
Subject: Making this code faster & nicer (ORing across dimensions of array?)

Posted by [Robin Wilson](#) on Sat, 22 Jan 2011 17:55:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I have written the function below, to find the endpoints of a line in a binary image using the HitOrMiss morphology. It is being called quite a lot in my program, and therefore I want to speed it up. I am also aware that the code is pretty horrible at the moment, as there are lots repeated lines - it definitely needs refactoring.

The question is: how should I best refactor this for speed and 'niceness' (in that order)?

My original plan was to make an 'miss' array which would be 3 x 3 x 8 and then do the MORPH_HITORMISS operations across these 8 sub-arrays. I'd then store the result in a multi-dimensional array and OR across the dimensions to get the output. However, I've found that you can't seem to do that with either the MORPH_HITORMISS function or the OR function. As this won't work, what should I do instead? Or is this function as good as it can be (I can't quite believe that...)

```
FUNCTION GET_ENDPOINTS, image
; Create output image
dims = SIZE(image, /DIMENSIONS)
output = intarr(dims[0], dims[1])

; Create the hit element
hit = intarr(3, 3)
hit[1,1] = 1

; Create the miss elements
miss1 = [ [0, 0, 0], [1, 0, 1], [1, 1, 1] ]
miss2 = [ [1, 1, 0], [1, 0, 0], [1, 1, 0] ]
miss3 = [ [1, 1, 1], [1, 0, 1], [0, 0, 0] ]
miss4 = [ [0, 1, 1], [0, 0, 1], [0, 1, 1] ]
miss5 = [ [1, 0, 1], [1, 0, 0], [1, 1, 1] ]
miss6 = [ [1, 1, 1], [1, 0, 1], [1, 0, 0] ]
miss7 = [ [1, 1, 1], [0, 0, 1], [1, 0, 1] ]
miss8 = [ [1, 0, 1], [0, 0, 1], [1, 1, 1] ]

; Do the morphology
res1 = MORPH_HITORMISS(image, hit, miss1)
res2 = MORPH_HITORMISS(image, hit, miss2)
res3 = MORPH_HITORMISS(image, hit, miss3)
res4 = MORPH_HITORMISS(image, hit, miss4)
res5 = MORPH_HITORMISS(image, hit, miss5)
```

```
res6 = MORPH_HITORMISS(image, hit, miss6)
res7 = MORPH_HITORMISS(image, hit, miss7)
res8 = MORPH_HITORMISS(image, hit, miss8)
```

```
; Combine the outputs
```

```
output = res1 OR res2 OR res3 OR res4 OR res5 OR res6 OR res7 OR res8
```

```
return, output
```

```
END
```

Cheers,

Robin
