Subject: Re: Reading an arbitrary profile from 2D FITS data
Posted by Craig Markwardt on Tue, 05 Jul 2011 04:26:06 GMT

View Forum Message <> Reply to Message

On Jun 29, 7:07 am, Bringfried Stecklum <steck...@tls-tautenburg.de>
wrote:

```
>  Balt wrote:
>>  Hi all,
>
>>  A seemingly very simple problem has me stumped: How do I extract into
>>  a vector a profile line from a 2D FITS data set? The IDL astro lib
>>  doesn't seem to contain such a function for 2D, only for 3D, and then
>>  only along a cardinal axis. That in turn is easy to do also but I need
>>  it to go from for example (x,y) 100,100 to 320,240.
>
>>  Any ideas?
>
>>  Cheers
>
>>  - Balt
>
>  Perhaps these few lines of code may help. Regards, Bringfried
>
>  ; extract profile from 2D image
>  ; input  - image
>  ;        - profile x and y start/end coords [x0,x1,y0,y1]
>  ; output - 1D profile
>
>  function profile,image,xy
>  ; check index bounds here...
>  if (xy[0] eq xy[1]) then return,image[xy[0],xy[2]:xy[3]]
>  if (xy[2] eq xy[3]) then return,image[xy[0]:xy[1],xy[2]]
>  ; non-trivial case
>  a=(xy[2]-xy[3])/(xy[0]-xy[1])
>  b=xy[2]-a*xy[0]
>  x=xy[0]+indgen(xy[1]-xy[0]+1)
>  y=round(a*x+b)
>  return,reform(image[[x],[y]])
>  end
```

It's the right idea.

The original question leaves open to interpretation what a "vector
profile" might be. I would suspect it would be equi-spaced in 2D
space. In which case it might be best to make a unit vector pointing
from A to B, and then INTERPOLATE() along multiples of that unit
vector.

Something along these lines (completely untested)

```
; extract interpolated profile from 2D image (equispaced)
; input  - image
;        - profile x and y start/end coords [x0,x1,y0,y1]
; n      - number of intervals between start/end point
; output - 1D profile (returns N+1 samples)
function profile,image,xy,n
  xya = xy[0:1] ;; Start
  xyb = xy[2:3] ;; End

  ; Distance from start to end
  dist = sqrt(total( (xyb - xya)^2 ))

  ;; Degenerate cases
  if n EQ 0 then return, image[xya[0], xya[1]]

  ;; Unit vector - there are N intervals from start to end
  unit = (xyb - xya)/n

  ;; Uniformly sampled points in X and Y
  xx = xya[0] + dindgen(n+1)*unit[0]
  yy = xya[1] + dindgen(n+1)*unit[1]

  ;; Interpolate
  return, interpolate(image, xx, yy)
end
```