Flamer

We have studied the use of repos with a fixed control matrix. Here is a way to generate a dynamic control matrix that shapes an image. First some static art:

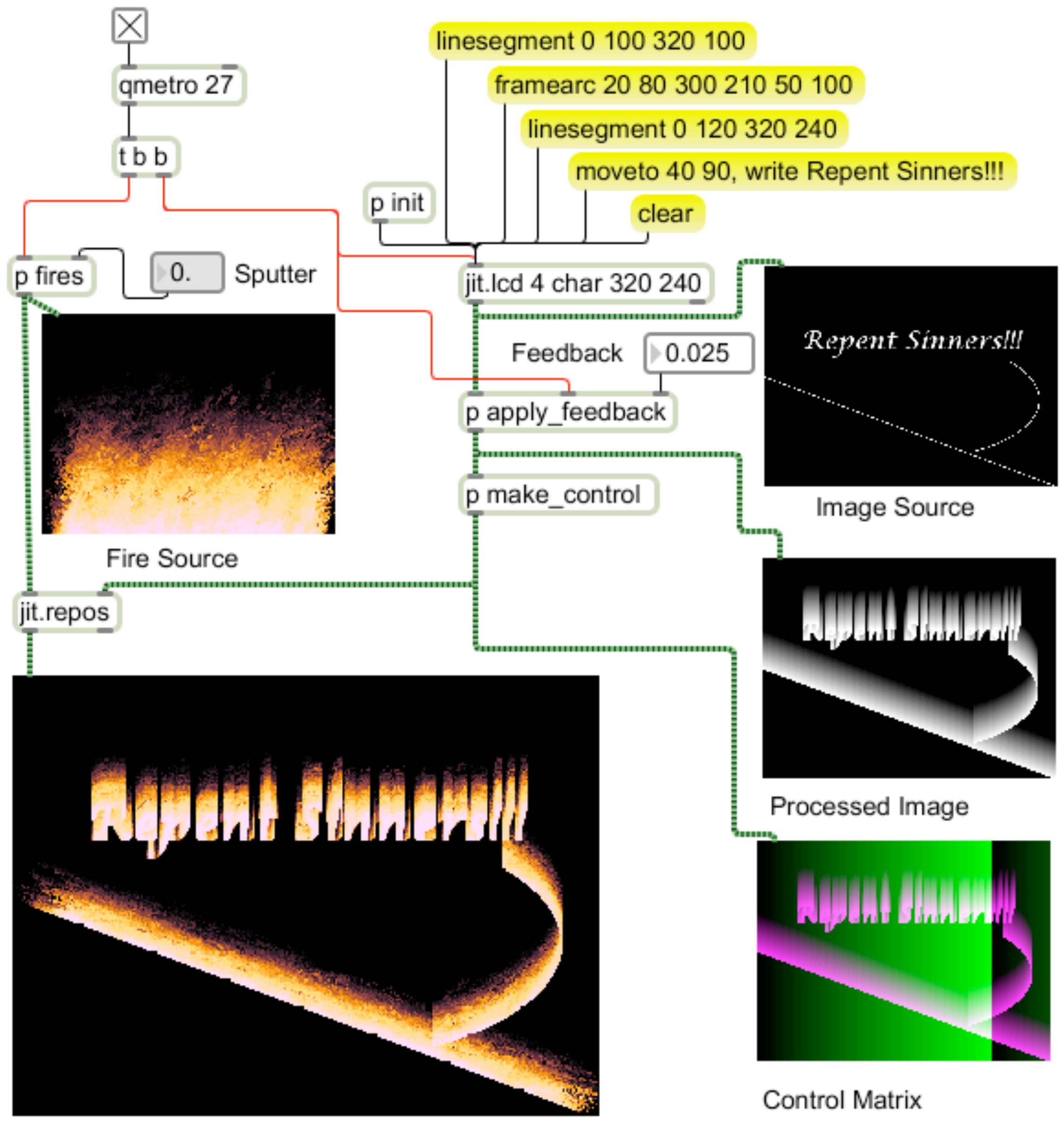


Figure 1.

There are two images in this patch—fire is the source image that will be run through jit.repos. The control image begins with a jit.lcd (with a variety of drawing commands) and is processed with feedback before being converted into a control matrix.

The fire is generated with jit.qt.effects number 15. The subpatch that does this is shown in figure 2.

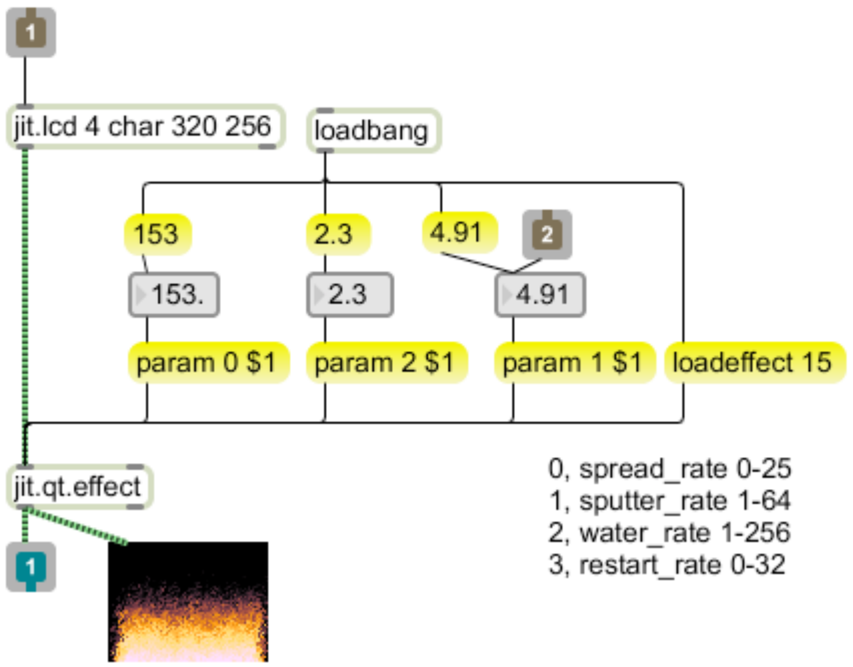


Figure 2. Fires subpatch.

Effect 15 is not very well documented on the Apple website. It seems to be an Easter egg--one of those things Apple engineers do for their own entertainment and then hide deep in an application regardless of any practical application. It is in fact a rather elegant particle system that mimics the thermodynamics of burning gas. The still picture does not do it justice—it appears hot enough to burn the screen.

QuickTime effects are buried within the jit.qt.effects object, so access to parameters is oblique. The only message we can send to jit.qt.effects is param with an argument for the parameter number and more arguments for the parameter value(s). Discovering the meaning of the parameters takes some detective work, but I have most of them listed in the Effect Spotter tutorial[1].

The param settings shown in figure 2 give a steady, even flame. Adjusting param 1 (sputter rate) changes the height of the flames—a value of 1 fills the window and a value of 64 douses the flames. Even though it isn't strictly an effect, number 15 requires an input matrix that it entirely replaces. Notice the size of this matrix—the height of 256 is crucial.

The jit.lcd is the origin of the control matrix. This part of the patch (figure 3) should hold no surprises. The init subpatch simply sets the background color to black and the drawing color to white.

[1] Apple keeps adding effects—it's hard to keep up.

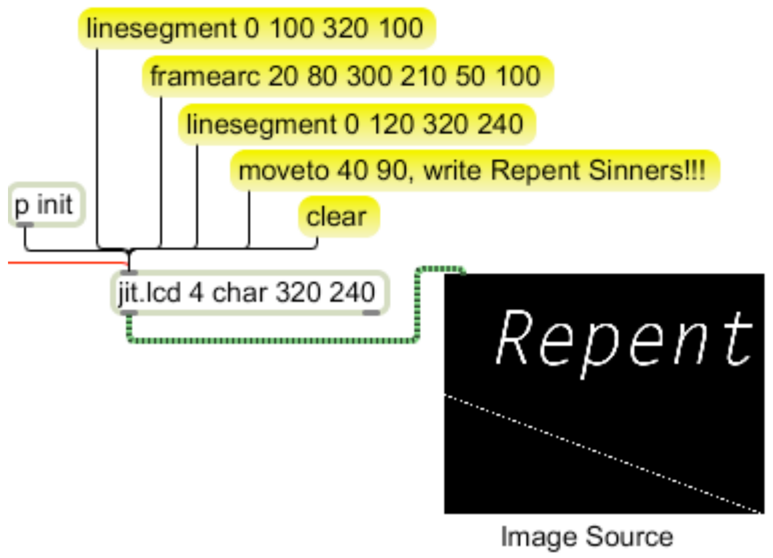
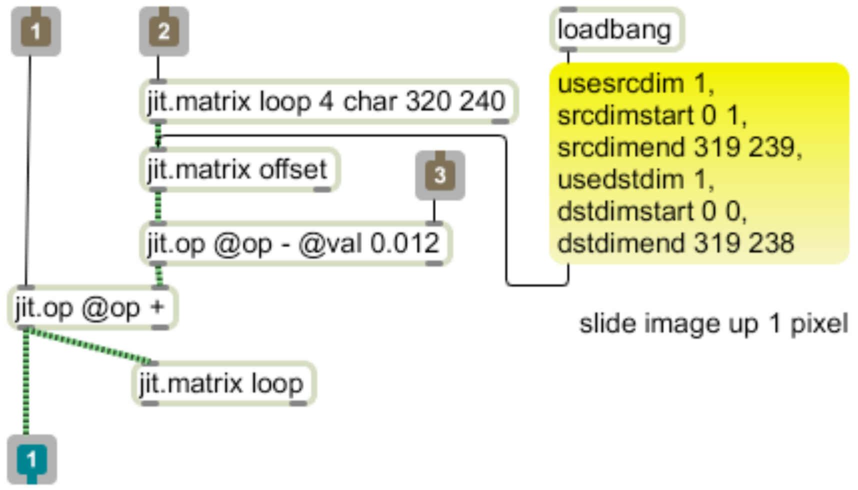

linesegment 0 100 320 100
framearc 20 80 300 210 50 100
linesegment 0 120 320 240
moveto 40 90, write Repent Sinners!!!
clear
p init
jit.lcd 4 char 320 240
Repent
Image Source

Figure 3. Jit.lcd is the source of control.

The lcd output is processed by a feedback loop as shown in figure 4. Note the characteristic paired matrices of the same name, in this case “loop”. The frame in the top loop matrix is banged down through a chain of processes before arriving at the lower jit.op where it is added to the new frame from inlet 1. The result of the add operation is output and stored in the “loop” matrix for the next frame.



```
usesrcdim 1,
srcdimstart 0 1,
srcdimend 319 239,
usedstdim 1,
dstdimstart 0 0,
dstdimend 319 238
```

Figure 4. The apply_feedback subpatch.

The feedback image is modified by the jit.matrix named “offset”. The combination of source dimensions and destination dimensions listed have the effect of moving the image one pixel up. This is made slightly dimmer by the jit.op with a subtract operation. The result is a gradient above the lines in the control image. This is adjustable by the subtract from feedback value. Figure 5 shows various settings. It is important to remember how the color of a pixel relates to the numeric value. A white pixel has a value of 255 and a black pixel is 0. The effect of the feedback on this stark black and white image is to produce a distribution of values progressively reduced by the val in the jit.op. With a setting of 0.012, the colors are 255 (the original line), 252, 249 and so on down to 0.

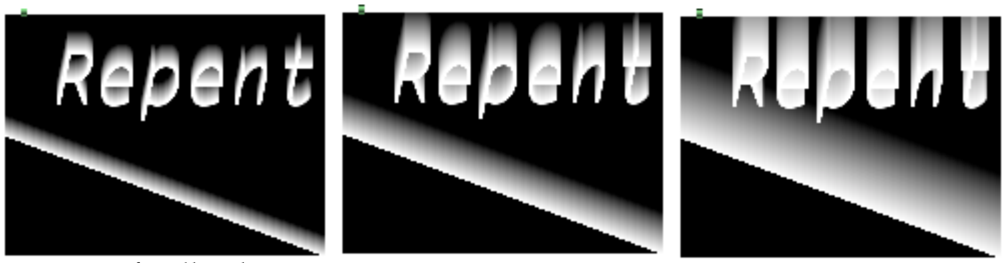
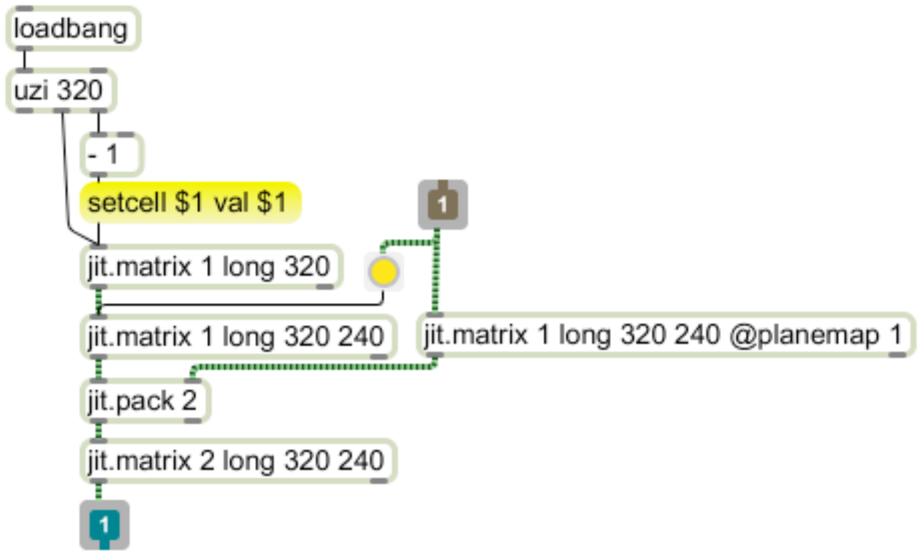


Figure 5. feedback setting 0.05, 0.025, 0.0125

The next step is to convert this into a control matrix for jit.repos. Figure 6 shows the mechanism. This stuffs the horizontal coordinates in the first plane of a matrix of type 2 long. The coordinates are generated by brute force with an uzi and setcell messages. There are many elegant jitter objects to do complex math such as jit.expr and jit.gencoord but they produce output as float matrices, and jitter will clip values to 255 when converting from float to long.



```
loadbang
uzi 320
- 1
setcell $1 val $1
jit.matrix 1 long 320
jit.matrix 1 long 320 240
jit.matrix 1 long 320 240 @planemap 1
jit.pack 2
jit.matrix 2 long 320 240
```

Figure 6.
The vertical coordinates in plane 1 are taken from the feedback image. Since the vertical dimension of the fire image is 256, there are no conversion issues. This matrix does the trick nicely, creating fire along the lines drawn in the jit.lcd. This is particularly nice with animated images that evolve slowly.

This effect can be applied to any image that is made up of white lines on a black background. Figure 7 shows a version that works with movies. It has also been expanded to a higher resolution. That's not difficult—every place that 320 appeared before now contains a 640 and every 240 is replaced with a 480. The offset numbers in the feedback loop also need to be changed. The height of the fire effect remains 256 however.

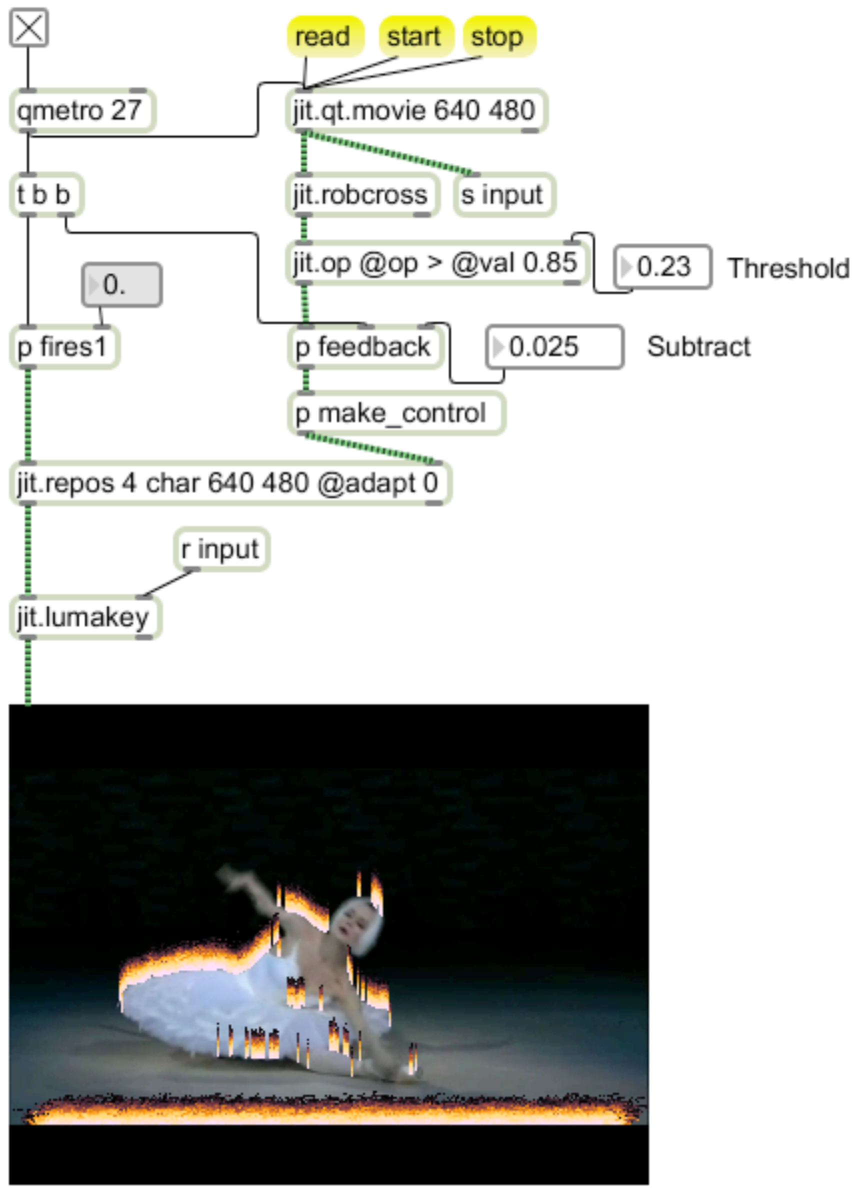


Figure 7.

The tricky part of this is extracting some lines for the fire to follow without turning the central character into a ball of flame. This begins with the jit.robcross object.

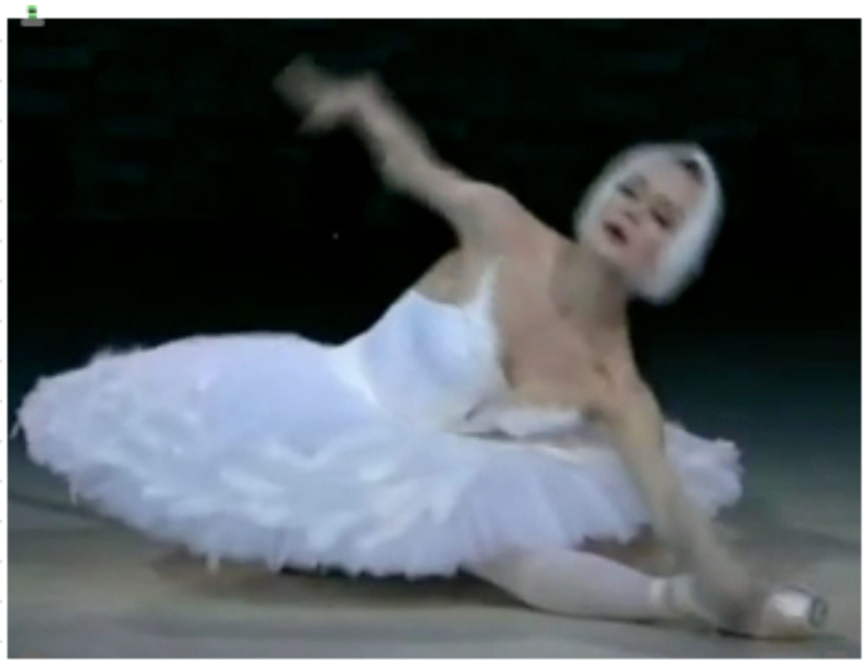


Figure 8
Figure 8 shows a full sized section of a movie frame before processing. This eventually be lumakeyed into the shaped flames.

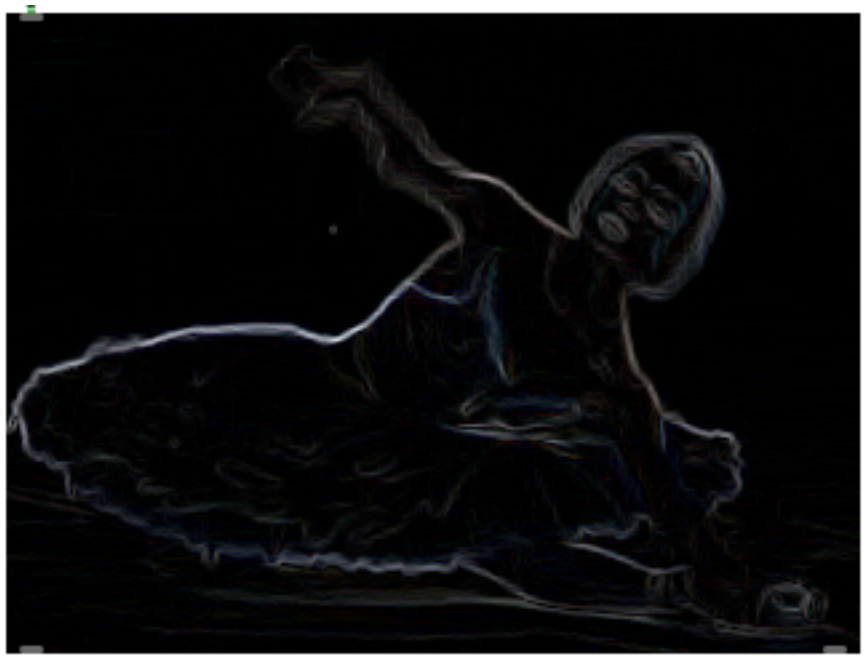


Figure 9.
Figure 9 shows the output of the jit.robcross object. This is an edge detector. It considers the difference between adjoining pixels and sets pixels with a large difference to white. Jitter has several edge detectors, and this is the least successful. Jit.sobel produces a complete outline of this figure, which is best for many applications, but which would detect too many edges for this application. In fact, even robcross gives more detail than we need. (We don't want flames

from the dancer's mouth and eyes.) To remove some detail, we use a jit.op with > as the op.

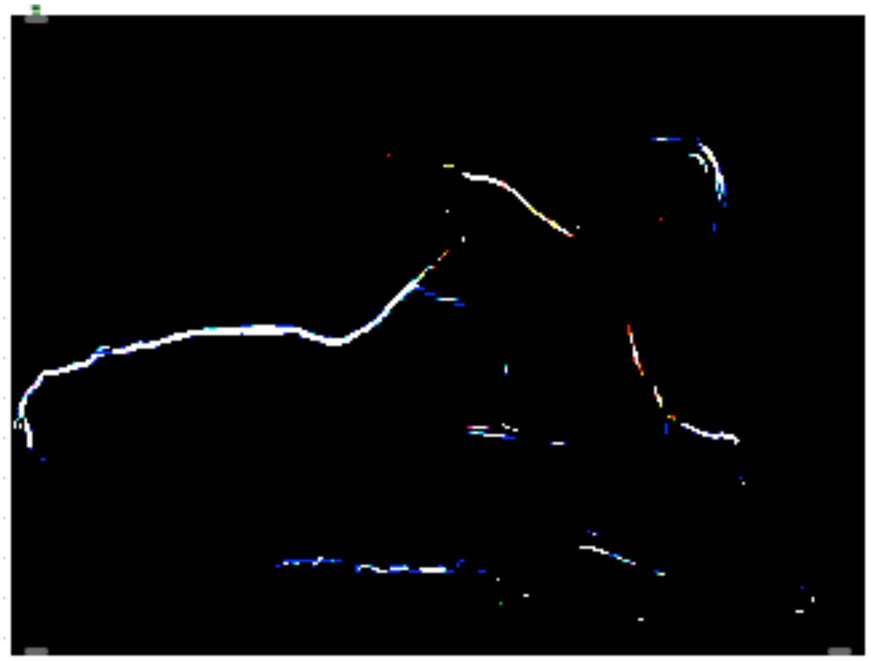


Figure 10.
The jit.op > sets pixels above a threshold to 1. This removes the faintest details and brightens what remains. The process works plane by plane, so some pixels are blue or red. This image is suitable for the feedback smear shown in figure 4.

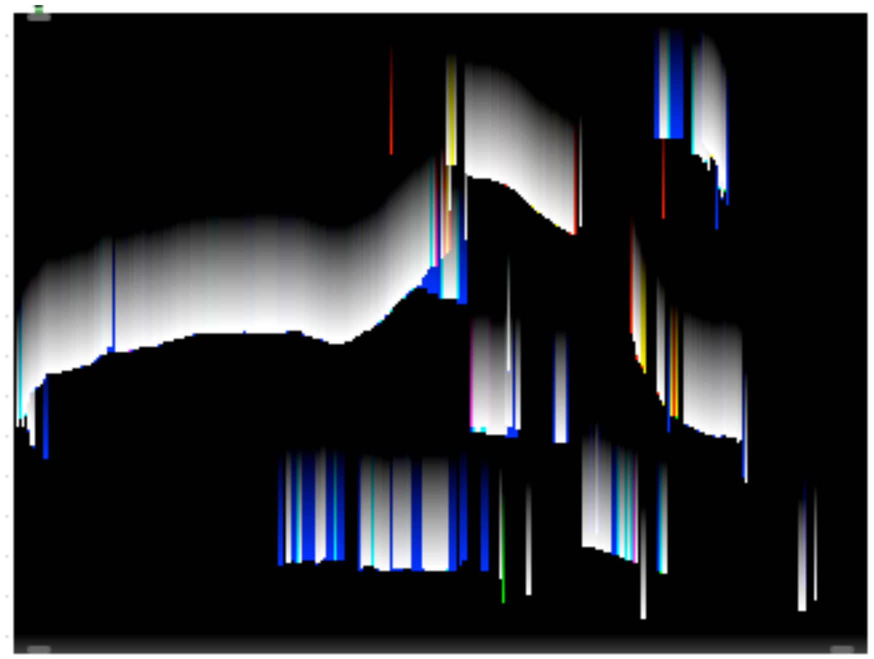


Figure 11.
Figure 11 shows the control image after feedback. The process of converting this to a 2 plane matrix will discard the blue and red lines.

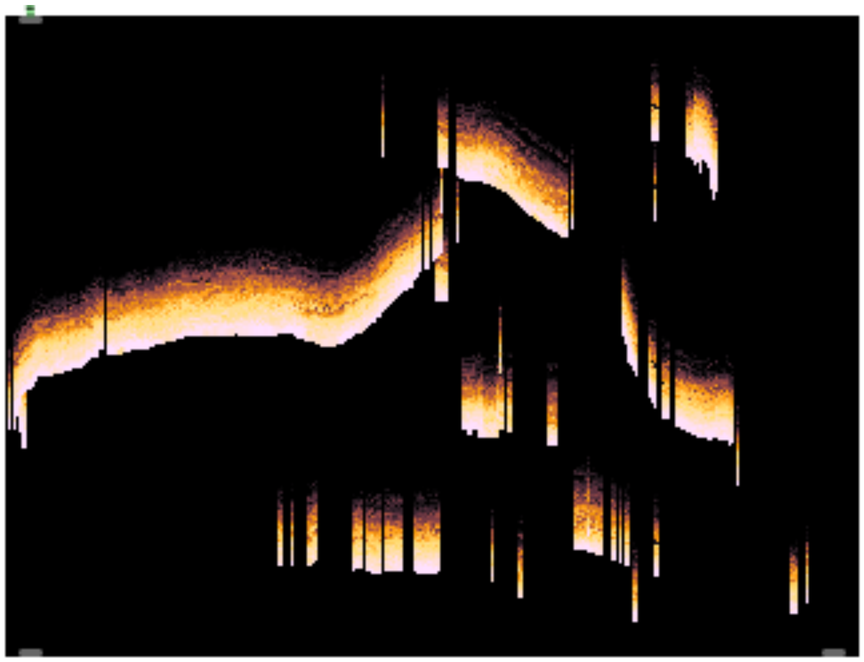


Figure 12.
Figure 12 is the shaped flame image. It takes some careful tweaking with the feedback control and the jit.op threshold to get an attractive flame. Figure 13 shows the final version created by a lumakey of the flame image and the original movie. (Since most of the flame image is black, lumakey is very effective.)

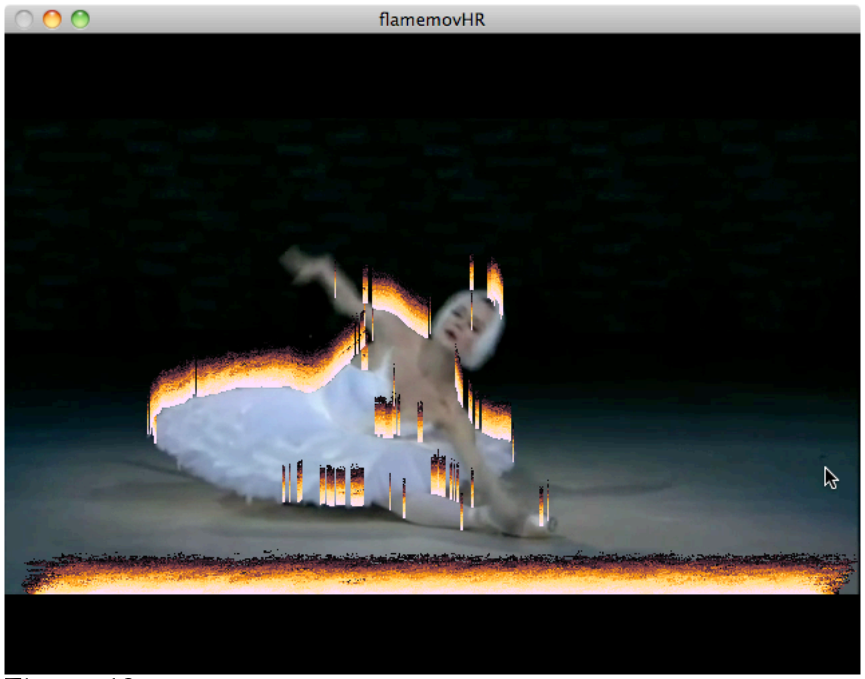


Figure 13.
Figure 14 is the final image. This does not show the dynamics of a moving image. Since the feedback smear only moves pixels up, any sideways motion leaves horizontal streaks. Also, since this is a movie, the changes from frame to frame may be very abrupt. This leaves afterimages of legs and arms when the dancer

executes quick moves or when there is a jump cut. Figure 14 shows some of the results.

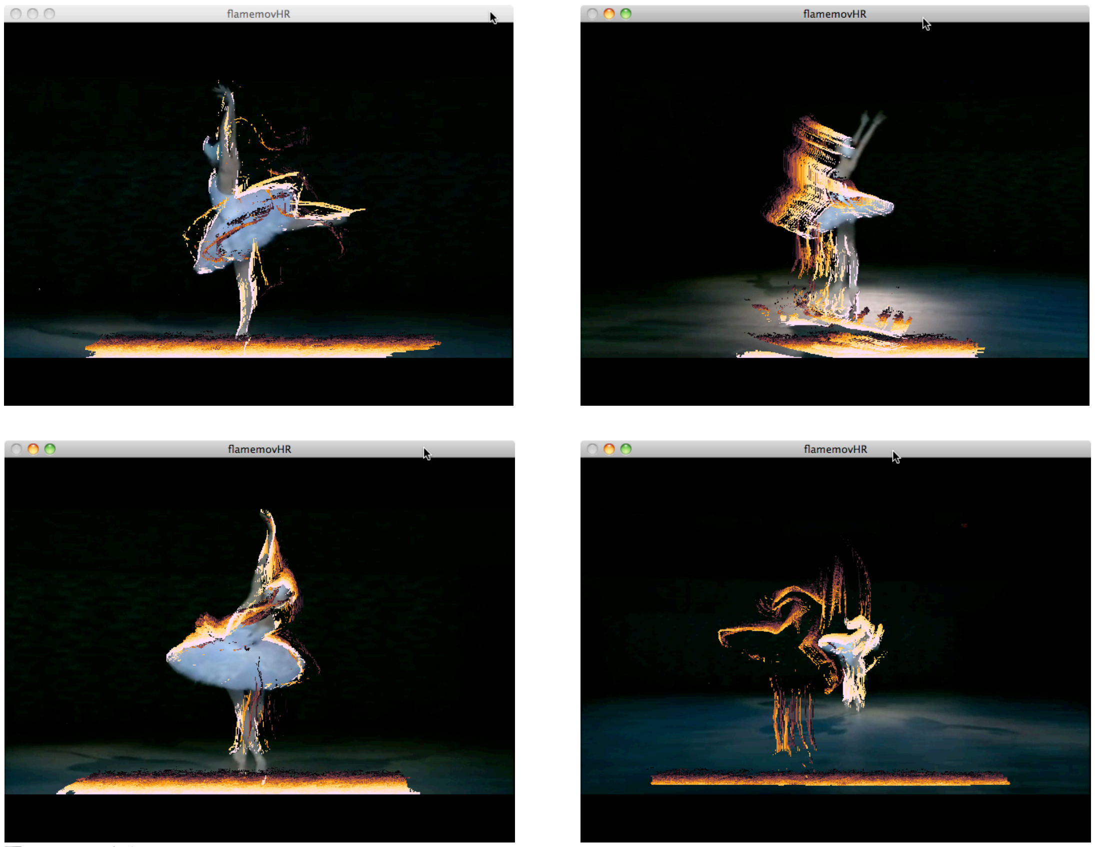


Figure 14.