Simple Motion Detection with Jitter

One of the most common applications of Max /Jitter in art is triggering installation activity based on the presence or motion of a visitor. There are many ways of doing this with security hardware (switches, IR sensors and the like) but the most flexible and easiest to set up are based on a video camera.

Hot Spots

You can easily watch a single pixel of the camera image with the getcell message. Trouble is, people wander around unpredictably and may not hit that particular pixel. A better strategy is to divide the area into zones or hot spots.

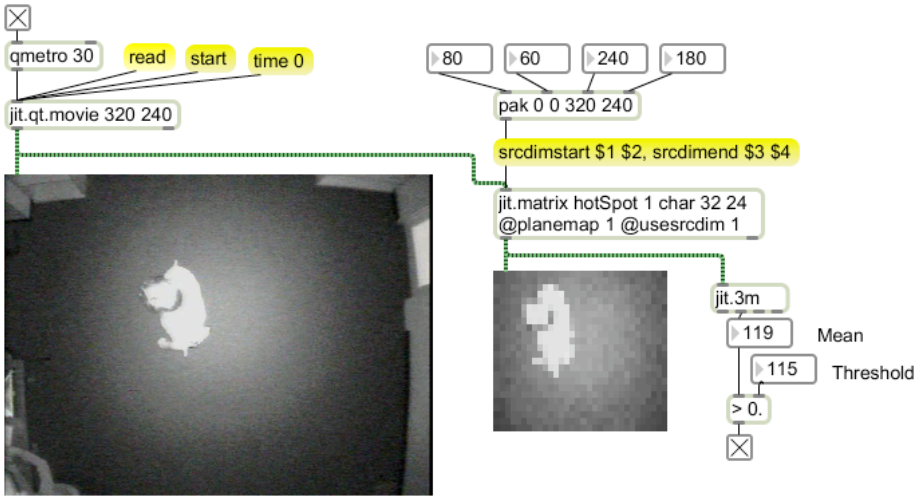

Figure 1.

Figure 1 shows the basic approach. The srcdimstart and srcdimend messages to a jit.matrix will focus the image on a small section of the original. Keeping this matrix small will speed up performance. In figure 1 the hot spot matrix is 32 by 24 and only responding to the red. (This image is infrared, so a person will register as white.) The jit.3m object gives a quick assessment of the mean brightness, which will change when a figure walks into it. You can use the [>] object (greater than) to detect the visitor's presence. Unfortunately, the mean provides a very narrow range of values, so this can be fussy to set, especially if the basic light level changes.

A bit more reliable scheme is to trigger on the difference between the maximum and minimum pixel value. This will be resistant to gradual changes in the lighting such as those from cloudy to clear. Figure 2 illustrates:

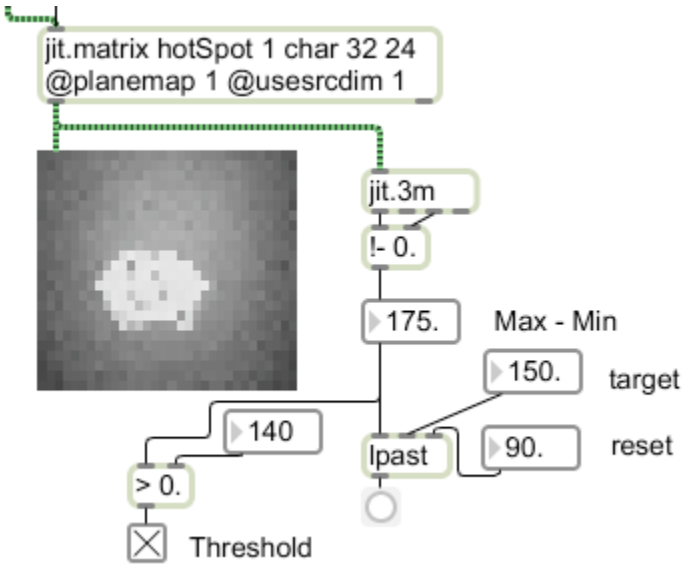

Figure 2.
The simplest way to detect presence in the zone is to set a threshold value. The [<] object produces 1s whenever the zone is occupied. This is a series of 1s, which need some processing in order to trigger actions. Usually we want to detect entry into a zone to trigger some process. A togedge on the [>] output will do this, but there is a better way. My Lpast object is similar to the stock past object in that it will only bang when the input rises to the target value. The difference is Lpast lets you determine the reset value the input must drop below before it can trigger again. (The difference between the target value and the reset value is hysteresis.) Since pixels are always varying somewhat, the past object is subject to multiple triggers as the visitor enters the zone.

Motion detection

To make the zone sensitive to motion instead of presence, we use frame differencing. Figure 3 shows the basic technique. The reduced image is fed to a second matrix placed to the left of the patch. A clone of the matrix (same name) is banged before the jit.qt.grab object and the old frame is sent to the right inlet of a jit.op. When the new frame comes in, only pixels that have changed will be bright. Jit.3m can be tapped for the max or mean. This will be more sensitive if you slow down the frame rate.

The flaw in this is if the visitor stands perfectly still, there will be no detection. In figure 4, the processing is extended to detect no motion for a specified period of time. The ticks are received from the qmetro and applied to a counter. This counter is reset to 0 anytime motion is detected in the zone. If the counter exceeds 100 (no motion for 3 seconds), we can assume the zone is empty. This technique can be applied to the master image to sense when there are no visitors at all and shut down.

Figure 4 also shows a slightly more efficient way to store the previous frame. The trigger object will route the current frame to the right of teh jit.op after the comparison is complete.

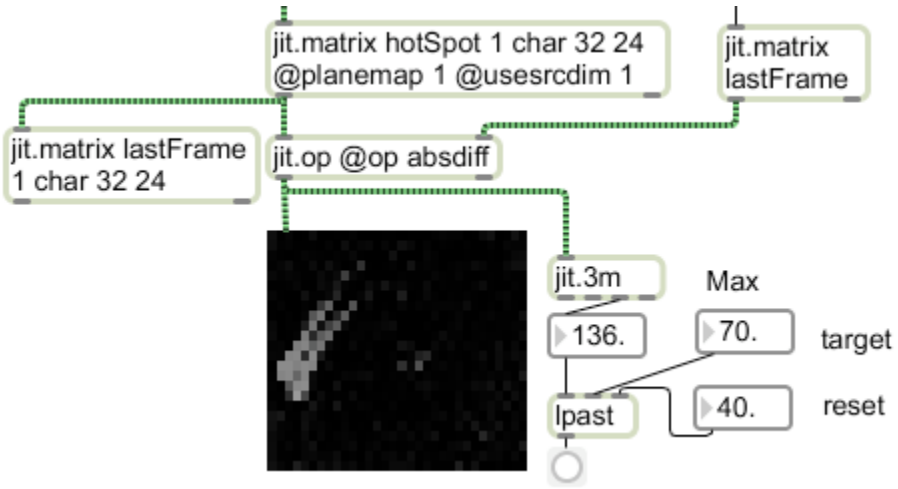

Figure 3. Frame difference technique

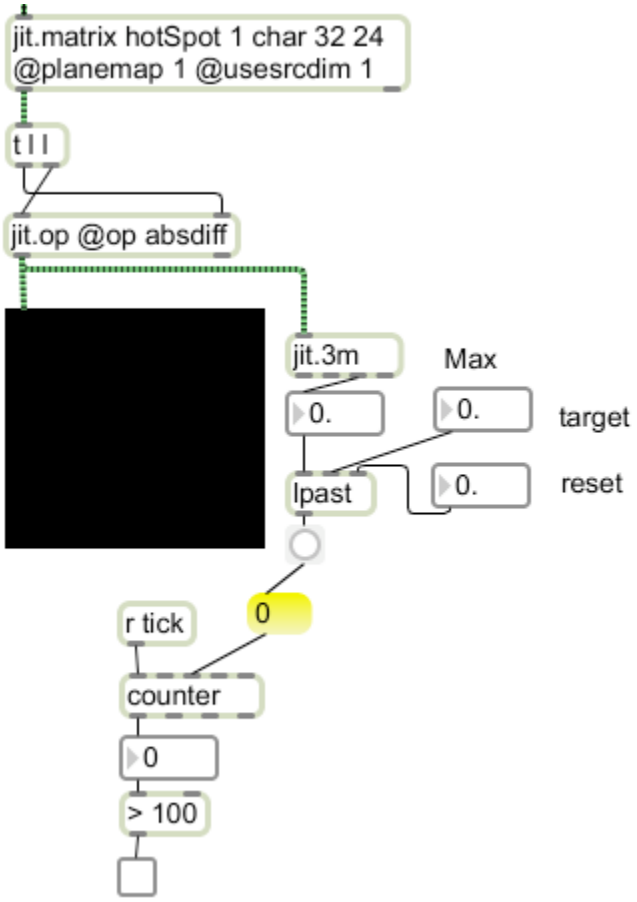

Figure 4. Refined frame difference

Counting to determine when motion has stopped is necessarily a slow process. If you anticipate your visitors may sit still for a long time but need the system to respond quickly to their exit, you can compare the current frame to a reference image. All you

have to do is capture a frame when the hot spot is empty. Figure 5 shows how to do this with the space bar. The bar opens a gate which is closed immediately after the matrix passes through.

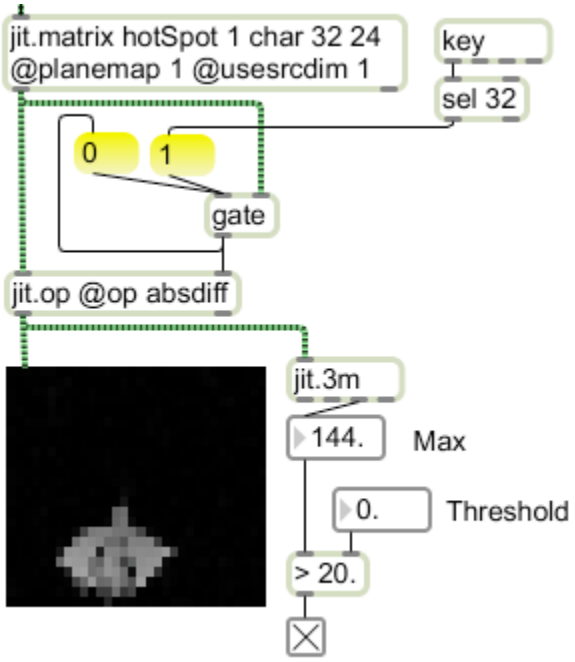

Figure 5. Comparison with base image

The base image is stored in the right inlet of the jit.op. This method is very reliable when the lighting is stable, but the base image should be refreshed fairly often. (Perhaps when the counter method shows no movement for a minute or more.)

Direction of Motion

Detecting direction of motion requires two zones that overlap, such as the red and blue areas marked in figure 6.

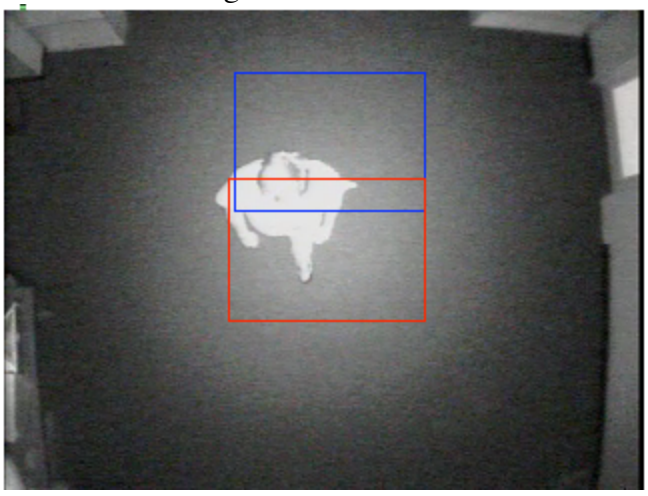

Figure 6.

The trick behind direction detection is simple. If the subject is in the blue zone when he enters the red zone, he is moving from blue to red. If the subject is in the red zone when he enters the blue zone, he is moving from red to blue. Thus if we use togedge on red to capture the state of blue, we know the direction. Figure 7 shows the mechanism.

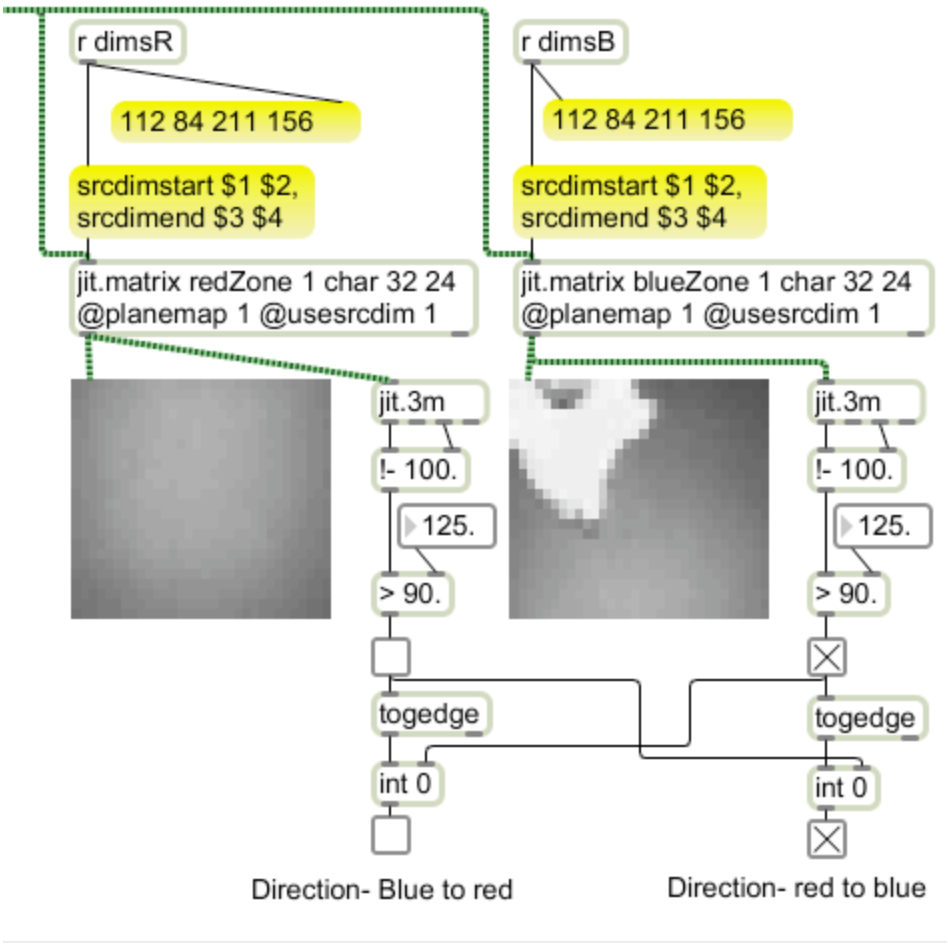

Figure 7.
This works best when the visitor's motion is constrained, perhaps by a doorway.

The ability to draw a rectangle on the master image and have that determine the size of the hot spots is a very useful trick. Here's how it's done;

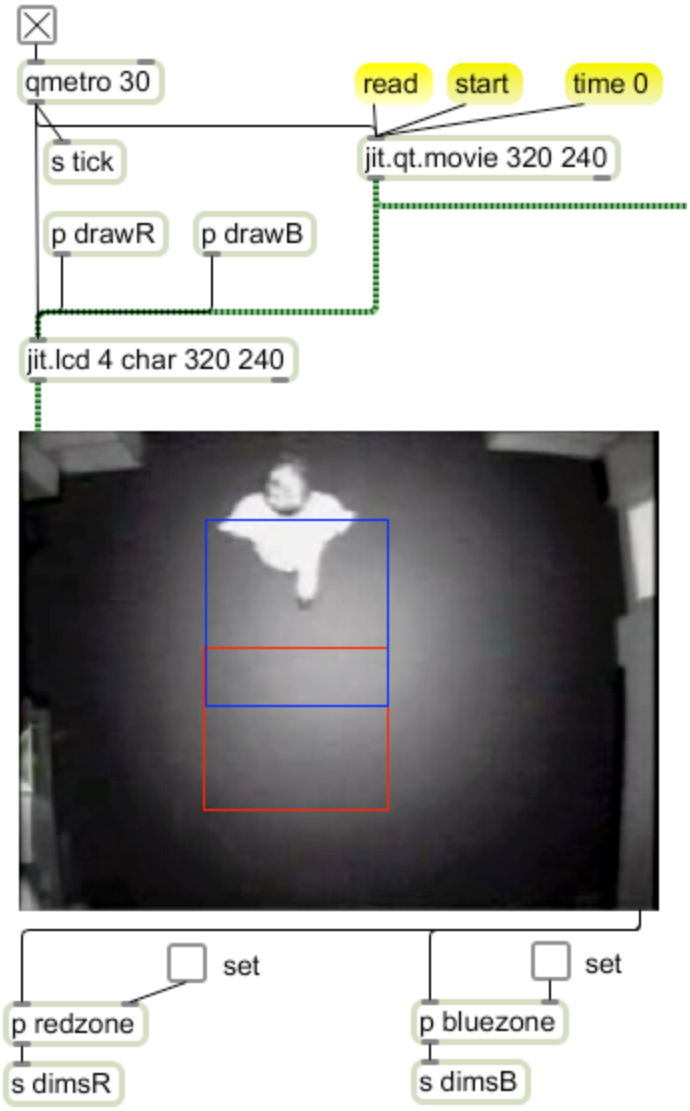

Figure 8. master image for figure 7.

The image from jit.qt.grab (or jit.qt.movie for practice) is sent via jit.lcd to the pwindow. If a matrix is input to jit.lcd, all drawing will occur on top of the image. (Matrix input is tantamount to clear.) When a user clicks in the pwindow, a mouse message is sent out the right outlet. This is formatted "mouse X Y button" with some following items for control keys and the like. Most of the messages are "mouse X Y 1" but when the mouse is released, a single "mouse X Y 0" marks the release point. These messages are processed in the redzone subpatch like this:

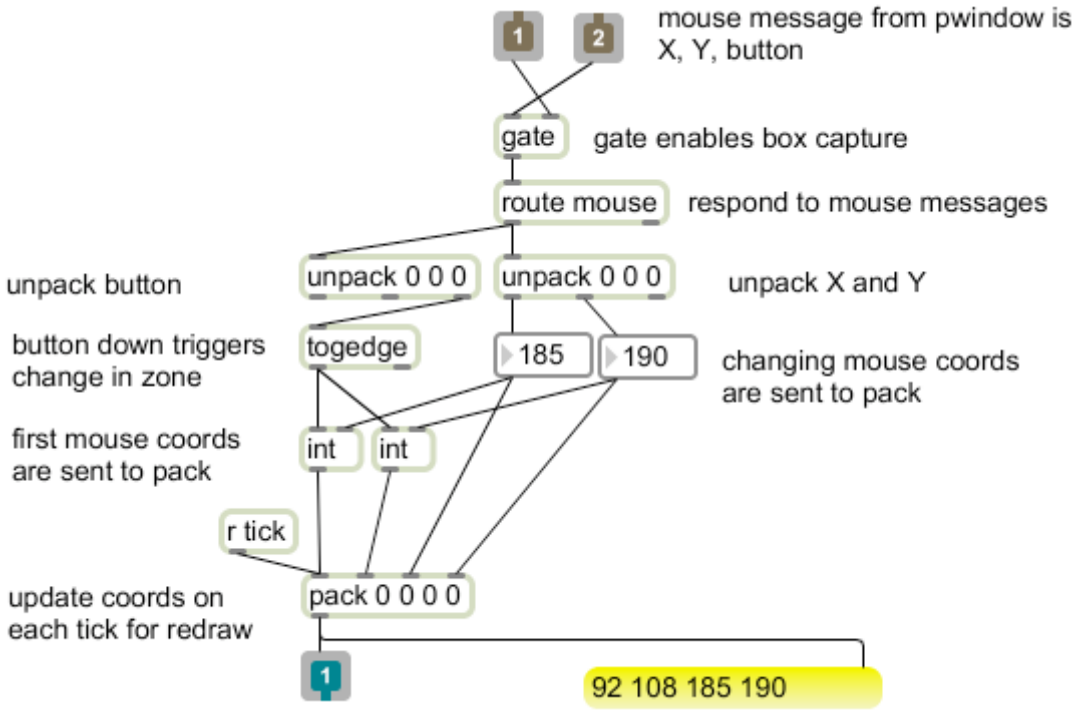

Figure 9. Mouse processing subpatch (redzone)

The gate is controlled by an external toggle to turn on box resizing. On the first message in the X and Y values are loaded to the int boxes and sent on to the first two positions of pack by the togedge responding to the 1 of the button state. After this, as long as the mouse is held down, the last two items in pack are updated but the first two are unchanged. When the mouse is released, the 0 in the mouse state primes togedge to fire on the next mouse down message. The result is a series of lists of the oredr left, top, right, bottom, just what is needed to draw a box and set the size of the spot matrices. These go to a framerect command fed to the jit.lcd and srcdimstart and srcdimend messages to the spot matrix.

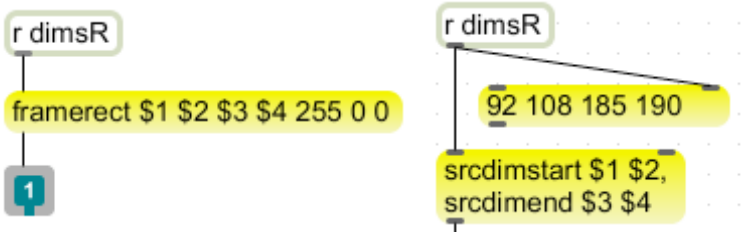

Figure 10. note how the framerect command specifies color.

Watching a Line

Sometimes we want to detect where objects cross a line. The next patch shows that, in the context of detecting fingers playing an invisible harp.

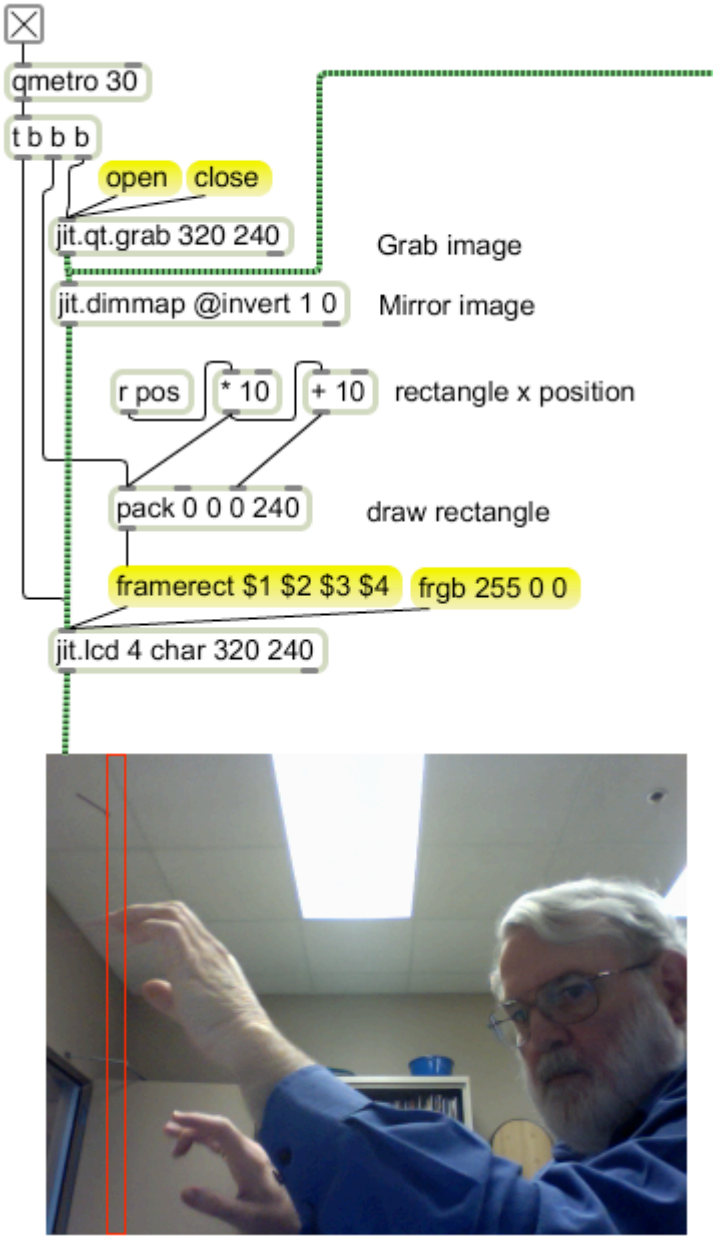

Figure 11.

Figure 11 shows half of the patch. This simply grabs an image from the computer camera, reverses the image to give a mirror effect (performance is much easier watching a mirror than a raw image.) and draws a red box over the zone that will trigger notes.

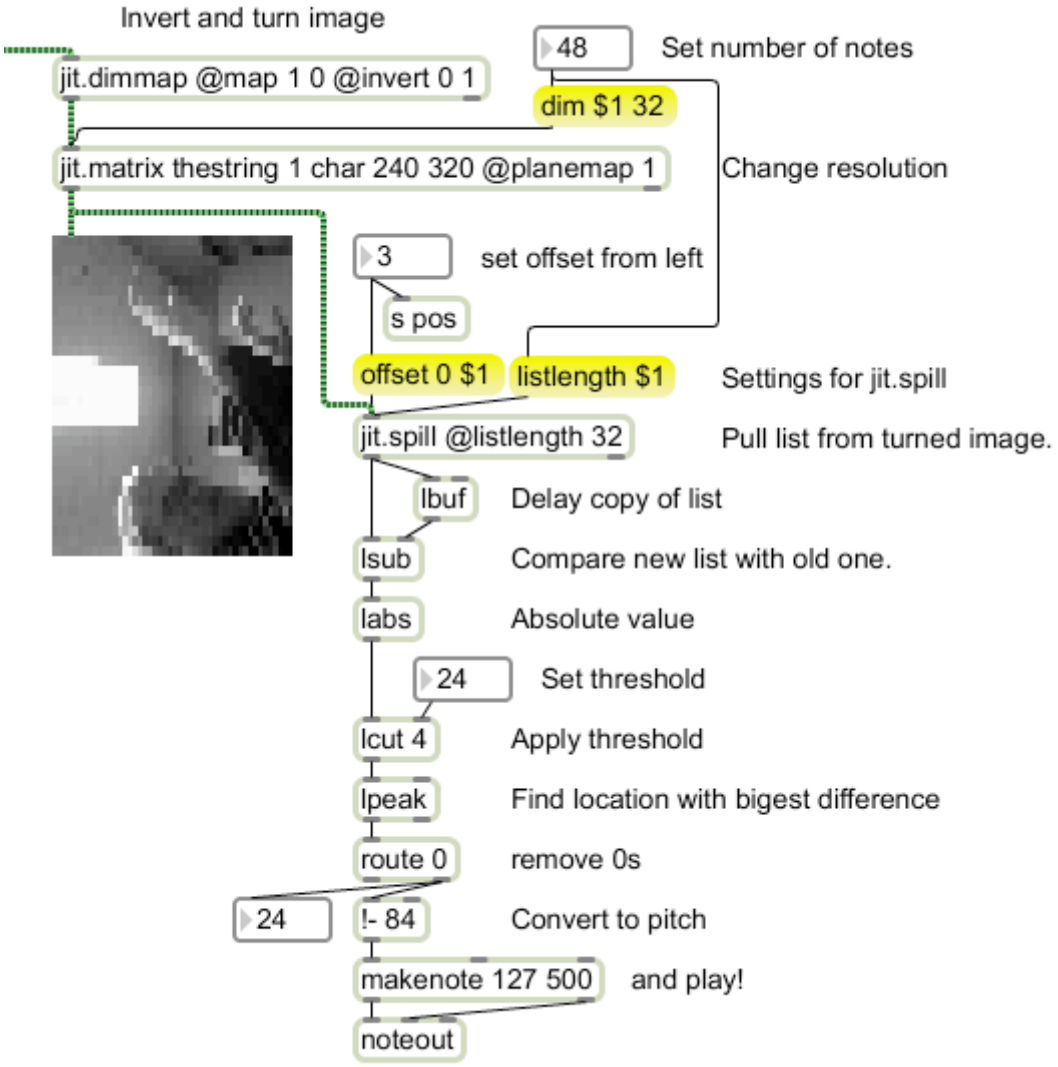

Figure 12.

Figure 12 shows how the image is processed to extract triggers. The key object for this is jit.spill which can convert a selected row of a matrix to a list. Since we are interested in a vertical line, we must convert columns to rows.