Example:Astrometry

Regarding to the reduction of astronomical images, ``astrometry *might mean slightly different things, depending on the context. The default interpretation of the term `astrometry* is to obtain the celestial coordinates (α: right ascension, δ:declination) of an object by analyzing the images. The most simple way to do this is to fit the (x,y) pixel coordinates of the centroid of the given object (star, minor planet, ...), and then do an (x,y) ↔ (α,δ) transformation. In the practice of the **FITSH** package, this transformation is done in two steps.

First, the celestial coordinates (α,δ) are converted to projected coordinates. For such coordinates, we use the notations (ξ,η) and define them as follows: at a certain pre-defined (α0,δ0) central coordinate, (ξ,η) = (0,0); otherwise these coordinates represent the projection of celestial coordinates on the tangential plane at (α0,δ0). The projection can be any of the normal spherical projection methods (e.g. orthographic, arc or gnomonic), and the projection scale can also be chosen to be arbitrary, however, both the projection type and the projection scale must be fixed during the astrometric process. For practical purposes, arc projection and degree-scaling is a good choice. For instance, in this case a field-of-view with the size of 1.3 × 1.3 degrees are mapped to the (ξ,η) = [-0.65,0.65] × [-0.65,0.65] domain.

Second, these projected coordinates are mapped to the pixel grid of the astronomical image by employing a polynomial transformation to the (ξ,η) coordinates. For smaller field-of-views (size smaller than a few arc minutes), linear mapping is adequate, and for larger ones, higher polynomial orders are required. Of course, the actual polynomial order depends on the pixel-level uncertainties of the centroid coordinates, the size of the detector itself and the optical distortions (caused by larger FOVs).

Astrometry of Varuna

Now we demonstrate how the astrometry of a target object can be done using the tasks of the **FITSH** package. This example relates to the example showing the optical photometry of Varuna.

First, we obtain the projected catalogue centered on the observed celestial field. We fetch the USNO-B1.0 catalogue from the internet, using the vizquery task of the cdsclient package. The output of the vizquery call is filtered somehow (some unnecessary headers and lines are removed, see the `egrep` and `tail` invocations). And then, the task grtrans is fed (with the output of vizquery) in WCS mode, that is intend to convert the celestial coordinates (α,δ) into projected coordinates (ξ,η), where the center coordinates are (α0,δ0) = (115.591,+25.863). The resulted projected catalogue is stored in the file `varuna.cat`.

```
#!/bin/bash

RA=115.591
DEC=+25.863

MAG=16
SIZE=1.2

SOURCE=I/284/out

 (     echo    "-source=$SOURCE"
       echo    "-out=USNO-B1.0,RAJ2000,DEJ2000,B1mag,R1mag"
       test -n "$MAG" && echo "Fmag=<$MAG"
       echo    "-c=$RA $DEC"
       echo    "-c.bd=$SIZE"
```

```
 ) | \
vizquery -mime=tsv -site=cds | \
egrep -v '^#|^[[:blank:]]*$' | \
tail -n +4 | \
grtrans \
        - --col-radec 2,3 --output - \
        --wcs arc,degrees,ra=$RA,dec=$DEC \
      --output varuna.cat
```

Having the projected catalogue, and the actual list of stellar profile pixel coordinates, the (ξ,η) ↔ (x,y) mapping can be obtained by the task grmatch (see also the related example):

```
#!/bin/bash
[...]
        grmatch --reference varuna.cat --col-ref 2,3 --col-ref-ordering -5 \
                --input $ASTR/$base.stars --col-inp 2,3 --col-inp-ordering -8 \
                --max-distance 1 \

                --order 3 --triangulation auto,unitarity=0.01,maxref=1000,maxinp=1000 \
                --weight reference,column=5,magnitude,power=2 \
                --comment --output-transformation $ASTR/$base.atrans
[...]
```

This match yields the transformation file `$ASTR/$base.atrans` which contains the coefficients of the (ξ,η) ↔ (x,y) transformation (that is, a third order polynomial transformation, see the command line switch `--order 3`).

Having both the coefficients for the projected ↔ pixel mapping (in the file `$ASTR/$base.atrans`) and the WCS information (center coordinates, projection type and projection scale), we can use two subsequent calls of the task grtrans to perform the (x,y) → (ξ,η) and the (ξ,η) → (α,δ) transformations:

```
#!/bin/bash

RA=115.591
DEC=+25.863

[...]

echo $x $y | \
grtrans \
        - --col-xy 1,2 --input-transformation $ASTR/$base.atrans --reverse --output - | \
grtrans \
        - --col-pixel 1,2 --output - \
        --wcs arc,degrees,ra=$RA,dec=$DEC
```

Thus, this script right above computes directly the celestial coordinates of the pixel coordinate ($x,$y) and prints it to the standard output. Like in the case of vizquery (and the expected input of grtrans in WCS mode), the celestial coordinates are represented in fractional degree format. This is especially important for right ascension, since traditionally this coordinate is measured in hours (1 hour is 15 degrees). The output of the second grtrans might be fed to a simple awk script that pretty-prints the celestial coordinates (hours, minutes, seconds for right ascension and degrees, arcminutes and arcseconds for declination):

```
#!/bin/bash [...] [...] | \ grtrans \ - --col-pixel 1,2 --output - \ --wcs arc,degrees,ra=$RA,dec=$DEC | \ awk \ '{ r=$1/15; h=int(r);r=(r-h)*60; m=int(r);r=(r-m)*60; s=int(r);r=(r-s)*100; f=int(r);if ( f<0 ) f=0; else if ( f>=100 ) f=99; printf("%.2d:%.2d:%.2d.%.2d ",h,m,s,f); r=$2; if ( r<0 ) { sc="-";r=-r; } else sc="+"; h=int(r);r=(r-h)*60; m=int(r);r=(r-m)*60; s=int(r);r=(r-s)*10; f=int(r);if ( f<0 ) f=0; else if ( f>=10 ) f=9; printf("%s%.2d:%.2d:%.2d.%.1d",sc,h,m,s,f); printf("\n"); }'
```