Subject: Re: Cell boundary program?
Posted by Richard Adams on Tue, 08 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

> The other method, convolution is also quick, but the returned indices are in
> scan-line order. In this case, all the boundary pixels are included. This
> method was suggested by ... uhoh, I forgotten who it was right now, sorry.
> Here's the steps as described on the newsgroup by the unknown author.
>
> bb = convol(Image, replicate(1,3,3),9,/center)
> Edges = Where(bb gt 0 AND bb LT 255)
> bb[*] = 0
> bb[edges] = 255 ; make this new image just edges
> bb = Image AND BB ; now keep just those edges inside your object
> perimeter = where(bb eq 255); these are indices to final outline
>
Actually I am guilty of the second method. I can add to that what I hope is an improved version. Pass the autotrace function a greyscale image and a threshold level or range and it returns to you a IDLgrROIGroup of all the outlines it finds. It can be slow on complex images. You could take out the trace_outline function to get just one outline, if you know where it starts. There are several possible improvements and maybe bug fixes - I haven't gotten to use it much since I wrote it. I could be made into a nice object with methods to get each outline by location or size, and it could be given an interactive widget to make selection easier. It also only uses 4-connectivity and could be better converted to 8-connectivity (you might lose some corners on the outlines). I'll may fix it if I get time, otherwise somebody else could :-)

I hope it is useful, Richard.

```
function autotrace, image, select_range

;
; Given an image (greyscale) and a selection range - either a single grey
level or a min/max range
; return a IDLgrROIGroup that corresponds to the 4-connected outline of each
selected region
;

   outlines = obj_new('IDLgrROIGroup')
   image_size = size(image, /dimensions)
   work_image = bytarr(image_size[0], image_size[1])
   if n_elements(select_range) eq 1 then selected = where(image eq
select_range, count) $
      else selected = where((image ge min(select_range)) and (image le
max(select_range)), count)
   if count eq 0 then return, outlines
```

```
   work_image[selected] = 255
   bb = convol(work_image, replicate(1,3,3),9,/center)
   edges = Where(bb gt 0 AND bb LT 255)
   bb[*] = 0
   bb[edges] = 255 ; make this new image just edges
   work_image = work_image AND bb ; now keep just those edges
   edges = where(work_image eq 255, count)

   repeat begin ; search for each outline
      edges = where(work_image eq 255, count)
      if count gt 0 then begin
         start = edges[0]
         new_roi = trace_outline(work_image, start)
         outlines->Add, new_roi
         endif
      endrep until count eq 0
   return, outlines
end


function get_neighbour_index, start, nx, ny, first_dir

; get indices for maze tracing.
; relative to direction of last step, look left, forward, right, back in
that preferred order
; first check that each step is in bounds for our image
; make an array of those indices
; shift array to get absolute direction into relative directions
; see which indices are valid
; remember which absolute directions these correspond to
   right = 0
   down = 1
   left = 2
   up = 3
   above = start lt nx ? -1 : start - nx
   below = start / nx eq ny - 1 ? -1 : start + nx
   to_right = start mod nx eq nx - 1 ? -1 : start + 1
   to_left = start mod nx eq 0 ? -1 : start - 1
   neighbours = [below, to_left, above, to_right]
   neighbours = shift(neighbours, -first_dir)
   directions = shift([right, down, left, up], -first_dir + 1)
   valid = neighbours ge 0
   return, [[valid], [neighbours], [directions]] ; [which are valid, their
indices, their directions]
end

function trace_outline, image, start
   image_size = size(image, /dimensions)
```

```
    done  = 0
    direction = 0 ; right to start
    current = start
    verts = [current]
    while not done do begin
       search = get_neighbour_index(current, image_size[0], image_size[1],
direction)
       s_valid = where(search[*, 0] eq 1, s_count)
       if s_count eq 0 then stop ; should not happen unless 1 pixel image!
       s_index = reform(search[*, 1])
       s_dirs = reform(search[*, 2])
       next_edge = where(image[s_index[s_valid]] eq 255, n_count)
       if n_count ge 1 then begin
          current = s_index[s_valid[next_edge[0]]]
          verts = [verts, current]
          direction = s_dirs[s_valid[next_edge[0]]]
          done = current eq start
          endif else done = 1
       endwhile
    image[verts] = 0
    x = verts mod image_size[0]
    y = verts / image_size[1]
    oOutline = obj_new('IDLgrRoi', x, y)
    return, oOutline
end
```