you wish the other channels to be aligned to and view the result.

Normally the transformation causing the misalignment can be modelled by a simple translation. In more complex situations it may be necessary to calculate rotation, scale and perspective changes by activating the relevant controls.

The quality of the interpolation of the misaligned channels can be modified by adjusting the filtering parameter. Low is the quickest but the image might look soft, aliased or discretised. Medium uses a bilinear filter, and high uses a sinc filter for the best results but at the expense of speed.

If the misaligned channels are noisy, it will be more difficult to estimate the transformation required to align them and this may adversely affect the results. The **noise** controls refer to

an optional median filter that can be applied per channel to reduce the noise on the reference images used internally to do the alignment. This can improve the results of the alignment and will not affect the noise on the output image.

Inputs

F_ColourAlign has two inputs, the first of which is the sequence to be aligned. The second is a **reference** clip that is used to calculate the transformation that is applied to the first clip. This means a denoised version of the input can be used to calculate the transformation to be applied to the original clip.

Parameters

The parameters for this plug-in are described below.

reference - The number colour channel to be used as a reference. The other colour channels will be aligned to this channel.

rotate - Switch this on if the misalignment transformation between the colour channels contains a rotation.

translate - Switch this on if the misalignment between the colour channels contains a translation.

filtering - Sets the filtering quality

- **Low** - low quality but quick to render
- **Medium** - uses a bilinear filter. This gives good results and is quicker to render than high filtering
- **High** - uses a sinc filter to interpolate pixels giving a sharper repair. This gives the best results but takes longer to process.

+noise - The parameters to control denoising of the images before alignment.

noiseRed - Median size used to denoise the red channel internally. Noise can sometimes cause incorrect results in channel registration. This median is applied to a reference clip so that the output will not have the median applied.

noiseGreen - Median size used to denoise the red channel internally.

noiseBlue - Median size used to denoise the blue channel internally.

+sampleRegion - The region of the image that the plug-in attempts to align between the colour channels.

sampleRegionMin - Set the bottom left of the sampleRegion

sampleRegionMax - Set the top right of the sampleRegion

+advanced - The lesser used refinement controls.

scale - Switch this on if there are changes in scale between the colour channels.

perspective - Switch this on if there are changes in perspective between the colour channels.

accuracy - It is rarely necessary to find the optimum transform for all pixels and so, to speed up the process, the accuracy can be reduced. High values will increase the processing time but may give you better alignment.

Examples

The footage used in the following example can be downloaded from our website. For more information, please see the Example Images section on page 17.

BelleColourAlign

1. FileIn BelleColourAlign.####.tif, and apply F_ColourAlign. The result is poor on the default settings, as in Figure 6.3, but we can fix that.

Figure 6.3 Poor result, caused by noise in the blue channel.

2. Look at the blue channel - there is a large amount of noise preventing the blue channel from aligning. Set the noiseBlue parameter to 1 in order to get rid of some of this noise for analysis, and you should get a result as in Figure 6.4. If you want to use an external noise rejection algorithm, apply this node to the clip and connect the noise reduced version into the second input.
3. Render the entire sequence to a flipbook.

Figure 6.4 Better result, after reducing the blue noise.

ColourMatte

This chapter looks at keying off arbitrary backgrounds using Furnace's plug-in F_ColourMatte. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_ColourMatte is designed to help an artist pull a matte of an object that has a soft edge, for example, fur, hair, or motion blurred objects that can appear on top of any background. The artist supplies a set of mattes (either as separate clips or using the internal roto tools) to define the foreground object to be keyed, and F_ColourMatte will refine these mattes to an accurate alpha.

Background

The image in Figure 7.1 is a typical example of an input to F_ColourMatte. The aim is to accurately build an alpha channel of the man in the foreground.

Figure 7.1 ColourMatte can be used to extract a matte from images like this.

Consider the compositing equation:

C=alpha*F+(1-alpha)*B

This describes how a pixel of rgb value C is composed of a foreground colour F, scaled by alpha and background colour B, scaled by 1-alpha. The aim of this plug-in is given C to deduce values for F,B and alpha. Obviously this is a very tricky problem

as for every three known variables (the red, green, and blue values of C), we must estimate seven unknown variables (the red, green and blue values of F and B, together with the value of alpha).

In order to solve this problem we require the artist to first generate a tri-map. A tri-map is a set of mattes that divides the image into three regions. The first region marks part of the image that is known to be foreground (alpha=1), see Figure 7.2. The second defines the part that is known to be background (alpha=0), see Figure 7.3. The remaining part of the picture, the unknown region, is the area over which we must calculate F, B and alpha. The tighter these foreground and background mattes can be painted the better, and faster the results.

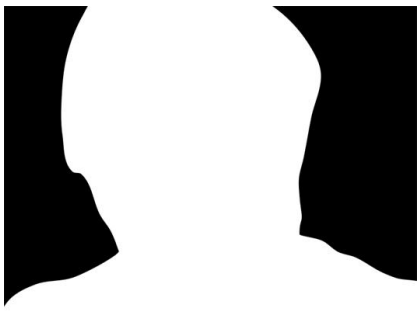


Figure 7.2 Foreground.

Figure 7.3 Background.

The algorithm works by taking pixels from the known foreground and background regions and trying them at the current pixel site. The colours that best fit the compositing equation whilst ensuring some form of smoothness to the foreground, background and alpha are chosen. These are then forced to fit the compositing equation and the process repeated in an iterative manner. Hence, it should be noted that if a colour close to the correct foreground and background do not exist anywhere nearby in the known foreground and background regions the algorithm is likely to fail.

If a clean background plate is available, this can be supplied as an input. In this case the number of unknowns are reduced dramatically leading to better results but it is important that the background plate is almost identical in colour and alignment to the background of the source image. If necessary, the clip should first be registered using F_Align.

The outputs of the plug-in are the unpremultiplied foreground (F), the alpha (a), and, optionally, the unpremultiplied background (B). Providing F and alpha separately means it is a trivial matter to composite the foreground into a different picture (see Figure 7.4).

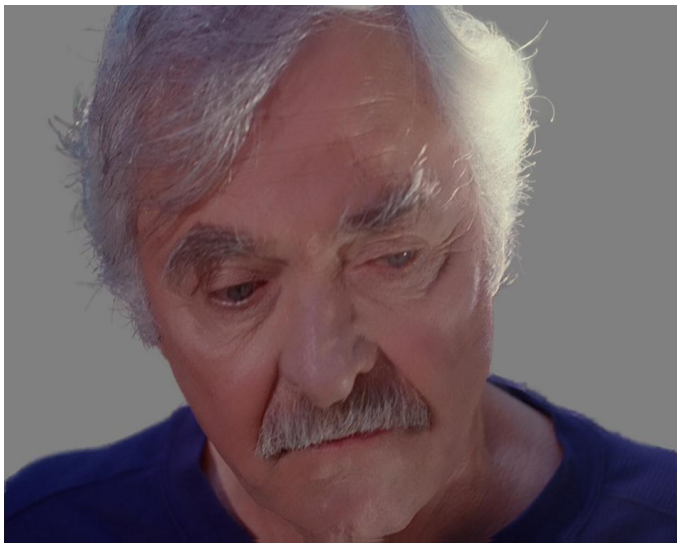


Figure 7.4 Foreground composited over mid grey using matte generated from F_ColourMatte.

Quick Start

Connect the source frame to the **source** input. F_ColourMatte has an internal roto tool, which is the same as that in F_RotoShape. View the source image and select the parameters for F_ColourMatte. Choose **Roto** from the **shapeSource** menu. While still viewing the source, draw a roto around the region in the image that is definitely foreground (alpha=1). Select the roto's edge points and draw an outer roto around the area that is definitely background (alpha=0). Alternatively, if you have them, you can supply mattes of these areas to the **foregroundMask** and **backgroundMask** inputs respectively.

The unpremultiplied foreground of the source frame should be displayed. View the alpha channel to see the alpha matte of the image. Try compositing this over a plain background to see the quality of the segmentation.

Try different **colourSelection** and **refinement** algorithms to improve the results. Also try increasing the number of **refinementPasses**.

Ideally the foregroundMask and backgroundMask input mattes should be drawn as tight as possible to the unknown region. If they are fairly loose it is worth running a few **matteOptimiser** passes to refine the mattes before calculation of the alpha. This should dramatically speed up the algorithm. To do this, increase the **levels** parameter under **matteOptimiser**.

Inputs

F_ColourMatte has three inputs. **Source** is the source image, **foregroundMask** is a matte specifying those regions which are definitely foreground pixels (alpha=1) and **backgroundMask** is a matte specifying those regions which are definitely background pixels (alpha=0). Optionally, a clean background plate, **knownBackground**, can also be supplied.

Parameters

The parameters for this plug-in are described below.

render - displays either the unpremultiplied foreground or the background. The alpha is automatically associated with the foreground.

- **Extracted Foreground**
- **Extracted Background**

shapeSource - where to get the foreground and background mattes from

- **Mask Clips** get the mattes from the inputs.
- **Roto** use the internal roto tools to define the foreground and background.

colourSelection - three possible techniques are used to choose initial guesses of the foreground and background colours at an unknown pixel

- **useAnyColours** chooses a pixel at random from the nearby foreground region for the unknown foreground colour and a pixel at random from the nearby background region for the unknown background colour.
- **useSimilarColours** forces a foreground colour to be chosen that is similar to the combined colour C.
- **useDiffuseColours** chooses pixels for the foreground and background by diffusing the colours in from the known foreground and background regions into the unknown region. This results in much smoother more consistent foreground and background colour guesses.

refinement - having chosen an initial guess for the foreground and background colours there are two possible refinement methods. errorMinimising is more suitable for more complex, rapidly changing alpha's and detailMatching is better suited to smoother alphas.

- **errorMinimising** takes the initial values for F and B and solves the compositing equation for alpha. An iterative process of refinement follows whereby the colours are modified in an attempt to get a better fit to the compositing equation whilst ensuring the foreground, background and alpha remain consistent and edges in the alpha correspond to edges in the original image.
- **detailMatching** takes the initial values for F and B and forces them to fit a system of equations that combines knowledge of both the compositing equation and structure in the image.

refinementPasses - the number of refinement iterations that take place. The greater the number, the better the output but the longer the processing time.

edgeErrorWeighting - how strictly edges in the alpha must correspond to edges in the input.

inputMasks - sets the mask channel to use.

- **useLuminance**
- **useAlpha**

+matteOptimiser - if the known foreground and background mattes are quite loose it is possible to use the matte optimiser to try and refine these mattes before the algorithm spends a large amount of time calculating alpha. This is achieved by running a number of crude passes of the algorithm that attempt to shrink the region of unknown alpha. At each pass any pixel with an alpha value of within levelsThreshold of 0 or 1 is set to 0 and 1 respectively.

levels - the number of iterations of matte optimisation. More levels will take longer but should produce tighter results.

levelsThreshold - any pixel with an alpha value within levelsThreshold of 0 and 1 is set to 0 and 1. Increasing levelsThreshold will shrink the size of the unknown region faster but is more likely to incorrectly set a soft alpha to 0 or 1.

foregroundMaskComponent - which component of the foregroundMask clip to use.

- **Alpha** - use the alpha channel of the foregroundMask clip.
- **Inverted Alpha** - use the inverse of the alpha channel of the foregroundMask clip.

- **Luminance** – use the luminance of the foregroundMask clip.
- **Inverted Luminance** – use the inverse of the luminance of the of the foregroundMask clip.

backgroundMaskComponent – which component of the backgroundMask clip to use.

- **Alpha** – use the alpha channel of the backgroundMask clip.
- **Inverted Alpha** – use the inverse of the alpha channel of the backgroundMask clip.
- **Luminance** – use the luminance of the backgroundMask clip.
- **Inverted Luminance** – use the inverse of the luminance of the of the backgroundMask clip.

Contrast

This chapter looks at image enhancement using F_Contrast. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_Contrast allows the user to extract hidden details from dark and bright areas of an image. Based on the adaptive contrast enhancement algorithm in Apical's Iridix product, F_Contrast boosts dark areas to give more balanced, memory-true images.

Most contrast enhancement (dynamic range compression) technologies are fixed and uniform, for example gamma correction.

Figure 8.1 Original with unbalanced contrast.

Figure 8.2 Original washed out with gamma correction.

However, F_Contrast's algorithm is retina-morphic. Retina-morphic contrast enhancement behaves like the human visual

system; it is adaptive, and it is spatially-varying. In other words, it automatically calculates a different curve transformation for each pixel in the image, based on an analysis of the scene content.

Figure 8.3 Original corrected with F_Contrast.

As a result, images processed using this technology are continually in balance across the entire image. Contrast and detail is preserved or enhanced both in dark and bright areas and true colour is preserved.

Because the processing mimics the human visual system, the images which are produced automatically look natural, without the need for complex calibration or parameterisation.

F_Contrast's algorithm is often employed "on-chip" in modern cameras. By shooting without this switched on, you can get flatter images which are more suitable for effects work. F_Contrast allows you to boost the impact of these flat images to match what was seen on set.

Quick Start

Connect the sequence you wish to enhance to the source input. F_Contrast will immediately attempt to balance the contrast. Control the ammount of enhancement using the **strength** parameter. The sensitivity of the plug-in to transform different areas of the image and the strength of the balance between processing dark and light areas can be controlled by the other parameters **variance** and **asymmetry** respectively.

Inputs

F_Contrast has one input. The input (**source**) contains the sequence to enhance.

Parameters

- The parameters for this plug-in are described below.

strength - Controls the amount of dynamic range compression performed by iridix. This can be made larger or smaller depending on the characteristics of the display and the quality of the source image. The lower the dynamic range of the display relative to the source, the larger should be this setting.

variance - Affects the sensitivity of the transform to different areas of the image and can be increased in order to emphasise small regions (e.g. faces). If this parameter is set to zero, the transform becomes spatially uniform.

asymmetry - Affects the strength of the transform in dark areas relative to bright areas of the image. The larger this parameter, the more the transform concentrates on the dark parts of the image. If set to around 80 and above, the transform concentrates almost completely on the dark areas, meaning that e.g. sky contrast is unaffected by the processing.

noiseSuppresion - Affects the strength of the transformation in very dark areas to prevent sensor or compression noise from becoming visible in the output image. A large value strongly restricts the transform, and should be use if source images are very noisy. The lower the value, the larger the enhancement capability for dark areas. This parameter can be set to a single value for all images, or to different values according to shooting conditions (exposure level, ISO speed, degree of source file JPEG compression).

inputType - To assist the algorithm select the image colour space. This can be

- **Video** - for images that are in the sRGB colour space.
- **Linear** - for images that are truly linear.

sRGB is an RGB colour space that was created to be a standard colour space for monitors, printers and the internet. It was originally created to match the way in which an 8-bit image was displayed on an 8-bit CRT monitor. More modern hardware, such as an LCD monitor or digital camera, often incorporates additional software or circuitry so that its output conforms to this standard as well. It is therefore best to assume, in the absence of any other information, that 8-bit input files will be in the sRGB colour space and you should choose **Video** as the input type.

DeBlur

This chapter looks at removing motion blur or out of focus blur from an image using the Foundry's F_DeBlur plug-in. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

Most blur removal algorithms simply apply a sharpen filter to the image, trying to boost image edges and give the impression of removing blur. The Foundry's F_DeBlur plug-in attempts a full solution of the deconvolution problem by taking the filter that originally caused the blur and applying the inverse filter to the image. In the case of motion blur, this has the effect of taking each pixel in the image that has been spread across a number of its neighbours and recombining it back to its original form.

The F_DeBlur algorithm is designed to remove global motion blur or out of focus blur. When deblurring a foreground object over an unblurred background, the algorithm should succeed on most of the foreground. However, it's likely that it will also introduce artifacts to the background from attempting to deblur parts of the image that weren't originally blurred. It's also likely that the parts of the foreground will contain similar effects for the same reason. In this situation, it will be necessary to composite the deblurred foreground out of the sequence and put it over the original background.

Figure 9.1 Input.

Figure 9.2 Output.

Quick Start

Connect the sequence to be deblurred to the source input and the same sequence delayed by one frame to the sourceMinusOne input. Position the sampleBox over the region of the image you wish to deblur. Initially select a small region, as the plug-in is computationally intensive and therefore easier

to tune using a small region. (Once you're happy with the results, swicth off **useDeBlurRegion** to process the entire frame.) For motion blur select set the **blurType** to **Motion** and set the width of the motion blur using **blurSize**. For focus blur set blurType to **Out Of Focus** and set blurSize to the diameter of the filter that caused the original blur. In both cases the size of the blur is very critical. Setting it to be too small will result in not enough blur being removed whilst setting it too big will produce large amounts of ringing and banding on the image.

For motion blur, F_DeBlur needs to know the direction of the global motion. This will be calculated automatically if **calculateAngle** is turned on, for which the temporal input **sourceMinusOne** must be connected. If **calculateAngle** is off, the angle is user specified via the **angle** parameter, which defaults to horizontal(0).

Once the blur type and size has been correctly set up try increasing **iterations** to 100. This should increase the amount of blur removed. If the image is particularly noisy or grainy try increasing **noiseEstimate**.

There are two fundamental problems with all deconvolution algorithms - amplification of noise and ringing on edges. In F_DeBlur, both are controlled to an extent by **noiseSuppression**, which regulates the correction applied in each iteration in flat areas of the image. To deal with the latter, the general approach is to apply bounds to the image after each iteration which limit the amount the image can be changed by the F_DeBlur algorithm. The magnitude (range) of the bounds is determined by **ringClamping**. In plain areas where no large variation in the image is expected, the algorithm applies stronger bounds to suppress ringing. Decreasing **ringClamped** will mean more stringent bounds are applied, which may have the adverse effect of flattening textures.

F_DeBlur also has a **suppressRinging** option to suppress ringing artefacts that may appear around edges. With **suppressRinging** turned on, the plug-in will do two passes over the image: the normal de-blurring pass, and a second pass which deblurs a highly blurred version of just the image edges. The second pass is designed to give good results in the regions where spurious rings tend to appear in the first pass, so the two resulting images can then be recombined in such a way as to reduce these artefacts.

Inputs

F_DeBlur has three inputs. The sequence to be deblurred, a time offset version of the same sequence, and an optional input, blur. By setting **output** to **Blurred** the sequence on the blur input will be blurred using the filter calculated by the deblur algorithm. This can be useful, for example, if you have removed blur from a sequence to comp an extra element in, and then wish to apply the original motion blur to the whole sequence.

Parameters

The parameters for this plug-in are described below.

blurType - Select **Motion** for removing motion blur and **Out Of Focus** for removing out of focus blur.

blurSize - The width in pixels of the blur filter for motion or the diameter of the filter for out of focus blur.

iterations - The number of iterations of the F_DeBlur algorithm. More iterations will give better results but take longer. Typically at least 100 iterations are required for a successful deblur but the default is set to 20 to allow quicker initial tuning of the algorithm.

noiseSuppresion - Increasing the **noiseSuppression** will cause the algorithm to concentrate on making the resulting image smooth by reducing noise amplification and ringing but less on removing blur. Decreasing **noiseSuppresion** should result in more blur being removed at the expense of increased ringing and noise.

ringClamping - Increasing ringClamping will result in a sharper image but with more ringing. Conversely decreasing ringClamping will result in textures in the image being flattened but with less obvious ringing.

output - Set to **DeBlurred** to view the deblurred source image. To re-apply the blur (to the third input), switch the output to **Blurred**. Make sure you have connected an image to the third (blur) input.

noiseEstimate - Is the amount of noise on the image. Increase if your image is very noisy or grainy and decrease if it is very clean.

calculateAngle - The algorithm tries to automatically work out the direction of the motion blur from frame to frame by calculating the global motion. If the motion is purely horizontal or you want to make the motion blur horizontal by pre-rotating the image you can turn off **calculateAngle**.

angle - If **calculateAngle** is turned off, this variable is used for the angle of the motion blur. This is measured in degrees.

suppressRinging - Set to true to suppress ringing artefacts that may be visible from the deblurring process.

useDeBlurRegion - whether to use the **deBlurRegion** or process the entire frame.

deBlurRegion - Selects the region of the image to be deblurred.

deBlurRegionMin - Set the bottom left position of the **deBlurRegion**.

deBlurRegionMax - Set the top right position of the **deBlurRegion**.

Examples

The images used in the following examples can be downloaded from our web site. For more information, please see the example images section on page 17.

LibertyBlurred

In this example, we will deblur a motion blurred image of the department store Liberty of London. This image is a difficult one, due to the black and white woodwork of the Tudor exterior, the stripes of which tend to produce ringing in the output. The input is shown in Figure 9.5.

Step by Step

1. FileIn the input image, and create an F_DeBlur node below it.
2. In order to see the results better, firstly set update to release, and then move the sample region up so that it's positioned over the stripes, as in Figure 9.3.
3. The result isn't that great, because the **blurSize** is wrong. The input image has a blur size of roughly 5 pixels, so increase the **blurSize** to 5. The result in the deBlurRegion should look like Figure 9.4.
4. To increase the amount of blur removed, increase **iterations** to 60.
5. There is still a small amount of noise in the result. Try setting **noiseSuppresion** to 250 to decrease this, which should also help the small amount of ringing that has occurred.

Figure 9.3 Input.

Figure 9.4 Output.

6. Finally, increase the **deBlurRegion** to be the same size as the input image. You should get a before and after like those in Figures 9.5 and 9.6.

Figure 9.5 Blurred image.

Fruit

In this example, we will deblur the fruit sequence, automatically inferring the blur direction.

Step by Step

1. FileIn the fruit sequence. Duplicate clip, and in the duplicate set the timeSlip to -1, and go to frame 20.
2. Create an F_DeBlur node, connecting the original clip to the first input, and the duplicate to the second. It's prob-

Figure 9.6 DeBlurred image.

ably worth positioning the **deBlurRegion** over a part of the image with some detail, for example the lettering in the blue label on the wine bottle in the bottom right of the frame as shown in 9.7.

3. Set the **blurSize** to 7. The image has begun to sharpen, as shown in Figure 9.8.

Figure 9.7 Input.

Figure 9.8 Output.

4. Increase **iterations** to 80 to get it even sharper.
5. This has introduced quite a lot of noise into the image, so increase **noiseSuppression** to 500, giving a result as in Figure 9.10.

Figure 9.9 Input.

Figure 9.10 Final Output.

DeFlicker1

When working in film you sometimes have to deal with shots that have a luminance flicker. Furnace has two different algorithms for handling flicker: F_DeFlicker1 and F_DeFlicker2. In general, it is difficult to quantify exactly when one works better than the other, so the suggested work flow is that you choose one and try it on the footage you want to deflicker. If it helps, try tuning the parameters to remove as much flicker as possible. If it doesn't seem very effective, you should try the other one.

This chapter concentrates on removing flicker using F_DeFlicker1. For details of how to remove flicker with F_DeFlicker2, please see the chapter on page 77. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_DeFlicker1 is designed to reduce spatially variable flicker from a sequence. Most current flicker reduction tools are global, that is, they try to reduce the same amount of flicker from the whole of the image. If only parts of an image are flickering this technique will fail as it will reduce the flicker from one part but introduce it in another. The difficulty in automatically correcting for spatial variable flicker is differentiating between flicker and motion. It is worth noting that this plug-in will reduce flicker in most cases, but it is unlikely to remove all traces of flicker.

In this plug-in we divide the image up into blocks and adjust the contrast and brightness in these regions to correct complex localised flicker. We also analyse motion vectors between the current and reference frames and use a global luminance change for blocks that have significant motion. In addition, the plug-in will allow you to analyse the flicker from one sequence and apply the same flicker to another. This can be particularly useful, for example, when trying to match the flicker on a CG element that is being composited in to a flickering sequence.

F_DeFlicker1 can deflicker relative to the previous frame of a sequence or to a specified frame. Note that network rendering is not possible when the reference method is set to **previousFrame**.

Quick Start

The only decision that's worth making at the start is whether to deflicker relative to the previous frame of the sequence or to a

specified frame. To decide this you'll need to look carefully at the flickering clip. If the flicker is in a part of the image that is not moving much, then choose clip as the **referenceMethod**. For example, you may have a locked off shot with action in the foreground, but the flicker is only in one of the corners in the static background. For most other circumstances, choose previousFrame. For example, if the camera is panning then everything is moving.

If you have chosen to deflicker to the previous frame, connect the sequence containing the flicker to the first (**source**) input of F_DeFlicker1 and render using the defaults. With this method you may see a drift in the mean luminance value during the render. You can correct for this by increasing the **feedback**, however, this will reduce the amount of flicker reduced.

If you have chosen to deflicker to a single image, connect the flicker sequence to the first (source) input and a single frame from the flicker sequence to the second (reference) input. Switch **referenceMethod** to clip and render.

Tuning

If the sequence still flickers there are one or two changes you can make to try to improve the result. To put this in context it's worth explaining a little more about the algorithm. F_DeFlicker1 fits a block based model to the image. We split the source and reference images into multiple blocks of size **blockSize**. We then try and modify the brightness and contrast (the flicker parameters) of the source block to make it equal to the brightness and contrast of the reference block. In practice, if there is motion in the sequence the pixels in the two blocks will not be the same which will result in incorrect flicker parameters. To minimise this effect, when the image is divided into blocks they are overlapped by a small region. To this region we apply the flicker parameters calculated for both adjacent blocks. If the results from these two de-flickers are not consistent we assume that the parameters have been calculated incorrectly due to motion and we discard them both, replacing them with the global brightness and contrast parameters for the whole image. The criteria for discarding parameters are set by **motionThreshold**.

If there is little information in the block, for example in a plain area, it is difficult to obtain reliable estimates for the flicker parameters. We check for this by discarding the parameters for any blocks with a variance below **weightThreshold**. Having obtained a reliable set of flicker parameters for each block these are then smoothed using a combing technique. The number of smoothing iterations is set by **smoothness**.

Copying Flicker

To copy the flicker onto another sequence, simply connect that sequence to the third (**copyTo**) input node and render. This works best if the source and destination clips have a similar luminance.

Warning! **Shake scripts containing F_DeFlicker1 cannot be distributed across multiple machines on a network render farm. See "Network Rendering" on page 22.**

Inputs

There are four inputs. The first (**source**) input is the clip that is flickering. The second (**reference**) input is an optional input for a reference clip and should be connected if the **referenceMethod** is set to clip. This would usually be a single frame but could also be a clip. The third (**copyTo**) input is used to copy the flicker from source input to this clip.

Some areas of luminance change in the image are not flicker but can cause the deflicker algorithm to produce poor results. The fourth (**globalFlickerMatte**) clip is used to remove these areas from the deflicker calculations. Black areas are ignored and white areas are included. Even when the **globalFlickerMatte** is in use, F_DeFlicker1 deflickers the entire image. To keep areas of the source image unchanged you must roto and composite the results of F_DeFlicker1 back over the original.

Parameters

The parameters for this plug-in are described below.

referenceMethod - Sets the comparison frame to DeFlicker1 against.

- **PreviousFrame** - removes flicker from the current frame by comparing it with the previous frame. Note, this won't work for distributed renders as it relies on the previous frame being in memory.
- **Clip** - removes flicker from the current frame by comparing it with the input frame of the clip attached to the second input (reference).

feedback - When the reference method is set to previousFrame, feedback trades off the colour drift against the amount of flicker removed. You should try and keep this value as low as possible to remove as much flicker as possible. If you're getting colour drift, increase this parameter although doing this will remove less flicker.

Warning! **If the reference method is clip, this parameter has no effect.**

blockSize - Sets the size of the blocks that are analysed for flicker. Smaller blocks will allow more complex flicker to be modelled but will produce less accurate and robust results and will take longer to render.

weightThreshold - Sets the threshold value for plain areas above which blocks are discarded due to the lack of reliable flicker data. If your results are poor, it's worth decreasing this value.

motionThreshold - Sets the threshold value below which localised deflicker is abandoned in favour of a global deflicker because of motion in the frame. The algorithm has difficulty distinguishing between luma flicker on an area of the image that has no movement and on areas that have movement but no flicker. This threshold determines at what point we force the deflicker to be global because there's too much movement. Setting the motionThreshold to 0 will turn off localised deflicker for all moving objects and apply global luminance changes to reduce the flicker. If you're getting errors (colour changes) around areas that are moving then you should lower the motionThreshold. If parts of the frame that are static are not being deflickered then you should increase the motionThreshold.

smoothness - Sets the number of times the flicker parameters are smoothed between blocks.

globalFlickerMatteComponent - Defines what to use to get the areas to exclude from the deflicker calculations.

- **None** - Don't exclude any areas.
- **Source Alpha** - exclude the area defined by the alpha channel of the source.
- **Source Inverted Alpha** - exclude the area defined by the inverse of the alpha channel of the source.
- **GlobalFlickerMatte Luminance** - exclude the area defined by the luminance of the global flicker matte.
- **GlobalFlickerMatte Alpha** - exclude the area defined by the alpha of the global flicker matte.
- **GlobalFlickerMatte Inverted Luminance** - exclude the area defined by the inverted luminance of the global flicker matte.
- **GlobalFlickerMatte Inverted Alpha** - exclude the area defined by the inverted alpha of the global flicker matte.

Examples

All the images for the following examples can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Hedge

In this example, we have a non-uniform and complex flicker over a moving leaf background.

Figure 10.1 Hedge.

Step by Step

1. FileIn hedge.####.tif and flipbook. The flicker is quite subtle so you're going to have to watch it loop for a while. While you're doing that, note that there is lots of movement in the leaves.
2. Select Furnace's F_DeFlicker1 and connect the output of the hedge to the first (source) input of the F_DeFlicker1 node.
3. Since there is lots of movement in the shot, set the F_DeFlicker1 **referenceMethod** to previousFrame and flipbook. Now compare it to the original.
4. You can improve the results still further by increasing the **motionThreshold** from 8 to 20.

Bus

This is an unusual example. The flicker here is a global luminance flicker over the moving bus and static background. Set the **referenceMethod** to either the previous frame or clip. It doesn't matter that much because we're going to ignore all motion anyway and only deflicker globally. To do this set the **motionThreshold** very low and render. Try rendering with **referenceMethod** set to clip and then **previousFrame**. Note

that the results are similar but for a slight luma gain over the sequence in the latter.

DeFlicker2

When working in film you sometimes have to deal with shots that have a luminance flicker. Furnace has two different algorithms for handling flicker: F_DeFlicker1 and F_DeFlicker2. In general, it is difficult to quantify exactly when one works better than the other, so the suggested work flow is that you choose one and try it on the footage you want to deflicker. If it helps, try tuning the parameters to remove as much flicker as possible. If it doesn't seem very effective, you should try the other one.

This chapter concentrates on removing flicker using F_DeFlicker2. For details of how to remove flicker with F_DeFlicker1, please see the chapter on page 71. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_Deflicker2 is used to remove flicker that is localised and dependent on the geometry of the scene, such as that caused by an unsynchronised fluorescent light in a shot. It works by calculating the gain between the current frame and each frame in a small window surrounding it. It then tries to adjust the gain so that it varies smoothly over this temporal window. This means it is better at reducing fast flicker than flicker which varies slowly over the image sequence, as the latter will already appear smooth over the window and F_DeFlicker2 will leave it largely untouched. (When you have slowly-varying flicker that you want to remove, F_DeFlicker1 is likely to do a better job, so you should probably try this first.)

The algorithm used by F_DeFlicker2 can introduce blurring in areas where there is rapid motion. If this happens, using local motion estimation to align the frames before deflickering them can help. However, this process is complicated by the fact that the presence of flicker can adversely affect the results of the motion estimation. F_DeFlicker2 therefore adopts a two stage approach to this problem. First, the normal deflickering process is performed. The motion vectors for the sequence are calculated on the resulting deflickered frames, then applied to the original frames in order to align them. The deflicker calculation is then performed on the aligned frames to give the final result. To use this approach, turn on **useMotion** in F_DeFlicker2.

Quick Start

Click on the F_DeFlicker2 file-in node and select an input sequence to deflicker. Render.

Inputs

There are nine inputs: the source frame, the four preceding frames and the four following frames.

F_DeFlicker2 also has a macro that can be used to connect a file sequence into its inputs automatically. In the Furnace tab page, this appears as an icon identical to that for F_DeFlicker2, apart from the green spot in the top right hand corner. This green spot is used across all Furnace plug-ins to denote a file-in node. When you click on the icon, you will be prompted to choose an input sequence. The appropriate frames from this sequence will then be connected to the nine source inputs for you.

Parameters

The parameters for this plug-in are described below.

blockSize - the size of the blocks used to calculate the flicker.

flickerClamp - use this to reduce flicker without removing it entirely; smaller values mean more will be left behind.

useMotion - turn this on to do a second deflicker pass using motion-compensated frames. This can improve results in areas where there is fast motion, where the initial deflicker pass can introduce blurring.

vectorDetail - determines the accuracy of the motion vectors used when useMotion is turned on. The maximum value of 1 will generate one vector per pixel. This will produce the most accurate vectors but will take longer to render.

windowSize - the size of the temporal window to use to remove flicker.

Example

In this example, we'll look at removing from a clip using F_DeFlicker2. The clip used here can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Step by Step

1. Click on the F_DeFlicker2 file-in node. Choose toDeflicker.##.tif as the input sequence.
2. Select the source node and render a flipbook. Notice the flickering fluorescent light in the window, which we're going to try to remove.
3. Select the in-scene preset on the F_Deflicker2 node.
4. Render a flipbook of the F_DeFlicker2 node's output.
5. View the two flipbooks side by side. Notice that the flickering from the light has been substantially reduced.
6. That's it.

DeGrain

This chapter looks at the tools available in Furnace to remove grain from an image. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

Furnace's F_DeGrain plug-in is used to remove grain from a sequence. The aim is to remove as much grain as possible whilst doing as little damage to the image as possible.

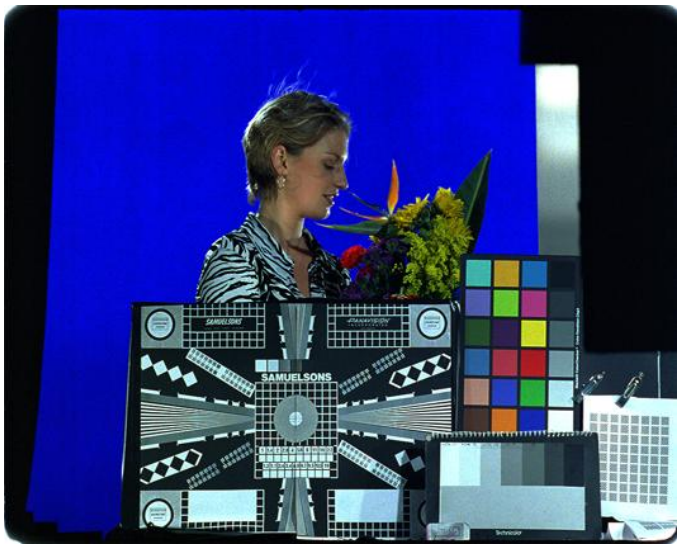


Figure 12.1 Test grain image supplied with tutorial examples.

F_DeGrain's spatial filtering involves averaging pixels within the same frame. This can lead to the blurring of the image and so, to keep this to a minimum, a wavelet based technique is used that decomposes the image into a number of different frequencies and scales before attempting to remove the grain. The high frequency spectrum can be isolated and processed using the Tune Small parameter. The low frequencies are processed using the Tune Large parameter. The internal degrain parameters are set automatically by analysing the image; however, tuning is also possible to help generate even better results.

Quick Start

Connect an image or sequence to the input node of F_DeGrain and position the selection box over a plain area of the image (Figure 12.3).

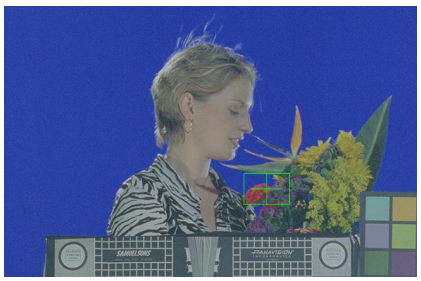


Figure 12.2 Bad sample position.

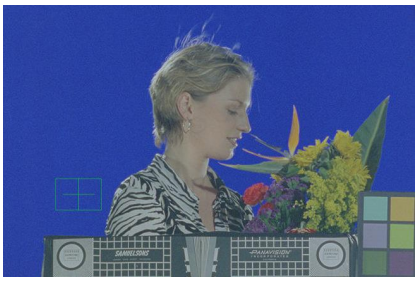


Figure 12.3 Good sample position.

Warning! **It is very important to position the selection box over a region with little image detail. Failure to do this will give poor results as the algorithm will think the image detail is grain and remove it.**

The sample selection automatically updates not only the sample rectangle parameters in the sample group but the frame at which the same should be made. Whenever the sample rectangle is altered the internal analysis of the grain in that region reoccurs. Sample rectangle analysis data is saved into scripts for distribution across render farms.

Fine Tuning

If either not enough grain has been removed or the picture has been soften by removing too much grain it will be necessary to fine tune the parameters. Increasing **tune** will remove more grain, reducing it will remove less. This is a fairly crude way of setting the parameters. Below this are more advanced controls.

The easiest way to find the optimal setting for F_DeGrain is to look at what is removed from the image. To do this, change the **output** parameter to **Grain**. This will display the grain that is being subtracted from the original image; if necessary, increase **exaggerateGrain** to make it more obvious. Only grain should be visible in this image. If you can see a lot of picture detail it means the degrainer is working too hard and removing too much of the image, which will lead to a soft result.

The degrainer works by decomposing the image into 4 levels - **Small**, **Medium**, **Large** and **Huge**. To set the first level select **detail** to **Huge** and adjust **tuneHuge** until the image is only just visible, and then repeat for **Large**, **Medium** and **Small**. Often the blue channel will contain more grain than the red and green. This can be checked by viewing the individual colour channels. If this is the case, increase **tuneBlue** until enough grain is being removed. When you are happy with the settings make sure **detail** is set to **All**.

Inputs

F_DeGrain has one input that is the image to be degrained.

Parameters

The parameters for this plug-in are described below.

detail - Sets which of the frequencies to process. In other words you can remove and process only the large grain leaving the others untouched. However, normally uou would remove grain throughout the frequency spectrum by selecting **detail All**.

- **All** small medium, large and huge grain is removed.
- **Small** only the small scale grain is removed.
- **Medium** only the medium scale grain is removed.
- **Large** only the large scale grain is removed.
- **Huge** only the largest scale grain is removed.

tune - this is the coarse adjustment control. Increasing tune will remove more grain and decreasing it will leave more in.

+fineTuning - these are the spatial fine tuning controls.

tuneRed - increases or decreases the amount of grain removed in the red channel.

tuneGreen - increases or decreases the amount of grain removed in the green channel.

tuneBlue - increases or decreases the amount of grain removed in the blue channel.

tuneSmal - increases or decreases the amount of small grain removed.

tuneMedium - increases or decreases the amount of medium grain removed.

tuneLarge - increases or decreases the amount of large grain removed.

tuneHuge - increases or decreases the amount of huge grain removed.

+sample - the selection box that marks the region of the image used to analyse the grain and to set the parameters automatically. It is important that this part of the frame contains no image detail, only grain.

sampleFrame - sets the frame from which the sample rectangle should be taken.

sampleRectMin - sets the bottom left position of the sample rectangle.

sampleRectMax - sets the top right position of the sample rectangle.

output - whether to output the degrained image or the grain that was removed.

- **Result** - output the result of degraining the source.
- **Grain** - output the grain that was removed from the source.

exaggerateGrain - when displaying the removed grain, increase this parameter to make it more visible.

Examples

The images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Rachael

FileIn rachael.jpg and apply the F_DeGrain node. You should

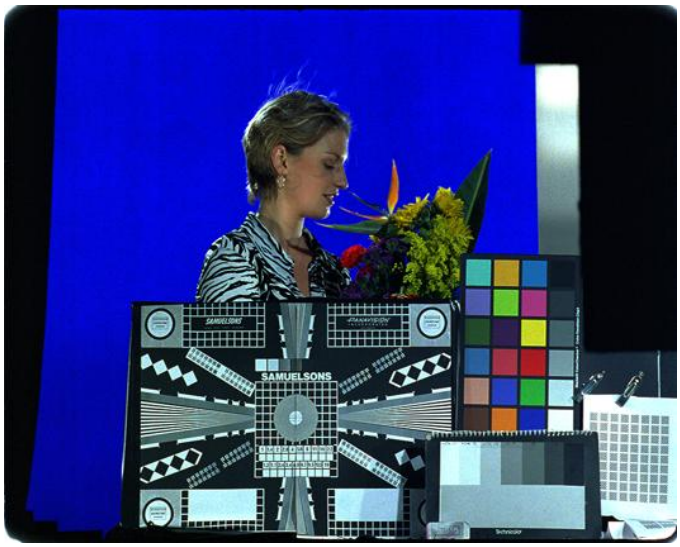


Figure 12.4 rachael.cin

practise looking at the grain and varying the **tune** parameter to increase or decrease the grain removed. Note how the fine hair detail is preserved. Try sampling and comparing the grain from light and dark areas.

Preserving Luminance

If you think the degraining is softening the image too much, the following example may help. It should help you retain the overall image sharpness whilst being able to remove a lot of grain.

Put the original and degrained images through separate ColorSpace nodes using RGB in and HLS out. Use a Copy node and copy the luminance channel from the original to the degrained image. This is done by putting "g" in the Copy node. Put the output of the Copy node back through a ColorSpace node using HLS in and RGB out. This tree has the effect of retaining the sharpness from the luminance of the image whilst losing the grain.

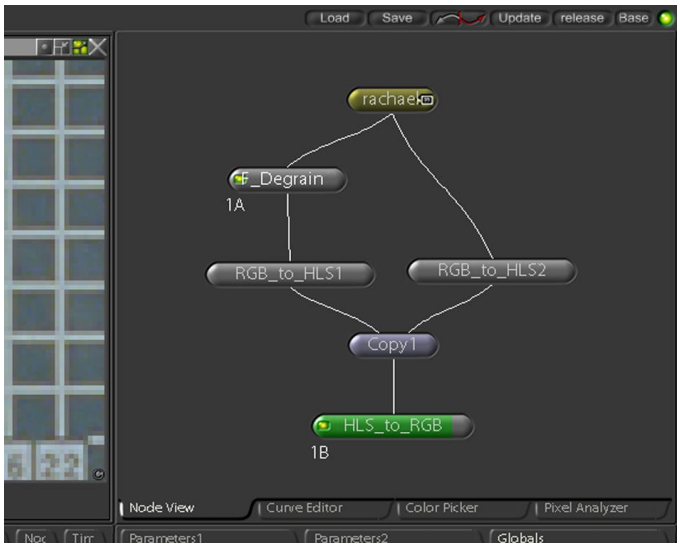


Figure 12.5 Shake tree showing luminance added back over the degrained image to preserve sharpness.

DeNoise

This chapter looks at removing noise or grain from an image sequence using Furnace's plug-in F_DeNoise. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_DeNoise is designed to remove noise or grain from a clip. Assuming there is no motion in a sequence, the best way to reduce the noise is to take an average across a number of frames (temporal averaging). The noise which is different on each frame will be reduced and the picture which is the same will be reinforced. Temporal averaging is far superior to averaging pixels from within the same frame (spatial averaging) as it doesn't soften the image. Unfortunately, if there is motion in the sequence, the averaged image will be blurred as the image appears at different locations in each frame. However, by estimating the motion in the sequence using The Foundry's advanced motion estimation technology, it is possible to compensate for any motion and so average frames temporally without introducing any blurring artefacts.

As F_DeNoise is a fully motion compensated noise reducer it is very good at removing noise or grain from a sequence. If you just have a single frame you should use the Furnace plug-in F_DeGrain. This is a powerful wavelet-based spatial noise reducer and is likely to give better results on single images.

Quick Start

Having imported the footage to be noise reduced, duplicate the input clip twice. Time slip the duplicated clips by one frame to either side of the current frame. Connect these to the sourcePlusOne and sourceMinusOne inputs. Select the **plateSize** of your input frame - note that this refers to the original size of the plate so even if you are working on a cropped part of a 2k plate, **plateSize** should still be set to 2k. F_DeNoise works by analysing the grain structure inside the on-screen sample box, so move this box over a plain area of the image. To get a good result it is important that this area is free from image detail, so no textures or edges. The output should will now show the denoised frame. If you are not satisfied with the results, try moving the sample box to a different, flat area of a frame. F_DeNoise will reanalyse the grain structure every time this box is repositioned, or the **plateSize** parameter changed.

To remove more noise simply increase the **tune** parameter. To view the removed noise, set the **output** parameter to **Noise**. The output from this will be the noise that has been subtracted from the original image by F_DeNoise. If necessary, increase **exaggerateNoise** to make the noise easier to see.

You can also remove different amounts of noise from the red, green and blue channels by altering the **tuneRed**, **tuneGreen** and **tuneBlue** parameters.

F_DeNoise has a **suppressRinging** option to suppress ringing artefacts that may appear around edges. With **suppressRinging** turned on, the plug-in will do two separate denoise passes over the image. The second pass is designed to give good results in the regions where spurious rings tend to appear in the first pass, so the two resulting images can then be recombined in such a way as to reduce these artefacts.

Inputs

F_DeNoise has four inputs, the current frame (**source**), the two frames either side (**sourceMinus1** and **sourcePlus1**) and an optional motion vector input (**vectors**).

F_DeNoise also has a macro that can be used to connect a file sequence into its inputs automatically. When you create an F_DeNoiseMacro node, you will be prompted to choose an input sequence, and the appropriate frames from this sequence will be connected to the three source inputs for you. The macro will additionally connect a dummy **vectorsIn** node into the vector input that can be easily replaced if desired.

Parameters

The parameters for this plug-in are described below.

plateSize - The algorithm automatically sets some parameters depending on the expected size of the noise and grain which can be related to the size of the image. As you may be processing a cropped region we do not necessarily know this from the image size. Select PAL/NTSC, 1K, 2K, or 4K depending on the original size of the scan.

tune - This adjusts the overall amount of noise or grain that is removed. Increase this value to remove more noise.

+fineTuning - These parameters allow you to remove different amounts of noise in each of the colour channels.

> **tuneRed** - increases or decreases the amount of noise removed in the red channel.

tuneGreen - increases or decreases the amount of noise removed in the green channel.

tuneBlue - increases or decreases the amount of noise removed in the blue channel.

+analysis - These parameters allow you to change the region used to analyse the grain, in order to improve the noise reduction.

sampleCentre - the position of the centre of the analysis region.

frameToAnalyse - the frame to analyse on.

suppressRinging - switch this on to remove the ringing that can be introduced near to edges in the denoised image.

output - Whether to output the denoised image or the noise that was removed.

- **Result** - output the denoised source image.
- **Noise** - output the noise that was removed from the source image.

exaggerateNoise - If you have chosen to output the noise, increase this parameter to make it more obvious.

Example

The footage used in the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Mike

This example, we'll use F_DeNoise to remove noise from a sequence. The clip used in this example can be downloaded from ou web site. For more information, please see the Example Images section on page 17.

Download File

MikeWire.tgz

Step by Step

1. Create an F_DeNoise node, and select MikeWire.#.tif as the input. Go to frame 9 (this is just a good example; any frame will do). This frame is shown in Figure 13.1 on the next page. Wire in the temporally slipped inputs.
2. Select PAL/NTSC for **plateSize.**

3. Whilst this is a reasonable result, we can do better. Move the analysis box widget to the top right of the frame to get a better sample region, as in Figure 13.1.
4. You should get a better result, as in Figure 13.3 on the next page. Render a flipbook of the sequence to see the results.

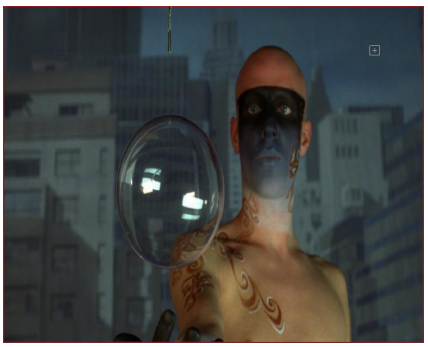


Figure 13.1 Original image

Figure 13.2 Zoomed original image.

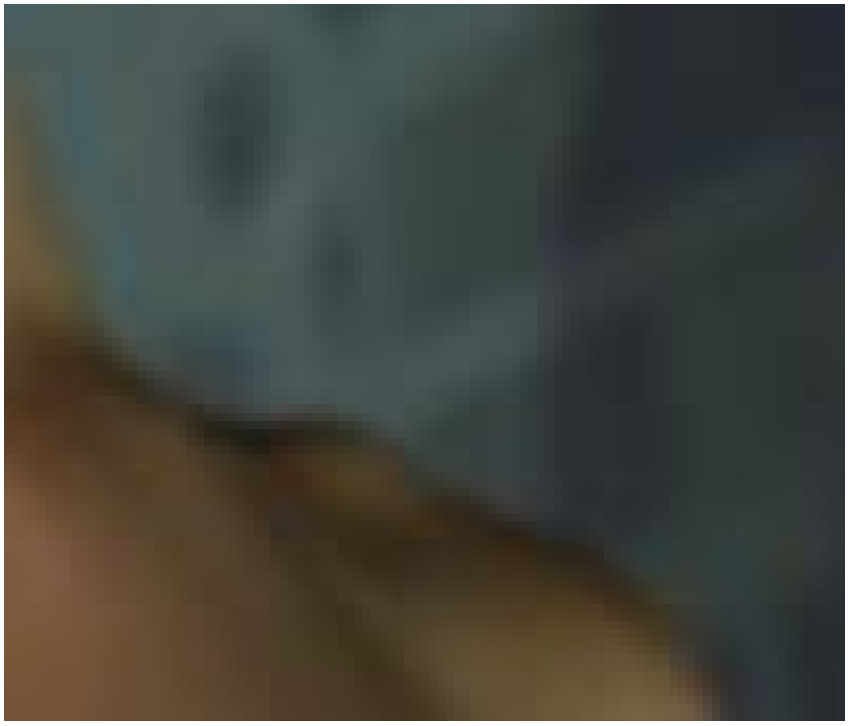


Figure 13.3 Output image with a good analysis region.

Depth

This chapter looks at calculating the depth channel (Z-channel) from a clip using Furnace's plug-in F_Depth. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

Z-depth compositing is usually associated with 3D computer generated images, since accurate depth information can be automatically generated. No such information is captured from live action footage. However, we have developed an algorithm that can extract a depth channel by looking at the parallax shifts in objects moving in a scene.

F_Depth uses The Foundry's advanced motion estimation technology to calculate the relative motion of objects in a sequence as this gives a reasonable approximation to depth. However, there are some restrictions. We assume there is no local motion and the focal point of the camera moves in space (i.e. the camera move is non-nodal). The depth is displayed as a grey scale image with white being closest to the camera and black furthest away. Don't expect the results to be as pin sharp as 3D generated Z-channels, but you should find them useful enough. So, what can you do with a depth channel? One simple use would be to apply a depth of field blur so that objects close to and far away from the camera appear out of focus with mid-range objects sharp (using Shake's ZBlur node, for example).

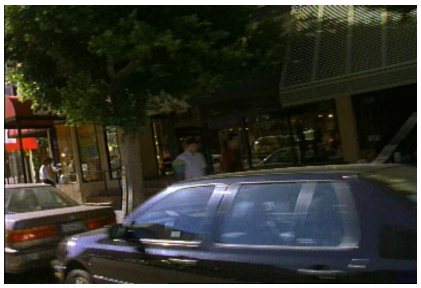


Figure 14.1 Input.

Figure 14.2 Output.

Quick Start

Connect the sequence from which you wish to calculate depth to the source input. Connect a frame delayed version of the same sequence to sourceMinus1, and a frame advanced version to sourcePlus1. If the sequence contains an obvious layer

which is the furthest from the camera, position the region box over the layer and turn on useBGAnalysis. This will force this layer of the image to be the background and calculate depth relative to this layer. Process the sequence to obtain the depth images.

Inputs

F_Depth has five inputs, **source, sourceMinus1**, **sourcePlus1**, **vectors** and a **matte**. The depth channel will be extracted from the **source** clip. **SourceMinus1** is this sequence delayed by one frame, and **sourcePlus1** is this sequence advanced by one frame. The **vectors** input is an optional motion vector input. If the motion in your input sequence has been estimated before and you have the motion vectors available, you can connect them to the **vectors** input. This will save processing time, as F_Depth will not then have to perform the motion estimation a second time. However, it is worth noting that F_Depth requires smoother motion vectors than most other Furnace plug-ins in order to produce good results; for example, if you were generating vectors for it using F_VectorGenerator, you would do so with **overSmooth** turned on (please see the chapter on Local Motion Estimation on page 265 for an explanation of what this means). Finally, you can supply a matte of the background in the image. If you connect this to the **matte** input and turn on **useBGAnalysis**, this will force this layer of the image to be the background and the depth elsewhere will be calculated relative to this layer.

F_Depth also has a macro that can be used to connect a file sequence into its inputs automatically. When you create an F_DepthMacro node, you will be prompted to choose an input sequence, and the appropriate frames from this sequence will be connected to the three source inputs for you. The macro will additionally connect dummy **matteIn** and **vectorsIn** nodes to the other two inputs that can be easily replaced if desired.

Parameters

backgroundRegion - defines the area to use as the background.

- **None** - don't set a background region.
- **Box** - use the backgroundRegionBox parameters below, or the on-screen box, to define the background region.
- **Source Alpha** - use the region defined by the alpha channel of the source as the background.

- **Source Inverted Alpha** - use the region defined by the inverse of the alpha channel of the source as the background.
- **Matte Luminance** - use the region defined by the luminance of the matte input as the background.
- **Matte Alpha** - use the region defined by the alpha of the matte input as the background.
- **Matte Inverted Luminance** - use the region defined by the inverted luminance of the matte input as the background.
- **Matte Inverted Alpha** - use the region defined by the inverted alpha of the matte input as the background.

useBGAnalysis - To calculate depth relative to a layer in the image, position the region box over the layer and turn on useBGAnalysis.

smoothOutput - Activates a post process to try to refine the edges of regions. Unfortunately, this may introduce artifacts in textured, but spatially flat areas.

normalizeOutput - By default, the output from F_Depth is normalized to give visual feedback. This information has been normalized spatially on a frame by frame basis, and so may introduce flicker as objects enter or leave the scene. If you wish to perform normalization yourself (for example by rendering the unnormalized output to find the largest value encountered and then normalizing by that), switch this off.

backgroundRegionBox - The region used to specify the layer in the image that the depth is calculated relative to. This region will be black in the depth matte.

backgroundRegionBoxMin - set the bottom left position of the background region.

backgroundRegionBoxMax - set the top right position of the background region.

Examples

All the images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Leicester Square

In this example, we will infer the relative depth of objects in a clip of Leicester Square in London, shown in Figure 14.4.

Step by Step

1. FileIn LeicesterSquare.####.tif and duplicate it twice, setting the timeslip parameter on the duplicates to -1 and +1. Go to frame 5.

2. Create an F_Depth macro node (the one with the green spot in the top right hand corner), and choose LeicesterSquare.####.tif as your input sequence. The output will look like Figure 14.3.

Figure 14.3 Initial output.

3. Whilst the result in Figure 14.3 is reasonable, we can make it better. We can refine the edges, and also increase contrast between foreground and background by positioning our backgroundRegion window better. Position the box so that it's in the centre but at the top of the input, as shown in Figure 14.4.

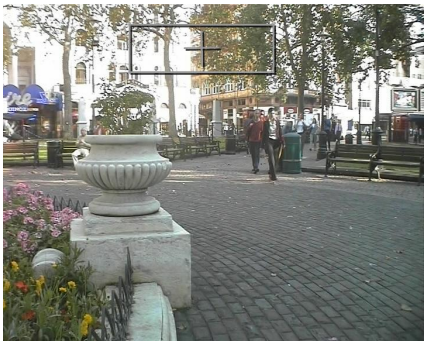


Figure 14.4 Background Selection.

4. On updating, you should get a result like the one in Figure 14.5.

5. For extra points, we're going to perform a blur based on this depth channel using Shake's ZBlur node. Create a Reorder node, connected to F_Depth, using "rrrrr" in the channels to put the depth into the Z-channel.

Figure 14.5 Improved output.

6. Create a Copy node, connecting the first input to the original non timeslipped sequence, and the second to the Reorder node. Copy the Z-channel by using "z" as the channel's parameter.
7. Create a ZBlur Node, and you should get a result like that in Figure 14.6.

Figure 14.6 Depth Blur.

DirtRemoval

This chapter looks at the removal of dust and dirt from images using Furnace's plug-in F_DirtRemoval. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_DirtRemoval will automatically detect and remove specs of dust and dirt from a frame. The plug-in works by looking for objects that appear for only one frame, after taking account of the motion in the sequence.

For example:

- A spec of dirt that appears for only one frame will be classified as dirt.
- A football being kicked across the image will not be classified as dirt because, after taking account of motion, it appears in each frame of the sequence.
- A vertical scratch in a sequence will not be classified as dirt as it appears in the same place in each frame. The scratch repair plug-in should be used to repair the sequence.
- Dirt on the camera lens or in the telecine gate will not be classified as dirt as it appears in the same place on each frame.

Having detected the location of the dirt, the algorithm produces a seamless repair by taking motion compensated pixels from the surrounding frames and interpolating them into the dirt region. In order to instantly see which regions have been repaired the pixels detected as dirt are marked in the alpha channel.

An alpha channel, marking the regions of dirt, can also be provided as in input to the plug-in, in which case no detection is done and only the regions specified in the alpha channel are repaired. This alpha channel could be generated by editing the automatic dirt detection from the plug-in, thus making a two-stage process, or from an infrared scan of the film that is available as an output from some film scanners.

Background

The main control provided by the plug-in is the set of **presets**. These control the trade off between falsely identifying dirt and

Figure 15.1 Before

Figure 15.2 After

failing to spot the dirt. Often, even if a region of image has been falsely detected as dirt, it will be repaired perfectly as the dirt will not have corrupted the motion in the region, allowing a high quality motion compensated repair. The **presets** are overall tuning controls that set nine parameter values. For the advanced user it is also possible to fine-tune these individual parameters for better results on specific sequences.

In order to understand how to tune the parameters it is first necessary to understand a bit more about the algorithms involved. F_DirtRemoval relives heavily on motion estimation to both detect the dirt and repair the image. Where the motion is complex, e.g. multiple objects moving fast in multiple directions, we are unable to correctly calculate the motion. This means both the dirt detection and repair will fail. In order to improve results in these regions we have a complex motion detector. This detector is designed to flag regions where we are unlikely to calculate the correct motion. In these regions, we detune the motion based dirt detector and add a spatial dirt detector. Only if both detectors flag dirt do we actually believe there to be dirt.

Quick Start

Click on the dirt removal macro and when prompted select the sequence to be cleaned. Set the **plateSize** to the original size of the scan (not the size of any cropped region you may be analysing) and render. View the result and the alpha channel to see where the dirt has been detected and removed and vary the preset applied as required.

If regions of motion have been incorrectly classified as dirt choose a lower preset or read the section on complex motion detection and tune the parameters. If dirt has been missed try choosing a higher preset or manually tuning the parameters under dirt detection.

Inputs

F_DirtRemoval has the following seven inputs:

1. **source** is the dirty sequence.
2. **sourceMinus2** is the source sequence offset temporally by -2 frames .
3. **sourceMinus1** is the source sequence offset temporally by -1 frame.
4. **sourcePlus1** is the source sequence offset temporally by +1 frame.
5. **sourcePlus2** is the source sequence offset temporally by +2 frame.
6. **dirtMask** is an optional mask indicating the position of the dirt in the sequence. If this input is connected no detection will take place and only the regions specified in this mask will be repaired.
7. **vectors** is an optional vector input to supply the motion vector fields for the input sequence (see the chapter on Local Motion Estimation on page 265 for more information about vector fields). If these have already been calculated elsewhere, connecting them to the DirtRemoval plug-in will save processing time by eliminating the need for it to repeat the motion estimation itself.

F_DirtRemoval also has a macro that can be used to connect a file sequence into its inputs automatically. When you create an F_DepthMacro node, you will be prompted to choose an input sequence, and the appropriate frames from this sequence will be connected to the five source inputs for you. The macro will additionally connect dummy **dirtMaskIn** and **vectorsIn** nodes to the other two inputs that can be easily replaced if desired.

Parameters

The parameters for this plug-in are described below.

presets - this is the main control for the plug-in which trades off the amount of dirt detected and repaired verses the number of false detections and incorrect repairs. For archive footage, you should typically choose **veryHigh**, whereas for a modern scan with isolated patches of dust you should choose **low** or **medium**.

output - as well as the repaired image it is possible to output a number of diagnostic images which are useful for tuning the parameters.

- **Source** - the original frame containing dirt.
- **Complex Motion Region** - the region of image flagged as having complex motion.
- **Motion Dirt** - the dirt detected using a motion based detection algorithm.
- **Spatial Dirt** - the dirt detected using a spatial median based filter.
- **Combined Dirt** - a combination of the two methods according to the complex motion region. Inside the complex motion region it is the dirt detected by both the motion and spatial based detectors. Outside the complex motion region it is just the dirt detected by the motion detector.
- **Dilated Dirt** - dilated Combined Dirt image used to make the repair.
- **Final Dirt** - for each piece of dirt in Dilated Dirt we attempt to make a repair. If this repair is very similar to the source image (within a tolerance defined by changeThreshold) we assume the dirt was incorrectly detected and discard it from the matte and repair. This modified matte is the Final Dirt. Red pixels indicate the dirt is in the complex motion region and therefore more likely to be unreliable. White pixels indicate dirt detected outside the complex motion region. By quickpainting this output you can quickly delete any false detections, or reinstate any that were erroneously discarded, before feeding the matte generated into a second pass of DirtRemoval as the dirtMask input.
- **Repair** - the repaired output image using the Final Dirt matte.
- **Repair Dirt In Alpha** - the repaired output image with the Final Dirt shown in the alpha channel. Any alpha in the source image will be removed.

plateSize - the algorithm automatically sets some parameters depending on the expected size of the dirt which is related to the size of the image. As you may be processing a cropped region we do not necessarily know this from the image size. Select PAL/NTSC, 1K, 2K, or 4K depending on the original size of the scan.

+dirtDetection - two dirt detection schemes are used. Generally a motion based detection algorithm is sufficient but in regions of complex motion this is aided by a spatial detection scheme.

detectionThreshold - this is the main threshold below which we set a pixel to be dirt and above which we assume it is an image feature.

dilate - this is the amount by which the pixels detected as dirt are grown to ensure that the whole region of dirt is correctly detected. Occasionally we only detect the centre of a piece of dirt and so without dilation we would correct the interior but leave a halo of dirt. If this is the case increase dilate until the whole of the dirt is removed.

changeThreshold - after repairing the dirt the repaired pixels are compared with the original pixels. If they differ by less than changeThreshold it is assumed that they were incorrectly flagged as dirt and replaced with the original pixels.

safetyFactor - in the regions of complex motion we need to be more cautious about detecting dirt based on motion. So rather than using the detectionThreshold used in the other regions we scale it by safetyFactor to be extra cautious.

medianSize - as well as using motion to detect dirt in regions of complex motion we additionally add a spatial check. This is based on filtering the image with a median filter to remove objects below a certain size. MedianSize sets the size of this median filter. Making it too large will increase computation time and falsely detect image objects as dirt, too small and dirt will be missed.

medianThreshold - after median filtering, pixels that differ by more than medianThreshold are flagged as dirt by the spatial detector.

dirtRejectThreshold - for any region of dirt flagged by the motion detector, if the percentage of pixels flagged as dirt by the spatial detector is over dirtRejectThreshold then the region is assumed to be dirt.

+complexMotionDetection - where complex motion exists in the sequence it is likely that the motion based dirt detection scheme will fail. A complex motion detector is required to flag these regions and allow a modified dirt detection scheme to be used. The complex motion detector works by dividing the image into blocks and looking at the consistency of the estimated motion for each block across five frames. Once each individual block has been assigned complex motion or not, a smoothing algorithm is applied to the block to generate the complex motion matte for the image.

detectComplexMotion - whether to use the complex

motion detector.

complexMotionThreshold - this threshold is the level below which we declare a block of image to be undergoing complex motion. A lower threshold will increase the amount of image flagged as complex motion.

complexMotionSmoothing - this is the amount of smoothing applied to the blocks of image that have been detected as moving with complex motion. More smoothing will increase the amount of image flagged as complex motion.

dirtMaskComponent - defines where to get the (optional) dirt matte from.

- **None** - don't provide any dirt.
- **Source Alpha** - use the alpha channel of the source as the dirt matte.
- **Source Inverted Alpha** - use the inverse of the alpha channel of the source as the dirt matte.
- **DirtMask Luminance** - use the luminance of the dirt input as the dirt matte.
- **DirtMask Alpha** - use the alpha of the dirt input as the dirt matte.
- **DirtMask Inverted Luminance** - use the inverted luminance of the dirt input as the dirt matte.
- **DirtMask Inverted Alpha** - use the inverted alpha of the dirt input as the dirt matte.

Examples

The images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

RollerBlade

This clip of a roller blader suffers from a lot of dust.

Step-by-Step

1. FileIn the roller blade clip. Play it and look at the dirt.
2. Load the F_DirtRemoval macro from the Furnace tab and select the roller blade clip again.
3. Set the plateSize to PAL/NTSC and render.

Figure 15.3 Roller blader.

4. Note that the alpha channel contains a matte of the detected dirt.

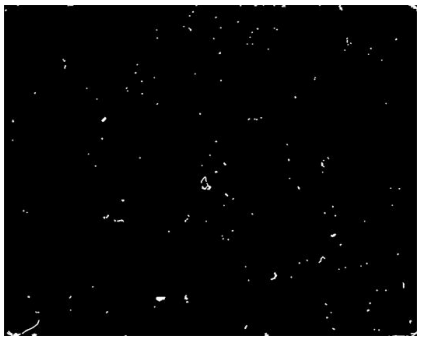


Figure 15.4 Dirt as a matte.

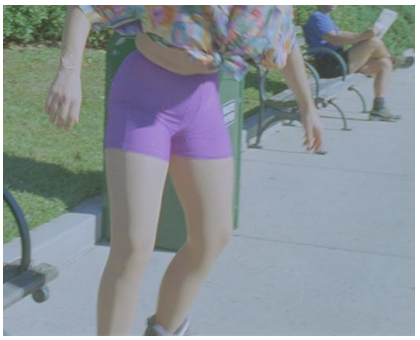


Figure 15.5 Repaired image.

If you wish to view and edit the detected dirt you can connect F_DirtRemoval into a QuickPaint node into another F_DirtRemoval node. The first dirt node generates the matte which can be edited by the paint node. The matte is fed into the second dirt node which uses that to repair the clip. If you spawn two viewers you can see the results of your painting in the clean plate. Figure 15.6 shows how the interface might look.

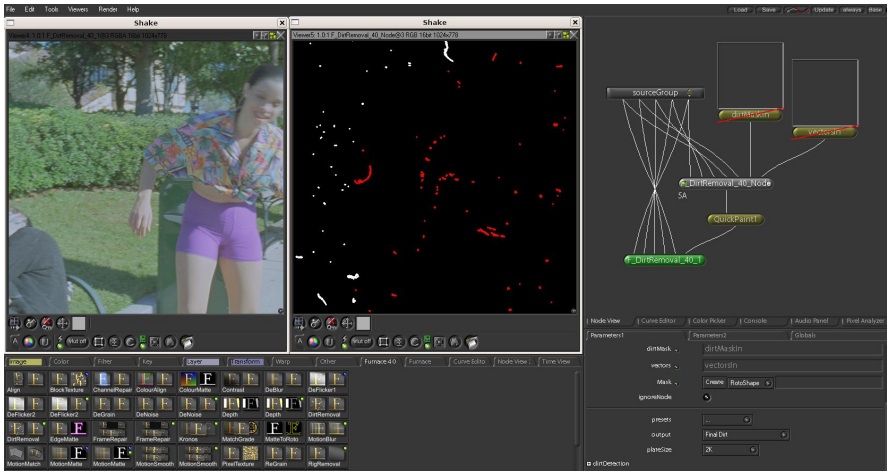


Figure 15.6 Roller blader. Workflow example for F_DirtRemoval.

EdgeMatte

This chapter looks at the creation of mattes from edge information in images using Furnace's assisted roto plugin F_EdgeMatte. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_EdgeMatte is a snap to tool designed to save you time whilst rotoing. Using its built-in roto tool, you simply need to draw a very loose roto around the object of interest, and F_EdgeMatte will convert it to a more accurate roto that has been snapped to the nearest edges in the original.

Background

Having been given the loose roto, the algorithm automatically finds the most likely edge position and displays it in purple, as shown in Figure 16.1. Selecting **outputRoto** will then turn this outline back into a standard Shake rotoshape (Figure 16.3 on the next page).

Figure 16.1 F_EdgeMatte

Although the algorithm uses spatial information only, the output rotoshapes operate as normal, allowing keyframes to be set, in exactly the same way as you would perform a manual rotoshaping job.

F_EdgeMatte uses colour edge information to try and fit the outline of an object. It will only work if a reasonable edge exists. The parameter **showGradient** displays the information

Figure 16.2 Input Roto

Figure 16.3 Output Roto.

the algorithm uses to find the edge. If no obvious edge is visible in this image, it is unlikely to give a good result. If the edge is very soft, such as hair, fur, or extreme motion blur F_EdgeMatte is not suitable and F_ColourMatte should be used instead.

The trick to this plug-in is to know when to stop adjusting the purple edge and to convert to a normal roto. This takes some practice.

Quick Start

Connect an F_EdgeMatte plug-in to the clip to be rotoed. Using the on-screen roto tool, roughly outline the object to be cut out. When the roto is built an outline of the object is displayed. Drag any of the points of the roto to force the outline to snap to the required edge. If you need to add more points to get a better fit, use Shift-Click on the line.

When you are happy with the purple (**edgeColour**) outline, select **outputRoto** and the edge will be converted to a roto-shape. The number of points in the output roto can be adjusted using **segments**. This output roto performs just like any other Shake rotoshape.

This rotoshape can be edited, if required. However, those changes will be lost on that keyframe if the input roto is subsequently adjusted at that frame. Edits to other output rotoshape keyframes will not be affected.

Warning! **If segments is changed, any edits to the output rotoshape you have made will be deleted on ALL keyframes.**

If the edge won't snap to the outline of the object look at **showGradient** to see if the edge is visible in the gradient image. If it is and the edge still wont snap to the object try increasing **snapToRange** and **pathAspect**. If you still can't get it to follow an edge you need to move on, so output the roto and adjust that.

It's worth also trying F_ColourMatte.

Finally, when rotoing complex shapes it is often best to divide the shape into smaller overlapping shapes and roto those. These mattes can be combined together later and give more flexibility if some parts are more difficult to roto than others.

Suggested Workflow

When using F_EdgeMatte to roto an object in a sequence it is recommended that you follow this workflow.

1. On the first frame, loosely roto around the object. Check that the purple edge snaps to the object. Add more points if required. If the plug-in won't follow a particular path dont spend too long trying to get it to work - it'll be quicker to fix it in the output roto.
2. Advance a few frames and move the shape to follow the object. The edge should snap to. You may need to move some of the individual vertices. Its worth at this stage having a quick look at the output roto. Switch this on and click back a few frames on the timeline to check the roto is following the object. If not, you may need to add more keyframes in between the ones you have already set.
3. Repeat this until the end of the sequence.
4. Click on outputRoto and look at the results on several frames throughout the sequence. If the roto isn't following parts of the edge very closely increase the number of segments.
5. Any edits you need to do to the output roto should be done now.
6. That's it.

Note **Saving an F_EdgeMatte script saves the input and output rotoshapes but not the purple edge path. Therefore, when loading a saved script, in order to successfully reset or change the number of segments of the outputRoto, you must first play through the timeline with the outputRoto toggle unset.**

Inputs

F_EdgeMatte has one input, the source, which is the clip to be rotoed.

Parameters

The parameters for this plug-in are described below.

outputRoto - switch this on and off to toggle between the edge path around the object and the output rotoshape. It is easier to correct any errors whilst viewing the outline path, but when you are happy with the shape select outputRoto to display the output rotoshape.

Note **Pressing this is quite safe. It does not affect any edits you may have made to the output roto.**

segments - specifies the number of new output segments required in between each input segment. For example, if **segments** = 2 then one vertex will be inserted between each pair of input vertices as shown in Figure 16.4. If the object has a complex edge, a greater number of output points in the rotoshape may be required to accurately model the shape. For simple shapes with smooth edges a smaller number may be sufficient.

Figure 16.4 2 Segments.

Figure 16.5 5 Segmnets.

Warning! **Changing the number of segments after edits have been made to the output roto will delete those changes on ALL keyframes.**

resetOutputRoto - press this to remove any changes that you have made to the output roto. All keyframes will be reset.

Warning! **Warning! Use with care!**

snapToRange - the algorithm looks around the input points to find the most likely edge position to start from. snapToRange specifies over how many pixels it searches. If this is set small the initial input roto must be quite tight, otherwise the edge will not be within the search range. If it is set large the initial input roto can be looser, but there is a greater chance of finding the wrong edge.

rectangleAspect - after snapping the input roto points to the edge, the algorithm finds the best path between them.

rectangleAspect specifies the search width of the path as a function of the path length. If the **rectangleAspect** is set too small (Figure 16.6) the path will be constrained to being close to a straight line, too big and it could wander off the correct edge onto another nearby edge.

Figure 16.6 Top rectangleAspect=0.5 Bottom rectangleAspect = 0.1

showGradient - display the image gradient information that is used to find an edge. If the edge is not visible in this image it is unlikely that the algorithm will be able to find it.

+edgeColour - the colour of the edge drawn by the algorithm.

+snapToColour - the colour of the 'snap to' vertices used by the algorithm.

FrameRepair

This chapter looks at replacing missing or damaged frames from clips using Furnace's plug-in F_FrameRepair. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_FrameRepair uses the Foundry's advanced motion estimation technology to quickly replace a damaged or missing frame by interpolating pixels from images either side.

Figure 17.1 Damaged frame (top centre) repaired by taking the previous and next frames to construct the missing frame.

Quick Start

To replace a missing or damaged frame, F_FrameRepair needs at least one good frame either side from which to create the repair. Duplicate the clip at least twice and connect one clip to the first (**source**) input. Depending on the frames you have available, connect the duplicated clips to two or more of the other source inputs. The **sourceMinusTwo** input is the source input after it has been shifted back two frames, **sourcePlusTwo** is the source input after it has been shifted forward two frames, and so on. Decide which of the surrounding frames you want to use for the repair and connect the clips to the appropriate inputs. Then use the timing tab on each clip to shift it back or forward the required number of frames. Finally, press the **toggleValidity** button to mark the current frame as invalid and in need of repair. The output of

the plug-in will now be a new frame generated from the inputs you supplied. This quality of this frame can be tuned using all the standard LocalMotionEstimation techniques (see the chapter on Local Motion Estimation on page 265 for more details). If only part of the frame is damaged, use a roto shape to comp the repaired section back over the original.

Inputs

F_FrameRepair requires at least three inputs to repair a frame and has eight optional inputs. **source** is the damaged or missing frame, while **sourceMinusTwo**, **sourceMinusOne**, **sourcePlusOne** and **sourcePlusTwo** are the four frames immediately surrounding it. At least one frame from either side must be connected in order for a repair to be done. In order to stop interference between any foreground objects and the background during motion estimation, a matte of the foreground objects can be optionally supplied. The **matte** input is the matte of the foreground for the source input, **matteMinusTwo** is the matte of the foreground for the **sourceMinusTwo** input, and so on. When using a foreground matte, the matte input corresponding to each of the connected source inputs should be connected.

F_FrameRepair also has a macro that can be used to connect a file sequence into its inputs automatically. When you create an F_FrameRepairMacro node, you will be prompted to choose an input sequence, and the appropriate frames from this sequence will be connected to the five source inputs for you. The macro will additionally connect dummy **matteIn** nodes to the other inputs that can be easily replaced if desired.

Parameters

The parameters for this plug-in are described below.

toggleValidity - press this to change the validity of the current frame. Note that this button will set key frames for the validity parameter below. Its exact operation will depend on the validity of the current frame and the frames surrounding it, but is designed to make the process of setting which frames need to be repaired as easy as possible.

validity - the validity of the current frame. A valid frame will be left alone, while an invalid frame (validity = 0) will be repaired.

The following parameters affect the local motion estimation used by F_FrameRepair. For an explanation of what they

do, please see the chapter on Local Motion Estimation on page 265.

vectorDetail -

smoothness -

filtering -

warpMode -

correctLuminance -

blockSize -

+tolerances -

matteComponent - where to get the (optional) foreground mattes to use for the motion estimation.

- **Source Alpha** - use the alpha of each source input.
- **Source Inverted Alpha** - use the inverted alpha of each source input.
- **Matte Luminance** - use the luminance of the matte inputs.
- **Matte Alpha** - use the alpha of the matte inputs.
- **Matte Inverted Luminance** - use the inverted luminance of the matte inputs.
- **Matte Inverted Alpha** - use the inverted alpha of the matte inputs.

Examples

The images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Carnaby Street

In this example, we will repair frame 4 of a 10 frame sequence which has some damage in the blue channel as shown in Figure 17.2 on the next page.

Step by Step

- Start Shake and FileIn the 10 frame clip carnaby.####.tif
- Look at frame 4 and see the damage.
- Duplicate the clip twice and wire the three clips into the F_FrameRepair node as shown in Figure 17.4 on the facing page.

Figure 17.2 Damaged frame. **Figure 17.3** Repaired frame.

- From the timing tab of the sourceMinusOne input clip, set the timeShift to 1 (the previous frame). For the sourcePlusOne input clip set the timeShift to -1 (the next frame).
- Go to frame 4 on the time line and press the toggleValidity button on the F_FrameRepair node to mark this frame as invalid and in need of repair
- Increase the vectorDetail to 1 and view the output of the F_FrameRepair node as shown in Figure 17.3.
- That's it.

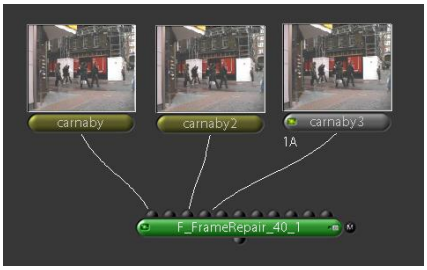


Figure 17.4 Node connections.

Kronos

Introduction

Kronos is Furnace's re-timer and is designed to slow down or speed up footage. It works by calculating the motion in the sequence in order to generate motion vectors. These motion vectors describe how each pixel moves from frame to frame. With accurate motion vectors it is possible to generate an output image at any point in time throughout the sequence by interpolating along the direction of the motion.

Background

Kronos contains a number of controls to allow you to trade off render time verses accuracy of vectors. Kronos uses Shake's built in time controls to generate arbitrary shaped speed curves. The plug-in will allow you to output both the re-timed image and the vectors used to generate the image. For more information on vectors generated by the Kronos macro see the Local Motion Estimation Chapter on page 265.

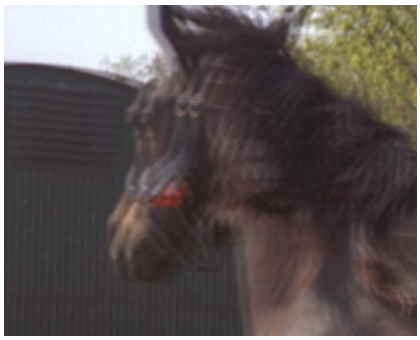


Figure 18.1 Simple mix of two frames to achieve an in-between frame

Figure 18.2 Kronos vector interpolation of the same two frames.

Quick Start

Kronos is a macro and initially prompts you to select a sequence to be re-timed. By default the speed control will be set to perform a half speed slow down. This is achieved by generating a new frame at position 0.25 and 0.75 between the original frames at 0 and 1. Frames are created at a quarter and three quarters instead of zero (an original frame) and a half so as not to include any original frames in the re-timed sequence. This avoids the pulsing that would otherwise be seen every other frame on a half speed slowdown.

Time Curves

To alter the speed open up the **sourceFrame** parameter, select and delete the current time curve. Next, make sure **sourceFrame** is keyframed. Then use the time bar to select an output frame and set **sourceFrame** to the input frame you want to appear at that output position. Repeat this for at least one more output position to get a linear time curve. For example, if we wish to do a 4 times slow down, move to frame 1 and set outputTime to 1, then move to frame 19 and set outputTime to frame 5. You can use all the normal shake time curve tools to create any time curve you might need. If the motion is speeded up, motion blur will be seen.

Alternatively, you can switch the **timing** popup to constantSpeed and use the **constantSpeed** control to set a single speed for the sequence.

Tuning Parameters

At this point you can render a re-timed sequence using the default parameter settings. Better results may be achieved by tuning Kronos using the Vector Generation Parameters described in the Local Motion Estimation Chapter on page 265.

Motion Blur without Retiming

You can add motion blur without retiming. To do this set the output time curve to be the same as the input time curve. Then set **shutterTime** (the output shutter time) to be a value greater than 1 (21.1 on page 129). Increase **shutterSamples** until you don't see multiple images (say 10). This will add motion blur along the direction of motion without retiming (21.4 on page 129). Alternatively, you can use the simpler F_MotionBlur plug-in, described on page 128.

Inputs

Kronos will prompt for an input sequence (**imageName**) when it is selected. It also has four optional inputs, described below.

- **warpName** If supplied, motion vectors will be calculated from this sequence and applied to the input sequence. This can be useful if, for example, your input sequence is very noisy, as too much noise interferes with the motion estimation, so you should supply a smoothed version of the sequence here.

- **matteName** This sequence will be used as a foreground matte. This can improve the motion estimation by reducing the dragging of pixels that can occur between foreground and background objects.
- **backgroundVectorsName** and **foregroundVectorsName** If the motion in your input sequence has been estimated before and you have the motion vectors available, you can supply one or more vector sequences to Kronos. This will save processing time, as Kronos will not then have to perform the motion estimation a second time. If your vectors were calculated using a foreground matte, you should supply the background and foreground vector sequences as **backgroundVectorsName** and **foregroundVectorsName** respectively. In this case you should also set **matteName** to the name of the foreground matte sequence used to create the vectors. If no matte was used in the creation of your vectors and you only have a single vector sequence, you can supply this sequence as either **backgroundVectorsName** or **foregroundVectorsName.**

Parameters

The parameters for this plug-in are described below.

imageName - this sequence is loaded on first starting Kronos and is the sequence from which motion vectors are extracted.

warpName - this sequence is warped using the motion vectors calculated from imageName.

matteName - this sequence is used to define the foreground object in cases where the background is being dragged along by the dominant foreground motion.

backgroundVectorsName - this sequence should contain vector fields written by F_VectorGenerator for the background of the sequence , if you wish to pre-calculate vectors rather than having Kronos calculate them internally.

foregroundVectorsName - this sequence should contain vector fields written by F_VectorGenerator for the foreground of the sequence , if you wish to pre-calculate vectors rather than having Kronos calculate them internally.

Note **If you do not wish to work with separate foreground/background motion layers, then you can just feed the vectors for a single layer into either the backgroundVectorsName field or the foregroundVectorsName field.**

fileControls - these are the standard file controls in shake.

method - sets the interpolation algorithm.

- **Frame** - the nearest original frame is displayed.
- **Blend** - a mix between two frames is used for the in-between frame. This is quick to render and useful when tweaking the timing on the curve before setting the method to motion.
- **Motion** - vector interpolation is used to calculate the inbetween frame.

timing - sets how to control the new timing of the clip.

- **Constant Speed** - select this if you wish to describe the retiming in terms of "double speed" or "half speed".
- **Source Frame** - select this if you wish to describe the retiming in terms of "at frame 100 in the output clip I want to see frame 50 of the source clip". You'll need to set at least 2 keyframes for the sourceFrame to retime the clip.

sourceFrame - this parameter is active only if timing is set to sourceFrame. Use this to specify the source frame at the current frame in the time bar. For example to slow down a 50 frame clip by half set the sourceFrame to 1 at frame 1 and the sourceFrame to 50 at frame 100. The default expression will result in a half-speed retime.

constantSpeed - this parameter is only active if timing is set to constantSpeed. Values below 1 slow down the clip. Values above 1 speed up movement. For example, to slow down the clip by a factor of two (half speed) set this value to 0.5. Quarter speed would be 0.25. This value can't be animated.

For descriptions of the following seven parameters, please refer to the chapter on Local MotionEstimation on page 265:

vectorDetail

smoothness

oversmooth

filtering

warpMode

showVectors

correctLuminance

output - sets the final output display for the re-timed image. Selecting anything other than **Normal** is only useful when doing a retime with separated motion layers.

- Normal - displays the motion interpolated image.

- Mask - displays the retimed matteName input
- Foreground - displays the retimed foreground layer - the background regions outside the matte input may show garbage.
- Background - displays the retimed background layer - the foreground regions inside the matte input may show garbage.

blockSize - please refer to the chapter on Local MotionEstimation on page 265

shutterTime - sets the equivalent shutterTime of the retimed sequence. A shutter time of 1 is equivalent to averaging over plus and minus half an input frame which is equivalent to a shutter angle of 360 degrees. A shutter time of 0.5 is equivalent to a shutter angle of 180 degrees. Imagine a grey rectangle moving left to right horizontally across the screen. Figures 21.1 and 21.2 show how **shutterTime** affects the retimed rectangle.

Figure 18.3 shutterTime 1

Figure 18.4 shutterTime 0.5

shutterSamples - sets the number of in-between images used to create an output image during the shutter time. Increase this value for smoother motion blur.

Figure 18.5 shutterSamples 2

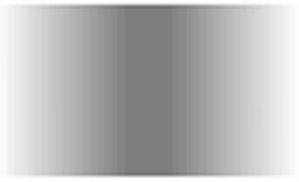


Figure 18.6 shutterSamples 20

automaticShutterTime - automatically varies the shutterTime throughout the sequence.

imageIsLog - if you are using the shutter parameters to apply motion blur and you are working with log images you should switch on **imageIsLog** so that the samples are correctly blended together.

+tolerances -

weightRed -

weightGreen -

weightBlue - for all of these parameters see the Local Motion Estimation Chapter on page 265.

matteChannel - the clip, matteName, can be used to help the motion estimation algorithm understand what is foreground and background in the image so that the dragging of pixels between overlapping objects can be reduced. White areas of the matte are considered to be foreground, and black areas background. Grey areas are used to attenuate between foreground and background. When the matteName input is filled with an appropriate clip, this popup controls how the pixel values in matteName are used to do the masking.

- **None** - don't mask.
- **Source Alpha** - use the alpha of the source input.
- **Source Inverted Alpha** - use the inverted alpha of the source input.
- **Matte Luminance** - use the luminance of the matte input.
- **Matte Alpha** - use the alpha of the matte input.
- **Matte Inverted Luminance** -use the inverted luminance of the matte input.
- **Matte Inverted Alpha** - use the inverted alpha of the matte input.

Examples

All the images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Taxi

This example, Figure 18.7 on the next page, shows a taxi driving past our offices in Soho. We'll retime this sequence to speed up the taxi at the start and slow down at the end.

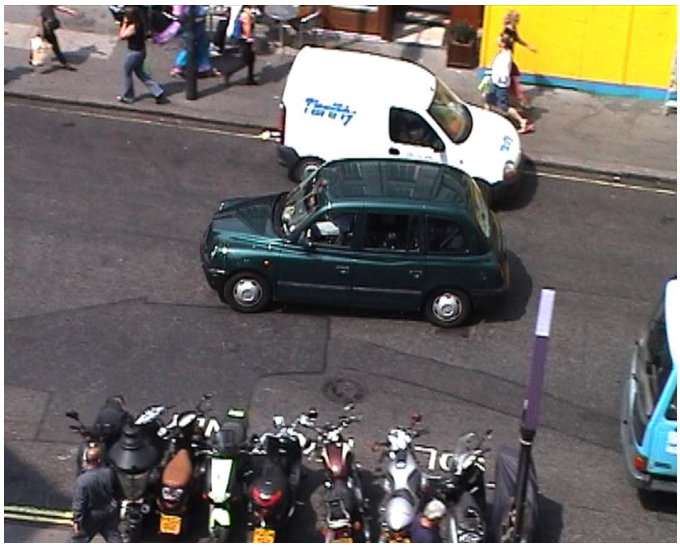


Figure 18.7 London taxi.

Step by Step

1. Start Shake and FileIn the taxi clip.
2. Flipbook the taxi clip to get a sense of the motion.
3. Select the Kronos plug-in from the Furnace tab. You are prompted to select a sequence to retime. Browse your directory structure again and load the Taxi.####.tif clip.
4. By default the clip will be slowed down to half speed. However, we wish to have a fast start and slow end. So at frame 1 set an sourceFrame key to be 1 and at frame 100 set an sourceFrame key to 50. Now to get that fast start, try setting the sourceFrame key to be 30 at frame 25. Use the Curve Editor to modify the animation graph.
5. To get a sense of the motion you can set the method to blend, render and adjust your timing curve from there.
6. When you're happy, set the method to motion. With the vectorDetail set to 0.2 you will see part of the road under the taxi being dragged by the taxi. To improve this increase this value to 1. To improve this further you'll need a matte of the taxi.

Taxi Matte

This example demonstrates how to use mattes to remove edge dragging.

Step by Step

1. Select Kronos from the Furnace tab and load in the taxi.

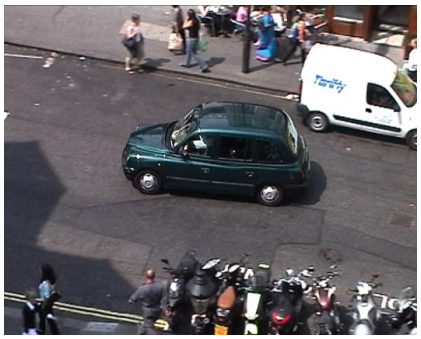


Figure 18.8 Taxi

Figure 18.9 Matte

2. Add the taxi matte by typing the path into the matteName text dialog or browsing for the clip.
3. Make sure foregroundMaskChannel is set to look at the correct channel for the matte. In this case it should be set to Luminance.
4. Since we're trying to reduce dragging of the road around the taxi we need lots of vectors, so set vectorDetail to 1.
5. Render.

MatchGrade

This chapter looks at automatic colour matching using Furnace's plug-in F_MatchGrade. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

It is often necessary to match the colours of one clip with those of another. When filming outside at different times of the day you will inevitably get colour and luminance differences that will have to be corrected if the sequences are to be composited or edited together.

You can, of course, use colour correction tools and trial and error to try and match the clips. But this tends to be time-consuming and requires some considerable skill. F_MatchGrade does it all for you by automatically modifying the colour histogram of an image to match a reference image.

This plug-in can also be used to add colour to black and white images.

Figure 19.1 Source image

Figure 19.2 Reference image

Figure 19.3 Output image.

Quick Start

Connect the source frame to the first input and the reference frame to the second input. View the output which should now match the look of the reference. Try increasing **iterations** if the match isn't close enough.

To apply a static transform to the first sequence, connect single frames to the **targetColour** and **sourceColour** inputs. F_MatchGrade will calculate the transform needed to match the sourceColour frame to the targetColour frame, and apply this transform to every frame of the **applyTo** sequence.

Inputs

F_MatchGrade has three inputs. The first, **applyTo** input is the sequence to which a colour transform will be applied. If only two inputs are supplied, this transform will make each frame of the **applyTo** sequence match the **targetColour** input**.** If a third, **sourceColour** input is supplied, F_MatchGrade will calculate the transform that matches the sourceColour input to the targetColour input and then apply that transform to each frame of the **applyTo** Sequence. This enables the connection of a single image into the last two inputs to ensure the transformation is temporally uniform, though generally F_MatchGrade is reasonably temporally consistent.

Parameters

The parameters for this plug-in are described below.

iterations - the number of refinement passes. More iterations should produce a better match but will take longer. This is an integer parameter, so animating it will not produce a smooth grade change but one with obvious steps. To achieve a smooth grade change, mix the output from F_MatchGrade with the original input sequence and animate the mix amount.

Examples

Mike

In this example, we will use F_MatchGrade to match the look of two clips, MikeWalking and MikeLightWand. The clips used here can be downloaded from our website. For more information, please see the Example Images section on page 17.

Download File

MatchGrade-Mike.tgz

Step by Step

1. FileIn MikeWalking.####.tif and MikeLightWand.####.tif – our goal is to make the image with the light stick lighter and more like the other. They are shown in 19.4 and 19.5.

Figure 19.4 Source image

Figure 19.5 Transfer image

1. Apply F_MatchGrade, using the walking clip as the first and third input, and light stick as the second. You should get the result in Figure 19.6

Figure 19.6 Output image

2. The result is slightly garish, so decrease the **iterations** parameter to 3, the result of which is shown in Figure 19.7.

Figure 19.7 Output image

MatteToRoto

This chapter looks at creating rotoshapes using Furnace's plug-in F_MatteToRoto. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_MatteToRoto is designed to take a matte from any source, for example Furnace's F_MotionMatte or F_ColourMatte plug-ins and turn it into a standard Shake rotoshape, thus making it easier to edit. The rotoshape is built by using an upper and lower threshold for the main and edge points of the output roto. If the input matte contains more than one disconnected object you will need to use a separate F_MatteToRoto plug-in for each object.

Rather than positioning the vertices at fixed intervals around the edge of the rotoshape, the plug-in can optionally attempt to position the rotoshape's vertices intelligently and so on areas of high gradient (corners) it will insert more vertices than on straight lines. The overall number of vertices used by the plug-in can be tuned by the user depending on how accurately you need the rotoshape to follow the matte.

For very complex shapes F_MatteToRoto can create a new rotoshape for each frame. Alternatively F_MatteToRoto allows you to create key frames at user specified positions. The position of the rotoshape on the in between frames will then be interpolated in exactly the same manner that you would expect from a standard Shake rotoshape. The plug-in also offers the ability to delete individual key frames or all the key frames.

Quick Start

Connect the matte image to the input of the plug-in. Position the crosshair near the edge of the matte you wish to turn into a rotoshape. The crosshair governs the position of the primary vertex. If the matte moves considerably, a feature should be tracked and the data applied to this point. Otherwise, the primary vertex can change sides and cause unwanted interpolation. Click on **createKeyFrames** to create a rotoshape for this frame. Either select **createAllKeyFrames** to create a key frame on every frame or scroll through the sequence creating key frames where necessary. Use the **deleteKeyFrame** or **deleteAllKeyFrames** button to remove key frames as necessary.

Inputs

F_MatteToRoto has just one input which is the matte image that is to be turned in to a roto.

Parameters

The parameters for this plug-in are described below.

createKeyFrame - manually creates a key frame on the current frame.

deleteKeyFrame - manually deletes the key frame on the current frame.

createAllKeyFrames - automatically generates a key frame every **skipFrames** frames.

deleteAllKeyFrames - deletes all key frames in the sequence.

position - sets the position of rotoshape. Position the cross hair near the edge of the matte you wish to turn into a rotoshape.

> **xpos** - the horizontal position of the start point
>
> **ypos** - the vertical position of the start point

upperThreshold - if the matte has a soft edge the plug-in thresholds using this value the matte to create the main vertices of the rotoshape.

lowerThreshold - if the matte has a soft edge the plug-in thresholds using this value the matte to create the main vertices of the rotoshape.

vertexDistance - the distance between vertices. Increase this for a more accurate rotoshape with more vertices.

numPasses - the number of attempts at planning the path around the roto. For more complex shapes, you may need to increase this parameter.

skipFrames - the gap between keyframes when **createAllKeyFrames** is pressed.

analyse - indicates whether the matte is in the luminance or the alpha channels of the input.

> **Alpha** - F_MatteToRoto will process the alpha of the input.
>
> **Luminance** - F_MatteToRoto will process the luminance of the input.

fill - whether to render the alpha channel.

Examples

The footage used in the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

BelleWalking

In this example, we will use F_MatteToRoto to create a roto from an inaccurate matte.

Step by Step

1. FileIn BelleMatte1-20.tif and view the matte in the alpha channel. Attach F_MatteToRoto.
2. Position the on-screen position widget as in Figure something. You can see from this image that the red outline for the upper threshold is OK, however, the blue one for the edge vertices is poor.

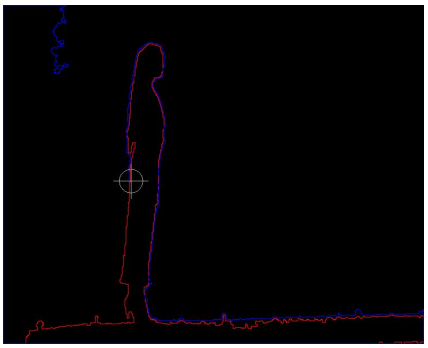


Figure 20.1 Output image

3. Set **lowerThreshold** to 0.2. The outside path is reasonable now, so click **createKeyFrame** to create a roto at that frame. It should appear as in Figure 20.2.

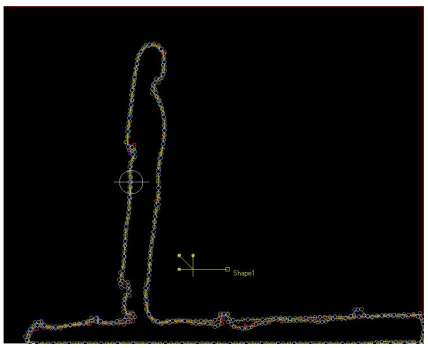


Figure 20.2 Output image

4. Got to frame 11, and create another keyframe. You should now be able to scroll through the timeline between frame 1 and 11, and get a reasonable result from the interpolation. If you wanted to create keyframes throughout the sequence, you could use the **createAllKeyFrames** button with a **skipFrames** of 1.

MotionBlur

This chapter looks at adding motion blur using Furnace's plug-in F_MotionBlur.

Introduction

MotionBlur uses the Foundry's advanced motion estimation technology to add realistic motion blur to a sequence. F_MotionBlur uses the same techniques and technology as the motion blur found in F_Kronos, but presents the controls in a less complex, more user friendly way. However if you need precise control over the motion vectors used for adding blur, or a large temporal range (i.e. a very high shutter time), you should use F_Kronos.

Quick Start

MotionBlur is a macro and initially prompts you to select a sequence to be motion blurred. Select a suitable **shutterTime**, depending on the amount of blur you wish to add. If you are working with log images turn on **imageIsLog**. Process the sequence to see the motion blurred result. Increasing **shutterSamples** will result in more inbetween images being used to generate the motion blur and so result in a smoother blur. If you can see that the motion blur has been created from a few discreet images, try increasing shutterSamples.

If your sequence is composed of a foreground object moving over a background the motion estimation is likely to get confused at the edge between the two. For better results try adding a matte of the foreground region to the **foregroundMatte** input. This will force the motion of the foreground to be calculated separately to the motion of the background.and so should produce less artefacts in the motion blur. Use **foregroundMatteComponent** to select which component of the matte to use.

Inputs

F_MotionBlur is a macro and will prompt for an input sequence when it is selected. Optionally select a matte of the foreground as the **matte** input to help improve the motion blur at foreground/background boundaries. Alternatively, you can supply pre-calculated motion vectors to the vector inputs, **backgroundVectors** and **foregroundVectors**. If you have separate vectors for the background and foreground you should

connect them to the appropriate inputs; if you have a single set of vectors you can connect it to either one.

Parameters

The parameters for this plug-in are described below.

shutterTime - sets the equivalent shutterTime of the re-timed sequence. A shutter time of 1 is equivalent to averaging over plus and minus half an input frame which is equivalent to a shutter angle of 360 degrees. A shutter time of 0.5 is equivalent to a shutter angle of 180 degrees. Imagine a grey rectangle moving left to right horizontally across the screen. Figures 21.1 and 21.2 show how **shutterTime** affects the retimed rectangle.

Figure 21.1 shutterTime 1

Figure 21.2 shutterTime 0.5

shutterSamples - sets the number of in-between images used to create an output image during the shutter time. Increase this value for smoother motion blur.

Figure 21.3 shutterSamples 2

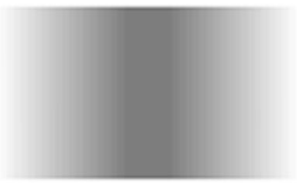


Figure 21.4 shutterSamples 20

vectorDetail - the amount of detail used for the motion estimation. The maximum value of 1 will produce the most accurate motion vectors, but will take longer to render.

imageIsLog - if you are working with log images you should switch on **imageIsLog** so that the samples are correctly blended together.

matteComponent - what to use as the (optional) foreground matte for the motion estimation.

- **None** - don't use a matte.
- **Source Alpha** - use the alpha of the source input.
- **Source Inverted Alpha** - use the inverted alpha of the source input.
- **Matte Luminance** - use the luminance of the foreground-Matte input.
- **Matte Alpha** - use the alpha of the foregroundMatte input.
- **Matte Inverted Luminance** -use the inverted luminance of the foregroundMatte input.
- **Matte Inverted Alpha** - use the inverted alpha of the foregroundMatte input.

Examples

All the images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

BelleWalking

This example, we'll use F_MotionBlur to add motion blur to the sequence.

Step by Step

1. Start Shake and FileIn the BelleMatte clip.
2. Flipbook the clip to get a sense of the motion. Got to frame 16 on the time line (this merely gives a better example than the initial frames).
3. Select the F_MotionBlur plug-in from the Furnace tab. You are prompted to select a sequence to motion blur. Browse your directory structure again and load the Belle-Walking.####.tif clip.
4. Set **shutterTime** to 10, and **shutterSamples** 10. You should get an image as in Figure 21.5 on the next page.

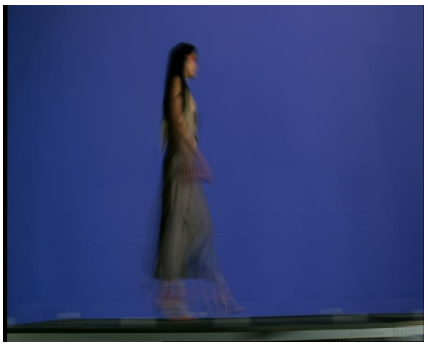


Figure 21.5 Output image

5. We can see the individual samples still, for example at the back of the dress and the righthand foot. Increase **shutterSamples** to 20 to sample more frames, and you should get a result as in Figure 21.6.

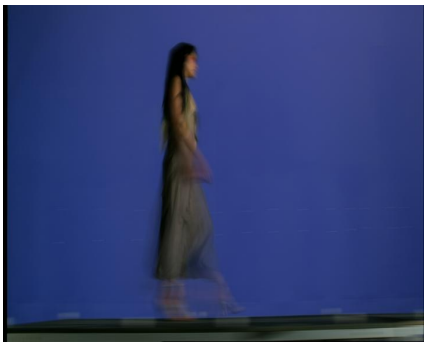


Figure 21.6 Output image

MotionMatch

This chapter looks at using the The Foundry's F_MotionMatch plug-in, which takes planar motion from one sequence and applies it to another. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_MotionMatch uses the Foundry's advanced global motion estimation to take planar motion from a reference sequence and apply it to another sequence. It can be used to do things like tracking a logo onto a moving background. For more of an overview of Global Motion Effects, and a description of the common way of working many of these effects have, please see the Global Motion Effects chapter on page 259.

F_MotionMatch is an Analysing Global Motion Effect, which pre-analyses a sequence for global motion and stores the calculated four corner pin as keyframed parameters. This analysis needs to be done before any useful output can be rendered. Once you've done the analysis, the keyframed four corner pins will be applied to the input sequence on any subsequent renders.

The analysis region is used to control which part of the reference sequence the motion is taken from. As with all the other Global Motion Effects, we are limited to working with motion that can be described by a four corner pin, so the area inside the analysis region should be moving in a planar way. F_MotionMatch also has a seed frame, which is the frame to which each of the other frames will be aligned in order to calculate the motion. The seed frame parameter will be set to the current frame whenever you change the analysis region. Because of this, the analysis widget differs slightly from those in the other Global Motion Effects, in that it will only appear solid when you are looking at the seed frames. On all other frames, it will have dotted lines, as is the convention across all Furnace plug-ins for a region which applies to a particular frame only.

Usually, you should choose a seed frame from near the middle of the sequence containing the motion you wish to match. This is so that the area covered by your analysis region will be common to as many frames as possible. Once this area goes out of shot, it will be impossible to perform the alignment.

Quick Start

Find two sequences: one you want to copy the motion from, and another you want to apply it to. Create an F_MotionMatch node. Connect the first sequence to its **Reference** input, and the second to its **Source** input. If necessary, change the **analysisRange** so that it contains all the motion you wish to copy. Move to a frame in the middle of this range, so that as many of the other frames as possible will have information in common with it. View the **Reference** sequence and position F_MotionMatch's **analysisRegion** over the area you wish to copy the motion from. Most of this area must be moving in a planar way, since we are limited to motion that can be described by a four corner pin. The analysis region should contain as much detail as possible, such as surface texture or edge information, to make it easier for F_MotionMatch to determine its motion between frames. Moving the **analysisRegion** sets the **seedFrame** parameter to the current frame; the **seedFrame** is the frame to which all others will be aligned in order to determine the motion in the reference sequence.

Now press **analyse**. F_MotionMatch will go through the sequence, aligning each frame to the seed frame and keyframing the resulting transformation into the four corner pin parameters. It will move forward through the sequence from the seed frame, then backwards from the same starting point. You'll see it setting keyframes along the timeline as it goes. Once it has finished analysing, you can render the output from the F_MotionMatch node. This will apply the keyframed four corner pins to the source input, transforming it so that it inherits the motion from the region you just analysed.

Inputs

F_PlanarPatcher has two inputs. The **Source** clip will have a four corner pin applied to it, so that it takes on some planar motion from the **Reference** clip.

Parameters

MotionMatch has all the Analysing Global Motion Effect parameters which are described in the 'Global Motion Effects' chapter on page 259. In addition, it has the following extra parameter.

seedFrame - this is the frame from which all motion is calculated. So if you want to position your source image in a particular place with respect to the reference image, this is the

frame you should be looking at when you set it up. This is also the frame you should look at when you position the analysis region. Note that when the region is changed, this parameter will automatically be set to the current frame.

Example

This section should be used in conjunction with the example images which can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Table

In this example we will be taking the motion from a moving image of a table, and applying it to a colour wheel. We will then composite the transformed colour wheel on top of the table image; because the motion is now the same, it will look as though it was part of the shot all along.

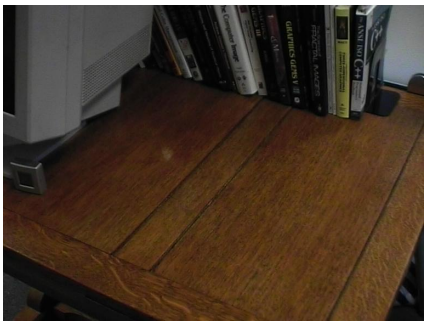


Figure 22.1 Textured wood table.

Step by step

1. Start Shake and FileIn Table.####.tif
2. Create a ColourWheel node and resize it to 100 by 100 pixels.
3. Go to the middle of the table sequence (frame 50).
4. Create a CornerPin node after the colour wheel. Use it to align the wheel with the table surface in this shot, as shown in Figure 22.2.

Figure 22.2 Colour wheel positioned on the table.

5. Create an F_MotionMatch node. Attach the output from the corner pin to the source input, and the table sequence to the second input. You'll see a message appear across the top of the window, warning you that the current analysis is invalid. Ignore this for the time being.
6. The colour wheel has an infinite time range, so change the analysisRange from "Source Range" to "Specified Range". The default range of 1 - 100 is fine in this example.
7. You should still be on frame 50; we're going to use this as the seed frame for F_MotionMatch. View the table sequence and the parameters for F_MotionMatch, so that you can position the analysis region. Move this region onto the surface of the table. This will automatically set the seed frame parameter, and to indicate that this is your seed frame the region will now be drawn with solid lines. Increase the size of the region so that it contains as much of the table surface as possible, as shown in Figure 22.3 - the more detail there is in the analysis region, the more precisely F_MotionMatch will be able to determine the motion.

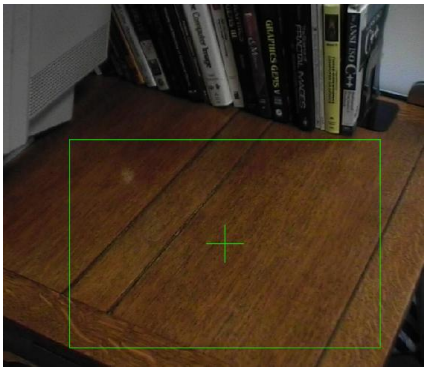


Figure 22.3 Analysis region.

8. Create an Over node. Connect the output from F_MotionMatch to the first input, and the table sequence to the second.

We're going to use this to composite the transformed colour wheel sequence onto the top of the table. Your tree should now look the same as that in Figure 22.4.

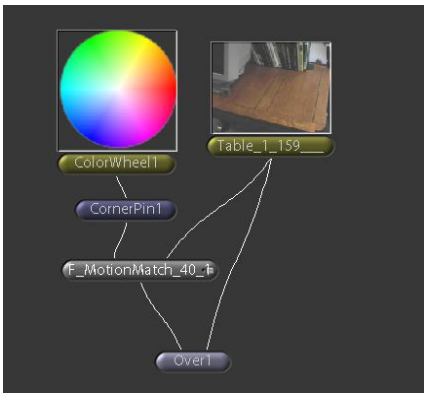


Figure 22.4 Shake tree.

9. Now press analyse in F_MotionMatch. The warning message will disappear, and F_MotionMatch will start analysing the motion in the sequence and keyframing the four corner pin. It moves forward through the sequence from the current frame, then backwards from the same place. You'll see it set keyframes along the timeline as it goes.
10. Finally, render a flipbook of the Over node to see the result. The colour wheel will now appear to move with the surface of the table, as if it had been present on the table when the shot was filmed.
11. That's it.

MotionMatte

This chapter looks at the automatic generation of mattes using Furnace's plug-in F_MotionMatte. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_MotionMatte uses a combination of The Foundry's advanced motion estimation and matte extraction technology to automatically segment a foreground object from its background and generate the foreground with background colour fringing removed, together with its associated alpha matte. There is no user input apart from selecting the sequence to segment.

The algorithm used by F_MotionMatte is highly complex and the processing times are very long; 30 seconds for a PAL frame would be typical. However, there are no tuning parameters for the plug-in and the proposed work flow is that the artist tries it on a few frames, and then if the results look useful background renders the sequence whilst having a cup of tea and a biscuit. If the results of the first few frames look unhelpful we suggest you move on and try the other tools in Furnace for assisted rotoscoping such as F_ColourMatte and F_EdgeMatte. Alternatively, if the results are close but still need tweaking a bit, you can do this easily by applying F_MatteToRoto to the results and then adjusting the resulting roto points as desired.

The **rolling** control is on by default; in this case the plug-in reuses results from previously calculated frames to improve the current frame calculation, both in terms of speed and quality. This control should be turned off if the render is to be distributed across multiple machines on a render farm.

Quick Start

F_MotionMatte is a macro and initially prompts you to select a sequence to be segmented.

For better results, paint a garbage mask of the foreground and use this as the second input clip in the plug-in. This gives the algorithm a more precise idea of which areas of the image are foreground and which are background.

Inputs

MotionMatte has 2 inputs:

1. the clip to be segmented, imageName, selectable from a file browser.
2. an optional matte input (matteName) as a garbage mask around the foreground region. If this input is not connected the algorithm will automatically detect the foreground region.

Parameters

The plug-in has only one parameter.

rolling - when on (default) this control indicates that the plug-in should use the results from previous frames to help improve the quality and speed of calculation of the current frame. If the render is to be distributed to a render farm, this control should be switched off, as the results will then be affected by the number of frames any given machine does in a single render run.

Typical results

The following images show some typical results, composited over grey. Note that the plug-in doesn't handle motion-blurred foregrounds well. The best results are for sequences where the foreground has a distinct motion which differs clearly from the background, and where the foreground is a single compact mass occupying approximately half the screen area.

Man holding baby

See 23.1 on the facing page.

Moderate result. Much of the matte is well defined, although the fast-moving motion-blurred regions and the non-moving arm are poorly defined.

Family

See 23.4 on the next page.

Very poor result. The motion of the foreground relative to the background is too small.

Quadbike

See 23.7 on page 140.

A good result. There is some erroneous pickup of background regions.

Figure 23.1 Original

Figure 23.2 Alpha result

Figure 23.3 Foreground result over grey

Figure 23.4 Original

Figure 23.5 Alpha result

Figure 23.6 Foreground result over grey

Figure 23.7 Original

Figure 23.8 Alpha result

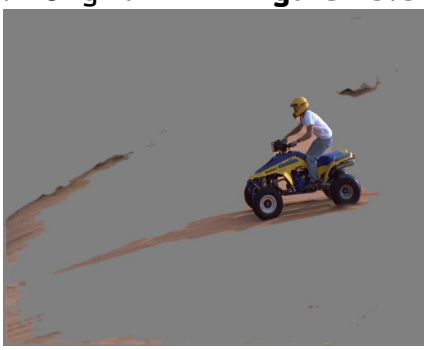


Figure 23.9 Foreground resulting over grey

Footballers

See 23.10 on the next page.

A good result. There is some poor definition of the matte where the footballers are arriving from off-shot on the left.

Figure 23.10 Original

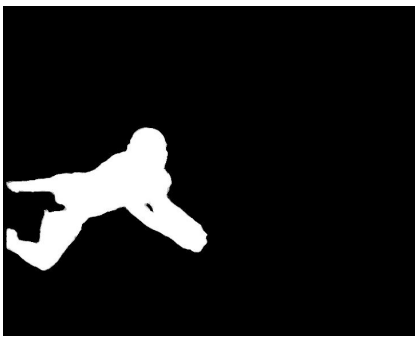


Figure 23.11 Alpha result

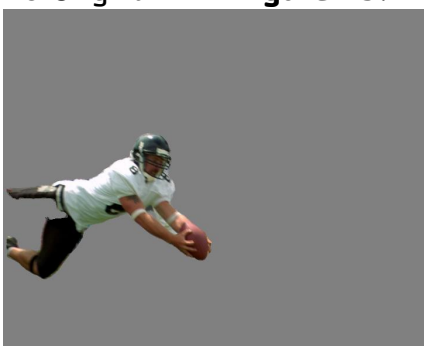


Figure 23.12 Foreground result over grey

MotionSmooth

This chapter looks at smoothing out boiling on clips using the Furnace plug-in F_MotionSmooth. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_MotionSmooth uses motion estimation to rebuild all frames in a clip. For each frame of the clip it aligns the frame before and frame after onto the current frame and then combines these three frames using a three way median filter. This process is very good at cleaning up temporal artefacts in a sequence. For example, it reduces noise and film grain, and if a prior algorithm has introduced temporal discontinuities such as flicker or boiling of parts of the image, these will be substantially reduced. Typical examples include flickery keys, boiling edge-detects and automatically generated Z-Depth mattes, such as the output from F_Depth.

Quick Start

Copy the clip you want to smooth twice, and connect the three clips to the first three inputs of the F_MotionSmooth node. Using the timing tab on the clips, shift the second input back one frame and the third input forward one frame. Render.

The quality of the motion estimation can be tuned using all the standard local motion estimation techniques. See Local Motion Estimation on page 265.

Inputs

F_MotionSmooth has three source inputs: the current (source), previous (sourceMinusOne) and next (sourcePlusOne) frames of the source clip. You can also supply an optional clip to smooth, which will then be smoothed using the motion vectors calculated from the source clip. Again, it is necessary to supply the current, previous and next frames in that order. A matte can also be provided to separate foreground and background regions and improve the motion estimation for occluded and revealed areas: the current, previous and next frames need to be connected and should correspond to those frames of the source. Finally, it is possible to supply F_MotionSmooth with motion vectors that have been calculated elsewhere, for example the output from F_VectorGenerator. This will save processing time, as F_MotionSmooth will not have to estimate

the motion again. If background and foreground vectors are available, they should be connected to the backgroundVectors and foregroundVectors inputs. In this case, a matte input must also be supplied, to tell F_MotionSmooth where to use the background vectors and where to use the foreground ones. If you have a single vector clip that was not calculated using a foreground matte, this can be connected to either of the vector inputs and it is not necessary to supply a matte.

Parameters

The parameters for this plug-in are described below and allow you to tune the way the local motion estimation is performed. See on page 265 for a more detailed explanation of how these parameters affect the motion vectors produced.

vectorDetail - Adjust this to vary the resolution of the vector field. The closer **vectorDetail** is to the maximum value of 1, the greater the processing time but the more detailed the vectors should be.

smoothness - The smoothness of the motion vector field. 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.

oversmooth - This is a computationally intensive smoothing operation that peforms a different vector-smoothing operation to normal. This generates highly smooth vector fields, at the expense of much lower detail.

filtering - sets the quality of filtering.

- **Normal** - use bilinear interpolation which gives good results and is a lot quicker than extreme.
- **Extreme** - uses a sinc interpolation filter to give a sharper picture but takes a lot longer to render.

correctLuminance - For clips where there is a global luminance shift, toggling this control on will allow F_MotionSmooth to take account of overall brightness changes between frames and should improve the motion estimation.

blockSize - This defines the width and height of the blocks used to generate the motion vectors. Smaller values will produce noisy data, larger values miss detail. This value should rarely need editing, but some sequences may benefit from using large block sizes to help the algorithm track regions better

where the algorithm isn't 'locking on' to the overall motion in the sequence.

tolerances - The tolerances allow you to tune the weight of each colour channel when calculating the image luminance, used for motion estimation.

- **weightRed**
- **weightGreen**
- **weightBlue**

matteComponent - where to get the (optional) foreground mask to use for motion estimation.

- **Source Alpha** - use the alpha of the source.
- **Source Inverted Alpha** - use the inverted alpha of the source.
- **Matte Luminance** - use the luminance of the matte.
- **Matte Alpha** - use the alpha of the matte.
- **Matte Inverted Luminance** - use the inverted luminance of the matte.
- **Matte Inverted Alpha** - use the inverted alpha of the matte.

PixelTexture

Introduction

Furnace includes a number of texture generation plug-ins. These are used to generate resolution independent images that have the same look and feel as a small sample region. F_PixelTexture is one such plug-in and should be used for replicating small scale textures, for example, tiny pebbles on a beach or a uniform fabric texture. If the source image has larger scale detail with noticeable shapes requiring some form of continuity across regions that are stitched together, F_BlockTexture should be used instead.

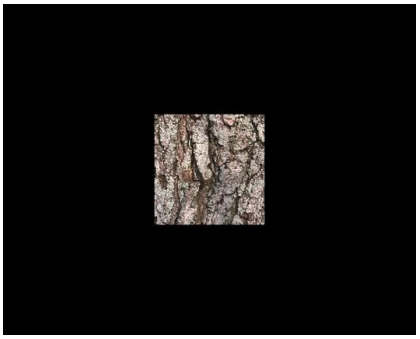


Figure 25.1 Small sample texture.

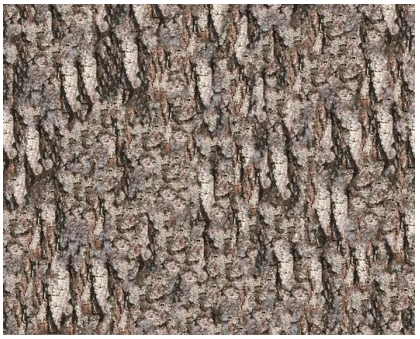


Figure 25.2 Large generated texture.

F_PixelTexture works by copying pixels from a source region in the original image in such a way that similar, but not necessarily adjacent, pixels are placed next to one another to try and generate more image with similar statistics to the source region.

It is also possible to supply a bias matte to this plug-in that inclines the pixel choice towards a particular colour. For example, if your source material contained pink flowers on a green background you could generate a bias matte with an arbitrary shaped pink region and arbitrary shaped green region. The new texture would then be generated such that pink flowers tended to appear in the regions defined by the pink areas in the bias matte and the green background tended to appear in the green regions.

Quick Start

Creating Textures

To create a texture from an image, simply connect the sample image to the first (**source**) input of the F_PixelTexture node.

View the input image and move the selection box over the area you want to use to generate the new texture. View the output of the PixelTexture node to see the result.

The size of the generated texture is taken from the image size of the source input. To create a large area of texture from a small sample swatch connect, say, a checker node to the first input and the sample image to second (**texture**) input. Make the checker any size you wish. View the second (**texture**) input and edit the parameters of the PixelTexture node. Move the on-screen box over the texture then view the output of the PixelTexture node.

Repairing Images

Connect the image you want to repair to the first (**source**) node of F_PixelTexture. Connect the texture image, that will be used to repair the source, to the second (**texture**) input. Connect a matte, Figure 25.4, that defines the region to be

Figure 25.3 Original Image.

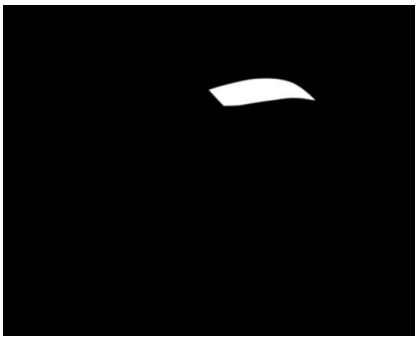


Figure 25.4 Matte of Barn

repaired to the third (**fillMatte**) input. Here, the original horse image has been fed into both the source input and the texture input. Move the sample region around the image until you get a good repair as show in Figure 25.5 where we have removed a barn by sampling the trees.

If there is an obvious colour gradient across the sample region switch on **removeGradient** and adjust **removeAmount** until no traces of the gradient can be seen in the new texture. If there is an obvious luminance change at the join between the new texture and the source image check on **luminanceMatch** and then adjust **lumianceBlur** until a better edge match is achieved.

To bias the output texture based on its colour, connect an image containing colours similar to the source texture to the fourth (biasMatte) input. This image would normally have full screen alpha so the effect is seen everywhere. If biasMatte is switched on the plug-in will tend to put pixels from the sample

Figure 25.5 Repaired Image.

Figure 25.6 Sample Texture, Bias Matte, Generated Texture.

texture into the locations in the biasMatte input that have a similar colour but only where there is full alpha. **ExaggerateBias** determines how rigidly the plug-in must follow the **biasMatte**. Increasing this value will follow the shape more.

Note **If you have zero alpha in your bias matte input, you won't see it working.**

Inputs

F_PixelTexture has four inputs. The first input (**source**) defines the size of the texture that will be generated. If it is the only input the texture will also be sampled from it. If the second input (**texture**) is connected it will be sampled rather than the source input. The third input (**fillMatte**) can be used to define the area that will be filled with the generated texture. This is then composited over the source input. Figure 25.7 shows the output of F_PixelTexture with a checker connected to the

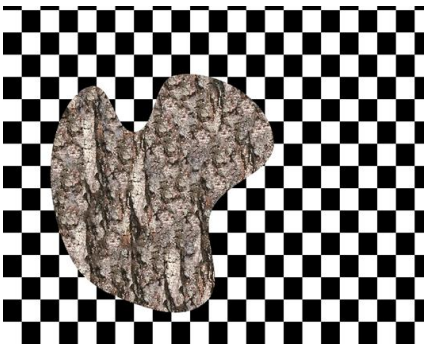


Figure 25.7 Masked bark.

first input, bark texture on the second input and a quickshape on the third. The fourth input (**biasMatte**) can be used to influence where the texture is generated. See Figure 25.6 on the preceding page.

Parameters

The parameters for this plug-in are described below.

fill - defines the area to fill with the generated texture

- **All** - fill the whole of the destination image, which will be the same size as the source image.
- **Source Alpha** - fill the region defined by the alpha of the source.
- **Source Inverted Alpha** - fill the region defined by the inverted alpha of the source.
- **FillMatte Luminance** - fill the region defined by the luminance of the fillMatte.

- **FillMatte Alpha** - fill the region defined by the alpha of the fillMatte.
- **FillMatte Inverted Luminance** - fill the region defined by the inverted luminance of the fillMatte.
- **FillMatte Inverted Alpha** - fill the region defined by the inverted alpha of the fillMatte.

seed - the random number used to start the texture generation process. Select another seed to see a different attempt at texture generation.

luminanceMatch - switch this on to match the luminance of the new texture with the luminance of the source region in order to help hide the join.

luminanceBlur - controls how much effect luminanceMatch has on the new texture.

bias - switch this on to position pixels according to the colour and alpha of the bias matte. If you don't have any alpha in the bias matte you won't see this working.

exaggerateBias - determines how rigidly the plug-in follows the colour guidelines defined by the bias matte.

removeGradient - if the source region contains a colour gradient across it the resultant texture will contain this gradient 'broken up'. RemoveGradient attempts to remove the gradient on the source material before generating the texture.

removeAmount - sets the amount of gradient that is removed by removeGradient.

smallerSegments - switch this on to copy smaller texture regions from the sample area. Let us consider the purple flower pattern. With smallerSegments off, whole flowers will be copied from the source texture. Switching this parameter on will force smaller patterns such as petals, but not whole flowers, to be copied. This can give better edges when trying to fit textures to a well defined bias matte.

sourceRegion - the sample area from the source input that will be used to generate the texture.

- **sampleRegionMin** - sets the bottom left position of the sourceRegion.
- **sampleRegionMax** - sets the top right position of the sourceRegion.

Examples

A variety of textures, including the horse and flowers, can be downloaded from our web site. For more information, please see the section on Example Images on page 17.

ReGrain

Introduction

Furnace's F_ReGrain plug-in is used to add grain to a sequence. It has been designed to sample an area of grain from one image and then to generate unlimited amounts of this grain with exactly the same statistics as the original. This new grain can then be applied to another image.

Background

Figure 26.1 Kodak 320.

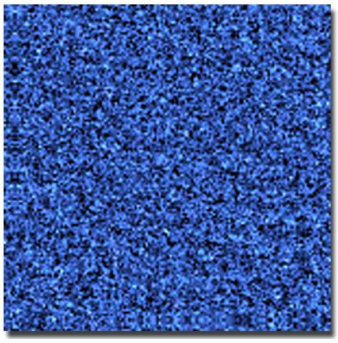


Figure 26.2 ReGrain.

Figure 26.1 shows an enlarged and exaggerated sample of grain from Kodak 320 film stock. Furnace's ReGrain was used to sample the original Kodak 320 stock and synthesize a plate of grain. The result is shown in Figure 26.2. Note that the grain characteristics closely match the original.

Similarly, Figure 26.3 is a sample from Kodak 500 film stock and Figure 26.4 shows this replicated using Furnace's ReGrain.

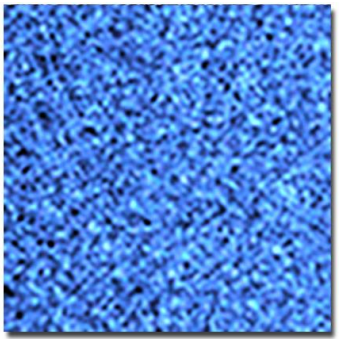


Figure 26.3 Kodak 500.

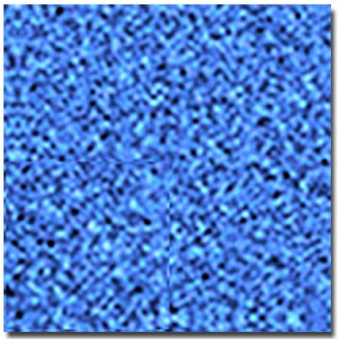


Figure 26.4 ReGrain.

Quick Start

Grain will be added to the first input and sampled from the second input. Connect the inputs to the F_ReGrain node as shown in Figure 26.5. You should be working at full resolution and not proxy resolution. ReGrain will not work at proxy resolution and will just pass through the source image. (See Proxy Resolutions on page on page 22.)

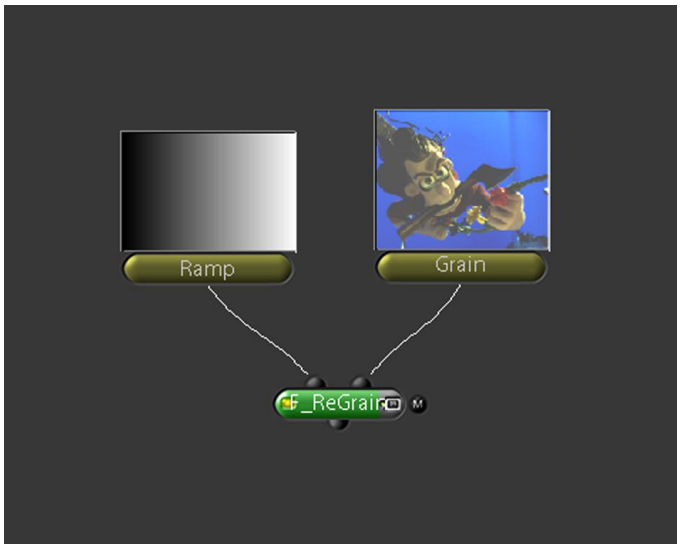


Figure 26.5 Node view showing sampled grain applied to a ramp.

Position the on screen sample region over an area of the sample image just containing grain and no picture detail. It helps to view the second input while editing the parameters of the F_ReGrain node. Your selection is important to get right. You should avoid any image detail or even a plain area that has luminance variations underneath the grain. The better this initial selection the better the result will be. See Figure 26.6 on the facing page. If you can't find a decent sample area on the frame, then try other frames from the same film stock. The default size of the sample area should be enough to gather information about the grain characteristics of your image. However, you may need to change its size and shape to fit over a plain area free of image detail.

Warning! **There is a minimum size of this sample area below which the statistical analysis of the grain will be unreliable. If the sample area you select is too small, you will see a warning message which prompts you to select a larger region. (See Proxy Resolutions on page on page 22.)**

The grain is sampled on a single frame which is set when you adjust the sample area (or by manual adjustment of the **grainSampleFrame** parameter). Although it is sampled on only one

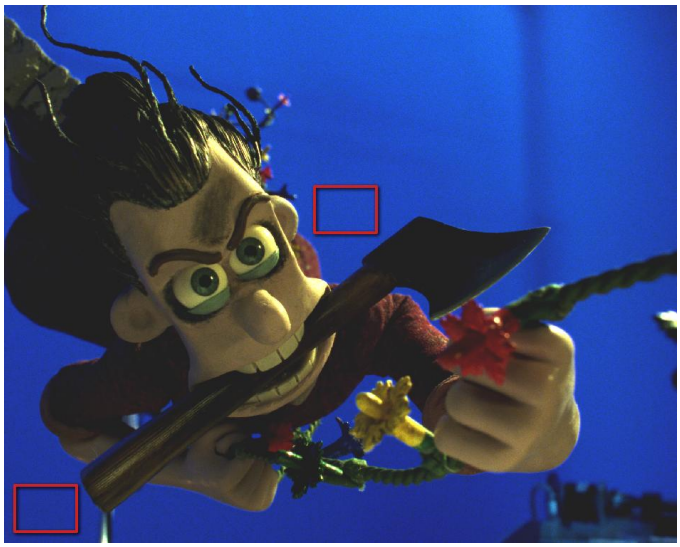


Figure 26.6 This shows two possible selection regions that contain no edge detail and little luminance variation.

frame, the algorithmically created grain will change from frame to frame but mirror the characteristics of the sample grain.

Once you have positioned the sample area, view the output of the F_ReGrain node to judge the results. The output will now contain the first input image with grain from the second input image applied. Both the size and the luminance of the new grain can be manually tweaked using **grainScale** and **grainGain** respectively.

Grain Stocks

To add grain from a standard film stock, only the first source input needs to be attached and the stock selected from the stock parameter list. 2K, 4K, aperture corrected and non aperture corrected stocks are included. Individual colour channels can be selected and adjusted using the **fineTuning** parameters.

Response

In its default setting, F_ReGrain adds the same amount of grain over the whole image. However, the amount of grain on an image is normally a function of luminance. Various parameters in the **response** group allow you to adjust how the amount of grain added varies with luminance. Pressing **sampleResponse** will cause the variation of the amount of grain with luminance to be calculated from the grain input, and checking on **useSampledGrain** will apply these curves to the grain added to the source. To view the sampled response curves, turn on **drawResponse;** an example is shown in Figure **26.7.** The

amount of grain added to the lowlights, midtones and highlights of the image can be adjusted using the **tonalResponse** parameters; the effect of adjusting these can also be seen on the response curves.

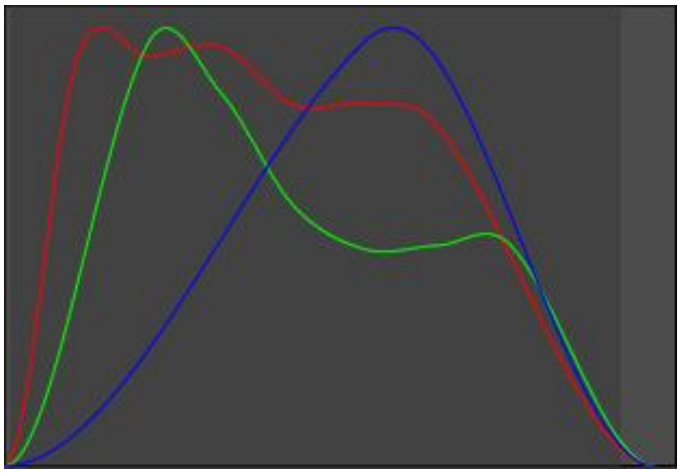


Figure 26.7 This shows an example of the grain reponse with luminance.

Checking the Result

To test that the new grain is the same as the old grain you can select **show – Grain Plate**.

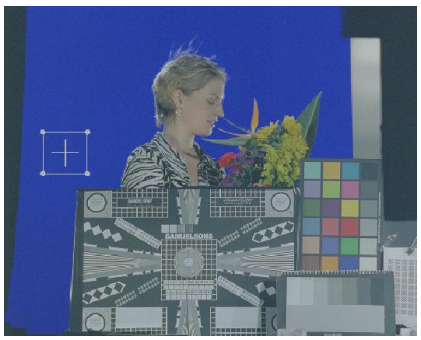


Figure 26.8 Good selection area...

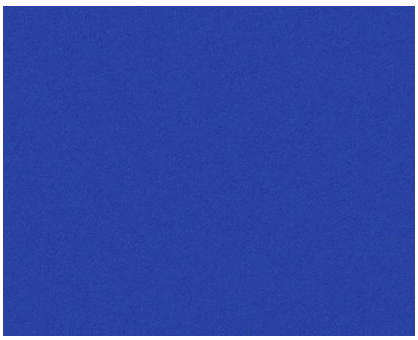


Figure 26.9 ...producing a good test plate of grain, free of artifacts.

This generates a sheet of grain with the same luminance level as the mean of the sample region. The sample region with the original grain is also displayed. If you turn off the on-screen controls it should be impossible to differentiate between the two regions. Figure 26.8 shows a good selection area giving a good test plate of grain in Figure 26.9. Figure 26.11 on the facing page shows a poor selection area since it contains image detail. Figure 26.11 on the next page shows the resulting test plate which clearly highlights the problem.

Figure 26.10 Bad selection area...

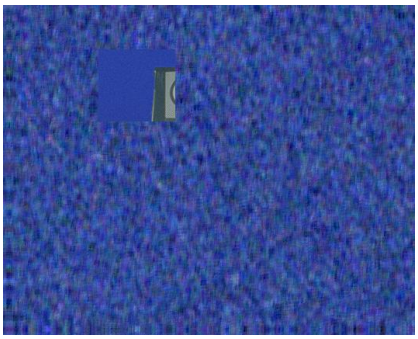


Figure 26.11 ...producing a poor result.

Proxy Resolutions

Grain manipulation at proxy resolution should be avoided as the results are unreliable. The grain selection area may be too small at proxy resolution to give a good result, and making this area larger may drag in unwanted detail from the image. If you try to use F_ReGrain at proxy resolution we simply pass the image through untouched and issue the following warning:

Cannot work at proxy scale.

We decided that this was preferable behaviour to doing poor grain replication at proxy resolution.

Note also that if you have your resolution set to base but have your pixelScale set to, say, 0.5 you are effectively working at proxy resolution and F_ReGrain will pass the image through untouched with a warning in the shell. PixelScale can be found in the useProxy settings on the Globals tab. You can, of course, crop the input clip and work with that rather than the proxy. There is a minimum size for the selection box, which is about 37x37 at base resolution. If the box you select is smaller you will get this warning along the top of the viewer,

Sample box is too small - please select a larger sample of the grain.

Inputs

There are two inputs. The first input is the image to which the grain will be added. The second input is the image from which grain will be sampled. If the supplied grain stocks are used only the first input needs to be connected. When a second input is connected, the plug-in will automatically switch to using grain sampled from this input. See Figure 26.5 on page 152.

Parameters

The parameters for this plug-in are described below:

stockType - selects whether the grain is sampled from the input image ("**Sample Grain**") or from a set of standard stocks. 2K, 4K, aperture corrected and non aperture corrected stocks are supplied. Although standard stocks are included it is recommended where possible that you sample from the film stock you are trying to match.

- **Sample Grain**- samples and reconstructs the grain characteristics from the second input.
- **<Stock><Exposure><Size>** - grain characteristics sampled from a supplied film stock. Common Fuji and Kodak stocks are supplied. The exposure can be under, over or, if left blank, non aperture corrected. The size is either 2K or 4K pixels. For example, FUJIF500 2K refers to the grain characteristics sampled from a 2K plate of Fuji Film 500 film stock non aperture corrected.

show - sets whether to render the result or a test image.

- **Result** - shows the input image with the grain applied.
- **Grain Plate** - shows a test image with the grain applied. This test image is composed from a section of the input image surrounded by a uniform solid colour sampled from the image with the grain applied (Figure 26.10 on the previous page). If the inner area is indistinguishable from the outer area then you have a good grain sample (Figure 26.9 on page 154.)

processRed - switch this on to process the red channel.

processGreen - switch this on to process the green channel.

processBlue - switch this on to process the blue channel.

grainScale - adjusts the size of the grain granules.

grainGain - adjusts the brightness per colour channel of the grain.

+fineTuning - the parameters under fineTuning allow detailed adjustment of the grain.

+gains - allows the brightness of the grain in the red, green and blue channels to be adjusted independently.

redGain - sets the brightness of the grain in the red channel.

greenGain - sets the brightness of the grain in the green channel.

blueGain - sets the brightness of the grain in the blue channel.

+scales - allows the size of the grain granules in the red, green and blue channels to be adjusted independently.

redScale - adjusts the size of the grain granules in the red channel.

greenScale - adjusts the size of the grain granules in the green channel.

blueScale - adjusts the size of the grain granules in the blue channel.

+response - the parameters under response allow the amount of grain added to be varied as a function of the image luminance.

+tonalResponse - allows the luminance of the grain to be adjusted in the lowlights, midtones and highlights independently.

lowGain - adjusts the gain of the grain in the lowlights.

midGain - adjusts the gain of the grain in the midtones.

highGain - adjusts the gain of the grain in the highlights.

useSampledResponse - switch this on to scale the brightness of the grain as a function of the luminance of the image.

sampledResponseMix - this control is usually set to 1. Decreasing it reduces the effect of the response curves until, at 0, they have no effect on the output.

sampleGrainResponse - press this to update the response curves from the current frame. Multiple presses accumulate the grain response rather than resetting every time.

resetGrainResponse - press this to reset the grain curves to their default (flat) response.

drawResponse - overlays the response curves on the bottom left corner of the viewer.

+grainSample - a selection box that marks the region of image used to analyse the grain. This part of the frame must contain no image detail, only grain. (Figure 26.8 on page 154).

sampleRegionX1 - sets the left position of the selection box.

sampleRegionY1 - sets the bottom position of the selection box.

sampleRegionX2 - sets the right position of the selection box.

sampleRegionY2 - sets the top position of the selection box

grainSampleFrame - sets the frame to sample the grain from.

Example

The image used in the following example can be downloaded from our website. For more information, please see the example images section on page 17.

Rachael

In this example, shown in Figure 26.12, we will degrain the image using F_DeGrain then, using ReGrain, sample the grain from the original image and reapply it to the degrained image. Switching viewer buffers between the original and regrained images will show different grain but with the same characteristics.

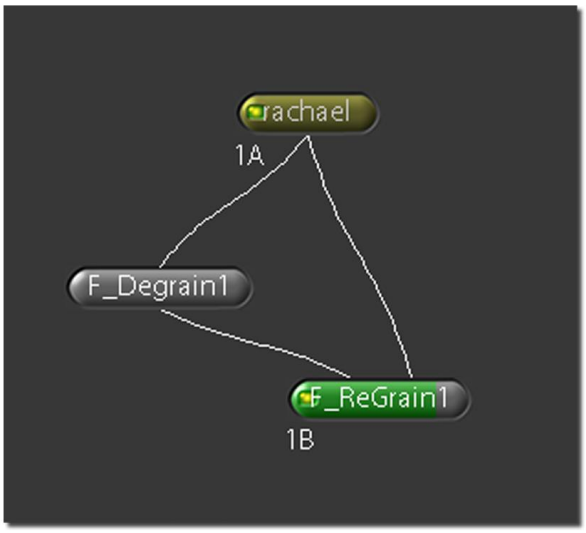


Figure 26.12 Test tree using F_DeGrain and F_ReGrain

Connect the output of Rachael.0001.jpg into the first input of the DeGrain node and the second input of the ReGrain node. Connect the output of the DeGrain node to the first input of the ReGrain node. Position the sample area on both nodes over a plain area and compare the original to the regrained. Note that although the actual grain differs on both images, the grain characteristics are the same.

Response

Replace the DeGrain node in the above example with a ramp. Switch **drawResponse** on and vary the response parameters and view the result. Note how the grain can be applied to mid-luminance values.

Now switch on **useSampledResponse,** and press **sample-GrainResponse** to update the response curves from the grain input. Note that the response curves are considerably more complex than those you can produce by adjusting parameters.

RigRemoval

This chapter looks at the removal of unwanted objects (rigs) from image sequences without accurate rotoscoping or keying to produce a clean plate. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

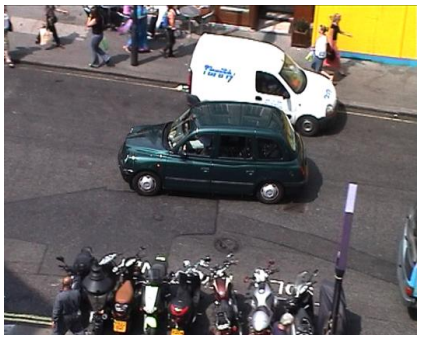


Figure 27.1 Before with taxi.

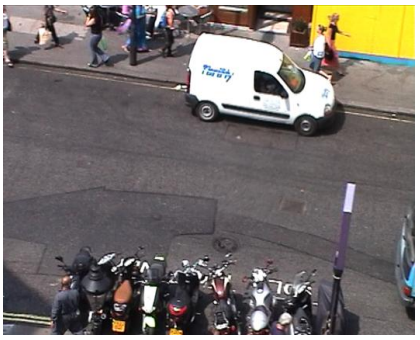


Figure 27.2 After applying RigRemoval.

Introduction

In this context we define a rig as a foreground element in a sequence that moves over a background element. The plug-in will only work satisfactorily if it is possible to model the background motion by a global 3D transform. For example, if the background contains multiple objects moving in different directions, the results will be poor. Typically, good results will only be achieved under the same circumstances that a skilled artist could generate, and track in, a clean plate in order to repair the sequence. However, this plug in should make the process quicker and easier. The rig removal algorithm works by estimating the background motion between successive frames, ignoring the foreground object, and then using the motion information to look forward and backward in the sequence to find the correct piece of background to fill in the missing region. The size of the missing region and the speed of the background dictate how far away from the current frame it is necessary to search to find the correct information.

Quick Start

F_RigRemoval requires the user to provide an image sequence and to define an area on each frame that will be repaired. If the image sequence has an embedded alpha channel, then that alpha can be used to define the area to be repaired. Alternatively, you can feed in a matte sequence to define this area.

Failing that, an on-screen rectangle can be used as shown in Figure 27.3. This should normally be keyframed to follow the object you wish to remove. To use the rectangle set repair to box. To use the alpha channel set repair to alpha.

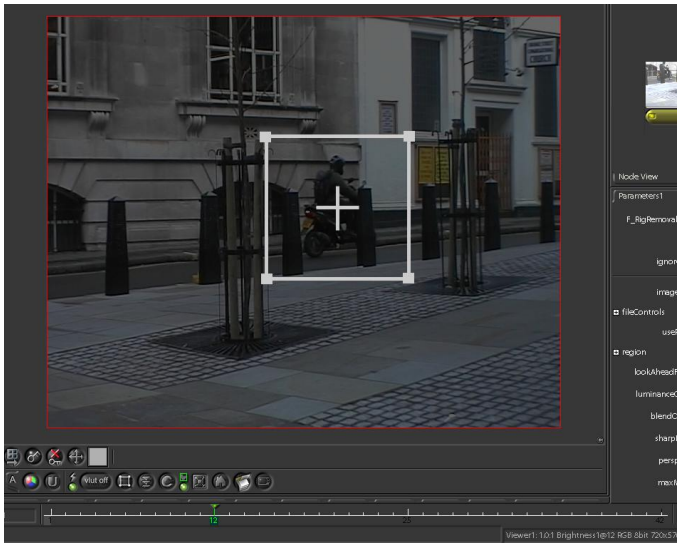


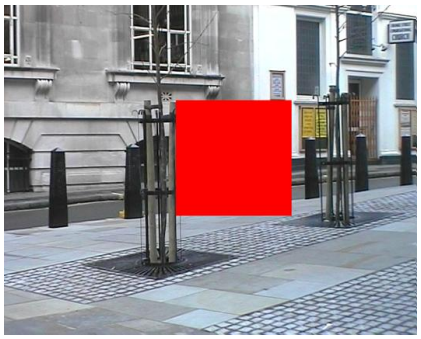
Figure 27.3 Rectangular region selector shown in grey.

In either case the region does not need to be the exact foreground region, just a rough outline, but you should avoid making it unnecessarily large as this will increase rendering time. Having defined the region to repair throughout the clip, set the parameter **numFrames** to the number of frames that the plug-in needs to analyse forwards and backwards to find enough data to repair the sequence. On the first frame this will be quite time consuming as the algorithm needs to estimate the motion between each pair of frames. Subsequent frames will be much quicker. If it has not been possible to replace all the foreground pixels, either because **numFrames** was set too low or the background information does not exist anywhere within the sequence, the pixels will be displayed in red. Try to adjust **numFrames** until no red pixels are visible and then render the sequence. In Figure 27.4 were using a box to define the pixels to replace and **numFrames** is set to zero. Increasing this value, as shown in Figure 27.5 on the following page, gathers pixels from other frames and improves the result. To completely remove the red pixels you'd need a **numFrames** value of 5.

Tip

RigRemoval is fairly slow to process which can make the initial keyframing of the rectangular region frustrating.

Sometimes the easiest way to adjust the region is to load up RigRemoval, and a FileIn node of the same sequence next



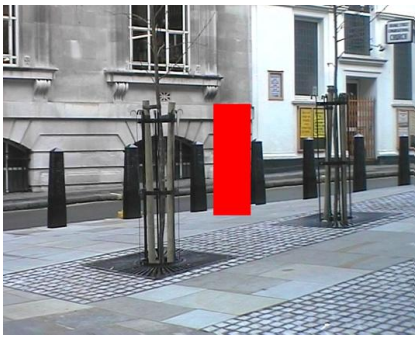


Figure 27.4 numFrames = 0. **Figure 27.5** numFrames = 3

to it. View the output of the FileIn node while editing the RigRemoval parameters, to save on the update speed. Animate the rectangular region over the foreground object you're trying to remove throughout the sequence. When you're happy with the region position, click back onto the RigRemoval output and wait for it to update. Slowly increase the numFrames parameter until the whole region is repaired, then check the output on other frames.

Occlusions

The algorithm used in F_RigRemoval is unable to differentiate between multiple foreground objects. If there is another foreground object in the sequence that moves through the background region that is being used in the repair, this second foreground object will also be cut up and used, resulting in an incorrect repair. To try and assist in these situations it is possible to mark regions of the image as not to be used for repair by setting their alpha value to mid grey. This will ensure that spurious bits of other foreground objects do not appear in the repair.

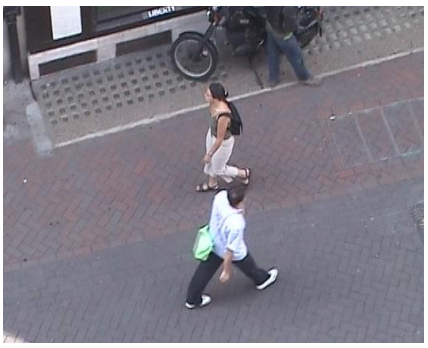


Figure 27.6 Original shot.

In Figure 27.6 we are trying to remove the woman in the centre of the screen as she walks from left to right down the street. At this frame a man walks in the opposite direction and her feet and his head overlap.

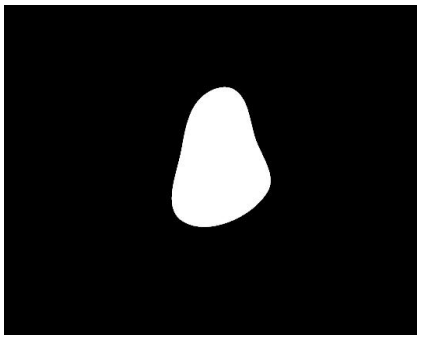


Figure 27.7

Figure 27.8

Figure 27.7 shows the normal matte for the woman and Figure 27.8 shows the result of using this in RigRemoval. Note that the man's head interferes with the repair and the reconstruction of the pavement is particularly bad, probably due to the man becoming the dominant motion source as the people occlude. To fix this we can adapt the matte to include a mid grey area over the man that tells the rig removal algorithm to ignore this in the repair. This matte is show in Figure 49 and the result is shown in Figure 50. Note that the repair on the pavement is improved and the man is simply clipped rather than being used in the repair.

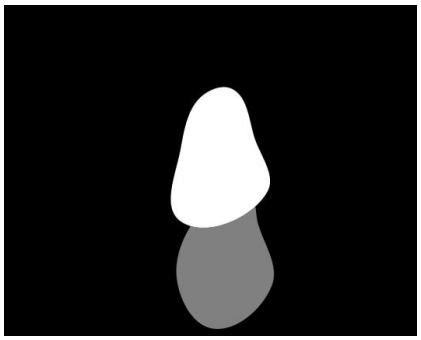


Figure 27.9

Figure 27.10

Inputs

RigRemoval will prompt for an input sequence when it is selected.

Parameters

The parameters for this plug-in are described below.

imageName - sets the name of the sequence to be processed by F_RigRemoval.

matteName - sets the name of the optional matte sequence that can be used to define the area to be repaired.

fileControls

This is the standard group of file controls in Shake. See the Shake documentation for more information.

repair - defines the area to repair.

- **Box** - repair the area inside a rectangular box, controlled by the box parameters below or the on-screen box.
- **Source Alpha** - repair the region defined by the alpha of the source input.
- **Source Inverted Alpha** - repair the region defined by the inverted alpha of the source input.
- **RigMask Luminance** - repair the region defined by the luminance of the rigMask input.
- **RigMask Alpha** - repair the region defined by the alpha of the rigMask input.
- **RigMask Inverted Luminance** - repair the region defined by the inverted luminance of the rigMask input.
- **RigMask Inverted Alpha** - repair the region defined by the inverted alpha of the rigMask input.

numFrames - sets the number of frames the algorithm should look forwards and backwards in the sequence to find the missing data. If you are getting red pixels then increase this value. See Figure 27.5 on page 162.

frameSpacing - if numFrames has to be set to a large number to make an effective repair, the rendering time can be prohibitive. frameSpacing will speed up the repair by not using every frame to fill the foreground region, effectively skipping frames, however, this may reduce the quality of the result.

maxMotion - defines the width of the border around the rig region that is searched in other frames to find the missing data.

luminanceCorrect - switch this on to correct for luminance changes from information taken from other frames. This is particularly important if the lighting changes throughout the sequence.

blendOverlap - the repair is built up using slices of information from other frames in the sequence. These slices can be overlapped and blended to give a more natural looking repair. This parameter controls how much the regions overlap. Increasing this parameter too much will degrade image sharpness.

filtering - sets the filtering quality.

- **Low** - low quality but quick to render.
- **Medium** - uses a bilinear filter. This gives good results and is quicker to render than high filtering.
- **High** - uses a sinc filter to interpolate pixels giving a sharper repair. This gives the best results but takes longer to process.

perspective - switch this on to correct for minor perspective changes.

direction - sets whether to search forwards, backwards or in both directions to find missing data.

- **Both** - searches before and after the current frame.
- **Forward** - searches frames after the current frame.
- **Backward** - searches frames before the current frame.

failMarkerAlpha - sets the level of transparency of the red pixels used to show where the repair has failed.

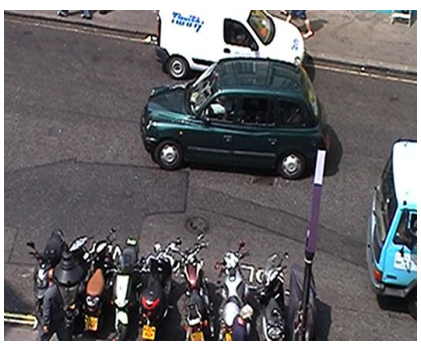
box - controls the size and shape of the rectangular area used in the repair.

Examples

The clips used in the following examples can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Taxi

This clip shows a taxi in London. There is an embedded alpha channel that roughly defines the taxi and will be used in this example to remove it from the scene. Figure 27.11.



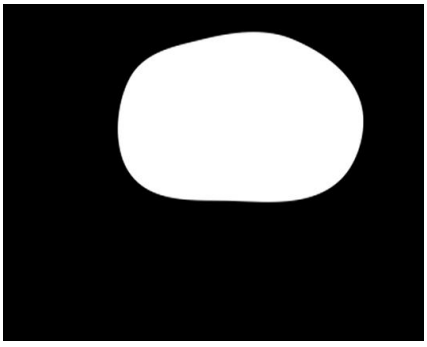


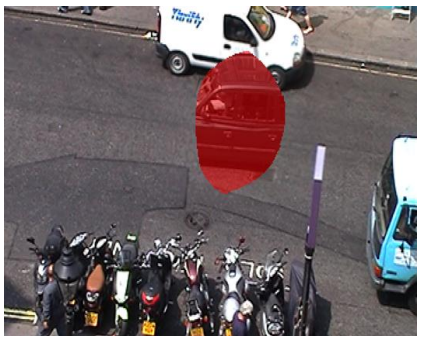
Figure 27.11 RGB Channels. **Figure 27.12** Alpha Channel

Step by Step

- FileIn the taxi clip. Play it. Have a look at the alpha channel.
- Load F_RigRemoval and choose the taxi clip as input.
- Look at frame 25. Set repair to alpha. With numFrames set to the default, 3, we get the image shown in Figure 27.13. The red area shows the pixels that cannot be found in other parts of the clip when searching forward and back 3 frame.

Figure 27.13 Fail area shown in red.

- Increase numFrames until the red shape fail marker disappears. See Figure 27.14 and Figure 27.15.



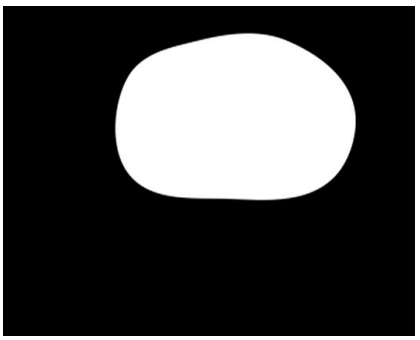


Figure 27.14 RGB Channels. **Figure 27.15** Alpha Channel

- Check a few other frames in the sequence to make sure the repair is good over the whole clip (no visible fail marker).
- Render.

Bike

This is a handheld panning shot of a motorcyclist riding along a street in London.

Figure 27.16 Before.

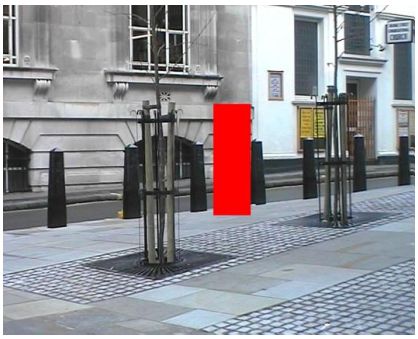


Figure 27.17 After.

Figure 27.16 shows the original clip with the motorbike in the centre. This is supplied in the examples/rigs directory. Figure 27.17 shows the clip with the bike removed.

Step by Step

- FileIn the bike clip and play. We're going to remove the first motorbike in the clip.
- Select F_RigRemoval. You will be prompted to load a sequence. Load the bike frames.
- There is no alpha channel so we'll need to keyframe the on-screen box over the motorbike in the sequence. For speed, view the FileIn bike clip and edit the F_RigRemoval parameters. Expand the box group and switch on keyframes for right, top, left and bottom. Position the on-screen rectangle over the bike on frame 1, move onto frame 10, reposition the box and repeat for frames 20, 30 and 40.
- Go back to frame 1, check repair is set to box and view the F_RigRemoval node. Increase the numFrames until the red pixels disappear. A value of 8 should do it.
- Render.
- You may see some red pixels creep into the shot. To fix this simply increase the numFrames to 12 and flipbook again.

Filling in the Gaps

Heres an excellent use of F_RigRemoval and F_Steadiness to fill in the gaps around an image after it has been stabilised. In Figure 27.18 on the next page we see that the image has been translated and rotated to stabilise it. This was done using F_Steadiness. Figure 27.19 on the following page shows the black area around the image filled with image data taken from elsewhere in the clip. This was done using F_RigRemoval. For a fully worked example, see London Eye on page 210.

Figure 27.18 Stabilised Image.

Figure 27.19 Filling in the Gaps.

RotoShape

This chapter looks at generating soft-edged shapes using the Furnace plug-in F_RotoShape. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_RotoShape is intended to be used in place of Shake's built-in RotoShape node, which sometimes processes edge points incorrectly.

Unlike Shake's RotoShape node, F_RotoShape has an optional input which can be used as a background for the resulting shape. When a clip is connected to the source input, the output will be the same size as the input. When no source clip is provided, the size of the output can be set via the **res** parameter, as is the case with Shake's RotoShape node.

In addition, F_RotoShape allows you to load and save rotos in raw format, which is not otherwise possible in Shake. This means that you can now import rotos from and export them to Autodesk® Systems Products. However, note that shapes from these applications which have had their axes tracked or animated will not contain this tracking data when imported into Shake, as it is saved separately from the roto points.

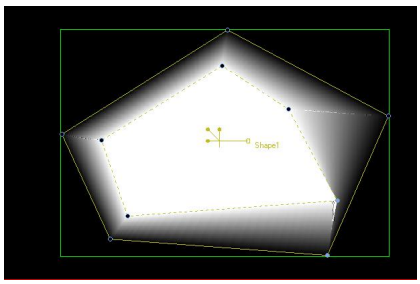


Figure 28.1 Output from RotoShape node showing errors at edge points.

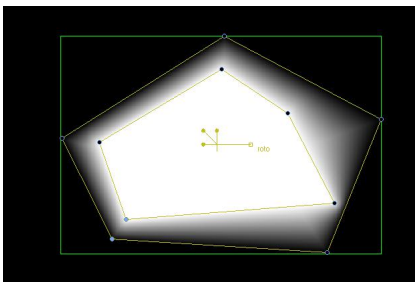


Figure 28.2 Output from F_RotoShape.

Quick Start

Create an F_RotoShape node. Choose an object to roto from an image sequence. View the sequence, select the F_RotoShape node's parameters and draw a roto around the object. The Adjust the edge points to get a nice soft edge. View the output from F_RotoShape: a soft-edged matte of your object.

Inputs

F_RotoShape has one optional input, **source**. When a sequence is connected to it, the resulting shape will be blended with the source according to the **blendMethod** chosen.

Parameters

The parameters for this plug-in are described below.

res - the resolution of the output image (when no input is provided).

falloff - how quickly the intensity of the shape fades away around the edges.

importRAWRoto - import raw format roto data.

exportRAWRoto - export roto in raw format.

blendMethod - how to mix the shape with the input image.

blendMix - the proportions to use for the mixing. 0 means just the background, while 1 is just the roto shape.

effectGain - the gain of the roto shape.

sourceGain - the gain of the background image.

Example

In this example we'll use F_RotoShape to selectively defocus an image.

1. File-in LisaBefore.tif.
2. Create an F_RotoShape node and connect the file-in node to it. We won't blend the shape with the input this time, but it's a convenient way to set the output size.
3. View the image of Lisa and select the parameters for F_RotoShape. Draw a roto around Lisa's face, as in Figure 28.3. Select edge points, and draw a roto around her hair, as shown. The output from F_RotoShape will be a soft-edged matte, as shown in Figure 28.4.

Figure 28.3 Lisa with roto.

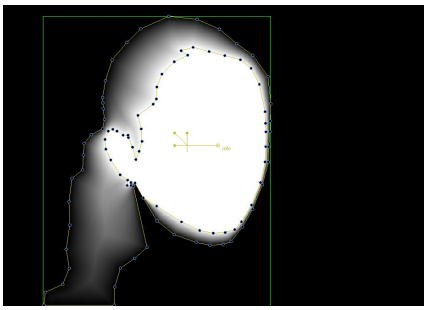


Figure 28.4 Matte.

4. Create a defocus node. Connect the image of Lisa to it and set pixels to 15. The output should look like Figure 28.5.

Figure 28.5 Defocused image.

5. Create a SwitchMatte node. Connect the image of Lisa to the first input and the output from F_RotoShape to the second.

6. Create a KeyMix node. Connect the defocused image to the first input, the output from SwitchMatte to the second input and the output from F_RotoShape to the third. Your tree should now look like Figure 28.6.

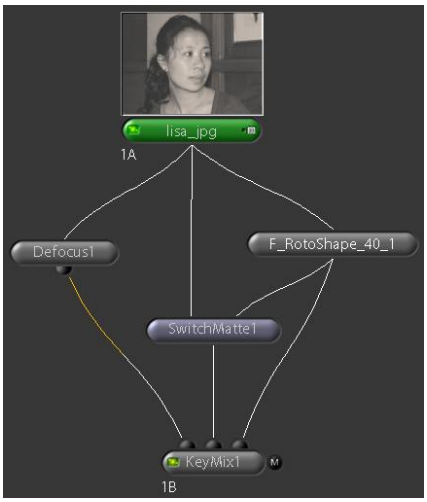


Figure 28.6 Shake tree.

The output from the KeyMix node is shown in Figure 28.7.

Figure 28.7 Final result.

7. That's it.

RotoTracker

This chapter looks at speeding up rotoscoping using Furnace's plug-in F_RotoTracker. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_RotoTracker can help speed up rotoscoping by tracking a roto round an object through a sequence of images. It works best on objects with strong edges, which allow points on the edge to be easily tracked.

Quick Start

Connect the sequence containing the object you wish to track to the input of F_RotoTracker. Set the **trackRange** parameter to the range of frames you wish to track it over. On the first of these frames, draw a roto around the object, taking care to position the roto points on the edges of the object to make them easier to track. Start the tracker by pressing the ▶ button in the tweaker panel. It will then track forwards from the current frame to the end of the track range, setting key frames as it goes. At any time, if one or more points on the roto seem to have wandered too far, you can interrupt the tracking and move them back to where you think they should be. This will set a user key frame, and you can then restart the tracker from your improved roto by pressing the ▶ button again. The other tracking controls in the tweaker panel are the same as those for F_Tracker, described on page 223, and are also outlined below.

If the tracking is not working well, try adjusting the parameters. See the Parameters section below for details of what they do.

Inputs

There is one input.

Controls

The following tracking controls appear on the Shake Viewer shelf, alongside the standard Shake rotoshape tools:

 **Create user keyframe** - creates a **user keyframe**.

 **Delete user keyframe** - deletes a **user keyframe**.

 **Track backwards** - plays backwards through the sequence tracking from frame to frame.

 **Step backward** - tracks backwards one frame.

 **Step forward** - tracks forward one frame.

 **Track forwards** - plays forwards through the sequence tracking from frame to frame.

 **Smart track** - tracks from begining to end of frame range in an intelligent order.

 **Delete track keyframes backwards** - deletes **track keyframes** backwards through the sequence until either a user key frame or the begining of the sequence is reached.

 **Delete track keyframe and step backward** - deletes a **track keyframe** and steps forwards one frame.

 **Delete track keyframe** - delete the current **track keyframe**.

 **Delete track keyframe and step forwards** - deletes a **track keyframe** and steps backwards one frame.

 **Delete track keyframes forwards** - deletes **track keyframes** forwards through the sequence until either a user key frame or the end of the sequence is reached.

 **Delete all track keyframes** - deletes all **track keyframes** from the sequence.

 **Delete all track And user keyframes** - deletes both **track keyframes** and **user keyframes**.

Parameters

The parameters for this plug-in are described below.

trackRange - the range of frames over which to track the object

trackRegionSize - the size of the regions to be tracked. There is a track region centred on each point of the roto.

springStrength - a high spring strength will act to preserve the distances between the roto points. Think of the lines connecting the roto points as springs - strong springs will pull the points back together if they try to move further away from each other, or push them further apart if they try to move closer together. Try a high spring strength if your object keeps roughly the same shape throughout the sequence. A low spring strength should help you to track objects whose shape changes or deforms.

temporalSmoothness - increase this to preserve the motion of the roto points through time.

Example

In this section we will look at an example that demonstrates the typical work flow for F_RotoTracker. The clip used in this example can be downloaded from our website. For more information, please see the Example Images section on page 17.

Step by Step

1. File-in the clip selection1_#####.tif.
2. Create an F_RotoTracker node, and connect the input clip to it.
3. Draw a roto around the ball in the first frame, as in Figure 29.1.

Figure 29.1 Initial Roto.

4. Start the tracker by pressing  and watch what happens.
5. After a while, one or more of the roto points will probably start to wander from the edge of the ball, like the point

circled in red in Figure 29.2. When this happens, click anywhere in the window to stop the tracker, and move the offending points back to where you think they should be. Then start the tracker again. Adjusting the roto sets a user key frame, so the tracking will continue from the new point positions. Hopefully, it should now track the ball well for several more frames before the points need adjusting again, though how long this takes will depend on the exact positions of your roto points.

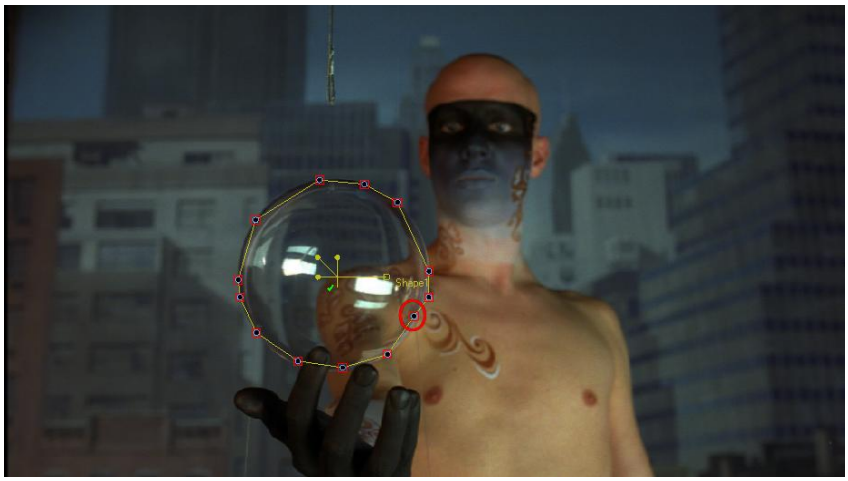


Figure 29.2 Wandering Point.

6. Finally, return to the first frame and press  to delete the track frames generated earlier. Increase the spring strength to 10, and start the tracker again. It will be slower this time, but notice how the distances between the points are preserved for longer. Also notice that this means the shape of the ball is preserved for longer.
7. That's it.

ScratchRepair

This chapter looks at the removal of scratches from images using Furnace's plug-in F_ScratchRepair. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_ScratchRepair is designed to remove scratches from film. Within Furnace, a scratch is defined as an artefact that appears on each frame, thus making it difficult to do a dirt removal style repair by taking clean pixels from the previous and next frame. If the scratch appears only on one frame, or intermittently, better results might be obtained by using F_DirtRemoval.

Figure 30.1 Image with scratches.

F_ScratchRepair uses a spatial repair algorithm which uses the values of nearby undamaged pixels to locally adjust the statistics of the scratched pixels. (In this respect it differs from F_WireRemoval, which ignores pixels inside the repair region, because it assumes all the image information is obscured by the wire.) The number of nearby pixels used in the algorithm trades off how much information is taken from the damaged pixels and how much interpolation is performed across the scratch. There is a gradient parameter that specifies the maximum angle of interpolation allowed across the scratch.

Figure 30.2 Before

Figure 30.3 After

Quick Start

Connect the clip with the scratch to the input of F_ScratchRepair. Choose the appropriate **scratchType** for your clip and position the on-screen scratch tool so that it covers the region you wish to repair. An example is given in Figure 30.4, which uses the **Curve Segment** setting. The scratch should disappear. If artefacts still exist try adjusting numPoints. If an edge passes oblique across the damaged region it may be necessary to increase **gradientWidthFactor**.

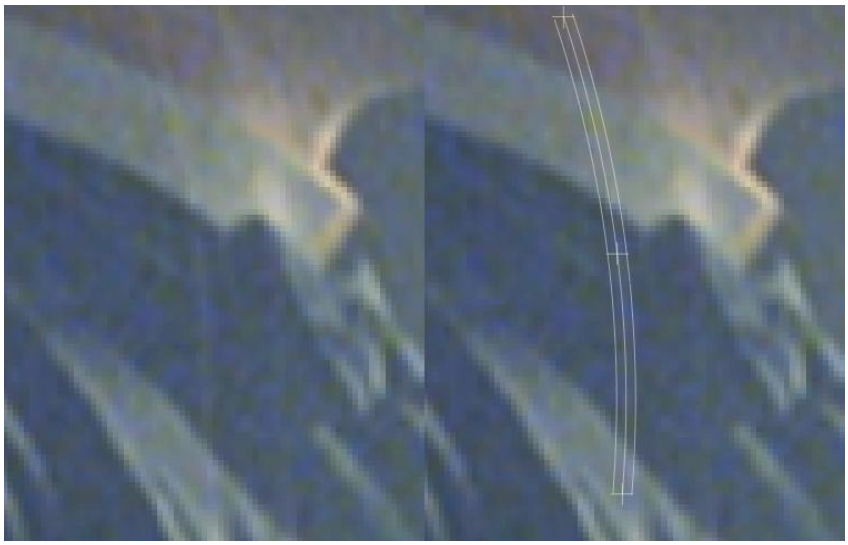


Figure 30.4 Example of positioning the widget.

To remove the scratch on subsequent frames it is necessary to ensure the scratch tool continues to be positioned over the scratch. If the scratch moves from frame to frame use a tracker, such as Furnace's F_Tracker, to track the position of the scratch and set the start, centre and end parameters.

Inputs

F_ScratchRepair has only one input: the clip containing the scratch.

Parameters

The region to repair can be set using the on-screen scratch tool, or adjusted using the **start**, **centre**, **end** and **scratchWidth** parameters.

output - the output can either be set to the repaired frame (repair) or a matte of the scratched region (repairMask).

- **Repair** - image with the scratch removed.
- **Repair Mask** - line matte defining the scratch. If you have successfully tracked the position of the scratch but failed to repair it, this matte may be useful in fixing the scratch using more traditional compositing methods.
- **Repair And Alpha Mask** - image with the scratch removed, plus a line matte defining the scratch in the alpha channel.

scratchType - changes the widget, and internal representation of the scratch, to enable the user to indicate more easily where the scratch is

- **Vertical** - Scratch is vertically aligned with the pixels. Many scratches are like this, and using this setting rather, than **Line Segment** or **Curve Segment,** will enable the plug-in to work quicker.
- **Horizontal** - the scratch is horizontally aligned with the pixels. Like **Vertical**, this option may be quicker that **Line Segment** or **Curve Segment**.
- **Line Segment** - the scratch is straight, but not aligned with the pixels in any way.
- **Curve Segment** - the scratch is curved. This setting allows the user to define a parabola that encompasses the length of the scratch.

start - the start position of the scratch.

centre - the centre of the scratch.

end - the end position of the scratch.

scratchWidth - sets the width of the repair to be made. Try to keep this value as small as possible.

spatialRepair - parameters to control the repair.

numPoints - the number of undamaged pixels used to alter the statistics of the scratched pixels. A value of 3 is similar to a simple interpolation across the damaged

region; values greater than 3 use increasingly more information from the nearby pixels to effect a more complex repair. If the scratch is completely opaque against a detailed background then try a value close to 3. For more transparent scratches with slowly varying backgrounds increasing numPoints should produce better results.

gradientFactor - defines the maximum angle of interpolation across the scratch. By default this is set to 45 degrees. Increasing gradientFactor will allow features that cross the scratch more obliquely to be correctly repaired but can also lead to spurious matches.

medianWidth - when reconstructing the scratched pixels rather than simply interpolating perpendicular to the scratch we choose the optimum angle across the scratch to interpolate a repair. This is done on a per pixel basis which can lead to a very noisy repair as the optimum angle can change rapidly. medianWidth is the width of a median filter that smooths the choice of angle in order to produce a less noisy repair.

filtering - sets the quality of filtering

- **Normal** use bilinear interpolation, which gives good results and is a lot quicker than extreme.
- **Extreme** use a sinc interpolation filter, which gives a sharper picture but takes longer to render.

ShadowRemoval

This chapter looks at removing shadows from images using Furnace's F_ShadowRemoval plugin. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_ShadowRemoval is designed to remove a shadow from an input image. It requires a matte of the shadow to be provided by the user. It is designed to be used when information is still present in the shadow region, only at a lower luminance, and performs no inpainting. It will not, therefore, work on footage where the shadow is completely black. The **shadowMatte** input allows you to use Shake's soft-edged fall-off roto capabilities to define the centre of the shadow, where the matte is saturated, and the penumbra region where the matte decays to zero outside the shadow. We recommend that you use Furnace's F_RotoShape plug-in to define these mattes, as it gives a smoother fall-off in the penumbra region, which in turn helps the algorithm filter the shadow out. If you use Shake's own RotoShape, you might find unexpected noise and artefacts in the repaired image.

Note that scripts containing F_ShadowRemoval nodes will not render correctly when distributed across a network.

Quick Start

Connect the source image into the first input, and connect an F_RotoShape node into the second input. Whilst viewing the source node, draw a roto so that the inner path defines the centre part of the shadow, where the luminance scaling is roughly constant. Move the outer path so that it's outside the shadow, as in Figure 31.1.

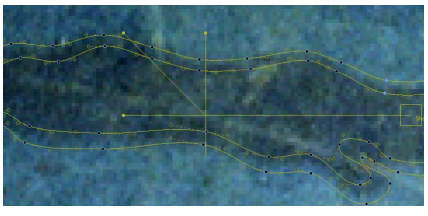


Figure 31.1 Example roto path to define where the shadow is.

Once you've finished doing this, process the F_ShadowRemoval node. If the results look noisy, try decreasing **alpha** (which

controls how edges within the shadow are scaled up). If the results still look too dark, try increasing **alpha**.

Inputs

F_ShadowRemoval requires two inputs, a source (**source**) image and a matte (**shadowMatte**), which defines the shadow and penumbra regions. If you have a shadow where the penumbra is occluded by something, make the inner and outer paths of the roto coincide, as in Figure 31.2.

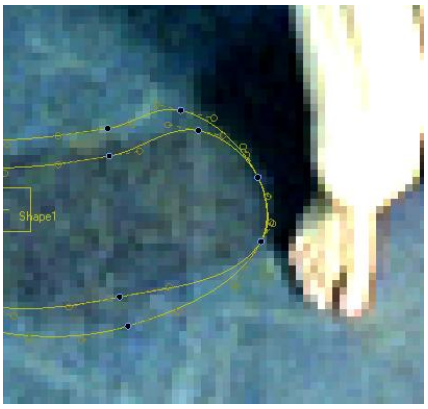


Figure 31.2 Shadow path around occluded objects.

Parameters

The parameters for this plug-in are described below.

alpha - controls the boosting of edges within the shadow. 0 leaves the edges alone, positive values boost edges and negative values lessen edge effects. If the result is too noisy, try decreasing alpha until you're satisfied with the results.

iterations - how many times we iterate the numerical process to try to get rid of the shadow. If you're not getting rid of the shadow in its centre, try increasing this value. It's set fairly conservatively by default, to give an idea as to how well the plug-in will work, so you may have to increase it to fully eradicate the shadow.

smoothing - controls how much we smooth the filtered gradients in the penumbra. If you're getting artefacts in this region, try increasing this value.

temporalSmoothing - uses the last result calculated to try to keep the filtered gradients consistent between frames. You may need to turn this on to get rid of boiling in the result. If this doesn't help, repair a single frame and track the repair in using one of the other Furnace plug-ins.

Warning! **Shake scripts containing F_ShadowRemoval with temporalSmoothing switched on cannot be distributed across multiple machines on a network render farm. See "Network Rendering" on page 22.**

shadowMatteComponent - which component of the shadowMatte to use.

- **Luminance** - use the luminance.
- **Alpha** - use the alpha.
- **Inverted Luminance** -use the inverted luminance.
- **Inverted Alpha** - use the inverted alpha.

Example

The footage used in the following example is available to download from our web site. For more information, please see the Example Images section on page 17.

Step by Step

1. Load the clip Shadows9316-9340#####.tif.
2. Add a Brightness node after the clip and set the value to 3.
3. Create an F_RotoShape node. While viewing the brightness node, draw a roto around the horizontal shadow in the first frame of the clip. Position the inner path around the central part of the shadow and the outer path around the penumbra, as shown in Figure 31.3. Notice that the two paths coincide on the right hand edge, where the shadow coincides with another shadow and the image is almost black. In this black region, there is not enough information available to allow us to remove the shadow.

Figure 31.3 Original image with roto.

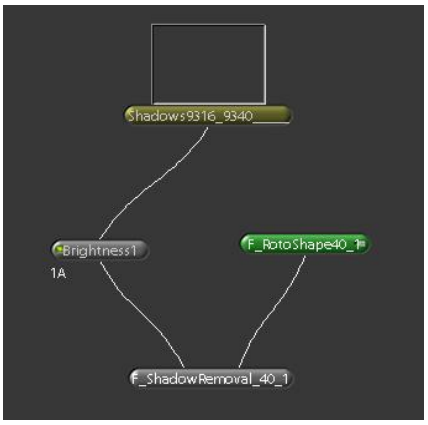


Figure 31.4 Node view.

4. Create an F_ShadowRemoval node. Connect the output from the Brightness node to the first input, and the output from the F_RotoShape node to the second input, as in Figure 31.4.

Figure 31.5 Shadow Removal with default parameters.

5. View the output from the F_ShadowRemoval node, which should look like Figure 31.5. The shadow has been removed, but the result looks noisy, so decrease **alpha** to 0.04.

Figure 31.6 Shadow Removal with alpha = 0.04.

6. The noise has now been decreased, but the centre of the shadow is still faintly visible, as in Figure 31.6. Increase

iterations to 300. The output should now look like Figure 31.7.

Figure 31.7 Shadow Removal with iterations = 300.

7. That's it.

SmartFill

This chapter looks at texture generation using Furnace's plug-in F_SmartFill. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_SmartFill is an intelligent fill tool designed to automatically fill a missing region in an image. For example, it could be used to remove an unwanted object from a scene or replace damaged pixels. The algorithm takes textures from spatially nearby regions to recreate the image whilst ensuring important structure is maintained. For example edges or lines crossing the missing region should be accurately continued and correctly join up.

Where there is a complicated underlying structure it is also possible to supply a guess image. This is a crude painting of the underlying regions. There needs to be no detail in the guess image as this will be supplied by the plug-in.

Remember the algorithm is inventing pixels that do not exist anywhere else in the image, so don't expect miracles: if there are no similar structures or objects it won't be possible to reinvent them. This tool is excellent at quickly removing objects giving you a good start before applying a final bit of touch-up with a paint tool.

Figure 32.1 Before

Figure 32.2 After

Background

In Figure 32.3 we have a single frame from a time-lapse sequence of London's Trafalgar Square. We have used F_SmartFill to remove two lampposts. The results are not perfect and need a small amount of touch-up paint work, but it does speed-up the process of creating the clean plate. You can have a go of this example if you wish.

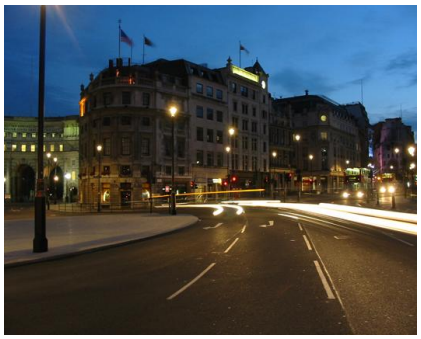


Figure 32.3 Before

Figure 32.4 After

The algorithm is only designed for repairing still images, making it ideal for the generation of clean plates.

Quick Start

Generate a matte of the missing or damaged region, Figure 32.5. Add this to the alpha channel of the source frame,

Figure 32.5 Original

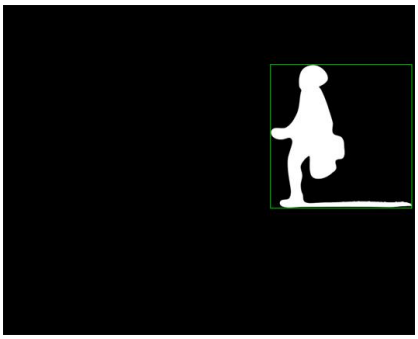


Figure 32.6 Matte

or connect it directly to the F_SmartFill plug-in. The region specified by the matte should be filled in using nearby textures and structures.

If suitable textures are a long way from the missing region in the source image, try increasing **searchSize**. If the texture has large repetitive structure increase **blockSize**.

If there are obvious structural elements in the missing region consider painting a guess image. This is just a crude colour image where the colours correspond to the colours of the structures in the missing region. Then try adjusting **guessInfluence** to determine how strictly the output follows this guess image.

Inputs

F_SmartFill has three inputs. **Source** is the frame containing the missing pixels that need to be repaired, **Guess** is an optional painted image showing the underlying structure of the

missing region and **Matte** is an optional matte showing the region to repair.

Parameters

The parameters for this plug-in are described below.

fill - what to use to define the missing region

- **Source Alpha** - use the alpha of the source. This is the default behaviour when no matte is supplied.
- **Source Inverted Alpha** - use the inverted alpha of the source.
- **Matte Luminance** - use the luminance of the matte. This is the default behaviour when a matte is supplied.
- **Matte Alpha** - use the alpha of the matte.
- **Matte Inverted Luminance** - use the luminance of the matte.
- **Matte Inverted Alpha** - use the alpha of the matte.

searchSize - specifies the size of the search region examined to find suitable pixels to fill the missing region. If the texture pattern is predominantly horizontal you should make the vertical size of the search region small. In Figure **??** we have a bollard that we wish to remove. You will note the dominant left-right texture in the image. To get a good repair as shown in Figure 32.7 we need a large horizontal search size and a small vertical search size. If we do the opposite we get a poor repair as shown in Figure 32.8.

Figure 32.7 Good

Figure 32.8 Bad

Note **searchSize is measured from the missing pixel to the edge of the search region. If the missing region is large, searchSize must be set large enough to cover both the missing region and a reasonable amount of undamaged image in order to find suitable repair pixels.**

Figure 32.9 blockSize=7

Figure 32.10 blockSize=1

blockSize - specifies the size of the block of pixels used to copy structure from the image to the missing region. In Figure 32.9 the block size is large and so the text on the building is copied well. In Figure 32.10 the block size is small and the text is not copied as well. Figure 133 shows the original image

Figure 32.11 Original Image

before the lamppost was removed with F_SmartFill.

guessInfluence - dictates how strictly the repaired image has to follow the guidelines shown in guess.

SmartPlate

This chapter looks at stitching images together into one large image using Furnace's plug-in F_SmartPlate. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_SmartPlate takes an image sequence and stitches the frames together to generate a single large image. It uses the Foundry's advanced motion estimation technology to calculate the best way to align one frame with another, using all the image information, not just a few tracking points.

Note that scripts containing F_SmartPlate will not render correctly when distributed across multiple machines on a network.

Background

In Figure 33.1 the left image shows a camera move across and down the building (Liberty on Carnaby Street in London). The sequence is about 60 frames long with just 4 images shown here. The image on the right is the plate that was constructed using F_SmartPlate.

Figure 33.1 F_SmartPlate.

If there are foreground elements in the sequence with different motion to the background these will appear blurred across the output plate. Alternatively the input frames can be combined using a median filter which has the effect of removing all

Figure 33.2 Sweeping camera move following a person walking down Carnaby Street

moving foreground elements entirely from the image, making it ideal for the generation of clean plates.

In Figure 33.2 there are three images from a 170 frame clip following a woman walking down Carnaby Street. F_SmartPlate is used to build a larger plate and with the median filter switched on, the moving foreground objects, including the woman, are removed as shown in Figure 33.3.

Figure 33.3 Output of F_SmartPlate.

F_SmartPlate has a large domain of definition but by default the output image is only shown as an image the size of the input. To see all of the output it is necessary to put a crop node of the required dimensions after the SmartPlate node.

The method used to combine frames is best understood by

considering that each frame has an alpha channel associated with it. The shape of the alpha channel is defined by **weight**, which can either be **flat** (a uniform value over the whole frame), **centre** (a Gaussian distribution positioned on the centre of the frame, or **Extreme Centre** (a Gaussian distribution with a much faster fall-off positioned on the centre of the frame). Pixels from each new frame are then added on to the clean plate weighted by **weight**.

FrameCombiner chooses between a number of different options as to how to combine the frames. **Blend** simply blends every overlapping pixel in the frame with the corresponding pixel in the output plate. **Limit Blend** blends the pixels in a similar way but stops blending frames when the alpha of the output reaches 1. This is to ensure that later frames don't overwrite earlier ones. For example, assume that a particular pixel is visible in a number of frames, after weighing with a Gaussian weight the alpha value of the pixel may be 0.2, 0.5, 0.8, 1, 1, 1, over frames 1-6. Only pixels in frames 1-3 would be used to build the output. Rather than averaging the pixels together to form the output, it is also possible to combine them with a median filter by choosing the **Median** option. This has the effect of removing locally moving foreground objects from the plate. Again **Limit Median** can be used to ensure later frames dont overwrite earlier ones.

You should note that if there is a large foreground motion in the clip you are using, the motion estimation may lock onto that rather than the background motion producing undesirable results. Using F_RigRemoval to remove the object before using F_SmartPlate may get you round the problem.

Putting the Camera Move Back

F_SmartPlate has an optional second input for the large plate. When this is connected and **Use Full Plate** selected from the **action** menu, SmartPlate no longer outputs the large plate, instead it calculates the transforms as usual and applies them to the second input.

This allows you to generate a plate in the usual way for painting, touch-up or other compositing work, then you pass this large plate back through SmartPlate to re-apply the original camera motion.

You should be aware of the following restrictions:

1. SmartPlate doesn't store the transform information in user-accessible variables.

2. The SmartPlate process unpeels the image sequence, flattening camera distortion to produce the large flat plate. The restore process doesn't reintroduce any camera distortion, so expect your output results to differ from the input.
3. SmartPlate locks on to the dominant motion in the image. We assume for this purpose that the original material contains little or no significant foreground motion. Large moving foreground objects will adversely affect the reconstruction of the background plate.
4. The offsets that were applied when you generated the large plate might have been lost when it was saved to a file, so it might be necessary to realign the large plate with the first frame of the input sequence before rendering.

Quick Start

Connect the input sequence to F_SmartPlate. Connect a crop node to the output of F_SmartPlate and set the size to the expected size of the output. View the crop node and play the sequence. An output plate should be progressively built up.

Note **To get a single image large clean plate you have to render the whole sequence then throw away all but the last frame.**

If the sequence contains significant zooms or perspective changes turn on **scale** or **perspective** respectively for better results.

Try experimenting with the ways the frames are combined together using **frameCombiner**. To remove locally moving objects select the **median** option.

If the sequence folds back on itself and you need the later frames to stitch on to the beginning of the sequence set **fitTo** to **currentPlate**.

Inputs

F_SmartPlate has two inputs, the sequence from which the plate is to be made and an optional second input, a previously generated full plate to which the camera move is to be reapplied.

Parameters

The parameters for this plug-in are described below.

action - sets whether to create a large plate or reapply a camera move on an input plate.

- **Full Plate** - takes the first input and creates a large plate from it. This is the default.
- **Lock** - builds a large output plate in the background then reinstates the original camera move. The effect of this is to remove foreground motion while preserving the camera move.
- **Use Full Plate** - takes a previously generated full plate from the second input and applies the camera move from the first input to it. (It might be necessary to realign the full plate to the first frame of the first input before processing.)

frameCombiner - selects the method used to combine frames into the output plate.

- **Blend** - mix the frames together weighted by the value of weight.
- **Limit Blend** - mixes the frames together, ensuring later frames do not overwrite earlier ones.
- **Median** - combines the frames using a median filter to remove local foreground objects moving independently of the background.
- **Limit Median** - combines the frames using a median filter, ensuring later frames do not overwrite earlier ones.

medianSamples - the number of frames used by the median filter in frameCombiner.

weight - the weighting given to each frame before it is combined with the output plate.

- **Flat** - a uniform weight in which all pixels are added equally.
- **Centre** - a weight specified by a Gaussian distribution. The pixels in the centre are weighted more than those at the edges.
- **Extreme Centre** a weight specified by a Gaussian distribution with a smaller standard deviation. The pixels at the centre are weighted much more than those at the edges.

filter - sets the filter type used for the pixel interpolation.

- **Box** - low quality but quick.
- **Gaussian** - higher quality but slower than box.
- **Sinc** - highest quality giving sharp images but slower than gaussian.

translateX - switch this off to ignore horizontal differences when lining up plates from a vertical camera move.

translateY - switch this off to ignore vertical shifts when lining up plates from a horizontal camera pan. Without this errors build up causing the long strip produced to curve away from a long rectangle.

rotate - by default the alignment process will attempt to match any rotational movement of the camera. Switch this off to ignore rotational moves.

scale - by default when aligning one frame with another only translational and rotational transforms are used. If there are significant scale changes in the sequence turn on scale. This will give improved results at the expense of longer computation time.

perspective - by default when aligning one frame with another only translational and rotational transforms are used. If there are significant perspective changes in the sequence turn on perspective. This will give improved results at the expense of longer computation time.

fitTo - how to align the frames when stitching them together.

- **Previous Frame** - align each frame to the previous one.
- **Current Plate** - align each frame onto the output plate. The advantage of this is that it allows frames to be stitched back on to earlier parts of the sequence. For example, consider an S-shaped camera move across a building. Current Plate must be selected to ensure the end of the S is correctly stitched back on to the top of it.

Example

The images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Liberty

In this example, we take a panning shot of the side of Liberty on Carnaby Street and build a large plate.

Step by Step

1. Start Shake and FileIn the Liberty clip. Play it to get a sense of the motion.
2. Load F_SmartPlate from the Furnace tab and connect the Liberty clip to the input.

Figure 33.4 F_SmartPlate.

3. Add a crop node after the F_SmartPlate and make it bigger. Try something like cropLeft -240, cropBottom -1500, cropRight 1500, cropTop 800.

Figure 33.5 Shake tree.

4. In the F_SmartPlate node switch off rotate and set fitTo to currentPlate.
5. Set the timebar to frame 1 and render the clip to a file called libertyplate.####.tif
6. Save the tree, we'll need it in the next example.

Liberty Banner

In this example, we will take the large plate created in the previous example and paint on it. Then we'll put the camera

move back.

Figure 33.6 Banner painted on output from F_SmartPlate

Step by Step

1. In a paint package draw a banner over the image libertyplate.0050.tif as shown in Figure 33.6. If you prefer, you can skip this stage and use the one in the download file.

2. FileIn the libertypaint.####.tif clip and feed it into a Move2D node and then into the second input of the F_SmartPlate node used in the first example, as shown in Figure 33.7.

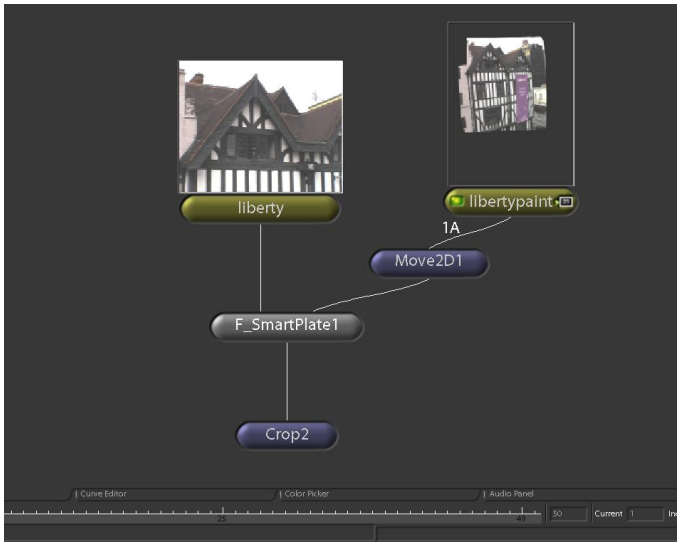


Figure 33.7 Putting the camera move back

3. The important thing here is that we've run it through a Move2D first. This simply attaches an xPan and a yPan which are linked to the bottom-left corners of the Crop node. In this way, we've simply offset things so that at frame 1 the output of the Move2D matches the original source material at frame1. From then on, SmartPlate continually applies the move, effectively playing back the original camera as before. So edit the parameters of the Move2D node and set the xPan to the expression Crop.cropLeft and the yPan to Crop.cropBottom.
4. View the F_SmartPlate node and render.
5. That's it.

Falling Snow

In this example, we'll take a shot with some falling snow and use SmartPlate to make the snow disappear.

Step by Step

1. File-in snow###.tif.
2. Render a flipbook and watch the sequence. As well as heavy snow falling, you'll see some traffic go past the building and a man walk into view.

Figure 33.8 Original image with falling snow.

3. Create an F_SmartPlate node and attach the input sequence to it.
4. Change the frameCombiner parameter to Median.
5. Render a flipbook of the SmartPlate node. Notice how the falling snow has disappeared after the first twenty frames, as in Figure 33.9. Also notice that neither the traffic nor the man walking appear in the output sequence.

Figure 33.9 SmartPlate output.

6. That's it.

SmartZoom

This chapter looks at superresolution using Furnace's plug-in F_SmartZoom. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_SmartZoom takes a sequence of different views of the same object and uses them to make a single image of higher resolution than any of the original views.

Given a number of different views of an object, all taken from slightly different positions, each one will contain different sub-pixel information about the object. Using the Foundry's advanced motion estimation technology it is possible to align this sub-pixel information from a number of images in order to make a higher resolution output. This differs from a sinc interpolation filter or sharpening filter, as the output will contain information from all the input frames, not just one, as in standard filtering operations.

Note that scripts containing F_SmartZoom will not render correctly when distributed across multiple machines on a network.

Background

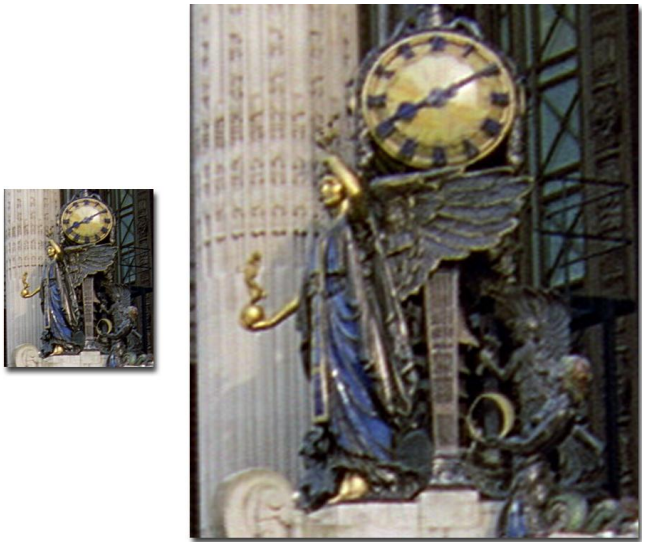


Figure 34.1 F_SmartZoom has been used to enlarge the image.

In Figure 34.2 the image is split vertically down through the clock face. On the left side F_SmartZoom has been used to enlarge the original image. On the right hand side a bilinear

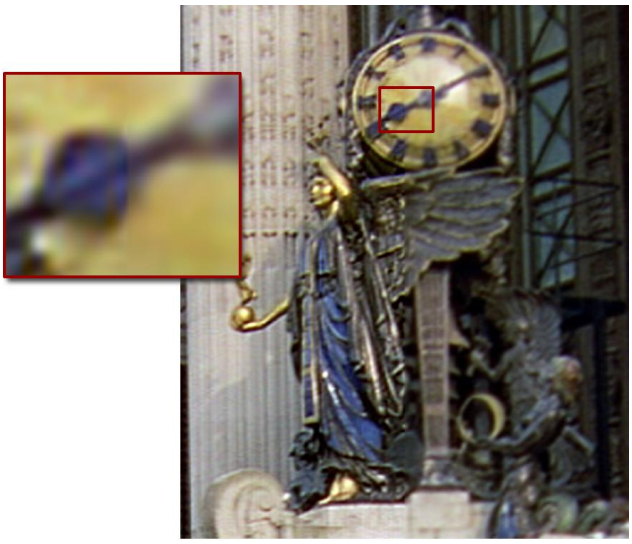


Figure 34.2 F_SmartZoom compared to Bilinear zoom.

filter has been used to do the same. Note that SmartZoom produces a much sharper result.

Quick Start

Connect the input clip to F_SmartZoom. Select how big the output plate should be using the resize factor. Make sure you are at the first frame in the sequence and then step through from frame to frame. The output of the plug-in should get progressively more detailed.

Tip **Since the output is always the last frame of the render, rather than render all frames and discard all but the last, you can render writing over each output frame as you go. From the command line try shake script test.shk -t 1-100 -fo test.tif -v**

If there is significant scale changes in the sequence turn on scale. Similarly if there are significant perspective changes, turn on perspective. If the picture still looks soft try switching filter from gaussian to sinc. If the image looks harsh try increasing softness very slightly and if ringing appears in the output increase errorThreshold by a small amount.

Inputs

F_SmartZoom takes only one input which is the sequence from which the high resolution plate is to be made.

Parameters

The parameters for this plug-in are described below.

resize - sets the integer scale factor. For example, a value of 2 will produce an output image of twice as wide and twice as high of the input. Note that only integer values are allowed. To scale to a non-integer value, set the value to the closest integer value and then use Shake's built-in resize node for the non-integer scale.

filter - sets the filter type used for the pixel interpolation.

- **Box** - low quality but quick.
- **Gaussian** - higher quality but slower than box.
- **Sinc** - highest quality giving sharp images but slower than gaussian.

scale - by default, when aligning one frame with another only translational and rotational transforms are used. If there are significant scale changes in the sequence, turn on scale. This will give improved results at the expense of longer computation time.

perspective - by default, when aligning one frame with another only translational and rotational transforms are used. If there are significant perspective changes in the sequence, turn on perspective. This will give improved results at the expense of longer computation time.

softness - when combining the sub pixel information from a number of frames it is possible that the resulting image, although containing more information, will look overly sharp or harsh. The softness parameter controls this harshness by applying a subtle filter to the information given by each additional frame. If this is set too high no useful information will be added and the resulting resized image will look like a simple bilinear interpolation of the original. If it is set too low the resulting frame may look unpleasantly sharp.

errorThreshold - another artefact that may occur when combining sub pixel information is ringing along any edges in the image. errorThreshold attempts to control this by clipping any overshoots that may occur when combing information from all the images in the sequence. If it is set too high then no additional information will be included and the resulting image will look like a simple bilinear interpolation of the original.

Example

The images for the following example can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Clock

In this example, we take a handheld shot of a clock and scale it to three times the original size. See Figure 34.1 on page 200.

Step by Step

1. Start Shake and FileIn the 20 frames of the clock clip. Play it to get a sense of the motion.
2. Load F_SmartZoom from the Furnace tab and connect the clock clip to the input.
3. Set the timebar to frame 1 and play (or render) the clip. Note that the frame sharpens over time. To output the clip render 10 frames and discard the first 9 or even better save the script and render to one frame (shake -script test.shk -t 1-10 -fo test.tif -v).

Splicer

This chapter looks at stitching images together using Furnace's plug-in F_Splicer. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_Splicer is designed to composite two images together and hide the join between the images. Rather than using a conventional alpha matte, it works by overlapping the two images and calculating optimum position for the edge that best hides the join.

F_Splicer is best at joining together images that contain a lot of texture but can also work well in other scenarios.

The plug-in is designed for joining together still images, for example to create clean plates. When it is applied to a sequence it will generate a new optimum join for each frame. These are likely to be in different positions on each frame, resulting in a highly visible moving edge. To avoid this, it is possible to enforce temporal consistency, but this is only likely to work on sequences with little motion.

Note that F_Splicer doesn't attempt to do any automatic image alignment. It simply joins a pair of images that have been positioned by the artist and attempts to hide the join. For automated image alignment try the F_Align or F_Steadiness plug-ins. Also, F_Splicer is also likely to give very different results at proxy resolutions, so it's best not to use it with proxies.

Also, note that scripts containing F_Splicer with the **temporalConsistency** parameter switched on will not render correctly when distributed across a network.

Quick Start

Connect the two images to be composited together to the two inputs. Connect a matte to the third input that specifies which part of the second source you require compositing over the first. Splicer will now try and find the optimum edge between the two images that hides the join.

Try increasing **pathWidth** to allow it to search for the optimum join over a larger area, or decreasing it to constrain the position of the join. Increasing **edgeBlend** may also help to hide the join by blending between the two images. Too high a

value will result in obvious softness. If you are working on a sequence turn on **temporalConsistency** before processing.

Inputs

F_Splicer requires three inputs. **Source1** and **Source2** are the images to be composited together. The third input, **Matte**, is a binary matte that specifies which part of **Source2** should be composited into **Source1**.

Parameters

The parameters for this plug-in are described below.

pathWidth - sets the width of the search region to find the best possible join between the source images. The larger the path the more better the opportunity for the algorithm to find a good path.

edgeBlend - the amount of blur applied at the join between the images. A small amount is good to help hide the join on some images but too much will result in an obvious softness.

temporalConsistency - switch this on to force temporal consistency between successive frames. Splicer is really designed to work with still frames not sequences. If you try and render a sequence Splicer is likely to find a new best edge path for each frame which will result in boiling along the edge when the sequence is played. Turning on temporalConsistency forces Splicer to use the same edge path on every frame. This is only likely to be helpful when there is little or no motion in the sequence.

Warning! **Shake scripts containing Splicer with temporalConsistency switched on cannot be distributed across multiple machines on a network render farm. See "Network Rendering" on page 22.**

luminanceMatch - switch this on to match the luminance of the new texture with the luminance of the source region in order to help hide the join.

luminanceBlur - controls how much effect **luminanceMatch** has on the new texture.

matteComponent - defines which component to extract from the matte input.

- **Luminance** - extract a single channel based on the luminance.
- **Alpha** - extract the alpha channel.

- I**nverted Luminance** - extract a single channel based on the inverted luminance**.**
- **Inverted Alpha** - extract the inverse of the alpha channel.

Examples

All the images for the following examples can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Hedge

In this example, we have shot of a hedge where we wish to fill the upper part of the frame where there is no hedge.

Figure 35.1 Hedge.

Step by Step

1. FileIn hedge.####.tif then generate a Move2D node below it. Pan the image upwards so that the hedge is positioned over the background we wish to fill.
2. Create a RotoShape that covers the area we wish to fill, as in Figure 35.2. Don't worry about being exact at the edges of the frame.

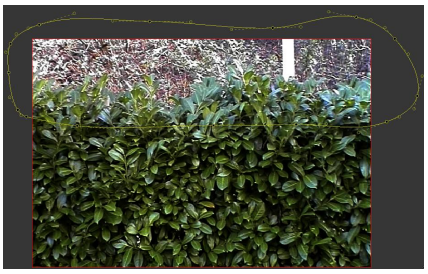


Figure 35.2 Hedge With Roto.

3. Create a F_Splicer node, and connect the original clip into the first input, the panned clip into the second and the RotoShape into the third. An example result is shown in Figure 35.3, but the actual result will depend on the RotoShape and pan used.

Figure 35.3 Resulting texture.

London Eye

In this example, we'll build an extension to County Hall, on the South Bank of the River Thames (in Figure 35.3).

Figure 35.4 London Eye.

Step by Step

1. FileIn londonEye.####.tif then duplicate it, setting timeSlip parameter on the timings tab to -15. This means the duplicate clip will output frame 16 when we're actually on frame 1.
2. Reflect the duplicated clip horizontally using a Flop node so that at the first frame there is an image such as that in Figure 35.5 on the next page
3. Draw a RotoShape looking loosely like that in Figure 35.5 on the following page. Create an F_Splicer node, connecting the original clip to the first input, the timeslipped, flopped one into the second and the RotoShape into the third.

Figure 35.5 London Eye Flopped with example RotoShape.

4. The resulting image is reasonable, however there is visible may be some softening of the left hand spire as in Figure 35.6 on the next page and the front of the building to right of the left hand boat. In our case, increasing the **pathWidth** to 80 (allowing the algorithm to find a joint further from the original matte) solved this, giving the result in Figure 35.7 on the facing page.

Figure 35.6 First Attempt.

Figure 35.7 Second Attempt with larger **pathWidth**.

Steadiness

This chapter looks at how to stabilise a shot using F_Steadiness. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_Steadiness uses Global Motion Estimation (GME) to calculate a four corner pin, so that camera motion within a single shot can be smoothed out over a range of frames or removed by locking to a specific frame.

For an overview of Global Motion Effects, and a description of the common way they work, please see the Global Motion Effects chapter on page 259.

Like all the other Analysing Global Motion Effects, F_Steadiness needs to analyse the input clip before it can render useful output. This analysis is done in the user interface of the plugin and keyframes a four corner pin which will stabilise the clip in subsequent renders. F_Steadiness, without having performed an analysis pass, will not do anything useful on render.

F_Steadiness can work in two ways. These are:

1. **Smooth** - A window of frames around each frame is analysed for motion and an average of that motion used to calculate the corner pin. Use this to keep the overall camera motion, but to smooth out sharp bumps and kicks.
2. **Lock** - A lock frame is specified and steadiness attempts to register each individual frame to that lock frame. Use this to completely remove camera motion from the sequence.

In lock mode, each frame in the clip must share a substantial amount of the scene with the lock frame, so you can't lock each frame in a 360 degree pan to the first one. However, in smooth mode, because F_Steadiness is only working on a small window of frames around the current frame, you can use it in shots which change completely over time.

The analysis region is used to control which section of the reference frame is being matched to each source frame. In lock mode, the reference is the lock frame, so you should position the analysis region when looking at the lock frame. In smooth mode, it looks at the incremental differences between frames, in which case you should place the analysis region in the area you want to appear with smooth motion.

Quick Start

This section gives a very brief outline of how to use the plug-in.

Smoothing Out Camera Motion

1. Create a new F_Steadiness node. By default it is in Smooth mode.
2. Find a shot that has some camera shake in it, connect that to F_Steadiness's single input.
3. Click on the **analyse** push button.
   (a) F_Steadiness will now start analysing each frame in the shot, figuring out the smoothing pin and writing it as keyframes to the corner pin parameters, cornerLL, cornerLR, cornerUL and cornerUR.
   (b) After a brief pause (while F_Steadiness calculates the transforms in the initial smoothing window), F_Steadiness will update the time-line and you will see the steadied image render in the viewer.
   (c) If you want to interrupt analysis, either hit the ESC key or clip with the left mouse button. The pins it has calculated until that point will be retained.
4. Play or scrub through the stabilised frames. If you want to make it smoother, increase the **smoothing** parameter and the corner pin will be recalculated immediately to give a smoother shot. You don't need to re-analyse the sequence if you do this; as F_Steadiness has kept the raw inter-frame transforms cached away, all it needs to do is re-write the keys for the average smoothing pin.
5. That's it!

Locking to A Frame

1. Create a new steadiness node and set the **mode** parameter to **Incremental Lock**.
2. Find a shot that has camera shake, but where all frames share scene information. Connect that to F_Steadiness's single input.
3. Choose a frame somewhere in the middle of the sequence that you want to lock to, and set the **lockFrame** parameter to that frame
4. Scrub back and forth through the shot and look for a region of the shot that is shared by all frames and doesn't

change much (for example, having people walk infront of it)

5. Whilst looking at the lock frame, position the onscreen widget for the **analysisRegion** over that region.
6. Hit analyse,
   (a) the effect will start to analyse, working first forward from the lock frame, then backwards from it, until all frames to be analysed are done,
   (b) the timeline will immediately update to give you feedback in the viewer.
7. That's it.

Inputs

F_Steadiness has a single input, which is the shot to stabilise.

Parameters

F_Steadiness shares all the Analysing Global Motion Effect parameters which are described in the 'Global Motion Effects' chapter. The parameters that are specific to F_Steadiness are described in this section.

mode - This parameter controls whether F_Steadiness is smoothing the shot or locking it to a single frame. It can be set to:

Smooth - in which case we are smoothing the shot for a window of frames described by the **smoothing** parameter,

Incremental - **Lock** in which case it will calculate the pin that takes each frame to the lock frame. This calculates the pin by doing working from the lock frame out to each frame, calculating the GME between each frame incrementally and accumulating it to create the corner pin.

Absolute - **Lock** this also calculates a pin that takes each frame to the lock frame. However, it does so by doing GME directly from the frame in question directly to the lock frame.

Incremental locks work better in some situations, whilst absolute locks work better in others. However, absolute locks, when they work, are typically more accurate.

smoothing - This controls the window of frames to average motion over when mode is set to **Smooth**.

lockFrame - This controls the frame which will be locked to when mode is set to either of the lock modes.

Examples

The images for the following examples can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Steadying A Walk Around Leicester Square

This is a small section of a hand held shot of a walk around Leicester Square in London. The full shot walks completely around the Square and so would be very hard to steady with traditional point tracking techniques, as points are continually coming into and out of shot.

Figure 36.1 Leicester Square.

Step by Step

1. Make a FileIn node that reads the LeicesterSquareWalk.#####.tif.
2. Create an F_Steadiness node and wire in the Leicester-SquareWalk shot.

- Hit analyse and wait as it goes through the sequence. After a brief initial pause, you will get feedback as it will render a steadied frame after it has been analysed.

1. Ta-da! A steadied shot. Scrub through the time line or play it to see the result.
2. Smooth it out more by changing the smoothing parameter to be '30' frames. Play the shot again and see the difference that made.
3. You can now render the result of the analysis.

Locking A Walk Around Leicester Square

We will take the last shot we have just smoothed the motion of and lock down the motion instead. Note that this would not work for the complete shot, because not all frames share scene information. However for this short section, it works well. This example will also show you how to use the analysis region to limit the area where GME occurs.

Step by Step

1. Make a FileIn node that reads the LeicesterSquareWalk.#####.tif.
2. Create an F_Steadiness node and wire in the Leicester-SquareWalk shot.
3. Change 'mode' to 'Incremental Lock'
4. Set lockFrame to 25.
5. Scrub through the timeline to frame 25.
6. Move and resize the overlay widget (not the four corner pin widget!) so that it is over the trees at the top, with it centred on the tree trunk in the middle. Have it fairly squat and broad, as in Figure 36.2, so that it misses most of the people and objects moving in the foreground.

Figure 36.2 Analysis Region

.

7. Hit analyse and watch as it runs through the sequence. It will be writing keyframes into the four corner parameters as it goes; you can see these if you later open shake's curve editor, as shown in Figure 36.3.
8. Done. You have steadied the shot and can now render the result of the analysis. If you play or scrub the shot you will see the trees at the top, where you placed the analysis box, stayed locked.

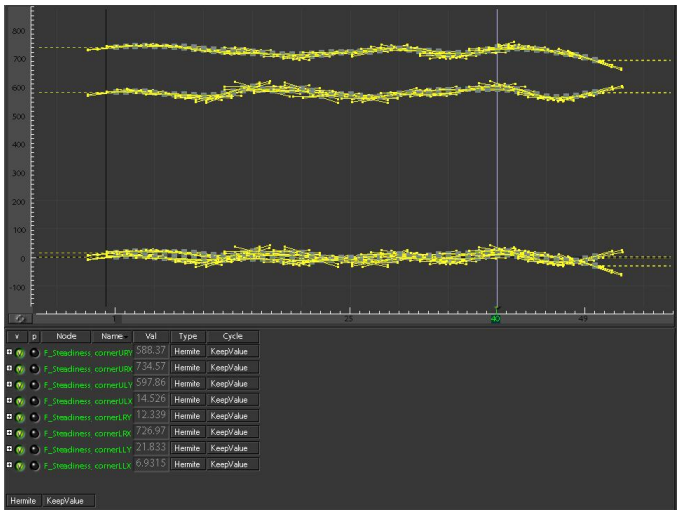


Figure 36.3 Animation Curves

London Eye

This is a handheld panning shot of the British Airways London Eye and London Aquarium from across the river Thames. In this example, we'll smooth out the camera shake as in the previous two examples. In addition, we will fill in the black gaps at the edges created when the frames are transformed.

Figure 36.4 London Eye.

Step by Step

1. FileIn LondonEye.####.tif and flipbook. Note the jerky camera pan.
2. Connect the shot to the input of a new F_Steadiness node and press analyse. It will go and analyse your 60 frames of input creating a keyframed corner pin that will steady your shot.

Flipbook to check the smooth motion. You will have noticed that the image will have been translated and rotated to smooth out the camera shake. This leaves black pixels around the

Figure 36.5 Frame 1 of the stabilised london eye clip

Figure 36.6 Frame 1 with edges filled using F_RigRemoval

image as shown in Figure 36.5. You can remove these by scaling up the image until no black edge pixels are visible, but this leads to a slight softening of the image. Alternatively, you can use F_RigRemoval to take image data from other frames in the clip and fill in the gaps. See Figure 36.6.

1. We need to modify the tree to generate a alpha channel for the areas we want to fill in. Add a SetAlpha node between the LondonEye input and the F_Steadiness node.
2. Add an Invert after the F_Steadiness node. Set Channels to `a'.
3. Add a DilateErode after the Invert node. Set Channels to a and xPixels to 2.5 and yPixels to 2.5.
4. Add a FileOut node after the DilateErode node and set the output name.

Figure 36.7 New nodes are shown in green.

5. Render the output clip.
6. Load F_RigRemoval and select the rendered clip. Increase the lookAheadFrames until the gaps round the image have been filled in. See Figure 36.8 and Figure 36.9. Since this is a pan, we dont need to search the clip in both directions to find the missing data. Set direction to backward.

Figure 36.8 lookAheadFrames=3.

Figure 36.9 lookAheadFrames=15.

7. Render.

Experimenting With The Limits of Global Motion Estimation

To understand the limits of what GME can do, use the same Leicester Square footage as before and try to perform the same lock, but this time onto the stone vase. Play with the accuracy setting, the different lock modes, the analysis region and turning on scale then perspective.

You will see that the heavy amount of parallax makes it harder for Steadiness to lock onto the vase; you need to position the analysis region carefully, enable analysis of scaling and increase the accuracy. The results are not as good as locking on the trees and due to the nature of that section of the shot, can never be.

With the example of locking the trees together, you could have done that by hand with a great deal of work. With the vase the parallax caused it to look very different from frame to frame, making it impossible to accurately match the vase at frame 1 to frame 25 with a simple pin. To do that some extremely complex 3D modelling and back projection would probably need to be done.

This should give you an idea of what GME is and is not capable of. Remember this: if you can do it with a four corner pin, then

Furnace's GME tools can as well, but without the painstaking effort of having to match the pin by eye. If you can't, it can't. There is no magic.

Tile

This chapter looks at the generation of intelligently tiled textures using Furnace's F_Tile plug-in. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_Tile is designed to intelligently repeat an image horizontally and vertically. It does this by overlapping copies of itself and then trying to make the join between them as invisible as possible. Obviously there are limitations, but in many cases the resulting tile is extremely plausible.

Figure 37.1 Conventional tiling with a vertical straight line join.

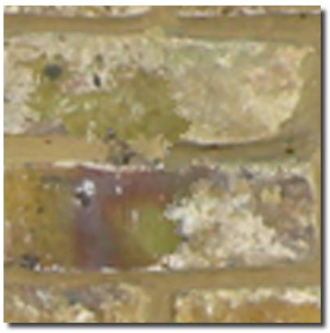


Figure 37.2 Furnace tiling with a vertical jagged line between tiles.

The plug-in is written so that it can take a small region of image and tile it both horizontally and vertically to generate a full image. It differs from Furnace's other texture plug-ins in that the new image has obvious horizontal and vertical repetition. However, it should be difficult to tell exactly where this repetition occurs.

Quick Start

Connect the image to be tiled to the source node of F_Tile and switch on fillOutputX and fillOutputY. That's it! This will give you a tiled output filling the original image size.

The two most interesting parameters are **overlap** and **blendSize**. This plug-in is fast, so switch update to always and adjust **overlap** to vary the amount adjacent tiles are overlapped. Look at how the best fit changes as you drag the overlap slider. A large overlap is more likely to produce a better join. The example below shows the effect of the overlap on some pebbles. The pebble image is supplied in the example

footage. The **blendSize** blurs the join between overlapping tiles. This can produce a better result on some images, but if increased too much will generate an obvious soft join between the tiles.

Figure 37.3 Overlap = 0 gives straight edges between tiles.

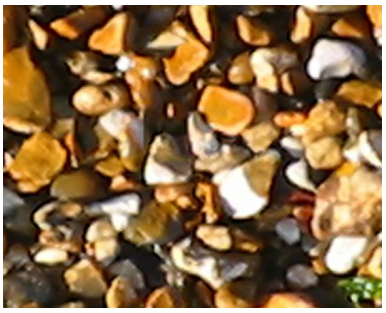


Figure 37.4 Overlap = 40 gives irregular edges between tiles.

If there is an obvious colour gradient across the tile, switch on **removeGradient**. Adjusting **removeAmount** controls the amount of gradient reduction. Too high a value will cause the image to loose quality and a setting of zero will give no gradient removal.

Figure 37.5 The wooden plank has been tiled vertically and repeated horizontally. The right-hand image shows the effect of switching on removeGradient

Large Textures

You may wish to use tile to create a larger image from a smaller sample patch. To do this you need to make your source image larger by inserting a window and a setDOD node between the sample image and F_Tile. Make the window larger with xRes and yRes and expand setDOD to match.

Inputs

F_Tile has one input that is the texture that will be sampled and tiled.

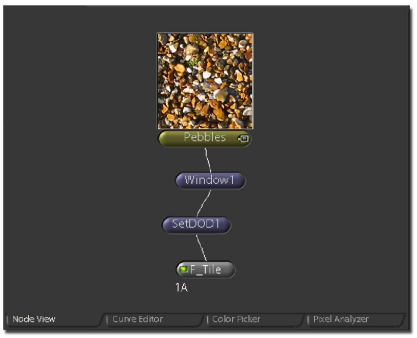


Figure 37.6 Tree showing the expansion of a 300x300 pixel pebble texture using window and setDOD nodes.

Parameters

The parameters for this plug-in are described below:

blendSize - controls how much overlapping edges are mixed together. This can be useful to disguise joins on some types of image. Large values will result in an obvious soft edge between tiles.

overlap - controls how much the tiles overlap. A large overlap is more likely to give a better join.

tileHorizontally - switch this on to use the algorithm to tile horizontally. When switched off the tiles will have straight left and right edges.

tileVertically - switch this on to use the algorithm to tile vertically. When switched off the tiles will have straight top and bottom edges.

fillOutputX - repeats the tiling horizontally to fill the width of the output image.

fillOutputY - repeats the tiling vertically to fill the height of the output image.

stretchToFit - switch this on to render a single repeating tile that can be used elsewhere for seamless texture mapping.

removeGradient - switch this on to remove colour gradients in the image which makes the tiling more obvious.

removeAmount - controls the amount of gradient reduction. Increase this to smooth off colour variations. Values close to one will cause the image to lose quality. A value of zero effectively switches off any gradient removal.

+bounds - defines the sample area that will be used to tile.

boundsMin - the bottom left corner of the sample area.

boundsMax – the top right corner of the sample area.

Examples

A variety of textures, including pebbles and planks, can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Figure 37.7 Top left is the original image, bottom left a tile with an overlap of zero giving a straight edge. The picture on the right has an overlap of 28 giving a seamless join.

Tracker

This chapter looks at the how The Foundry's F_Tracker plug-in operates. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_Tracker is designed to be both easier to use and more robust than the standard Shake tracker. Its data can be used to enhance other Foundry algorithms.

Note

Quick Start

Connect the sequence you wish to track to the input of F_Tracker. Postion the tracking cross hair over the object you wish to track. Adjust the size of the box so it contains just the object to be tracked. A **user keyframe** will automatically be created when you adjust the tracking region in this way. Press  . The object will be tracked through the sequence creating **track keyframes** at each frame.

If you want to track the object over a range of frames rather than the whole sequence, use  to set the start frame of the range and  to set the end frame of the range.

If the tracker loses the object being tracked, press any key to stop the track. Go back to the frame where the track first went wrong and move the selection box to the correct position. Create another **user keyframe** by pressing  , delete all **track keyframes** forward from this point by pressing  and then set the tracker off again by pressing .

The F_Tracker Algorithm

F_Tracker uses an intelligent algorithm to build up a statistical model of the track. It is therefore able to determine on its own the type of track being attempted and the required search region. This means that there are no parameters that the user needs to tweak.

Should the algorithm fail to successfully follow an object, the interface has been designed to allow the user to quickly and efficiently correct the track.

The User Interface

Each F_Tracker node may contain a number of individual tracks. Each track has a track region box in the Shake Viewer workspace and corresponding parameters in the Shake Tweaker. The track region box is a coloured box which determines the area to be tracked - the track region. It is surrounded by icons containing additional information (discussed below). As the track region box is dragged around (using its central cross hair) a zoomed representation of the image beneath is displayed, allowing greater placement precision.

The Controls

Most of the controls for F_Tracker are situated in the Shake Viewer shelf at the bottom of the Viewer workspace. These include the controls to edit keyframes, track, set the time range etc.

Figure 38.1 Tracker controls.

These controls only apply to the currently selected or 'in focus' track. This track is highlighted by an arrow at the top left corner of the track region box. To focus on a particular track either click on its track region box or change one of its parameters.

Figure 38.2 Track region box with 'in focus' icon.

Time Range

Each track has an individual time range. By default this is the length of the input clip. It can be edited in one of two ways - either by typing the range in the relevant box in the Tweaker, or by using the Viewer workspace controls. To set the start of your time range click the **set start range** control (). Similarly, to set the end of the time range click the **set end range** control (). Information about the time range is also displayed by icons at the bottom left corner of the track region. A tick means the track is within its valid time range and a cross means the track is outside its time range. There are also icons indicating the start and end of the time range.

Figure 38.3 (left to right) Start range icon, in range icon (tick), end range icon and out of range icon (cross).

User and Track Keyframes

F_Tracker makes a clear distinction between keyframes which the user has set (**user keyframes**) and those that the algorithm has determined (**track keyframes**).

User keyframes supply the track template (i.e. the region to be tracked) and are always considered to be correct. They are denoted in the gui by a key symbol next to the track region (see the first image in figure 38.3).

User keyframes can be created using the **create user keyframe** control ( ) and deleted using the **delete user keyframe** control ( ). **User keyframes** are automatically created when the user manually moves the tracking points.

Hint **Whilst the algorithm will build up knowledge of the track path as it goes, supplying a few user keyframes throughout the sequence before tracking may aid the learning process.**

Track keyframes fall into three categories:

1. If the algorithm successfully finds the location of the track and decides that it is reliable enough to update the track template, a **reliable track keyframe** is created. This is denoted in the gui by a solid track region box.

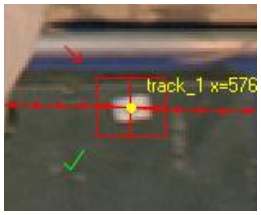


Figure 38.4 Reliable track keyframe.

2. If the algorithm successfully finds the location of the track but decides that it is not reliable enough to update the track template, a **successful track keyframe** is created. This is denoted by a densely stippled track region box. This may occur, for example, when tracking a subject that passes behind another object. As the subject starts to become occluded there may still be enough of it

visible to accurately find its location, however updating the template would be a bad idea as it would contain large amounts of the occluding object.

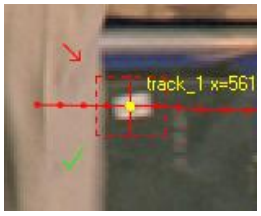


Figure 38.5 Successful track keyframe.

3. If the algorithm fails to successfully find the location of the track, an **unsuccessful track keyframe** is created. This is denoted by a sparsely stippled track region box. This might occur when tracking a subject that becomes completely occluded. After tracking, the position of

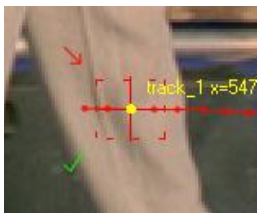


Figure 38.6 Unsuccessful track keyframe.

unsuccessful track keyframes can be adjusted by interpolating the positions of the **reliable** and **successful track keyframes** using the **interpolate unsuccessful track keyframes** control ( ).

It is therefore possible to delete erroneous **track keyframes** without losing your **user keyframes**. This can be done in a variety of ways.

Single **track keyframes** can be deleted using the **delete track keyframe** control ( ), the **delete track keyframe and step backwards** control ( ) and the **delete track keyframe and step forwards** ( ) control (the latter two, as you might expect, delete the **track keyframe** and then advance the timeline by a single frame in the relevant direction - this allows you both speed and control when deleting **track keyframes**).

You can also delete multiple **track keyframes**. The **delete track keyframes backwards** control ( ) and the **delete track keyframes forwards** control ( ) allow you to delete the **track keyframes** from the current frame forwards or backwards until either the next **user keyframe** or the beginning/end of time range (whichever occurs first). The **delete all track**

keyframes control ( ) simply deletes all the **track keyframes** in the sequence.

Finally, the **delete all track and user keyframes** control ( ) deletes both the **track keyframes** and **user keyframes** in the sequence.

Tracking

Once the **user keyframes** are set, the actual tracking can be carried out in a number of ways:

1. The most usual way to track would be using the **track backwards** ( ) and **track forwards** ( ) controls. These play through the sequence tracking the 'in focus' track.
2. The **step backwards** ( ) and **step forwards** ( ) controls allow you to track a single frame at a time. This is particularly useful if there is a part of your sequence where the track is difficult, allowing you to track and adjust as necessary.
3. The final method is to use the **smart track** control ( ). This chooses the most intelligent order to do the tracking based on the **user keyframes** supplied. For example, given a 20 frame sequence and **user keyframes** at frames 1, 10 and 20, F_Tracker will first attempt a track at frame 2, then 9, then 11, then 19 and so on, expanding outwards from known frames.

Offset Tracking

F_Tracker has the ability to offset the track from the track region. This is particularly useful if there are no good tracking points on the subject you really wish to track yet good ones on a nearby object displaying the same motion. In order to offset the track, simply hold the 'ctrl' key as you drag the central crosshair. The track offset can be reset by clicking the **reset offset** control ( ). Offset tracking also allows you to adjust the track point during the track without introducing a discontinuity in your tracking data. The new track point will seemlessly follow the existing points even though it is tracking an entirely different part of the image.

Inputs

F_Tracker has one input which is the sequence containing the object to be tracked.

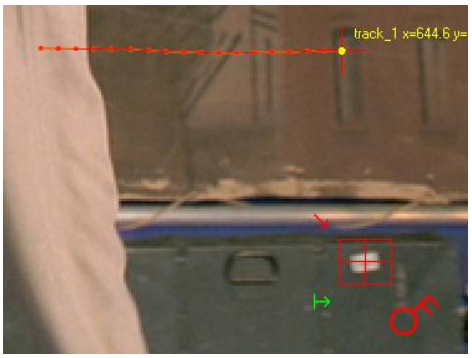


Figure 38.7 An offset track.

Controls

The following controls appear on the Shake Viewer shelf:

 **Toggle the on-screen tools** - toggles through the configurations for the on-screen tools.

 **Set start range** - sets the start of the range of frames to be tracked.

 **Set end range** - sets the end of the range of frames to be tracked.

 **Create user keyframe** - creates a **user keyframe**.

 **Delete user keyframe** - deletes a **user keyframe**.

 **Track backwards** - plays backwards through the sequence tracking from frame to frame.

 **Step backward** - tracks backwards one frame.

 **Step forward** - tracks forward one frame.

 **Track forwards** - plays forwards through the sequence tracking from frame to frame.

 **Smart track** - tracks from begining to end of frame range in an intelligent order.

 **Delete track keyframes backwards** - deletes **track keyframes** backwards through the sequence until either a user key frame or the begining of the sequence is reached.

 **Delete track keyframe and step backward** - deletes a **track keyframe** and steps forwards one frame.

 **Delete track keyframe** - delete the current **track keyframe**.

 **Delete track keyframe and step forwards** - deletes a **track keyframe** and steps backwards one frame.

 **Delete track keyframes forwards** - deletes **track keyframes** forwards through the sequence until either a user key frame or the end of the sequence is reached.

 **Delete all track keyframes** - deletes all **track keyframes** from the sequence.

 **Delete all track And user keyframes** - deletes both **track keyframes** and **user keyframes**.

 **Interpolate unsuccessful track keyframes** - adjusts the positions of **unsuccessful track keyframes** by interpolating the positions of the **reliable** and **successful track keyframes**.

 **Reset offset** - resets the offset between the track and the track region.

Parameters

+track_1 - the default 1st track. It is possible to add additional tracks by using the **add** button. Subsequent tracks will have the same parameters as the first; these are outlined below. Both the name of the track and the colour of the track region box can be edited.

track_1Range - the time range of the track.

track_1X - the horizontal postion of the track at the current frame.

track_1Y - the vertical position of the track at the current frame.

track_1RegionCentreX - the horizontal postion of the track region at the current frame.

track_1RegionCentreY - the vertical position of the track region at the current frame.

track_1RegionSizeX - the width of the track region. (Note this is not the track search region which is calculated automatically.)

track_1RegionSizeY - the height of the track region.

add - adds a new track.

delete - delete the current tracker.

save - saves tracking data as a text file.

load - loads tracking data from a text file.

Examples

All the images for the following examples can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Michael

In this example we wish to track the eye of a man walking towards the camera. Throughout the sequence the eye becomes larger in shot, undergoes changes in lighting and moves with a rather unpredictable path. In order to track this we will use F_Tracker's **smart track** feature.

To do this we will set a number of **user keyframes** spread fairly evenly over the sequence (about 5 in the 125 frames). The keyframes set are shown below:

Now simply press the **smart track** control. This should give a successful track.

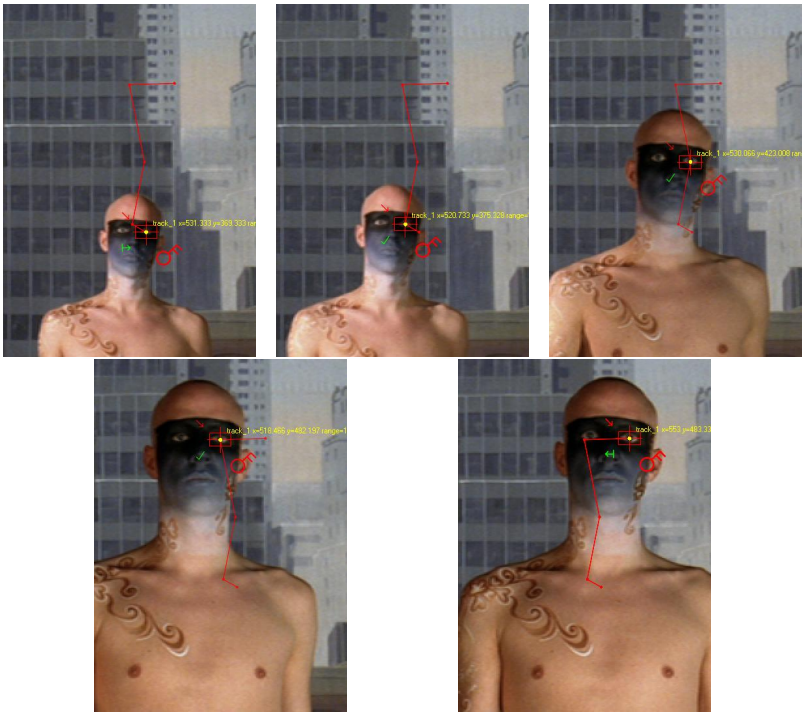


Figure 38.8 (left top to bottom right) User keyframes set at frames 1, 30, 61, 91 and 125 (the exact frames chosen are fairly arbitrary).

Figure 38.9 The successful track.

VectorConverter

This chapter looks at using Furnace's F_VectorConverter plug-in to convert to or from the Furnace motion vector format. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

In the Furnace vector format, pairs of motion vector fields are stored as floating point RGBA images. The two motion vector fields stored at each frame represent the (x, y) offsets which map the previous (backward) and next (forward) frames onto the current one. These offsets represent a displacement that can be applied to each pixel in the previous or next frame in order to construct an approximation to the current frame. The displacement from the previous to the current frame is stored in the red (x displacement) and green (y displacement) channels and is known as the **backward vector field**. The displacement from the next to the current frame is stored in the blue (x displacement) and alpha (y displacement) channels, and is known as the **forward vector field**. The displacement is measured in pixels at the current resolution.

For more information about vector fields and how to produce them, see the chapters on Local Motion Estimation on page 265 and the Furnace plug-in F_VectorGenerator on page 236.

F_VectorConverter allows you to convert from other vector formats into the Furnace format by offering an interface that allows you to rearrange, scale, offset and invert the channels from a pair of input vector images. It can also be used to convert from the Furnace vector format to another vector format of your choice.

Background

The best way to understand how F_VectorConverter works is to consider an example, so suppose we have an 8-bit vector clip containing a forward vector field only, where the forward displacements are stored in the red and green channels. Also, suppose we know that the maximum displacement in this vector clip is 50 pixels in either direction, so the true range of displacements -50 to 50 has been transformed to the range 0 to 254 for storage in our 8-bit image.

The first thing to do is to connect the first input vector clip. The red and green channels of our vector clip contain the vectors from each frame to the next one, so to obtain the vectors

from the previous frame to the current one we timeshift the vector clip backwards by one frame, then connect this to the **vectorsFromPreviousFrame** input of F_VectorConverter. The default behaviour of F_VectorConverter is to take the information from the red and green channels of the first input, which is fine in this case. Also, the motion is in the right direction so there is no need to invert the channels.

For the second input, we make a copy of the vector clip. This time, there is no need to timeshift it, so we set the time offset to zero. We then have a clip containing the vectors from the current frame to the next one, whereas what we want are the vectors from the next frame to the current one. Since there is no backward motion information in the clip, we simply invert these vectors to give the best available approximation to the vectors we require. So we connect the copied clip to the **vectorsFromNextFrame** input of F_VectorConverter. Again, the default behaviour is to take the information from the red and green channels, and in this case they will also be inverted by default, which is what we require.

Next, both clips will need to have their current values shifted and scaled so that they represent displacement in pixels in our floating point format. In the 8-bit clip, we assume that the zero point is in the middle of the range (this is usual), which means it is at 127. F_VectorConverter has a parameter for each input which is the current zero point, so we set both of these parameters to 127.

Now all that is left is to scale the values. To do this we divide them by 254, the size of the current range, and multiply by 100, the size of our desired range. This is equivalent to multiplying them by 100/254 = 0.3937. F_VectorConverter also has a scale parameter for each input clip, so we set both scales to this value, 0.3937. The output from F_VectorGenerator will now consist of the information from the original vector clip, transformed into the Furnace format, and is ready to be used in any of the other Furnace plug-ins which can take a vector input, for example F_Kronos.

Quick Start

The exact use of the F_VectorConverter plug-in will depend very much on the format your vectors are in, and the format you wish to convert to. The default behaviour when its two vector inputs are connected is to store the values from the red and green channels of the first input in the red and green channels of the output, and to invert the values from the red and green channels of the second input and store them in the blue and alpha channels of the output respectively, so in

theory connecting the two inputs might be all you need to do. However, depending on the exact conversion you wish to perform, you might need to rearrange the channels, adjust the inversion or scale the input values. For more advice on how to do this, please refer to the background section above or to the parameter descriptions below.

Inputs

F_VectorConverter has two inputs, **vectorsFromPreviousFrame** and **vectorsFromNextFrame**. If the vectors from either frame to the current one are not available, it is possible to supply motion vectors from the current frame to that frame instead and invert them, in order to obtain an approximation to the vectors we require.

Parameters

The parameters for this plug-in are described below.

+firstVectorInput - parameters for transforming the vectorsFromPreviousFrame input into the required format.

> **backwardXChannel** - the input channel from which to obtain the motion in x between the previous frame and the current one.
>
> **invertBackwardX** - whether to invert the backwardXChannel.
>
> **backwardYChannel** - the input channel from which to obtain the motion in y between the previous frame and the current one.
>
> **invertBackwardY** - whether to invert the backwardYChannel.
>
> **backwardScaleFactor** - amount to multiply the current pixel values by, i.e. the size of the desired range of values divided by the size of the current range of values. For example, to transform an 8-bit image with a range of 0 to 254 to a floating point image with a range of -1 to 1 this would be 2/254 = 127.
>
> **backwardZeroPoint** - The value in the input clip that represents a displacement of zero pixels, so for an 8-bit image with a range of 0 to 254 and the zero value in the middle of the range this would be 127.

+secondVectorInput - parameters for transforming the vectorsFromNextFrame input into the required format.

forwardXChannel - the input channel from which to obtain the motion in x between the next frame and the current one.

invertForwardX - whether to invert the forwardXChannel.

forwardYChannel - the input channel from which to obtain the motion in y between the next frame and the current one.

invertForwardY - whether to invert the forwardYChannel.

forwardScaleFactor - amount to multiply the current pixel values by, i.e. the size of the desired range of values divided by the size of the current range of values. For example, to transform an 8-bit image with a range of 0 to 254 to a floating point image with a range of -1 to 1 this would be 2/254 = 127.

forwardZeroPoint - The value in the input clip that represents a displacement of zero pixels, so for an 8-bit image with a range of 0 to 254 and the zero value in the middle of the range this would be 127.

VectorGenerator

This chapter looks at producing images containing motion vector fields using Furnace's plug-in F_VectorGenerator. For more information about motion vector fields, please refer to the chapter on Local Motion Estimation on page 265. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

Many of the Furnace plug-ins rely on motion estimation to produce their output. This means there is some overlap in the work they do: if you have more than one Furnace effect in your tree with one or more of the same inputs, they might well be performing identical motion estimation tasks. F_VectorGenerator is a utility plug-in designed to save processing time by allowing you to perform the motion estimation separately, so that the results can then be reused by other Furnace effects.

In general, once you have generated a sequence of motion vector fields that describe the motion in a particular clip well, they will be suitable for use in most of the Furnace plug-ins which can take vector inputs. However, F_Depth is unusual in that it requires highly smooth vectors (which would normally be generated with the **overSmooth** parameter turned on) in order to produce good output. Such vectors will lack detail and are unlikely to be useful for a plug-in such as F_Kronos, which uses the motion vectors to generate intermediate frames.

Output

The output from F_VectorGenerator is an RGBA image containing two sets of motion vectors for each frame, otherwise known as motion vector fields. The red and green channels contain the backward vector field: the x and y offsets per pixel that, when applied to the previous frame in the sequence, allows you to reconstruct an approximation to the current frame. Similarly, the blue and alpha channels contain the forward vector field: the x and y offsets needed to transform the next frame into an approximation to the current one. The closeness of the approximation can be controlled by altering the plug-in's parameters.

Quick Start

The plug-in has three required inputs: the current source frame and the previous and next frames of the source. These can be

set up manually or by using the VectorGenerator macro, which when selected will prompt you to choose an input sequence and will then set up the three source inputs with the correct time offsets for you. The quickest way to obtain a pair of motion vector fields is to click on the macro, choose an input file and use the default parameters.

Inputs

As well as the three source frames described above, the plug-in takes an optional foreground matte for each source frame. This can be used to help the motion estimation algorithm inside F_VectorGenerator understand what is foreground and background in the image, so that the dragging of pixels between overlapping objects can be reduced. White areas of the matte are considered to be foreground, and black areas background. Grey areas are used to attenuate between foreground and background.

Parameters

The parameters for this plug-in are described below. These parameters all control the Local Motion Estimation algorithm used inside F_VectorGenerator, so please see the chapter on Local Motion Estimation on page 265 for a more detailed description of their effects.

outputRegion - when a matte input is supplied, determines whether the motion vectors corresponding to the background or the foreground regions are output.

- **Foreground** the vectors for the foreground motion are output
- **Background** the vectors for the background motion are output

vectorDetail - determines the accuracy of the motion vectors. The maximum value of 1 will generate one vector per pixel. This will produce the most accurate vectors but will take longer to render.

smoothness - high smoothness will mean the output vector fields are less detailed, 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.

overSmooth - is a computationally intensive smoothing operation that peforms a different vector-smoothing operation

to normal. This generates highly smooth vector fields which are especially suitable for use in F_Depth.

blockSize - adjusts the blockSize used to calculate the vectors. Varying the blockSize will cause the motion estimation to lock on to different parts of the image.

+tolerances - By default we analyse motion based on the brightness of the image. Specifically this is the mean of the red, green and blue channels. However, we can bias this using the redWeight, greenWeight and blueWeight parameters. For example, if we set the red and green weight parameters to zero, we will only look for motion in the blue channel.

weightRed - - sets the contribution the red channel will make in the calculation of the motion vectors.

weightGreen - - sets the contribution the green channel will make in the calculation of the motion vectors.

weightBlue - - sets the contribution the blue channel will make in the calculation of the motion vectors.

matteComponent - where to get the (optional) foreground mask to use for motion estimation.

- **Source Alpha** - use the alpha of the source.
- **Source Inverted Alpha** - use the inverted alpha of the source.
- **Matte Luminance** - use the luminance of the matte.
- **Matte Alpha** - use the alpha of the matte.
- **Matte Inverted Luminance** - use the inverted luminance of the matte.
- **Matte Inverted Alpha** - use the inverted alpha of the matte.

Example

Taxi

In this example we will generate motion vectors for a clip showing a taxi driving past our offices in Soho, then use them to reconstruct a frame of the clip using Furnace's plug-in F_VectorWarper (see the chapter on F_VectorWarper on page 240).

Step by Step

1. FileIn Taxi.##.tif and duplicate it twice, setting the timeslip parameter on the duplicates to +1 and -1 respectively.

Go to a frame where you can see the taxi, for example frame 27 (shown in Figure 40.1).

2. Create an F_VectorGenerator node and connect the three clips to the three inputs: the clip with no time slip should be connected to the first input, the clip time-slipped by +1 to the second input, and the final clip to the remaining input. The output should look like Figure 40.2.

3. To show the output vectors are a good estimate of the motion in the clip, we can use the Furnace plug-in F_VectorWarper. Create an F_VectorWarper node and connect the output from F_VectorGenerator to the first input of the F_VectorWarper node. Connect the clip time-slipped by +1 to the second input, and the clip time-slipped by -1 to the third input, as shown in Figure 40.3. The output should look like the taxi image in Figure 40.3.

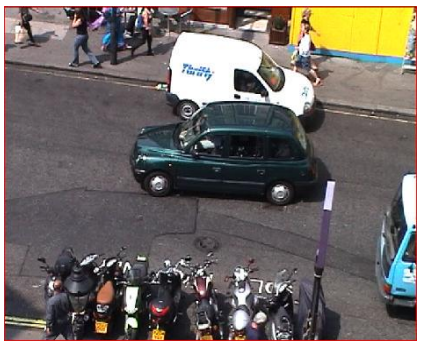


Figure 40.1 Taxi.

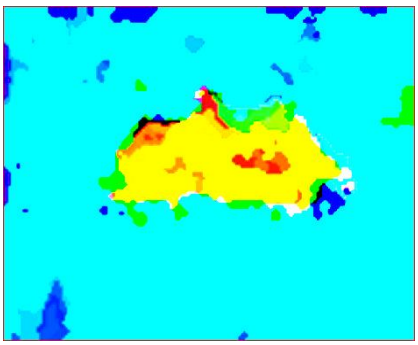


Figure 40.2 Motion vectors for the taxi sequence.

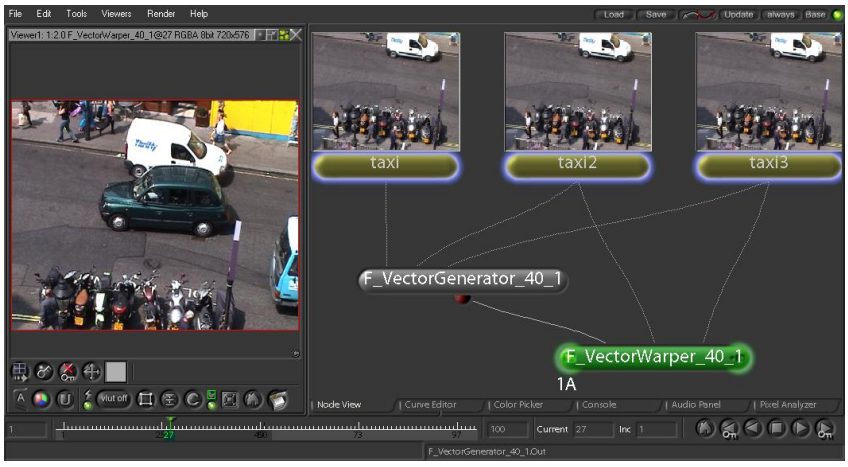


Figure 40.3 Using the vector fields in F_VectorWarper.

VectorWarper

This chapter looks at warping one frame of a sequence onto another using a previously generated motion vector field. For more information about motion vector fields and how to produce them, please see the chapters on Local Motion Estimation, on page 265, and F_VectorGenerator, on page 236. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

F_VectorWarper reconstructs one image from another using a vector field that describes the motion between the two images. It can be used as a quick way to check the output from the Furnace plug-in F_VectorGenerator. Given the adjacent frames and the vector field that describes the motion between this frame and the adjacent frames, the output from F_VectorWarper is an approximation to the current frame. If the resulting warped image is a reasonable approximation to the true image, then you can be sure that the vector field produced by F_VectorGenerator is a good estimate of the motion and is therefore suitable to use as an input to other, more complex Furnace effects such as F_Kronos.

F_VectorWarper can also be used to test whether or not motion vector fields from another application have been correctly converted to the Furnace vector format. Again, if the resulting image looks reasonable you can feel confident that the conversion has been performed correctly.

See the chapter on F_VectorConverter on page 232 for information about how to convert from one vector format to another.

Background

In the Furnace vector format, two motion vector fields are stored at each frame. The backward vector field represents the offsets that would be needed at each pixel in order to warp the previous image onto the current one. Similarly, the forward vector field describes the motion needed to warp the next image onto the current one. We can therefore use the backward vector field to reconstruct the current frame from the previous one, or the forward vector field to reconstruct the current frame from the next one. It is important to note that in most cases these reconstructions will not be exact, since the vector fields are likely to be only an estimate of the

actual motion between the frames. (The only situation in which your vector fields might model the motion between the images precisely is if both have been generated from a 3D application.)

Quick Start

Connect a vector clip to the Vectors input, and the previous and next images from your sequence to the SourceMinus1 and SourcePlus1 inputs respectively. Decide which frame you wish to reconstruct from. To reconstruct from the preceding frame, set the warp popup to SourceMinus1. To reconstruct from the following frame, set the warp popup to SourcePlus1. You will then obtain a reconstruction of the current frame from your chosen frame. This can be compared to the actual image at the current frame in order to give you an idea of how well the vector field you provided describes the motion between the current frame and the input frame.

Inputs

A motion vector clip and the source frames to reconstruct from.

Parameters

The parameters for this plug-in are described below.

warp - use this parameter to tell the plug-in which frame to use in the warp. At each frame, there are two sets vector fields: a backward vector field describing the motion from the preceding frame of a clip to the current one, and a forward vector field describing the motion between the next frame and the current one. If you chose to warp the SourcePlus1 frame, the plug-in will use the forward vectors from the Vectors clip. Conversely, if you warp the SourceMinus1 frame, the plug-in will use the backwardVectors from the Vectors clip.

- **SourcePlus1** use the forward vectors to warp the following frame.
- **SourceMinus1** use the backward vectors to warp the preceeding frame.

filtering - sets the filtering quality

- **Normal** uses a bilinear filter. This gives good results and is quicker to render than high filtering
- **Extreme** uses a sinc filter to interpolate pixels giving a sharper repair. This gives the best results but takes longer to process.

Example

For an example demonstrating the use of F_VectorWarper, please see the chapter on F_VectorGenerator on page 236.

WireRemoval

This chapter looks at the removal of wires from images using Furnace's plug-in F_WireRemoval. For hints, tips, tricks, and feedback please visit http://support.thefoundry.co.uk.

Introduction

Many effects movies feature complicated stunts that require an actor to be suspended by wires for their safety. These wires need to be digitally removed.

There are many ways of doing this including painting the wires out frame by frame and replacing large parts of the background to cover the wires. The method used depends on the type of image under the wire. Furnace is particularly good at replacing the painting method but it also includes tools to assist in clean plating when new backgrounds have to be tracked into place.

Background

Clean Plates

The use of clean plates in wire removal is very common and gives good results in certain situations.

Consider a scene shot with an actor suspended from wires and then the same scene shot again without the actor. This second sequence is called the clean plate. The wires from the first shot can be painted out using pixels from the clean plate leaving the actor suspended in thin air.

Shooting a clean plate if the camera is locked off is easy. If the camera moves then motion control rigs can be used to exactly reproduce the first pass. But it doesn't always work, particularly if the scene is shot against backgrounds that don't look the same on the second pass, such as clouds, sky or smoke. Motion control rigs are also expensive and that makes them a rarity. Often a clean plate is not shot during the filming and the compositor is left to create one by merging together unobstructed pixels from many frames. This single frame can then be tracked into place to cover the wires.

Furnace

Furnace's wire removal plug-in should make the process of wire removal much easier. It is particularly good at removing

wires over heavily motion blurred backgrounds or wires over smoke, dust or clouds. It can be used to remove each wire in a sequence or to quickly create a clean plate which can be then tracked into place.

The reconstruction of the background behind the wire can be done spatially, in other words using only information from the current frame. Alternatively, motion estimation techniques can be used to warp frames from before and after onto the current frame so that, where available, background pixels from these frames can be used to improve the repair. Our reconstruction methods are unique in that they remove the wire without removing and reapplying the grain. They are also tuned to remove long thin objects, leaving other objects untouched. For example, if transient foreground objects cover the wire, they will be left untouched.

Reconstruction Methods

Our four reconstruction methods are:

- Spatial
- Spatial With Global Motion
- Spatial With Local Motion
- Clean Plate

The spatial method takes the background information from adjacent pixels in the current frame and the cleanPlate method takes the information from a separate clean plate input. The other methods are temporal and attempt to get background information from frames either side of the current frame. Where this information is not available, for example because the wire covers part of the same region in one or more of the other frames, the spatial method will be used for the repair.

The spatial method is fastest. It uses a slope-dependent filter that interpolates across the wire at the most likely angle, given the image behind the wire. It works well when the wire is over backgrounds such as sky, clouds or smoke, and can also cope with some backgrounds where there is a noticeable gradient, such as the edge of a roof, building or car. If this fails and the wire is moving relative to the background, you should try one of the temporal methods. This looks on frames from before and after the current one for likely pixels to use to repair the region, and uses the spatial method in areas where there are none available.

Where traditional clean plates are possible or give better results than using F_WireRemoval to repair the wire on each

frame, you can supply a clean plate as the fourth input to the plug-in. It will then automatically matchmove it to repair each frame. If the overall luminance of the background is different to that of the clean plate or varies during the sequence, turn on **luminanceCorrect**. The luminance of the background pixels used for the reconstruction will then be adjusted before the repair is made. Of course, various Furnace tools can be used to create a clean plate, including an F_WireRemoval pass on a single frame where the repair has been successful.

Quick Start

To get you started let's consider a simple wire removal and describe what you need to do.

1. FileIn a clip that needs wires removed.
2. Load F_WireRemoval from the Furnace tab.
3. Connect the clip to the first input of the F_WireRemoval node.
4. Use the on-screen tools to draw a region that defines the position and shape of the wire to be removed.
5. Start the tracker so that it follows the wire during the clip.
6. Choose a repair method that removes the wire. Increase the **expandWidth** until the wire disappears.

Positioning the On-Screen Wire Tool

To make this easier, the output of the F_WireRemoval node should be set to **source** while you position the wire tool. That way, you will be able to see the wire you're trying to remove but won't have to wait for the node to repair the wire every time you change its position.

First choose the number of points that are needed to describe the wire: either three, for straight lines and simple curves, or five, for more complex shapes. This can be done by changing the **wireType** parameter or toggling the button in Shake's tweaker panel, which will look like  or  , depending on which wire type is currently selected. Position the on-screen wire tool so that it roughly fits the wire you want to remove. Then press the snap-to button  in the tweaker panel, which will find the edges of the wire and adjust the removal region to fit it more closely. It does this by locating the areas of highest gradient, which should correspond to the edges of

the wire. However, in practice the edges of the wire will be slightly soft, so the resulting region might not cover the whole width of the wire. To correct for this you can adjust the **expandWidth** parameter which expands the region outwards, over all keyframes, to ensure the entire width of the wire is covered.

Tracking

F_WireRemoval incorporates a tracker that will track the region to be repaired through the image sequence. Positioning the repair region sets a user keyframe for the current frame, so after you have done this once you can start the tracker by pressing  . Check that it has correctly tracked the wire across the other frames and adjust the track if necessary; the tracker controls in the tweaker panel are the same as those for F_Tracker, which are explained in the chapter for F_Tracker on page 223 and are also outlined in the Controls section below. Any points initially positioned off the image will remain off the image.

If it is not possible to track the wire automatically, the wire can be manually keyframed. Adjusting the on-screen wire tool on each frame will automatically set user keyframes, or alternatively they can be set by pressing  on the Shake viewer shelf.

Inputs

F_WireRemoval has four inputs. The first input (source) node should contain the wire to be removed. The second and third input (sourceMinusOffset and sourcePlusOffset) nodes are used in the **spatialWithGlobalMotion** and **spatialWithLocalMotion** repair modes and should be the same as the first input but offset temporally by setting the timeShifts to +offset and -offset respectively. If you're using either of these modes, you should ensure that the value of the **temporalOffset** parameter is the same as the size of the offsets on the input clips. The fourth input node is for an optional clean plate node, used by the **cleanPlate** repair mode which will warp this clean plate onto the current frame and use the warped image to reconstruct the background behind the wire.

Controls

The following controls appear on the Shake Viewer shelf:

 **Toggle display mode** - toggle the display mode for the on-screen wire tool: show the points and lines, show just the points or hide both points and lines.

 **Number of points** - changes the number of points used to describe the wire.

 **Create user keyframe** - creates a **user keyframe**.

 **Delete user keyframe** - deletes a **user keyframe**.

 **Snap to** - finds the edges of the wire and snaps the edges of the region onto them.

 **Track backwards** - plays backwards through the sequence tracking from frame to frame.

 **Step backward** - tracks backwards one frame.

 **Step forward** - tracks forward one frame.

 **Track forwards** - plays forwards through the sequence tracking from frame to frame.

 **Smart track** - tracks from begining to end of frame range in an intelligent order.

 **Delete track keyframes backwards** - deletes **track keyframes** backwards through the sequence until either a user key frame or the begining of the sequence is reached.

 **Delete track keyframe and step backward** - deletes a **track keyframe** and steps forwards one frame.

 **Delete track keyframe** - delete the current **track keyframe**.

 **Delete track keyframe and step forwards** - deletes a **track keyframe** and steps backwards one frame.

 **Delete track keyframes forwards** - deletes **track keyframes** forwards through the sequence until either a user key frame or the end of the sequence is reached.

 **Delete all track keyframes** - deletes all **track keyframes** from the sequence.

 **Delete all track And user keyframes** - deletes both **track keyframes** and **user keyframes**.

Parameters

The parameters for this plug-in are described below.

trackRange - the range of frames to track the wire over.

output - sets the output mode for the plug-in.

- **source** - output the untouched source image. Use this output mode to position the on-screen wire tool over the wire you wish to remove.
- **repair** - output the repaired source image, with the wire removed from under the on-screen tool.
- **wireMatte** - renders a matte for the wire. This may be useful if the wire has been tracked but cannot be repaired using F_WireRemoval and other techniques have to be used.

+wire - parameters to set the type and position of the wire to be removed.

wireType - choose the number of points needed to describe the wire you wish to remove.

- **Thee Point** - choose this if your wire is straight or a simple curve.
- **Five Point** - choose this if your wire has an s-shaped curve.

points - the points used to define the wire (this can be three or five points, as above).

startInnerWidth - the width of the wire at one end.

endInnerWidth - the width of the wire at the other end. This allows you to make your repair region wider at one end than the other, which is useful for cases where there is motion blur on the wire.expandWidth increase this parameter if you have used the snap-to control to find the edges of the wire and some pixels on the outside have been missed. This will expand the width of the repair region along its entire length, and for all key frames.

+repair - parameters to control how the repair is made.

repairMethod - sets the algorithm used to remove the wire from under the grain.

- **Spatial** - this method uses a slope dependent filter that interpolates across the wire at the most likely angle, given the image behind the wire. It uses information from the current frame only.
- **Spatial With Local Motion** - this method uses local motion estimation to align the sourcePlusOffset and sourceMinusOffset inputs onto the current frame. If the wire is not in the same place with respect to the background in these two frames, it can then use background information from them to fill in the background behind the wire in the current frame. Where no such information is available, for example if the wire covers part of the same region in all three frames, the spatial repair method above will be used. This is useful for sequences where the wire is moving and where the motion in the rest of the scene is non-uniform, for example if there are objects moving in the area surrounding the wire.
- **Spatial With Global Motion** - also aligns the sourcePlusOffset and sourceMinusOffset inputs onto the current frame, but uses global motion estimation. Again, it gets background information from these two frames where it can and uses the spatial repair method to fill in the rest. This is useful for sequences where the wire is moving and the motion in the rest of the scene is fairly uniform, for example as a result of a camera pan along an otherwise stationary scene.
- **Clean Plate** - choose this method if you have a clean plate you wish to use for the repair, or if F_WireRemoval does not do a good job of removing the wire from each frame. Connect the clean plate or single frame with the wire removed to the cleanPlate input; F_WireRemoval will then align this frame to the frame to be repaired in order to reconstruct the background behind the wire.

filterSize - is a trade off between the amount of grain removed and the blurring of image along the direction of the wire. If the wire you are trying to remove has detail within it (for example, a steel wire in which the twisted threads are reflecting light) then the algorithm may leave these alone thinking that they are grain. In this situation you should decrease the filter size. A value of zero will definitely remove the artifacts but also the grain, which would have to be put back using Furnace's ReGrain.

temporalOffset - the time offset of the additional frames to use for the spatialWithLocalMotion or spatialWithGlobalMotion methods. You should ensure that the value of temporalOffset is the same as the size of the time shift applied to the sourcePlusOffset and sourceMinusOffset inputs.

luminanceCorrect - turn this on for methods other than spatial repair where there are global luminance shifts between one frame of the sequence and the next, or between a frame of the sequence and a clean plate you are using for the repair. With luminanceCorrect turned on, F_WireRemoval will adjust the luminance of the background pixels it uses to the correct level before doing the repair.

luminanceBlockSize - change this value if luminance correction is not working as well as it should. The luminance correction uses overlapping blocks and matches the luminance within those blocks; changing the size of the blocks will change the regions covered by each block and could result in a better match.

Examples

This section should be used in conjunction with the example images which can be downloaded from our web site. For more information, please see the Example Images section on page 17.

Clouds

These time-lapse clouds over a row of terraced houses would be tricky to clean plate unless an entirely separate sequence of clouds were available. This shot would normally be painted. However, Furnace does a good repair on it using the spatial reconstruction method.

Step by Step

1. Start Shake and FileIn the WireClouds.####.tif .
2. Flipbook the clouds. Note that the wires have already been stabilized (using F_Steadiness) and will not therefore need to be tracked in this example.
3. Connect the clouds to the first (source) input of the F_WireRemoval node. Make sure you're working at full resolution. (Figure 42.2 on the facing page).

Figure 42.1 Clouds over houses

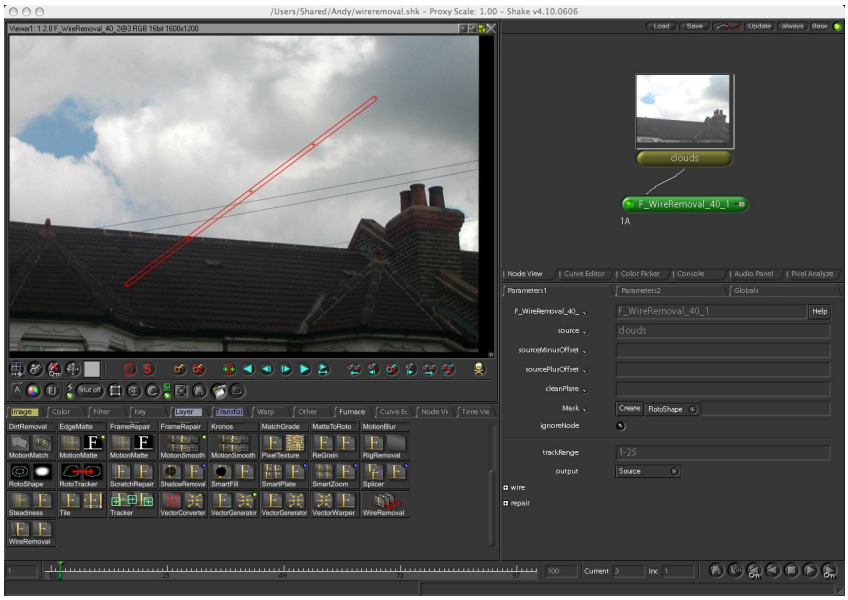


Figure 42.2 Shake UI showing F_WireRemoval.

4. Now we need to help F_WireRemoval find the wire we wish to remove. F_WireRemoval has a widget to specify which wire to remove. The wire widget can accommodate curved as well as straight wires. As our wire is straight a 3 point widget will be sufficient. Click on  to toggle to a simpler 3 point wire removal widget.

5. Place one of the end positions of the widget on the left most point of the wire to remove. **N.B. The wire pases**

infront of the roof and we want to repair that too. (Figure 42.3) Place the other end of the wire widget on the right most point of the wire to remove. (Figure 42.5). And position the mid point of the wire widget on the wire wire. (Figure 42.4)

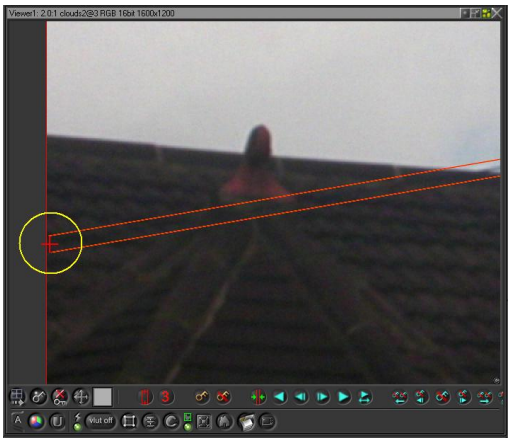


Figure 42.3 Start Position

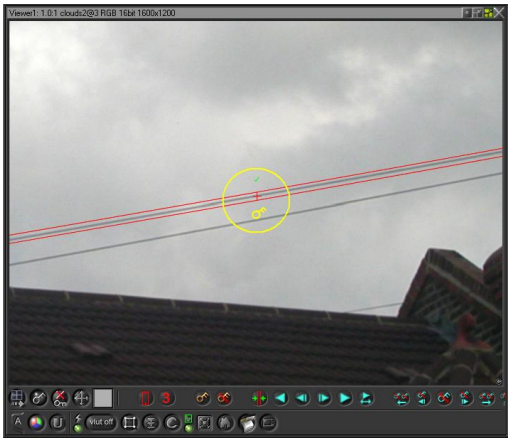


Figure 42.4 Mid Position

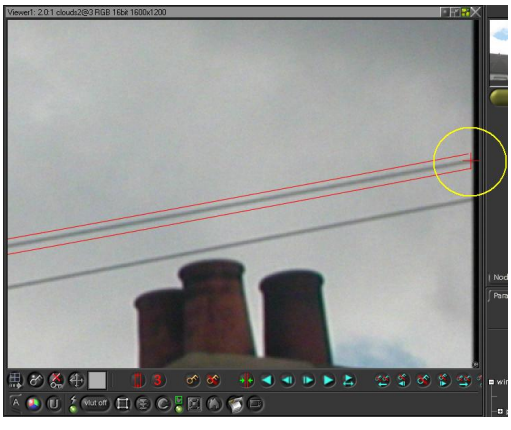


Figure 42.5 End Position

6. To see the repair select F_WireRemoval node and set **output** to **Repair** instead of **Source** and hide the wire removal widget by clicking  twice. **N.B. Clicking once removes the parallel lines but not the point crosshairs.**

7. This is a spatial repair. F_WireRemoval has three other methods of repair and our current repair can be improved by these. Duplicate the **clouds** FileIn twice. Now we have three clouds FileIn nodes called **clouds**, **clouds2 (A)** and **clouds3 (B).** In the **Timing** tab of the **clouds2** FileIn, set **timeShift** to 1 and connect it to the second input of F_WireRemoval (sourceMinusOne). In the **Timing** tab of the **clouds3** FileIn, set **timeShift** to -1 and connect it to the third input of F_WireRemoval (sourcePlusOne).

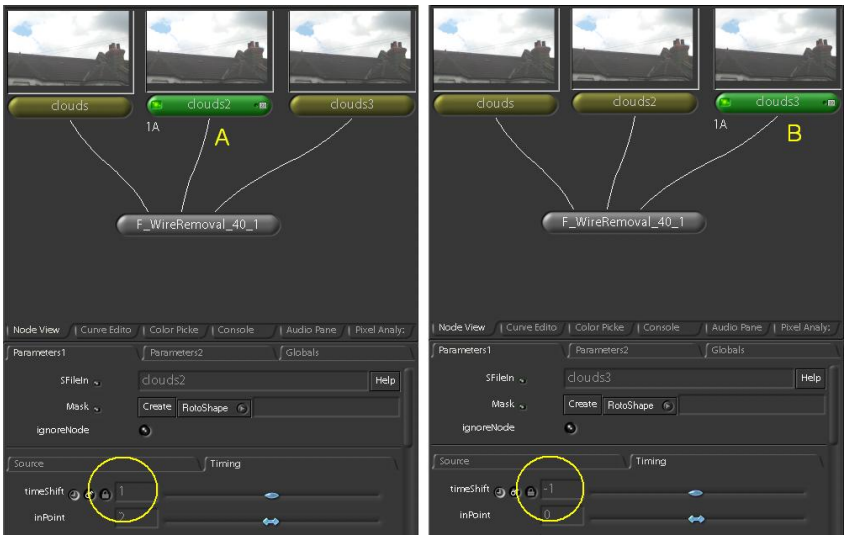


Figure 42.6 Timeshift source -1 and source +1

8. Select **repairMethod "Spatial With Global Motion"** to use a superior repair method. Frame 1 may not appear any better than regular **Spatial** repair. However, frame 2 has performed a superior fix to the wire over the roof. Any slight inaccuracies in the positioning of the wire can be made less significant by increasing F_WireRemoval's **expandWidth** parameter in the **wires** folder.

9. Render the sequence and note that the **Spatial** algorithm has done a good job of repairing the wire. If the wire repair appears lighter than the surrounding clouds check on **luminanceCorrect** and this will improve the repair.

Bricks

On this shot of a wire over a brick wall (Figure 42.7) we use **Spatial With Global Motion** and **Clean Plate** repair methods. We will see how a **Spatial With Global Motion** repair can be

Figure 42.7

used to feed back in to the clean plate input of F_WireRemoval to provide a seemless repair of the slowly moving wire.

Step by Step

1. FileIn the brick clip (WireBricks.####.tif)
2. Apply F_WireRemoval from the Furnace tab and connect the output of the brick clip to the first input of the F_WireRemoval node.
3. Now we need to help F_WireRemoval find the wire we wish to remove. F_WireRemoval has a widget to specify which wire to remove. The wire widget can accommodate curved as well as straight wires. As our wire is straight a 3 point widget will be sufficient. Click on  to toggle to a simpler 3 point wire removal widget.
4. Place one of the end positions of the widget on the left most point of the wire to remove (Figure 42.8 on the next page). Place the other end of the wire widget on the right most point of the wire to remove (Figure 42.10 on the facing page). And position the mid point of the wire widget on the wire wire. (Figure 42.9).

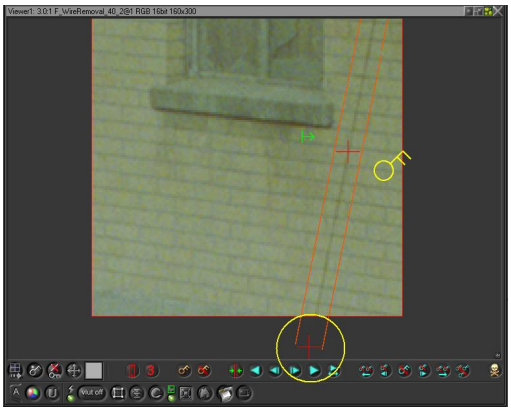


Figure 42.8 Start Position

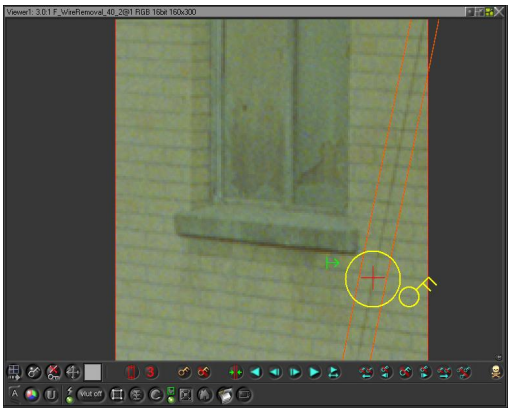


Figure 42.9 Mid Position

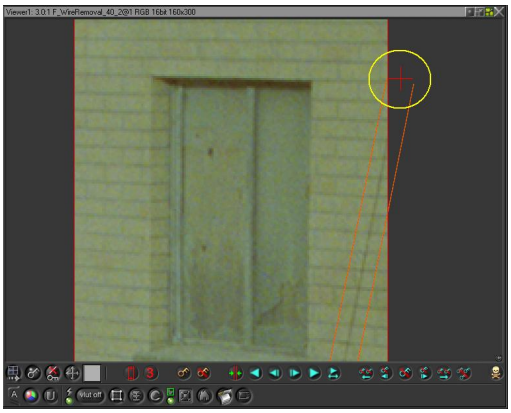


Figure 42.10 End Position

5. To see the repair select F_WireRemoval node and set **output** to **Repair** instead of **Source** and hide the wire removal widget by clicking  twice. **N.B. Clicking once removes the parallel lines but not the point crosshairs.**

6. This is a spatial repair. F_WireRemoval has three other methods of repair and our current repair can be improved by these. Duplicate the **brick** FileIn twice. Now we have three brick FileIn nodes called **brick**, **brick2 (A)** and **brick3 (B).** In the **Timing** tab of the **brick2** FileIn, set **timeShift** to 1 and connect it to the second input of F_WireRemoval (sourceMinusOne). In the **Timing** tab of the **brick3** FileIn, set **timeShift** to -1 and connect it to the third input of F_WireRemoval (sourcePlusOne).

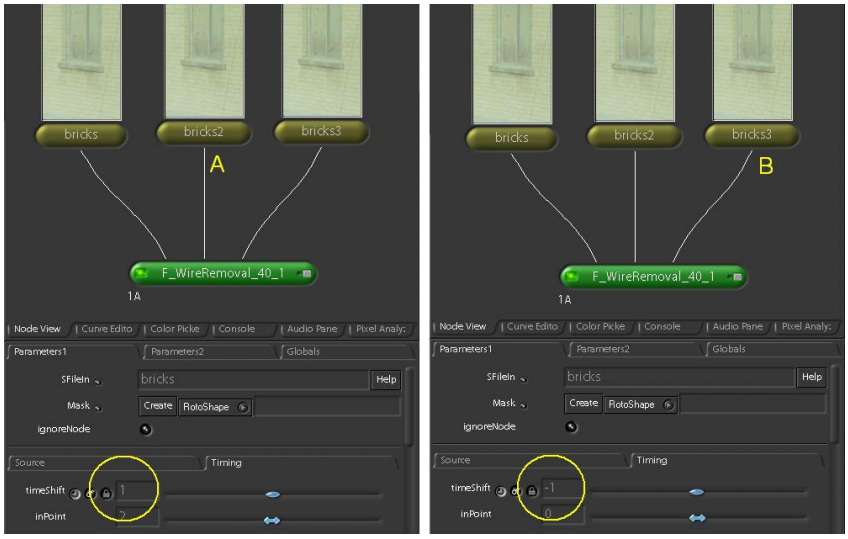


Figure 42.11 Timeshift source -1 and source +1

7. Now we have access to the frames before and after the current frame we need track our wire through the sequence. To do this click  to step forward a frame and track the wire in the process. For this sequence the tracker locks well to the wire, however on other examples it might be necessary to manually adjust some of the tracking results by moving the wire widget slightly. Allow the wire to be tracked for 40 frames.

8. Select **repairMethod "Spatial With Global Motion"** to use a superior repair method. Frame 1 may not appear any better than regular **Spatial** repair. Any slight inaccuracies in the positioning of the wire can be made less significant by increasing F_WireRemoval's **expandWidth** parameter in the **wires** folder.

9. Render the sequence and play the results. You will note that the repair, although good on individual frames, is not that impressive when played. This is because the temporal reconstruction is being hampered by the slow moving wire in the previous and next frames. When F_WireRemoval can't find a clean piece of footage in the

surrounding frames to perform the repair, it reverts to a **Spatial** repair. The majority of the wire is being repaired spatially and that is what your eye can see.

10. To overcome this we need to utilise the **Clean Plate** repair method. There are many ways to achieve a clean plate for this wire repair, but the simplest is to use F_WireRemoval itself. We can help F_WireRemoval overcome the slow moving wire by setting the **timeShift** in **bricks2 (A)** and **bricks3 (B)** to 20 and -20 respectively. Now set **temporalOffset** in the **repair** folder of F_WireRemoval to 20. This indicates to F_WireRemoval that the previous and next inputs are 20 frames before and 20 frames after the current frame. Because in these frames the wire is in very different positions the amount of repair that is **Spatial** is significantly reduced.

11. To see the repair select F_WireRemoval node and ensure **output** is set to **Repair** instead of **Source** and hide the wire removal widget by clicking  twice. Select frame 20 to give F_WireRemoval a valid previous and next frame. This is the best clean plate F_WireRemoval can produce.

12. Attach a **FileOut** node to F_WireRemoval and export the result of frame 20 to a file on disk.

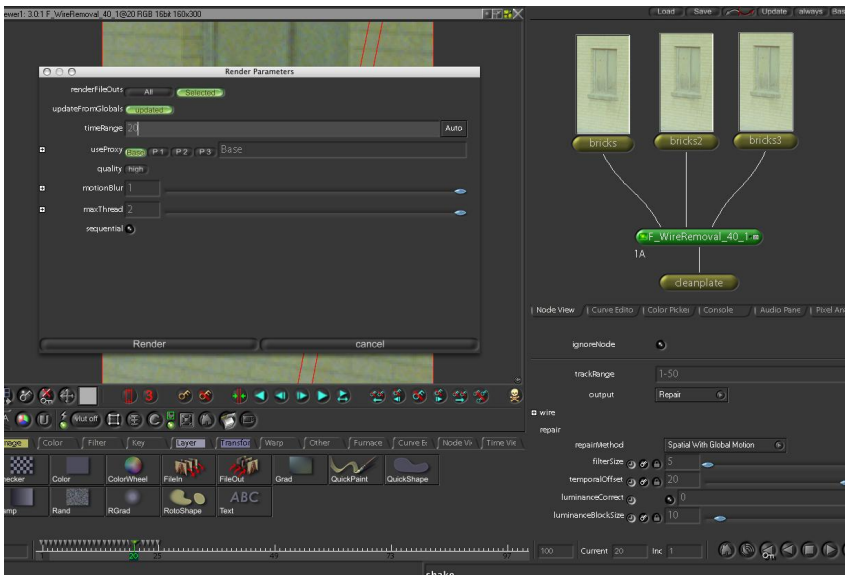


Figure 42.12 Exporting Clean Plate

13. Use a **FileIn** to reload the clean plate file from disk and connect this input to F_WireRemoval's fourth input. The fourth input is for supplying a clean plate from which F_WireRemoval can construct the wire's repair.

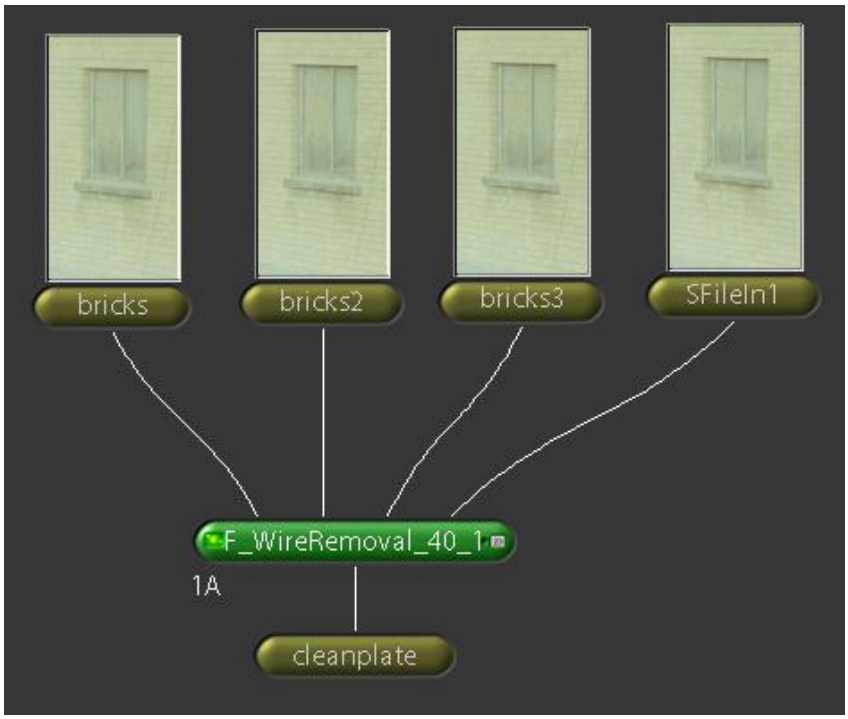


Figure 42.13 Reimporting the Clean Plate

14. Now that we have a clean plate and it is connected set **repairMethod** to **Clean Plate**. This will use the clean plate to perform the reconstruction. Render 40 frames in a flip book. You will notice the repair is superior to that achieved with **Spatial With Global Motion** and that when played the repair cannot be seen.

Global Motion Estimation

Introduction

Furnace has several effects based on global motion estimation (GME). They all calculate a four corner pin, which finds the best fit of one image onto another, then apply that pin. These effects differ in how that corner pin is calculated and to which image that pin is applied. This chapter describes the general idea behind these plug-ins; for more detailed information on each plug-in, please read their individual chapters.

These effects are:

- **F_Align** - which registers one clip to a similar reference clip, by finding a corner pin from each source clip frame to the corresponding reference clip frame. For example, you can use this to align two separate but similar steady cam shots of the same scene.
- **F_Steadiness** - which removes motion from a single clip, by calculating a pin that either locks all frames in the clip to a single reference frame, or smooths that motion out over a window of frames. For example, you can use this to remove camera shake.
- **F_MotionMatch** - which makes one clip move like another, by analysing the motion in a reference clip and applying that to the source clip. For example, you can use this to track in a logo onto a moving background.
- **F_ColourAlign** - which uses GME to split the colour planes in a frame into three separate images and to bring them back into alignment. This is used to fix colour registration problems.
- **F_SmartPlate** - which uses GME to create a large background plate, by aligning separate frames in a panning/tilting shot and blending them together.
- **F_SmartZoom** - which uses GME to create a super-resolution image, by aligning separate frames from a near static shot and blending them together.

What is Global Motion Estimation?

Global motion estimation is a technique that attempts to map one image onto another with a simple four corner pin. This

differs from local motion estimation (LME), which attempts to find where each individual pixel in the image is in the other image. (For more information about local motion estimation, please see the chapter on page 265). GME is much cheaper to compute than LME, but gives you less information about the image. Nevertheless, it is still very powerful for a variety of applications.

The GME engine can be told what type of motion to expect, this can be a combination of any of:

1. **translation** - which allows the four corners to translate by the same amount,
2. **rotation** - which allows the corners to rotate about their centre,
3. **scale** - which allows the size of the area defined by the corners to change,
4. **perspective** - which allows the angles at the corners to change, so that the area defined by them is no longer a rectangle.

The more types of motion you allow, the more expensive the motion estimation becomes. For many scenes, rotation and translation are sufficient.

The GME effects have an accuracy control, which controls the amount of work the Foundry's GME engine does to calculate the global motion. Typically, the higher this is, the better the estimation, but the more expensive it is.

Limitations of GME

As stated above, global motion estimation simply calculates a four corner pin to transform one image onto another. This means that GME can't be used to match two images where there is heavy parallax, very complicated foreground motion, changing objects and so on.

The best way to think of what GME can do is that if you can do it with a four corner pin, it can; if you can't, it can't. However, GME will take the pain out of hand matching pins frame by frame.

The Analysing Global Motion Estimation Effects

F_Align, F_MotionMatch and F_Steadiness work in a similar way, which is distinct from the way the other GME based effects

work. These three effects calculate a four corner pin for each frame and save it into the corner pin parameters. These pins are then used during the render to move the source image.

In previous versions of Furnace, F_Align and F_Steadiness effects were contained within one plug-in called F_Steadiness. We have split them out and made them simpler to use.

Using Them

These effects analyse images over a range of frames to figure out their four corner pins. This is done in response to the user pressing the 'analyse' button in the effects control panel. During analysis, the effect will run through a range of frames adding keys to the corner pin parameters. These corner pins are then applied to the source clip to render a frame.

During render, Shake will not allow an effect to fetch images at frames other than the current one. This means that F_Steadiness and F_MotionMatch have to have a separate analysis pass that happens interactively; they can't compute the corner pin during render. However, F_Align, because it only ever needs the two current frames from each clip, can compute the corner pin on the fly (but not keyframe it!). This leads to a slightly different mode of operation for the following effects:

- F_Steadiness and F_MotionMatch
  - need to have an analysis run before they render useful output,
  - will always use the value in the corner pin parameters when rendering the output image.
- F_Align
  - no need to have the analysis run to render output, but doing so will give you a key-framed corner pin,
  - during render it will use the value of the corner pin parameters only if there is a keyframe there, otherwise it will analyse on the fly during render. This means that analysis will speed up later renders, as just rendering the corner pin is much cheaper than calculating it.

Some parameters to the effect control how the effect performs GME analysis, and some only affect the rendering. If you ever modify one of these parameters, then any analysis you may have performed will be out of date. To let you know, an overlay warning will be posted whenever this happens. You don't have to re-analyse and your renders will still look at the keyed corner pins.

If you have not modified a parameter that affects analysis (the warning overlay will let you know), pressing 'analyse' will only re-analyse a frame if there is no key on the corner pin at that time. This avoids redundant re-analysis if you have interrupted the analysis or extended the analysis range. However, if you want to force re-analysis, press 'clearAnalysis' and all keys will be deleted.

These three effects have an analysis region rectangle parameter which is used to specify which area of the reference image should be analysed during GME. So, for example, with Steadiness set to 'lockMode', this is the area inside the lock frame that a match will be sought for. The documentation for each plug-in documents exactly how to use the analysis region.

Controls

The controls common to all GME plug-ins are described below. They are grouped into two sections: ones that determine how analysis is carried out, and ones that control the rendering of the output.

Controls That Affect Analysis

The following parameters and controls affect the analysis of the four corner pin.

analyse - This is a push button which will trigger an analysis of the input clips and calculate a corner pin. To interrupt the analysis, either click again with the left mouse button or hit the escape key. Escaping analysis will not delete the corner pin keys that have already been calculated.

clearAnalysis - Pressing this pushbutton will delete all keyframes from the corner pin parameters, allowing you to force a re-analysis if you feel the need to.

analysisRange - this controls the range of frames any analysis will be run over. It can be one of:

- **Specified Range** - which will look at the parameters **analysisStart** and **analysisStop** for the range of frames to analyse,
- **Source Clip Range** - which will automatically determine the range of frames to analyse from the length of the input clip.
  - Note that not all nodes in Shake have a bound frame range; for example, the default state of the ColourWheel has an infinite range of frames. If the effect is wired to such a node and the "analysisRange" is

set to "Source Clip Range", then an warning overlay is posted and it will refuse to analyse the inputs. In such a situation, you have to set this parameter to "Specified Range".

analysisStart - the first frame to analyse from if "Analysis Range" is set to "Specified Range"

analysisStop - the last frame to analyse from if "Analysis Range" is set to "Specified Range"

scale - a toggle that indicates whether the calculated corner pin can include a scaling factor.

rotate - a toggle that indicates whether the calculated corner pin can include rotations.

translate - a toggle that indicates whether the calculated corner pin can include translations in x and y.

perspective - a toggle that indicates whether the calculated corner pin can include perspective transforms.

accuracy - this controls the time/accuracy trade off in the GME engine. The higher this is, the slower it goes, but you have a better likelihood of a good result.

analysisRegion - this is the region analysed to calculate the four corner pin. This is especially useful when doing any form of frame locking, in which case, go to the lock frame, look at the reference clip and position the box over the area you want locked.

renderDuringAnalysis - if set, this toggle will cause the effect to update the time line and render a freshly analysed frame so you can see the progress of the effect. Doing so will slow down the analysis somewhat, so toggle this off to speed up the general analysis.

Parameters That Affect Rendering

These following parameters control how a GME effect renders the four corner pin. Some of them are set during the analysis pass.

cornerLL - the lower left of the corner pin calculated by the analysis pass.

cornerLR - the lower right of the corner pin calculated in the analysis pass.

cornerUL - the upper left of the corner pin calculated in the analysis pass.

cornerUR - the upper right of the corner pin calculated in the analysis pass.

pinOrigin - the rectangle that is set during the analysis to specify the origins of the four corner pin. The corners of this rectangle are what are moved to the matching pin positions. This is set to the size of the input clip at the frame where the analysis was triggered.

invert - if set, then the inverse of the calculated four corner pin is used during render.

filtering - this controls the quality of the rendering.

- **low** - perform nearest neighbour filtering
- **medium** - perform bilinear filtering
- **high** - perform sinc filtering

Widgets

All the Analysing GME effects have two on-screen widgets, one to provide feedback and one to set up the analysis region.

Analysis Region Widget - This is a rectangle widget which you use to set the analysis region over the reference image.

Four Corner Widget - This is a widget that shows the state of the four corner pin that has been calculated. You can change it by grabbing any of the corners and tweaking the shape of the pin. To give you more feedback as to what frames have been analysed, it will be drawn solid if there is a key in the corner pin at the frame being displayed, otherwise it will be drawn dashed.

Local Motion Estimation

Introduction

A lot of the tools in Furnace make use of Motion Estimation technology. Motion Estimation tends to fall into two areas: Global Motion Estimation and Local Motion Estimation. In this chapter we look at the parameters for Local Motion Estimation (or LME), which is the per-pixel motion analysis used in tools such as Kronos.

Background

The easiest way to understand LME is to think in terms of vector fields. A vector field for an image in a sequence has the same dimensions as the image, but contains an (x,y) offset per pixel. These offsets show how to warp a neighbouring image onto the current image. Clearly, as most of the images in a sequence have two neighbours, each can have two vector fields. These are called the 'forward motion vectors' where they represent the warp of the image in front of the current one, and 'backward motion vectors' where they represent the warp of the image behind the current one.

This is an approximation to what is really going on in an image sequence. A single vector field can be though of as representing a warp or a morph - sometimes referred to as a 'rubber sheet' warp and cannot truly represent multiple overlapping motions. This effect can be seen where moving foreground objects appear to drag the background. To help cope with this restriction, a number of the tools allow the use of two vector layers per frame, one representing foreground motion and one representing background motion, where a Matte input is used to identify the layer separation. In addition, see the **occlusions** option in the parameter descriptions below, which attempts to improve the rubber-sheet effect when separate vector layers aren't being used.

LME is an expensive compute operation; most of the tools which use LME have a choice between generating vector fields on the fly and using pre-calculated vector fields. When using pre-calculated vector fields, these are typically generated by the F_VectorGenerator tool, although they may also come from other third-party sources.

The vector generation process has a number of tuning parameters which can be used to adapt the vectors to suit particular

sequences, as well as to trade off render time verses accuracy of vectors.

Using Pre-Calculated Vector Fields

LME is an expensive compute operation. For this reason, most of the plug-ins which use LME allow you to choose between generating vector fields on the fly and using pre-calculated vector fields. Pre-calculated vector fields can be generated by the F_VectorGenerator tool, described on page 236, and may also come from third-party sources. These can then be passed into the following effects as inputs:

- F_Depth
- F_DeNoise
- F_DirtRemoval
- F_Kronos
- F_MotionSmooth

Therefore, if you are going to be using more than one of these effects in your script, it might be worth generating the vector fields beforehand with F_VectorGenerator, so that they can be reused.

Most of the time, a vector field that produces good output in one of these effects will work well in the others as well. However, F_Depth is a special case, as it generally requires smoother vector fields than the other plug-ins in order to produce good output.

Parameters

Not all tools using LME have all of these parameters. They can be categorized into two groups - one for the generation of the vectors, and one for their use in picture warping.

Vector Generation Parameters

vectorDetail - Adjust this to vary the resolution of the vector field. The larger **vectorDetail** is the greater the processing time, but the more detailed the vectors should be. A value of 1.0 will generate a vector at each pixel. A value of 0.5 will generate a vector at every other pixel. For some sequences, a high **vectorDetail** near 1.0 generates too much unwanted local motion detail; often a low value is more appropriate.

smoothness - Vector 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.

oversmooth - This is a computationally intensive smoothing operation that peforms a different vector-smoothing operation to normal. This generates highly smooth vector fields, which is useful for the F_Depth tool, but may sacrifice a lot of required detail for other LME operations.

blockSize - The vector generation algorithm subdivides the image into small blocks, and separately tracks them. **blockSize** defines the width and height of these subdivisions. Smaller values will produce noisy data, whereas larger values may produce data that is lacking in detail. This value should rarely need editing; some sequences may benefit from using large block sizes to help the algorithm track regions better where the algorithm isn't 'locking on' to the overall motion in the sequence.

tolerances - For efficiency, much of the LME is done on luminance only - i.e. using monochrome images. The tolerances allow you to tune the weight of each colour channel when calculating the image luminance. These parameters rarely need tuning. However, you may, for example, wish to increase the red weighting **weightRed** to allow the algorithm to concentrate on getting the motion of a primarily red object correct, at the cost of the rest of the items in a shot.

- **weightRed**
- **weightGreen**
- **weightBlue**

correctLuminance - LME is highly dependent upon the idea that the brightness of objects doesn't vary through a sequence. Where brightness varies rapidly - for example a highlight moving across the bodywork of a car - the motion calculation will perform poorly. The luminance of a shot can come from other sources too - such as an overall flicker problem. In these cases where there is a global luminance shift, toggling this control on will allow the LME algorithm to take account of overall brightness changes between frames.

Picture Warping Parameters

filtering - sets the quality of filtering when producing in-betweens (F_Kronos) or replacement frames (F_FrameRepair).

- **normal** - use bilinear interpolation which gives good results and is a lot quicker than extreme.
- **extreme** - uses a sinc interpolation filter to give a sharper picture but takes a lot longer to render.

warpMode - sets how to control the new timing of the clip.

- **simple** - this is the quickest option, but may produce less than optimal results around moving objects and image edges.
- **normal** - this is the standard option, with more optimal treatment of moving objects and image edges.
- **occlusions** - this is an advanced option that can improve the results when not doing a separated picture build with multiple vector layers and mattes. It attempts to reduce the level of background dragging that occurs between foreground and background objects.

showVectors - switch this on to display the vectors on the screen.

Vector Field Representation

In Shake, the Furnace tools which produce or consume vectors use vector fields encoded into floating-point RGBA images. This means we can encode two vector fields per frame. One field's (x,y) values are in r and g, and the other field's values are in b and a. The (r,g) pair represents the offsets required to fetch pixels from the previous frame to match pixels in the current frame (the backward vectors). The (b,a) pair represents the offsets required to fetch pixels from the following frame to match pixels in the current frame (the forward vectors).

As an example, Figure 44.1 on the facing page shows the Shake interface. A VectorGenerator node is being used to generate the forward and backward vectors for the taxi sequence. The central FileIn node is the current frame; the two on either side are timeshifted forwards (sourceM1_ on the right) and backward (sourceP1_ on the left) by one frame. Underneath, an IDisplace node, whose controls are visible, is being used to warp the forwards frame (sourceM1_) to make it look like the current frame (sourceIn), using the forwards vectors in

channels (b,a). The **scale** parameters of the IDisplace node are set to (-1,-1), and the xChannel is set to B, with the yChannel set to A. Alternatively, you could use an F_Warper node, which can do a similar job to the IDisplace node, but has a simpler and more intuitive interface.

For more details on the use of the F_VectorGenerator node and the F_VectorWarper node, please see their respective chapters.

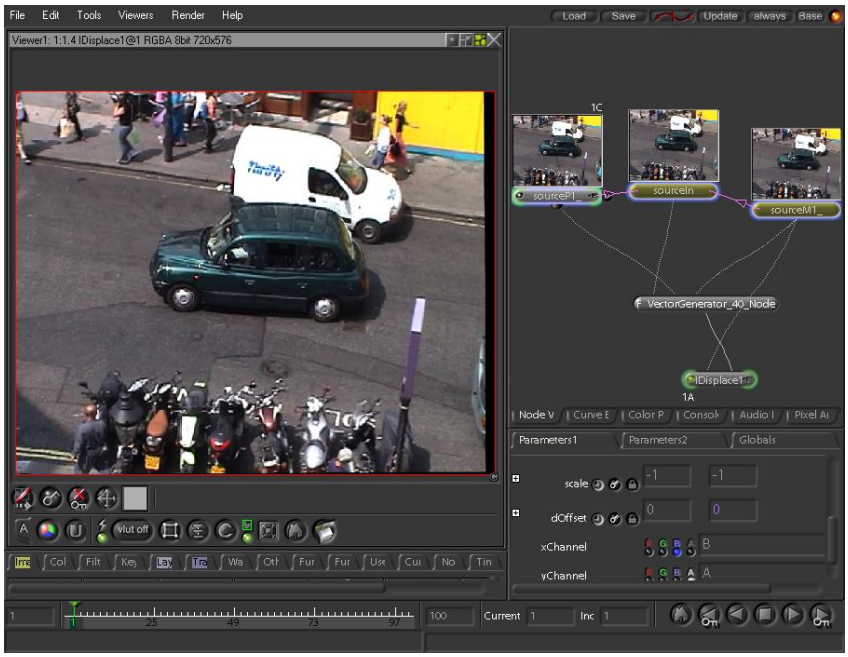


Figure 44.1 Using the vector fields in IDIsplace.

Appendix A

Release Notes

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

Furnace 4.0v4

This is a maintenance release of Furnace for Shake.

Release Date

10th March 2008.

Requirements

1. Shake 4.10 on Mac OS X and Linux (Fedora Core 4). Shake 4.00 on Linux (Fedora Core 4).
2. Foundry FLEXlm Tools (FFT) (4.0v1 or later) for floating license support.

Bug Fixes

1. F_DirtRemoval - BUG ID 1289 - The dirt mask input previously stopped F_DirtRemoval from doing any dirt repairs, this has been fixed.
2. F_WireRemoval - BUG 2108 - Previously when a script was reloaded the manually altered parameters in F_WireRemoval were over written, this has been fixed.
3. F_WireRemoval - BUG 1133 - You can pick up and move the wire removal widget using one click and drag instead of moving each of the points on the line separately.

Furnace 4.0v3

Not released to public.

Furnace 4.0v2

This is a maintenance release of Furnace for Shake.

Release Date

3 April 2007.

Requirements

1. Shake 4.10 on Mac OS X and Linux (Fedora Core 4). Shake 4.00 on Linux (Fedora Core 4).
2. Foundry FLEXlm Tools (FFT) (4.0v1 or later) for floating license support.

Bug Fixes

1. F_Align - BUG ID 1164 - produces incorrect alignment at proxy resolutions when analysis has not been carried out. This has been fixed.
2. F_ChannelRepair - BUG ID 1167 - crashes after a CornerPin node. This has been fixed.
3. F_ColourMatte - BUG ID 1169 - incorrect output and alpha when roto inputs are the same. This has been fixed.
4. F_Contrast - BUG ID 1155 - right side of image set to black for 8 and 16 bit images with alpha. This has been fixed.
5. F_Contrast - BUG ID 1168 - crash with differing input and output DOD's. This has been fixed.
6. F_DeFlicker1 - BUG ID 1234 - reference clip output incorrect. This has been fixed.
7. F_RotoShape - BUG ID 1048 - instability causing a crash under some circumstances. This has been fixed.

Furnace 4.0v1

This is a major new release of Furnace for Shake including new plug-ins and improvements.

Release Date

19 February 2007.

Requirements

1. Shake 4.10 on Mac OS X and Linux (Fedora Core 4). Shake 4.00 on Linux (Fedora Core 4).
2. Foundry FLEXlm Tools (FFT) (4.0v1 or later) for floating license support.

New Features

1. F_Contrast.
2. F_DeFlicker2.
3. F_RotoShape.
4. F_RotoTracker.
5. F_ShadowRemoval.
6. F_VectorConverter.
7. F_VectorGenerator.
8. F_VectorWarper.

Improvements

1. New motion estimation engine.
2. Motion vector inputs into plug-ins to utilise external motion vector assets.
3. Performance improvements.
4. Unified operation and appearance of on-screen tools.
5. F_ChannelRepair - temporal controls removed for simpler workflow.
6. F_ColourMatte - roto shapes have been added to speed workflow.
7. F_DeBlur - optional deringing parameter added.
8. F_DeGrain - temporal controls removed for simpler workflow.
9. F_DeNoise - optional deringing parameter added.
10. F_Kronos - algorithmic changes have reduced the number of parameters needed.
11. F_MotionRepair - split into F_MotionSmooth and F_FrameRepair for simpler workflow.
12. F_PlanarPatcher - reworked as F_MotionMatch with simpler workflow.
13. F_ReGrain - now able to reproduce grain response with respect to exposure.
14. F_RigRemoval - matte input added.
15. F_ScratchRepair - now deals with curved scratches and linear scratches at arbitrary angles.

16. F_Steadiness - split into F_Steadiness and F_Align for simpler workflow.
17. F_WireRemoval - new widget, with explicit and variable width wires. 3 and 5 point wires. New correction methods making use of clean plates and motion estimation technology and wire tracking.
18. F_Tracker - toggle to control widget display and improved offset tracking.

Bug Fixes

1. Known Bugs and Workarounds 2, 3 and 4 in Release Notes for Furnace 3.0v1 are no longer valid.

Known Bugs and Workarounds

1. Nodes containing roto parameters cannot handle multiple rotoshapes. As a workaround, use multiple nodes (1 per rotoshape required).
2. The roto timeline on some plug-ins is not being refreshed automatically. It can be forced by moving the control context to another node, then back again.
3. On Mac OS X, installing Furnace more than once results in multiple shortcuts to /usr/local/foundry/FLEXlm being created as /Applications/TheFoundry/FLEXlm, /Applications/TheFoundry/FLEXlm/FLEXlm and so on.

Furnace 3.0v4

This is a maintenance release of Furnace for Shake to fix an optimization bug on Linux.

Release Date

8 September 2006.

Requirements

1. Shake 3.50/4.00 on Mac OS X and Linux (RH9).
2. Foundry FLEXlm Tools (FFT) (4.0v1 or later) for floating license support.

Bug Fixes

1. F_WireRemoval - crash if reconstructMethod set to gradient on a floating point image. This has been fixed.

Furnace 3.0v3

This is a maintenance release of Furnace for Shake to support Shake 4.10 and Intel Macs.

Release Date

1 June 2006.

Requirements

1. Shake 3.50/4.00/4.10 on Mac OS X (Intel and PPC) and Linux (RH9).
2. Foundry FLEXlm Tools (FFT) (4.0v1 or later) for floating license support.

Bug Fixes

1. F_Kronos - BUG ID 694 - crash if blockSize set to 2 or lower. This has been fixed.
2. F_DeNoise - BUG ID 704 - unable to copy and paste nodes with names preceding F_DeNoise in the alphabet when an F_DeNoise node is present in the workspace. This has been fixed.
3. F_DeBlur - BUG ID 706 - crash if image to deblur is a plain texture (i.e. a single constant colour). This has been fixed.
4. F_DirtRemoval - BUG ID 707 - incorrect dirt detection over plain areas (i.e. a single constant colour). This has been fixed.
5. F_MatchGrade - BUG ID 709 - does not copy the source alpha channel to the destination. This has been fixed.
6. F_DeNoise - BUG ID 710 - under some circumstances the analysis region is not saved correctly. This has been fixed.
7. F_SmartPlate - BUG ID 713 - memory leak when frameCombiner set to median or limitMedian. This has been fixed.
8. F_Tracker - BUG ID 738 - message 'error: : (R2) No such file or directory' is printed to the console when an F_Tracker node is used. This has been fixed.
9. F_Tracker - BUG ID 740 - crash when loading a script where the track name has been changed. This has been fixed.

10. F_MatchGrade - BUG ID 750 - produced 'colour bleeding' incorrect results when copyFrom and copyTo images are a different size to the source image. This has been fixed.
11. F_Tile - BUG ID 752 - the Help button on Macs referenced an incorrect link. This has been fixed.

Furnace 3.0v2

This is a maintenance release of Furnace for Shake to fix an install issue on OSX.

Release Date

23 February 2006.

Requirements

1. Shake 3.50/4.00 on Mac OS X and Linux (RH9).
2. Foundry FLEXlm Tools (FFT) (4.0v1 or later) for floating license support.

Bug Fixes

1. An issue with installations of previous versions of Furnace on OSX installed in the default Shake location has been fixed. Old files (furnace.bundle, furnace.h) will be renamed with a .superceded in order to prevent clashes between the two versions if both are installed to /Applications/Shake.
2. Fixed a bug where some popup parameters didn't work correctly.
3. F_Degrain - temporal switch now works correctly.

Furnace 3.0v1

This is a major new release of Furnace for Shake including new plug-ins and improvements.

Release Date

16 January 2006.

Requirements

1. Shake 3.50/4.00 on Mac OS X and Linux (RH9).
2. Foundry FLEXlm Tools (FFT) (4.0v1 or later) for floating license support.

New Features

1. F_ColourAlign.
2. F_DeBlur.
3. F_DeNoise.
4. F_Depth.
5. F_MatchGrade.
6. F_MatteToRoto.
7. F_MotionBlur.
8. F_MotionMatte.
9. F_PlanarPatcher.
10. F_Splicer.
11. F_Tracker.

Improvements

1. FLEXlm - the plug-ins are licensed with FLEXlm.
2. Progress Bars - plug-ins now display a pop-up to indicate their progress (Shake 4.00 and later only). These can be disabled by setting the environment variable FOUNDRY_PROGRESS_BAR to 0.

Bug Fixes

1. Various floating license bugs affecting sites running very large numbers of licenses have been fixed with the introduction of FLEXlm licensing.
2. F_Kronos - BUG ID 179 - anomalous output from Kronos occasionally seen "error: (R2) No such file or directory". This has been fixed.
3. F_RigRemoval - BUG ID 353 - clips floating point images. This has been fixed.
4. F_ChannelRepair - BUG ID 338 - did not support floating point images correctly. This has been fixed.
5. BUG ID 472 - Plug-ins running on a multiprocessor Red Hat Linux system using a kernel of 2.4.20 or newer would crash unless the environment variable LD_ASSUME_KERNEL was set to 2.4.19. This has been fixed.
6. F_Kronos - BUG ID 602 - retiming some clips at proxy resolution can give (obviously) wrong results. Working at full resolution is a workaround. This has been fixed.

Known Bugs and Workarounds

1. F_EdgeMatte - BUG ID 77 - sometimes the cache is wrong. Deleting the cache from the command line will fix the problem.
2. F_WireRemoval - BUG ID 51 - temporal inpainting output is wrong.
3. F_WireRemoval and F_DeGrain. WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignoring the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.
4. User Guide. Appendix C needs updating.

Furnace 2.0v4

This is a maintenance release of Furnace for Shake to support Shake 4.00.

Release Date

July 2005

Requirements

1. Shake 4.00 on Mac OS X and Linux, or Shake 3.50 and 3.01 on Mac OS X, Linux and Irix, or Shake 2.51 on Windows.
2. Foundry License Mangager 3.1 or later. (Built with Licensing Library 3.0v7v4)

New Features

1. The Shake help button now launches a basic html help page.
2. The version number is now included in the parameter list of the plug-in.

Improvements

Improved timing of the memory allocation enabling more robust handling of large images.

Bug Fixes

There are no fixed bugs.

Known Bugs and Workarounds

1. F_EdgeMatte - BUG ID 77 - sometimes the cache is wrong. Deleting the cache from the command line will fix the problem.
2. F_WireRemoval - BUG ID 51 - temporal inpainting output is wrong.
3. F_RigRemoval - clips floating point images.
4. F_WireRemoval and F_DeGrain. F_WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignoring the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.
5. User Guide. Appendix C needs updating.
6. F_Kronos - BUG ID 179 - anomalous output from Kronos occasionally seen "error: (R2) No such file or directory".

Furnace 2.0v3

This is a maintenance release of Furnace for Shake to fix some bugs.

Release Date

March 2005

Requirements

1. Shake 3.01 or 3.50 on Mac OS X, Linux and Irix.
2. Foundry License Manager 3.1 or later. (Built with Licensing Library 3.0v7v3)

New Features

There are no new features.

Improvements

There are no improvements to existing features.

Bug Fixes

1. Multiple Ethernet cards on Mac OS X. Although multiple ethernet cards have been supported on OS X for a while, additional interfaces of a non ethernet type are being returned by OS X. These non ethernet interfaces have a null machine address and cause the plug-ins to crash. The

licensing has been updated in this release to safeguard against null machine addresses and machine addresses of insufficient length.

2. Licensing - BUG ID 108 - Plug-ins would crash if there was no license server running. This has been fixed.
3. F_ScratchRepair - BUG ID 344 - would crash if used on floating point images. This has been fixed.
4. F_ColourMatte - BUG ID 339 - did not support floating point images correctly. This has been fixed.
5. F_ReGrain - BUG ID 287 - the test plate background is scaled incorrectly for 16 bit and floating point images. This has been fixed.
6. F_WireRemoval - BUG ID 249 - produces incorrect output when followed by a crop node. This has been fixed.
7. F_SmartZoom - BUG ID 202 - floating point values were converted to 16 bit. These are now being handled correctly.
8. F_SmartPlate - BUG ID 201 - floating point values were converted to 16 bit. These are now being handled correctly.
9. F_SmartFill - BUG ID 200 - floating point values were converted to 16 bit. These are now being handled correctly.
10. F_SmartPlate - BUG ID 341 - when using any of the median modes the internal cache is not deleted when the node is deleted or when Shake exits. This has been fixed.
11. F_BlockTexture - BUG ID 206 - crashed under the following circumstances. Apply F_BlockTexture, turn temporal consistency off, go to the next frame, switch temporal consistency on and you'd get a crash. This has been fixed.

Known Bugs and Workarounds

1. F_EdgeMatte - BUG ID 77 - sometimes the cache is wrong. Deleting the cache from the command line will fix the problem.
2. F_WireRemoval - BUG ID 51 - temporal inpainting output is wrong.
3. F_RigRemoval - clips floating point images.
4. F_WireRemoval and F_DeGrain. F_WireRemoval with rolling repair and temporal degrain may render incorrectly if

there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignoring the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.

5. User Guide. Appendix C needs updating.
6. F_Kronos - BUG ID 179 - anomalous output from Kronos occasionally seen "error: (R2) No such file or directory".

Furnace 2.0v2

This is a maintenance release of Furnace for Shake to fix some bugs. Shake 2.51 on Windows is no longer supported.

Release Date

26 November 2004

Requirements

Shake 3.50 on Mac OS X, Linux (RH9). Shake 3.01 on Mac OS X, Linux (RH7, 8, 9). Foundry License Manager 3.1 or later (recommended).

Built with licensing library 3.0v7.

New Features

There are no new features.

Improvements

1. Notes on how to install Furnace on Mac OS X to a non-default directory (ie not within the Shake directory structure) are included in the Introduction.

Bug Fixes

1. F_DeGrain - BUG ID 175 - the 'tuneRed' control affected the blue and the green channels on the output. This has been fixed.
2. F_DeGrain - BUG ID 194 - the default DeGrain setup after an analyse was poor, and increasing the tuning to improve the result lead to clipping artifacts in the output. This has been fixed.
3. F_WireRemoval - BUG ID 196 - in float space, WireRemoval clipped the output for values below 0 or above 1. This has been fixed.

4. F_ScratchRepair - BUG ID 156 - at proxy resolution, the result would be slightly inconsistent with the result at full resolution. This has been fixed.

Known Bugs and Workarounds

1. F_EdgeMatte - BUG ID 77 - sometimes the cache is wrong. Deleting the cache from the command line will fix the problem.
2. F_WireRemoval - BUG ID 51 - temporal inpainting output is wrong.
3. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.
4. F_WireRemoval and F_DeGrain. WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignorning the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.
5. User Guide. Appendix C needs updating.
6. F_Kronos - BUG ID 179 - anomolous output from Kronos occasionally seen "error:: (R2) No such file or directory".

Furnace 2.0v1

This is a major new release of Furnace for Shake including new plug-ins and improvements.

Release Date

28 September 2004.

Requirements

Shake 3.50 on Mac OS X, Linux (RH9). Shake 3.01 on Mac OS X, Linux (RH7, 8, 9). Shake 2.51 on Windows. Foundry License Manager 3.0v1 or later (recommended).

Built with licensing library 3.0v7.

New Features

1. DirtRemoval.
2. ColourMatte.
3. EdgeMatte.
4. SmartZoom.

5. SmartPlate.
6. MotionRepair.
7. SmartFill.
8. ScratchRepair.
9. ChannelRepair.

Improvements

There are no improvements to existing features.

Bug Fixes

There are no bug fixes.

Known Bugs and Workarounds

1. F_EdgeMatte - BUG ID 77 - sometimes the cache is wrong. Deleting the cache from the command line will fix the problem.
2. F_WireRemoval - BUG ID 51 - temporal inpainting output is wrong.
3. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.
4. F_WireRemoval and F_DeGrain. WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignoring the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.

Furnace 1.1v3

This is a maintenence release of Furnace on Shake with many improvements to existing features and bug fixes.

Release Date

29 September 2004.

Requirements

Shake 3.50 on Mac OS X, Linux (RH9). Shake 3.01 on Mac OS X, Linux (RH7, 8, 9). Shake 2.51 on Windows. Foundry License Manager 3.0v1 or later (recommended)

Built with licensing library 3.0v7.

New Features

There are no new features.

Improvements

There are no improvements.

Bug Fixes

1. Licensing. If the plug-ins tried to get a license from a machine that wasn't running a foundry license server, the plug-in would crash. This has been fixed.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.
2. F_WireRemoval and F_DeGrain. WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignorning the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.

Furnace 1.1v2

This is a maintenence release of Furnace on Shake with many improvements to existing features and bug fixes.

Release Date

Not available as a general release.

Requirements

Shake 3.50 on Mac OS X, Linux (RH9). Shake 3.01 on Mac OS X, Linux (RH7, 8, 9). Shake 2.51 on Windows. Foundry License Manager 3.0v1 or later (recommended).

Built with licensing library 3.0v6

New Features

There are no new features.

Improvements

There are no improvements.

Bug Fixes

1. F_DeGrain - Bug ID 27 - the analysis box failed to give the correct results at proxy resolution. This has been fixed.
2. Licensing. Plug-ins would not release a floating license immediately after quitting Shake, but would wait until the server timeout. This has been fixed so that licenses are given back to the server on quitting Shake.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.
2. F_WireRemoval and F_DeGrain. WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignorning the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.

Furnace 1.1v1

This is a maintenence release of Furnace on Shake with many improvements to existing features and bug fixes.

Release Date

Not available as a general release.

Requirements

Shake 3.50 on Mac OS X, Linux (RH9). Shake 3.01 on Mac OS X, Linux (RH7, 8, 9). Shake 2.51 on Windows. Foundry License Manager 3.0v1 or later (recommended).

Built with licensing library 3.0v3.

New Features

There are no new features.

Improvements

1. F_WireRemoval. Curved wires can now be removed using Shake's standard roto controls in the F_WireRemoval node. Previously curved wires had to be removed in several straight line segments.

2. F_WireRemoval. Multiple wires can be removed in one node.
3. F_Kronos. Mattes have been added to improve edge dragging on moving foreground objects in retimed sequences.
4. F_Kronos. The ability to use motion vectors from one clip to warp another has been built into this node.
5. F_Kronos. This node now supports the correct blending of log images when applying motion blur.
6. F_Kronos. Sequences can now be retimed by specifying a speed rather than mapping output to input frames.
7. F_Steadiness. Now has low, medium and high filtering options.
8. F_Steadiness. The parameter "windowed" has been renamed to "useSampleRegion" for clarity.
9. F_DeGrain. The tuning of Level 4 grain is now accessible to the user.
10. F_DeGrain. Now uses smarter tiling to optimise memory usage on larger images.
11. F_RigRemoval. Areas that cannot be replaced were highlighted in opaque red. An alpha slider (failMarkerAlpha) has been added to make this transparent.
12. Licensing. There have been a number of changes to the licensing code to improve reliability when checking out licenses from the server for sites with large numbers of floating licenses.
13. Licensing. Furnace nodes that are unable to get a license are now shown in red.

Bug Fixes

1. F_DeGrain - Bug ID 333 - various memory allocation errors have been fixed.
2. F_DeFlicker - Bug ID 339 - load a clip, set the proxy to P2, add a F_DeFlicker node and set blockSize to < 8 and Shake would crash. This has been fixed.
3. F_DeGrain - Bug ID 341 - level 4 was not configurable. This has been fixed.
4. F_BlockTexture - Bug ID 352 - mattes were being ignored if temporalConsistency was on. This has been fixed.

5. F_Steadiness - Bug ID 358 - after analysing long clips (200+ frames) the analyse button remains on despite attempts to switch it off. This has been fixed. The previous workaround was to right click in the node view window.

6. F_Steadiness - Bug ID 363 - an animated selection box was not used correctly when calculating image registration. This has been fixed.

7. Licensing - Bug ID 31 - ignored nodes do not generate license requests.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.

2. F_WireRemoval and F_DeGrain. WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignorning the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.

Furnace 1.0v8

This is a maintenence release of Furnace on Shake with some bug fixes.

Release Date

29 September 2004.

Requirements

Shake 3.50 on Mac OS X, Linux (RH9). Shake 3.01 on Mac OS X, Linux (RH7, 8, 9). Shake 2.51 on Windows. Foundry License Manager 3.0v1 or later (recommended).

Built with licensing library 3.0v7.

New Features

1. Licensing - if no license is found no processing will take place and a dialog will popup (excluding Shake 2.51 on Windows) to warn the user. This information is also written to the shell and the license.log file.

Improvements

1. Installation - on Mac OS X the plug-ins and icons are installed directly into the Shake application folder so that the environment variables NR_INCLUDE_PATH and NR_ICON_PATH no longer need to be defined to see Furnace. Previously, they were installed to /Applications/TheFoundry.
2. Documentation - F_Steadiness chapter re-written with more examples and more pictures.

Bug Fixes

1. Licensing. Plug-ins would not release a floating license immediately after quitting Shake, but would wait until the server timeout. This has been fixed so that licenses are given back to the server on quitting Shake.
2. Licensing. There have been a number of changes to the licensing code to improve reliability when checking out licenses from the server for sites with large numbers of floating licenses.
3. Licensing - Bug ID 31 - ignored nodes do not generate license requests.
4. Licensing. Furnace nodes that are unable to get a license are now shown in red.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.
2. F_WireRemoval and F_DeGrain. WireRemoval with rolling repair and temporal degrain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignorning the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.

Furnace 1.0v7

This is a maintenence release of Furnace on Shake to support Red Hat Linux 9 on Shake 3.01 and Furnace for Shake 3.50 on Irix, Linux and OS X.

Requirements

Shake 3.50 on Mac OS X, Linux (RH9) or Irix. Shake 3.01 on Linux (RH9). Foundry License Manager.

New Features

There are no new features.

Improvements

There are no improvements to existing features.

Bug Fixes

1. Red Hat 9. Furnace has been recompiled to work with Shake 3.01 on Red Hat Linux 9.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.
2. DOD. WireRemoval with rolling repair and temporal de-grain may render incorrectly if there is a node (DOD) up or downstream that affects the size of the input to the Furnace plug-in. Ignorning the DOD or switching off the temporal part of the plug-in would be a temporary workaround until this bug is fixed.

Furnace 1.0v6

This is a maintenence release of Furnace on Shake to fix a a persistent licensing bug and support Shake 3.01.

Requirements

Shake 3.01 on Mac OS X, Linux or Irix. Shake 3.00 on Mac OS X or Linux. Foundry License Manager.

New Features

There are no new features.

Improvements

There are no improvements to existing features.

Bug Fixes

1. Floating Licenses - Bug ID 343 - On Mac OS X and Linux only, extended renders or interactive sessions where the license reverts back to the floating license server would eventually cause these plug-ins to become unlicensed. The previous release (Furnace 1.0v5) fixed one cause of this bug, but since its release another manifestation of the same bug (343) has been found and fixed.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.

Furnace 1.0v5

This is a maintenence release of Furnace on Shake to fix a a licensing bug on Mac OS X.

Requirements

Shake 2.50, Shake 2.51 and Shake 3.00 on Mac OS X. Foundry License Manager 2.0v1.

New Features

There are no new features.

Improvements

There are no improvements to existing features.

Bug Fixes

1. Floating Licenses - Bug ID 343 - On Mac OS X only, extended renders or interactive sessions where the license reverts back to the floating license server would eventually cause these plug-ins to become unlicensed. This has been fixed.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.

Furnace 1.0v4

This is a maintenence release of Furnace on Shake to fix a number of bugs and support Shake 3.

Requirements

Shake 2.50 or Shake 2.51 on Irix, Linux, NT or Mac OS X. Shake 3.00 on Mac OS X, Irix or Linux. Foundry License Manager 2.0v1.

New Features

There are no new features.

Improvements

1. Licensing - Furnace license timeout warnings have been added to all plug-ins to help clients spot pending license loss. Warnings are given for all licenses with less than 30 days to run. Licenses with 5 days to run are given an enhanced warning, and a license which expires "tomorrow" is reported as such. Reported as a bug, Bug ID 331.
2. Floating Point Support - Kronos now supports floating point input clips.

Bug Fixes

1. Kronos - Bug ID 340 - The output bit depth of Kronos didn't always match the input bit depth. This has been fixed.
2. Kronos - Bug ID 336, 360 - Kronos clipped the output when retiming floating-point images. This has been fixed.
3. Kronos - Bug ID 334 - Kronos could crash when using floating point iff files if the DOD on the image files was less than the whole image. This has been fixed.
4. Kronos - Bug ID 335 - Essentially the same as Bug ID 334 - DODs on the input sequence were handled incorrectly. This has been fixed.
5. ReGrain - Bug ID 321 - When adding grain to a floating point image, if the input image had values in it smaller than 0, and 'useFilmicResponse' was checked on, then overflow pixel values could be seen on the output. This has been fixed. Additionally, the application of the response curve itself was often incorrect. These issues have been fixed.

Known Bugs and Workarounds

1. Floating Point Support - RigRemoval does not support floating point file formats. All other plug-ins work at floating point.

Other Changes

1. Kronos is a macro. Vectors are passed from node to node in the macro encoded as 16bit images (See "Coding vectors between nodes" on page 252.). For floating point images however (such as floating point iff inputs) the vectors are passed from node to node as pure floating point images, unencoded. The snippet of code in

Appendix C which is used for decoding the 16bit images is not necessary when looking at the vectors generated by floating-point images.

Furnace 1.0v3

This is a maintenence release of Furnace on Shake to fix a floating license bug and support Shake 3.

Requirements

Shake 2.50 or Shake 2.51 on Irix, Linux, NT or Mac OS X. Shake 3.00 on Mac OS X, Irix or Linux. Foundry License Manager 2.0v1.

New Features

There are no new features.

Improvements

There are no improvements to existing features.

Bug Fixes

1. Floating Licenses. A potential floating license deadlock existed on Linux and OSX distributions of Shake which could cause intermittent licensing failures or freezes of the Shake interface. This has been fixed.
2. RigRemoval - this node would crash if the lookAhead-Frames parameter was set to a value larger than 40. This has been fixed.
3. Kronos - when motion-blurring, the Kronos macro incorrectly blended the alphas of the motion-blurred temporal samples, leading to an over-exposed alpha channel in the output. This has been fixed.

Known Bugs and Workarounds

1. ReGrain - Bug ID 321 - negative values in the source image to ReGrain can cause bright pixelation artifacts in the ReGrain output, when useFilmicResponse is checked on. As a workaround, turn off the useFilmicResponse boolean, and blend between the ungrained and ReGrained footage, using a luminance-related mask which peaks around the midtones of the image. You should be able to get a reasonable mask by passing a monochromed version of the ungrained footage through a bell-shaped lookup, where the bellshape peaks at 1.0 around your midtones, and fades off to zero on either side.

2. Floating Point Support - Kronos and RigRemoval do not support floating point file formats. All other plug-ins work at floating point.

Furnace 1.0v2

This is a maintenence release of Furnace on Shake to fix a few bugs.

Requirements

Shake 2.50 or Shake 2.51 on Irix, Linux, NT or Mac OS X. Foundry License Manager 2.0v1.

New Features

1. Furnace is now available on SGI IRIX

Improvements

There are no improvements to existing features.

Bug Fixes

1. WireRemoval tracking on Linux would crash if the tracking were allowed to cycle back to frame 1 before the 'tracking' parameter was toggled off. This has been fixed.
2. RigRemoval might occasionally generate corrupt output when the 'blendCorrect' parameter was set to a non-zero value. This has been fixed.
3. DeGrain - the matte shown by 'showMotionMatte' when doing temporal degraining had horizontal and vertical banding artifacts on the OSX and NT builds, and hence less-than-optimal temporal degraining. This has been fixed.
4. BlockTexture - there was a possibility of an infinite loop occuring when using a matte input to limit the region of the output filled by the generated texture. This has been fixed.
5. Steadiness was not using the cache correctly when used with a 'referenceMethod' of 'staticFrame', causing unnecessary reprocessing in the GUI. This has been fixed.

Furnace 1.0v1

This is the first release of Furnace on Shake.

Requirements

Shake 2.50 or Shake 2.51 on Linux, NT or Mac OS X. Foundry License Manager 2.0v1.

New Features

Not applicable.

Improvements

Not applicable.

Bug Fixes

There are no known bugs.

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. 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. UPGRADES/ENHANCEMENTS.

As long as Licensee pays the Annual Support Fee (if applicable) in connection with the Annual Upgrade and Support Program (offered under a separate agreement from The Foundry and its authorized resellers), The Foundry or its authorized reseller will provide Licensee with access to upgrades and updates, if any, made available by The Foundry. All upgrades and updates provided to Licensee by The Foundry shall be covered by this Agreement, unless such upgrade or update contains a separate license.

SECTION 7. 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 8. 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, NONINFRINGEMENT, 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 warranties 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 9. 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 10. 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 11. 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 12. 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 13. 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 14. 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 15. 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 16. 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 Â§52.227-14 ALTS I, II and III (June 1987); (ii) FAR Â§52.227-19 (June 1987); (iii) FAR Â§12.211 and 12.212; and/or (iv) DFARS Â§227.7202-1(a) and DFARS Â§227.7202-3. The Software is the subject of the following notices: * Copyright (c) 2007 The Foundry Visionmongers, Ltd.. All Rights Reserved. * Unpublished-rights reserved under the Copyright Laws of the United Kingdom.

SECTION 17. SURVIVAL.

Sections 2, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18 and 19 shall survive any termination or expiration of this Agreement.

SECTION 18. 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 19.
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.

Index