Subject: Re: 3D registration
Posted by Mike[2] on Fri, 30 Mar 2007 14:46:46 GMT
View Forum Message <> Reply to Message

On Mar 29, 7:44 am, "Bita" <rahm...@sbox.tugraz.at> wrote:
> Hi,
> I am looking for examples of the implementation for 3D registration by
> Roger Woods.
> My problem: how should I start the iteration and how can I calculate
> the next iteration step.

I've done woods-like registration, as well as mutual information based registration with IDL. To drive the fitting process I use tnmin (http://cow.physics.wisc.edu/~craigm/idl/fitting.html). I calculate the objective functions using my own volume-to-volume interpolation routines which are essentially just wrappers using t3d and IDL's interpolation routines. This is simple in IDL, but not particularly fast. Extracting image values at common points either takes lots of memory (N points by (x,y,z,c) homogeneous coordinates) or lots of CPU (which is still slow because of the need to loop within IDL). I've settled on doing it slice by slice, which allows me to run in less than many Gbytes of RAM, but it is a bit slow because of looping over slices.

I've noticed a couple of things - the first is independent of whether the algorithm is implemented in IDL or C or whatever. That is that 3D registration requires a pretty good starting point. In practice I have always had to do an approximate hand registration to get started. The Woods algorithm is also quite sensitive to thresholds, depending on the relative noise level in the image.

Second, I've found that tnmin seems to easily get good results for translation parameters, but the MI algorithm that I'm using seems to be rather insensitive to rotations. I don't know why that is, but I've been looking at it this week, so I thought I'd mention it. It may be a simple scaling issue, but I suspect it has more to do with choice of origin and voxel size. I have not investigated this with woods, nor have I implemented Woods' multiple histogram bins.

Here's how I use tnmin for this:

```
parinfo = replicate( { value:0.D, $
                       fixed: 0, $
                       limited: [0,0], $
                       limits: [0.D, 0.D], $
                       step: 0.0D, $
                       mpside: 2 }, $
                     9 )
```

```
;; Specify initial values from manual registration:
parinfo[*].value = [dx, dy, dz, xrot, yrot, zrot, xscale, yscale,
zscale]

result = tnmin('obj_fnc', parinfo[*].value, $
            functargs={img1:img1, img2:img2}, $
            iterargs={iva:iva}, $
            parinfo=parinfo, $
            iterproc='obj_iterproc', $
            /maximize, $
            autoderivative=1)
```

Here, img1 and img2 are image objects that I pass in that handle the interpolation and obj_fnc is the routine that calculates the objective function that I'm maximizing (or minimizing as the case may be).

Mike
