Zen Mirror: A case study in video processing

The concept of the Zen Mirror patch is simple:

The user approaches the computer monitor and sees a reflection of the room but not himself. If he holds very still, his reflection will fade in in front of the background. Any motion makes the image disappear again.

This works best with a display that contains the camera. The installation has two constraints:

- The screen and camera must be set up so only one person at a time is in camera range.
- The background must be fixed, with no moving items.
- Lighting must be constant.

The main patch is simple. The camera image is obtained with jit.qt.grab. This image is routed three ways:

- Through a gate that can be briefly opened (via space bar) to transfer the image to a jit.matrix named background.
- To a subpatch that delays the image by three frames.
- To a jit.xfade object that can select the camera image or background.

A motion detection subpatch controls the jit.xfade object.

The result is flipped horizontally (to mirror the image) and displayed full screen in a jit.window. Figure 1 shows the four images involved.

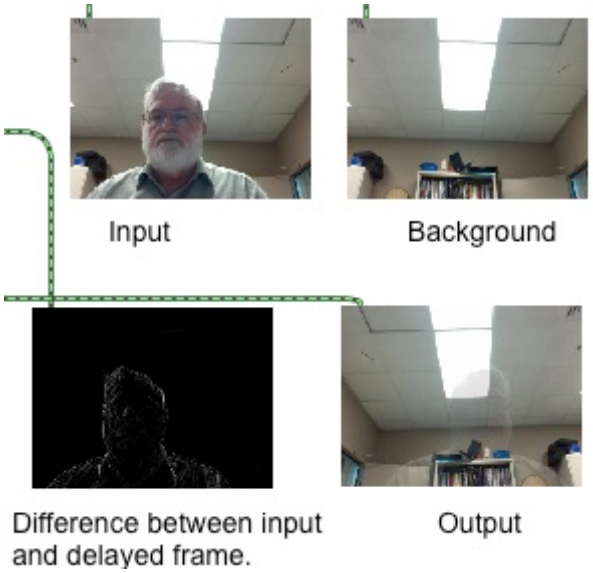

Figure 1.

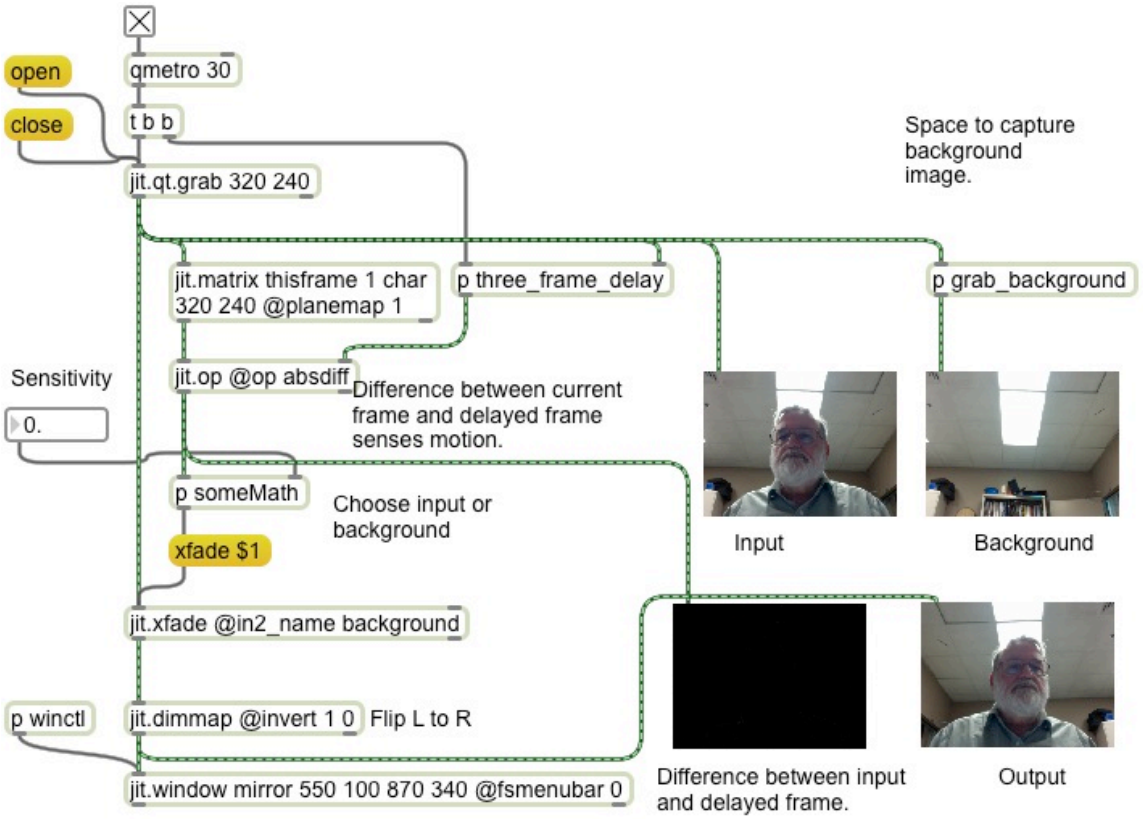

Figure 2.

Figure 2 shows the main patch. The objects at the upper left bring in the image frames. The contents of the grab_background subpatch are shown in figure 3.

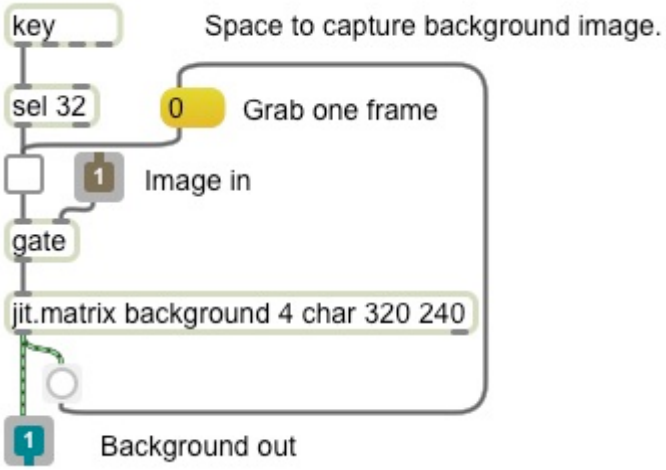

Figure 3.

Figure 3 is an example of what I call a "mousetrap" patch. A gate object is controlled by a toggle. (In this case the toggle is set by the spacebar.) When the gate is open, a frame from inlet 1 will be loaded into the jit.matrix named background. When the

matrix emerges from background, a 0 is sent to the toggle to close the gate. This lets exactly one frame into background. Triggering with the space bar makes it easy to lock the background without appearing in the image. The outlet in figure 3 merely connects to a monitor window for the background. This image will be available elsewhere by name.

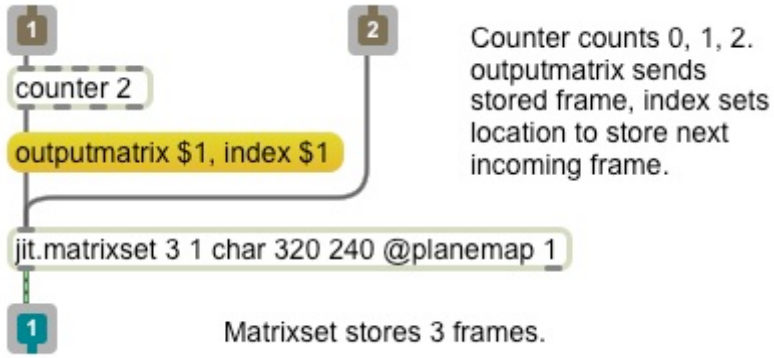

Figure 4.

Figure 4 shows how to delay frames. This is based on the jit.matrixset object, which can store multiple matrices. The counter cycles through 0, 1 ,2, 0 ... Each cycle produces two messages: *outputmatrix n* commands the matrixset to send the matrix located at *n*, and *index n* will direct the next frame to arrive to be stored at that same location. Thus a frame that came in on step 0 will be held for three cycles before it is sent out. The trigger object on the qmetro in the main patch ensures that the counter will step before each new frame arrives. Note that jit.matrixset has the planemap attribute set to 1. This means we are capturing only the red layer.

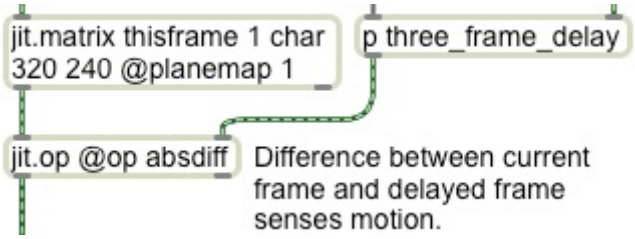

Figure 5.

The delayed frame is compared to the current frame with jit.op absdiff. The red layer is extracted from the current image for this. The frames will only differ if the image is moving. (See figure 6.)

Figure 6.

This image is now processed in the someMath subpatch. The point of this subpatch is to generate a control value for jit.xfade, so it has to reduce the image in figure 6 to a number between 0 and 1. An xfade value of 1 will show the background, 0 will show the current image.

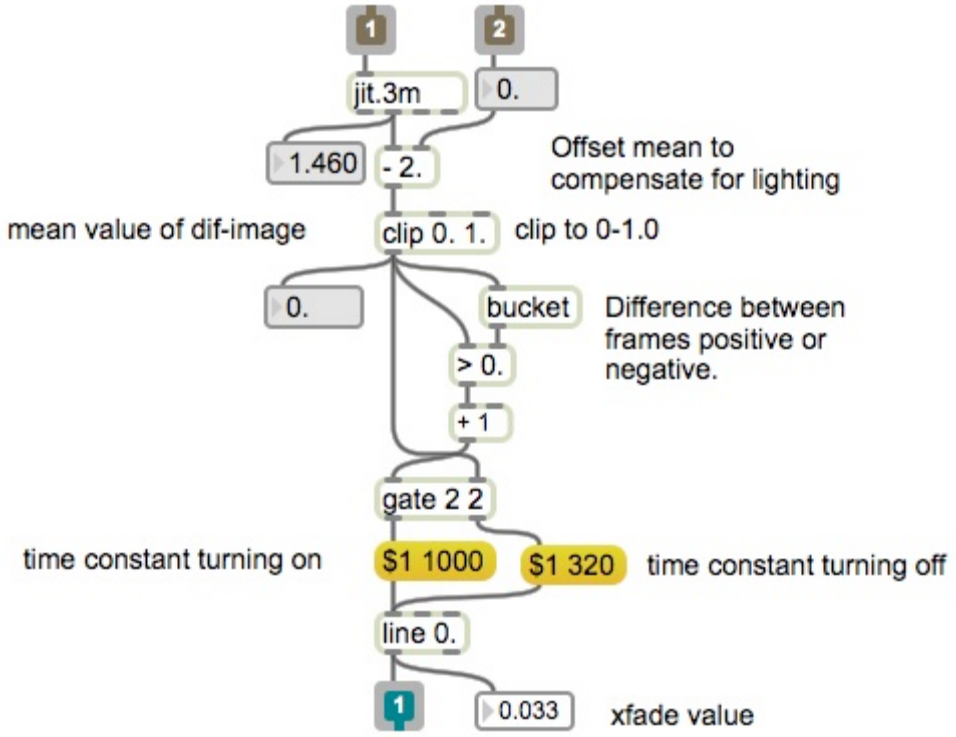

Figure 7.

The math in figure 7 should be fairly clear. The jit.3m object reports the mean value of a matrix in the range 0-255. This value is going to vary with lighting, so something is subtracted to ensure the control will go below 0. (There will always be one or two pixels lit, so the mean will always be positive.)

The control value is clipped to remain between 0 and 1. This could control the fade directly, but the transitions would be too sudden. A line object will slow it down, but I wanted the face to fade faster than it appears. The bucket object and > will detect if the control is increasing or decreasing. Bucket shifts values out when new ones arrive, so the output is one step behind the input. The > object compares the current

control with the previous value, so if the control is increasing, the > object will output 1. Another 1 is added to this to select which message to send the control to. These have different time constants for line. Line then produces the value that controls jit.xfade.

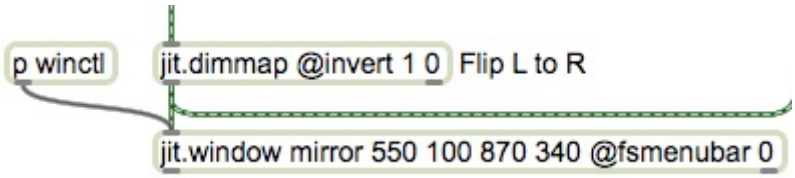

Figure 8.

All that remains is to mirror the image and display it. Jit.dimmap will flip the image in the horizontal direction with the arguments shown in figure 8. Note that the jit.window object has the fsmenubar attribute set to 0. This will hide the menus and dock when the window is expanded. I routinely connect the fullscreen command to the escape key as shown in figure 9.

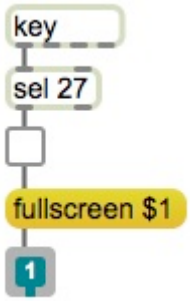

Figure 9.

Figure 9 is encapsulated in the winctl subpatch because I copy it into almost all of my patches.

Zen Mirror is one of my most popular installation patches. The only issue that has ever come up is the need to update the background to accommodate gradual changes in lighting. It is not hard to automatically grab the background from time to time when no activity is detected for several minutes.

You are free to install it somewhere as long as you credit me.