A note on jit.findbounds

Jit.findbounds is not the most sophisticated tracking object out there, but it is a quick and simple solution that is useful in some well controlled situations. The point of jit.findbounds is to identify the area of a specific color in an image. Given a color specification, the output is the upper left and lower right corners of a box that bounds all pixels of that color. This is what the helpfile wants us to do:

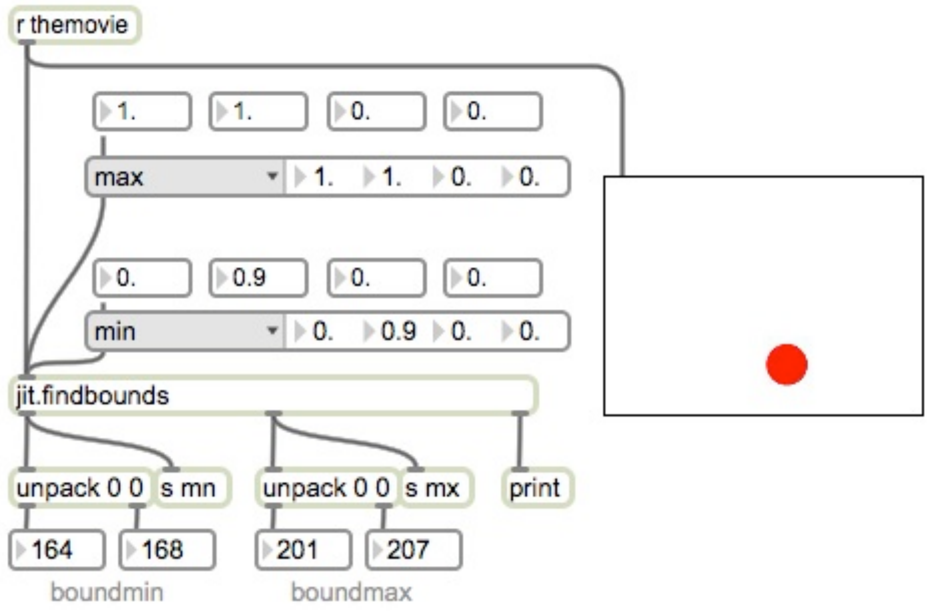


Figure 1.
The color to find is specified as a range of pixel values from 0 to 1.0. The color format is ARGB. Since we never know what the alpha values of a movie are going to be (jit.qt.movie sets them to 1 if there is no alpha) we allow the full range most of the time. For the other colors, the min and max values must bracket the target color.

If the color is simple, such as the solid red ball in redball.mov (found in your cycling74/patches/media folder) it can be set up by hand as in figure 1. This is seldom an effective option, however. Figure 2 shows how to define the color with a swatch. The target color is chosen by swatch. (Swatch does not need to be in old style mode here.) The minimum and max values are calculated by lsub and ladd objects., adding or subtracting the tolerance value. Vexpr could be used instead. The tolerance should be as small as possible, as wide tolerances will increase the chances of mistracking.

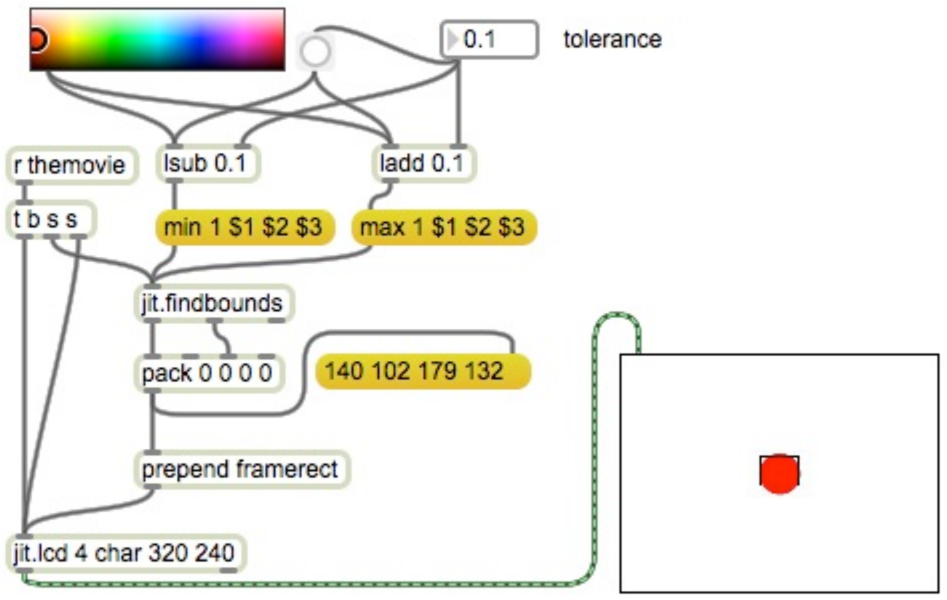


Figure 2.
Figure 2 also adds some objects to display the bounds that are found, by drawing a rectangle in jit.lcd. Note that the trigger object (t b s s) sends the matrix to jit.lcd first, to clear the old image. Then findbounds gets the matrix so it can do its work before a bang sends the results to the pwindow.

The swatch is still not an ideal way to set the color, because you have to match shades by eye. The best approach is to use suckah, which provides color information where the mouse is clicked. Suckah is dropped over the pwindow that displays the movie. Once the patch is locked, a click on the window produces the color under the mouse.

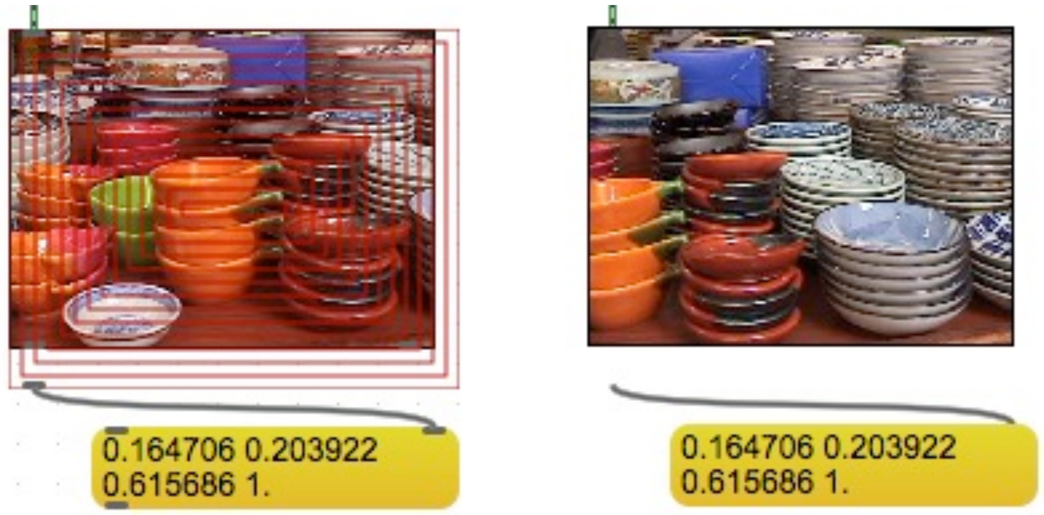


Figure 3. Suckah with patch unlocked and locked.

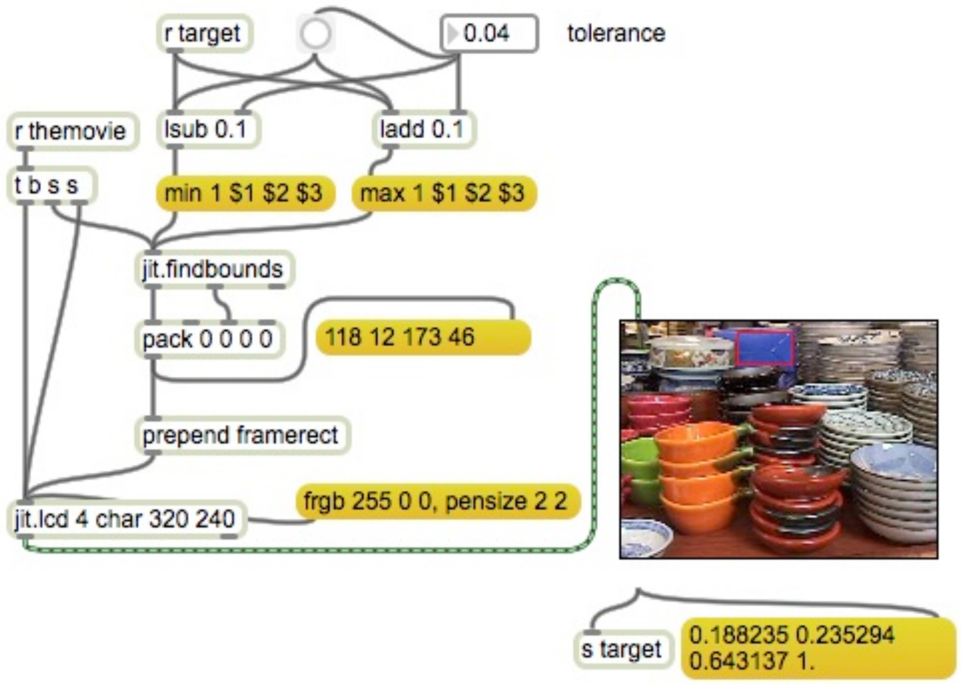
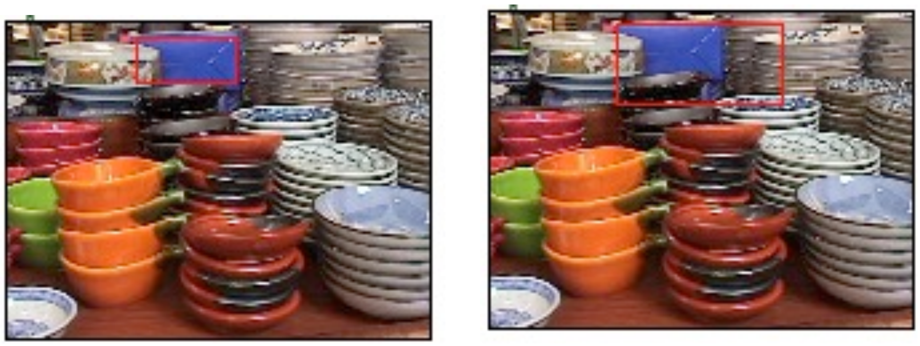


Figure 4.

Figure 4 shows a patch with suckah providing the target color. Note that we still need to generate a min and max setting as before. The target color was chosen by clicking the blue item, which is nicely outlined by a red frame. The tolerance was very tight, 0.04. Figure 5 shows what happens when tolerance is increased.



Tolerance 0.6 Tolerance 0.13
Figure 5.

A tolerance of 0.16 encompasses most of the window. Of course very tight tolerances make it difficult to capture any pixels at all. (Findbounds outputs -1 for all values when no matching pixels are detected.) it is particularly difficult when the image area is not lit consistently-- the object of interest often changes color enough to get lost.