Subject: Re: Accurate/fast interpolation
Posted by Mike[2] on Tue, 04 Sep 2007 14:47:20 GMT
View Forum Message <> Reply to Message

On Sep 4, 7:32 am, Steve <f...@k.e> wrote:
> What I am trying at the moment is to set one image as a common
> reference, covert all the others to sub pixel positions on that
> reference frame and then use triangulate and trigrid to interpolate
> image values onto this common reference frame. This seems to work but
> is painfully slow [trigrid is fine triangulate takes many seconds].

> I just wondered since my data is nearly on the right grid to start with
> if there were a quicker way to do this?

One way is to use interpolate afte transforming the homogeneous coordinates of the points of your image with the !p.t matrix. Suppose your image data is in an [Nx,Ny] array. You can set up the !p.t matrix with something like

```
saved_pt = !p.t
t3d, /reset
t3d, rotate=[0.0, 0.0, angle], translate=[dx, dy, 0]
matrix = !p.t
!p.t = saved_pt
```

This sets up !p.t as a transformation matrix for rotation by angle around the z-axis (or is that -angle?) with translations of dx and dy along x and y (again, I may be dropping a sign here). Your array of homogeneous coordinates will be [Nx*Ny,4], call it p0. Then you can transform those points with the !p.t matrix:

```
p1 = p0 # matrix
```

and use the new points (in the rotated and translated coordinate system) to interpolate the image like this:

```
new_image = reform(interpolate(image, p1[*, 0], p1[*, 1], p1[*, 2]),
Nx, Ny)
```

I have never timed this against triangulate and trigrid, but I suspect it will be faster.

Mike