
Subject: Re: IDL 8.2, read pixel value along a surface
Posted by [Helder Marchetto](#) on Tue, 21 May 2013 14:44:49 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tuesday, May 21, 2013 4:20:25 PM UTC+2, a.clo...@gmail.com wrote:

> Hello Everyone !
>
>
>
> I use IDL 8.2 and my problem is the following :
>
>
>
> I have a FITS Image that i know how to read. I choose 2 points and read the pixel value along
the line between them.
>
> My problem is, i want to do the same (i.e read the pixel value) but along a surface , (the same
line but with some pixel wide)
>
> I thought of using interpol, and add several one but it seems to be very long.
>
>
>
> Any ideas ?
>
>
>
> thanks !

Hi,

I've done this and I have seen two approaches. In my case, I construct all the pixel values that would be on the line (surface) of a given width using the same pixel interval as the raw data, then I interpolate. It might sound not very efficient, but IDL is good if you work with arrays.

The other approach is to rotate the image around one of the two reference points and then cut a section of the image.

The two methods differ slightly depending in the performance. If I remember correctly, the rotation method is slightly slower for "thin" surfaces and comparable for thicker regions.

I have some untested/uncommented code that performs the first method and the code for the second method is hidden in some old hard-drive.

The procedure below can actually calculate a given line made of more than two points. The input parameters should be:

Img: n x m image

Pts: 2 x p points in the form [[Pt0x,Pt0y],[Pt1x,Pt1x],[Pt2x,Pt2x],...]

InWidth: Pixel width

The averaged values will be returned in xRes, yRes

You can test the procedure like this:

```
;Test code
a=dist(400)
pts = [[0.0,0.0],[300.0,300.0]]
w = 10
GetCS_Image, a, pts, w, xres=x, yres=y
plot, x,y
;*****
;
```

Here is the code:

```
PRO GetCS_Image, Img, Pts, InWidth, Distances = Distances, $
      xRes = xRes, yRes=yRes, PosRef=PosRef ;Output variables

IF N_ELEMENTS(InWidth) EQ 0 THEN Width=1.0 ELSE Width= ((ROUND(InWidth) MOD 2) NE
1 ) ? (FLOAT(InWidth+1.0)>1.0) : (FLOAT(InWidth)>1.0)
IF N_ELEMENTS(Img) EQ 0 THEN Message, 'GetCS_Image: Please supply an image',
/NoName
IF N_ELEMENTS(Pts) LT 4 THEN Message, 'GetCS_Image: Please supply two points for the
cross-section', /NoName

FltHalfWidth = (Width-1.0) / 2.0
nPoints      = (SIZE(Pts, /DIMENSIONS))[1]
nLines       = nPoints-1
FloatPts     = FLOAT(Pts)
ShiftedFloatPts = SHIFT(FloatPts,0,-1)
Distances    = SQRT(TOTAL(((ShiftedFloatPts-FloatPts)[*,0:(nLines-1)])^2, 1))
CumulDistances = TOTAL(Distances,/CUMULATIVE)
WidthArr     = FINDGEN(Width)
OnesArr      = WidthArr*0.0+1.0
Angle        = ATAN((shiftedfloatpts-FloatPts)[1,0:-2],(shiftedfloatpts-Flo atPts)[0,0:-2])
NormAngle    = Angle+!PI/2.
CosNormAngle = COS(NormAngle)
SinNormAngle = SIN(NormAngle)
CosAngle     = COS(Angle)
SinAngle     = SIN(Angle)
CeiledDist   = CEIL(Distances)
FOR I=0,nLines-1 DO BEGIN
  xNorm = (WidthArr-FltHalfWidth)*CosNormAngle[I]+FloatPts[0,I]
  yNorm = (WidthArr-FltHalfWidth)*SinNormAngle[I]+FloatPts[1,I]
  xNorm = xNorm # (FLTARR(CeiledDist[I])+1.0)
  yNorm = yNorm # (FLTARR(CeiledDist[I])+1.0)
  xlocArr = OnesArr # FINDGEN(CeiledDist[I])*CosAngle[I]
  ylocArr = OnesArr # FINDGEN(CeiledDist[I])*SinAngle[I]
  profile = TOTAL(INTERPOLATE(Img, xlocArr+xNorm, ylocArr+yNorm, CUBIC=-0.5,
MISSING=0.0),1)/Width
  IF I EQ 0 THEN BEGIN
    nPos = N_ELEMENTS(Profile)
```

```
xRes = LIST(FINDGEN(nPos), /EXTRACT)
yRes = LIST(Profile, /EXTRACT)
PosRef = LIST(nPos)
ENDIF ELSE BEGIN
  nPos = N_ELEMENTS(Profile)
  xRes.Add, FINDGEN(nPos)+CumulDistances[l-1], /EXTRACT
  yRes.Add, Profile, /EXTRACT
  PosRef.Add, nPos+PosRef[l-1]
ENDELSE
ENDFOR
xRes = xRes.ToArray()
yRes = yRes.ToArray()
PosRef = PosRef.ToArray()
END
```

Hope it helps. Let me know if you have improvements... I once in a while use this.

Cheers,
Helder
