Subject: Shear transformation with Poly_2d
Posted by Helder Marchetto on Mon, 18 Jun 2012 15:15:48 GMT
View Forum Message <> Reply to Message

Hi,
I'm not going to post a question, rather a solution... The reason is that I was fighting with this a few hours and I thought it would be nice to show a solution to anybody who might care about this.
I was a bit annoyed that of the affine transformations (scaling, rotation, translation and shear) one is missing... The transformations like scaling and rotation can be accessed using ROT that then calling POLY_2D that actually does the work. Translation can be done (easily) with shift (although this is only possible using integer translation, for non integer translations, the procedure described below may also be used...).
What is missing is of course shear. If you look in text books you find a matrix for shear that looks like:
ShearVertical = [[1,0,0],[Vert,1,0],[0,0,1]]
where Vert is the degree of vertical shear.
I thought it would be nicest to get this sorted by using this matrix, but I just couldn't get this to work without loops or having to rewrite basic math code... not nice and given that IDL is well suited for image processing I thought that there has to be an IDL way for this.

So here we go. POLY_2D give in the help indications on how to use it and some examples (of course not shear).
If you want to implement a shear transformation, then you would have to do the following.

```
Img = DIST(200)
WINDOW, XSIZE=405,YSIZE=200
TVSCL,Img
VertShear = 20.0
KX = [[0.0, 0.0],[1.0, 0.0]]
KY = [[VertShear, 1.0],[-VertShear/100.0, 0.0]]
TVSCL,POLY_2D(img,KX, KY,cubic=-0.5), 205, 0, /DEVICE
```

This works fine if you want to shear the image in the middle. If you want shear somewhere else, I could only manage that using POLYWARP.
In this case, you set some coefficients that act as transformation points (4 is the minimum number of points):

```
s = SIZE(Img,/DIMENSIONS)
Offset = [50,0]          ;In pixels
XI = [0, 0, s[0]-1, s[0]-1]+Offset[0]
YI = [0, s[1]-1, 0, s[1]-1]+Offset[1]
XO = XI
VertShear = 20.0
YO = YI + [-VertShear, -VertShear, VertShear, VertShear]
```

Then with POLYWARP you can retrieve the coefficients KX and KY and use them as above.

```
POLYWARP, XI, YI, XO, YO, 1, KX, KY
```

```
TVSCL,POLY_2D(Img,KX, KY,cubic=-0.5), 205, 0, /DEVICE
```

The shearing point is now moved 50 pixel to the right (positive).
There is probably an easy way to obtain these coefficients in a smooth mathematical way... Happy to hear suggestions!

Cheers and happy shearing!

Helder