Subject: What could cause disappearing array?
Posted by robintw on Tue, 19 Jan 2010 21:54:25 GMT
View Forum Message <> Reply to Message

Hi,

I've got a very strange problem in one of my IDL programs. I have two nested FOR loops with some processing happening in the inside loop, and after a bit of processing one of my arrays seems to disappear.

I've checked this with some strategically placed help statements, when it seems to disappear it there (comes up with the right dimensions etc with the help command) at the bottom of the loop, but by the time the loop starts again it is showing as undefined.

I was about to tell you how many iterations of my loop this occurred after (by looking at the value of i and j when it crashed) but after it's crashed I find i and j to both be equal to 0 - even though it's gone through most of the loop already.

I can't see anywhere in the code that I'm assigning anything to i or j. I've left the FOR loop to look after them itself, yet somehow they end up as 0.

When I comment out all of the code inside the for loop then the loop runs perfectly to the end, and i and j don't get reset to zero part way through.

Does anyone have any ideas what on earth might be causing this? I've really got no idea!

The code for this function is below. If you need any of my code from other functions called by this function then just let me know.

Cheers,

Robin

```
PRO TRY_PLACING_REGION, WholeImage, edge_map, window_size
  COMMON SegData, seg_id, segment_image

  d = (window_size - 1) / 2

  image_dims = SIZE(WholeImage, /DIMENSIONS)

  ns = image_dims[0]
  nl = image_dims[1]
```

```
  FOR i = 0L, nl - 1 DO BEGIN
    FOR j = 0L, ns - 1 DO BEGIN
      print, i
      print, j
      help, segment_image

      j_lower_limit = j-d
      IF j_lower_limit LT 0 THEN j_lower_limit = 0

      j_upper_limit = j+d
      IF j_upper_limit GT ns - 1 THEN j_upper_limit = ns - 1

      i_lower_limit = i-d
      IF i_lower_limit LT 0 THEN i_lower_limit = 0

      i_upper_limit = i+d
      IF i_upper_limit GT nl - 1 THEN i_upper_limit = nl - 1

      image_moving_window = WholeImage[j_lower_limit:j_upper_limit,
i_lower_limit:i_upper_limit, *]
      edge_moving_window = edge_map[j_lower_limit:j_upper_limit,
i_lower_limit:i_upper_limit]
      segment_moving_window =
segment_image[j_lower_limit:j_upper_limit, i_lower_limit:i_upper_limit]

      ; Go to next possibility if moving window contains edges
      if total(edge_moving_window) NE 0 THEN CONTINUE

      ; If any of the window's pixels are already in a segment then go
to next possibility
      if total(segment_moving_window) NE 0 THEN CONTINUE

      inequality_result = INEQUALITY_FUNCTION(image_moving_window, [1,
1, 1])

      ; If one of the pixels had a inequality value greater than 1 then
continue
      if inequality_result EQ 0 THEN CONTINUE

      ; If we've got to here then it's a suitable seed
      print, "Found a suitable seed"

      ; Mark the seed as a segment
      segment_image[j_lower_limit:j_upper_limit,
i_lower_limit:i_upper_limit] = seg_id

      help, segment_image
```

```
      seg_id++
    ENDFOR
  ENDFOR

  ENVI_ENTER_DATA, segment_image
END
```