Subject: Re: Efficient finding of locations where two 'on' pixels are next to one another
Posted by jeanh on Thu, 13 Jan 2011 17:56:28 GMT
View Forum Message <> Reply to Message

On 13/01/2011 12:28 PM, Robin Wilson wrote:
> Hi,
>
> I'm working with a binary image which is the output of another procedure
> I've run. I am trying to write a routine to post-process this image, and
> clean it up a bit. As part of this, I want to find the indices of pixels
> that are set to 1 (ie. 'on') that are next to other pixels that are set
> to 1, but only in the X direction. To make that a bit clearer, there is
> a diagram below:
>
>
> (0) . . . * * . .
> (1) . . . * . . .
> (2) . . * * * . .
> (3) . . . * . . .
> (4) . . . * . . .
>
> In this example I would want the indices of both 'on' pixels in Row 0,
> nothing in Row 1 (as the pixel is not touching anything in the X
> direction), all three in Row 2, and nothing in Rows 3 and 4.
>
> At the moment all I can think of some kind of ghastly FOR loop with
> loads of state-checking. I'm thinking that there must be a nicer (that
> is more efficient and cleaner code) way...is there?
>
> Cheers again,
>
> Robin

You can try using "shift"

```
pix = where(img eq 1 and shift(img,-1,0) eq 1)
pix = [pix,pix+1] ;include the index of all valid pixels
pix = pix[uniq(pix,sort(pix))] ; remove duplication caused by the
previous line
```

You handle the behavior at the edge

Jean
PS: untested, double check that I didn't got confused between +1 and -1
in the shift()