Subject: Re: Avoiding FOR loops (version googleplex.infinity)
Posted by MichaelT on Mon, 07 Apr 2008 16:19:34 GMT
View Forum Message <> Reply to Message

You can do it without WHERE and EXTRACT. As long as your the number of surrounding elements and your 2d Byte array is not too large.
Otherwise you may get an "unable to allocate memory: to make array" message. Basically, arrays of indexes are generated so that you can directly compare your original array with its 5x5 = 25 shifted versions. The positions at the border are not considered.

==========code begins==========

```
;Generate a byte array with random numbers, ranging from 0 to 9
nx = 300
ny = 200
b = byte(randomu(s, nx, ny) * 10)

;Define your search "radius". Your search area is 5x5 so the search
radius (sr) is 2
sr = 2

;width and height of your search area is (sr * 2 + 1)
nsr = (sr * 2 + 1)

;Generate an array containing the index deviations from your central
index (for x and y indexes)
;Example for x
;-2, -1, 0, 1, 2
;-2, -1, 0, 1, 2
;-2, -1, 0, 1, 2
;-2, -1, 0, 1, 2
;-2, -1, 0, 1, 2
;y is simply the transposed of x

vx = (LINDGEN(nsr) - sr) # (LONARR(nsr) + 1l)
vy = Transpose(vx)

;Reform the (5x5) array so that it becomes a vector of length (25)
vx = reform(vx, nsr^2)
vy = reform(vy, nsr^2)

;Now replicate this for each element of your byte array (omitting the
sr=2 positions at each border)
vxs = replicate({a: vx}, nx - 2*sr, ny - 2*sr)
vx = vxs.a

vys = replicate({a: vy}, nx - 2*sr, ny - 2*sr)
```

```
vy = vys.a

;Now the x- and y-indexes of your byte array are generated, again
omitting the positions at the border.
;So it starts at position sr=2 and runs through position nx-1-
sr=nx-1-2
ix = (lonarr(ny - 2*sr) + 1l) # (lindgen(nx - 2*sr) + sr)
iy = (lindgen(ny - 2*sr) + sr) # (lonarr(nx - 2*sr) + 1l)

;Replicate this as often as there are elements in your 5x5 window =
nsr^2 = 25.
ixs = replicate({a: ix}, nsr^2)
iys = replicate({a: iy}, nsr^2)

;Transpose the array so that it has the same dimensions as vx and vy
ix = transpose(ixs.a)
iy = transpose(iys.a)

;Now the shifted positions are generated simply by adding the index
deviations to the index numbers
ixv = ix + vx
iyv = iy + vy

;b[ixv, iyv] eq b[ix, iy] results in an array containing 1 where a
shifted position is equal to the central position otherwise 0.
;This is summed over your 25 shifted positions: total(result, 1)
;In the end you have to substract 1 from each element as the central
position is compared to itself as well and contributes to the sum.

bn = total(b[ixv, iyv] eq b[ix, iy], 1) - 1l

;Location [0, 0] in the bn-array then corresponds to the value for [2,
2] in the b-array, due to the border problem

;Print example to check:
print, bn[0, 0]
print, b[0: 4, 0: 4]

end
==========code end=======================
```

I hope it will work in your case and should be quicker than a loop.

Michael
