Installation (windows)

Download and install python – version 2.3 – from www.python.org
Download and install the Numeric extensions – from numpy.sourceforge.net (version >23, be sure to get Numeric, not numarray)
Download and install the matplotlib plotting package from matplotlib.sourceforge.net
Download and install the opengl extensions pyopengl.sourceforge.net

Finally – get ImageD11 from:

After it is installed, you should find a script "ImageD11_gui.py", probably located in c:\python23\scripts\

Double click it to start the application (click OK for the message which appears at the start)

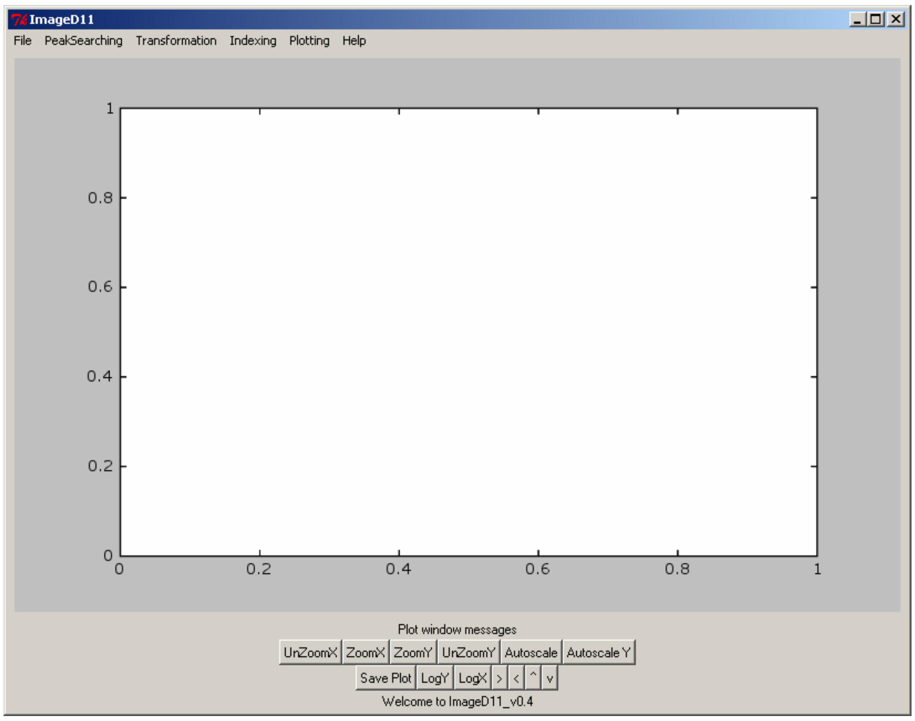
Short summary:

1: Peaksearch a rotation series of images via the script peaksearch.py in c:\python23\scripts to make a .pks file
2: Read this into the gui on the peaksearch menu ("Read pks file")
3: Sequentially click on each menu item in the peaksearch menu till you get to the bottom and save a ".flt" file.
4: Read this .flt file in from the Load filtered peaks item in the "Transformation" menu
5: Either load or edit the transformation parameters to be approximately correct
6: Choose the "fit" menu item in transformation to optimize tilts/distance/wavelength
7: compute g-vectors from the "transformation" menu after the parameters are optimal
8: Save the g-vectors, probably into a ".gv" file
9: Load the g-vectors in the indexing menu (load g-vectors)
10: Inspect them via a 3D plot if you like (Indexing plot x/y/z)
11: Assign the g-vectors to "powder rings" via the indexing menu
12: Check the output in the black window to see if you like the peak assignments
13: Check the parameters being used for indexing (indexing -> edit parameters)
14: Generate trial orientations via the indexing menu
15: Score the trial orientations via the indexing menu (might take some time if there are lots of peaks)
16: Try to reassign the g-vectors to rings and generate more trial orientations with different indexing parameters if you have lots of peaks unindexed
17: Write out the orientations found via "Indexing -> write indexed peaks", might take some time if there are lots of peaks
18: … interpret your results …

And now in agonizing detail:

Worked example for ImageD11
Essentially you should just work your way down each of the 3 menus in turn…

Once the gui has started up you should see something like the following (after clicking OK for the message):

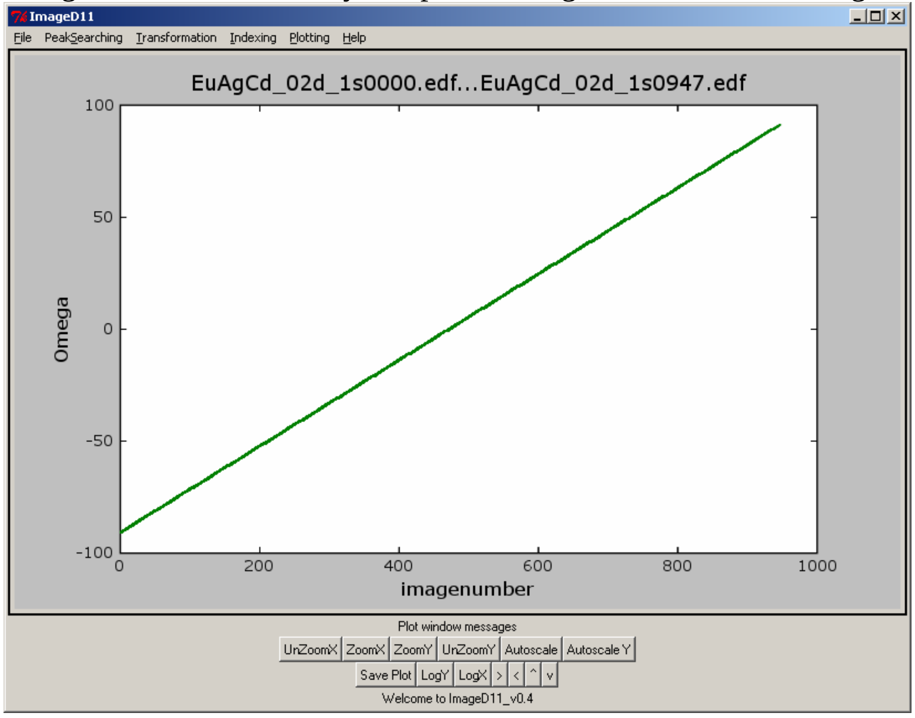

ImageD11
File PeakSearching Transformation Indexing Plotting Help
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Plot window messages
UnZoomX ZoomX ZoomY UnZoomY Autoscale Autoscale Y
Save Plot LogY LogX > < ^ v

Welcome to ImageD11_v0.4

Example 1: eu.pks

The file eu.pks contains peaks from a single crystal, so there is only one grain to index. Load this file in via the “Read pks file” menu item highlighted above.

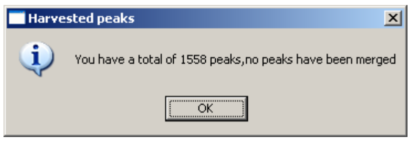
The gui should then show you a plot of image number versus omega angle:


ImageD11
File PeakSearching Transformation Indexing Plotting Help
EuAgCd_02d_1s0000.edf...EuAgCd_02d_1s0947.edf
Omega
100
50
0
-50
-100
0
200
400
600
800
1000
imagenumber
Plot window messages
UnZoomX ZoomX ZoomY UnZoomY Autoscale Autoscale Y
Save Plot LogY LogX > < ^ v

Welcome to ImageD11_v0.4

In this case we see the data were collected as a single scan of the omega (=samome) angle. If there were several scans then you select the range of images you want to process by drawing a box with the mouse.

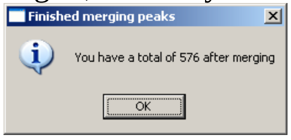
Once the range is selected, select “Harvest peaks” from the peaksearching menu and OK the message (after reading it)


Harvested peaks

You have a total of 1558 peaks,no peaks have been merged

The program has now placed the peaks corresponding to the scan range you wanted into memory. It tells you how many peaks you have.

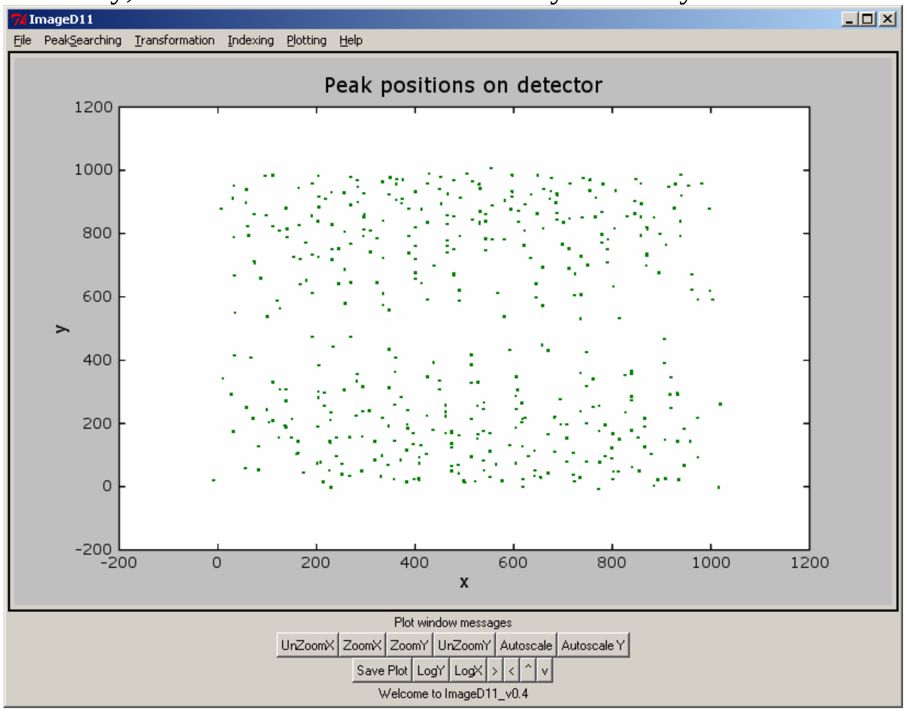
Now select “Merge peaks” and the program will combine together peaks with adjacent omega angles, and tell you how many peaks are left.


Finished merging peaks

You have a total of 576 after merging

Next select “filter peaks” and the program will plot out the final peaks found in terms of their x and y co-ordinates.

Normally, x is the “fast” index in the binary file and y is the slow index.



Unfortunately no filtering operations are currently implemented, but you might imagine that you want to throw out certain peaks at this stage (ones at the edge of the detector, or depending on peak intensity or shape). Please feel free to make the program do this.

Finally, you need to save these peaks so save doing all this again – conventionally the file extension is “.flt” so you know they are filtered peaks, but you can use anything you like. Save them via the “peaksearching -> save good peaks” menu item.

The file looks like this inside:

```
# xc yc omega
527.216740 166.256114 -90.400000
100.510514 537.215449 -89.921794
```

```
870.462833 819.030435 -89.800000
870.515946 819.223412 -89.600000
934.018285 21.624610 -89.000000
399.926021 760.183138 -88.400000
399.903309 760.197801 -88.200000
```
[etc]

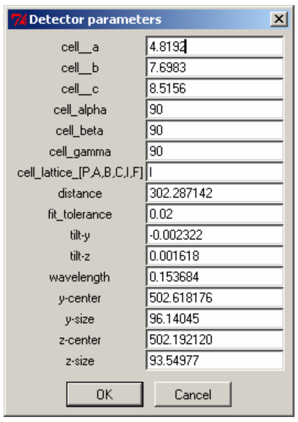
These are the x,y,omega positions of the peaks found, dumped into ascii format. All other information is lost for now (sorry).

Now you are ready to try to transform the peaks into reciprocal space. Do this by working through the “transformation menu”.

First “load filtered” peaks from the transformation menu.

Then “load parameters” from the file eu2.pars (supplied)

Check/edit the parameters via “edit parameters”:

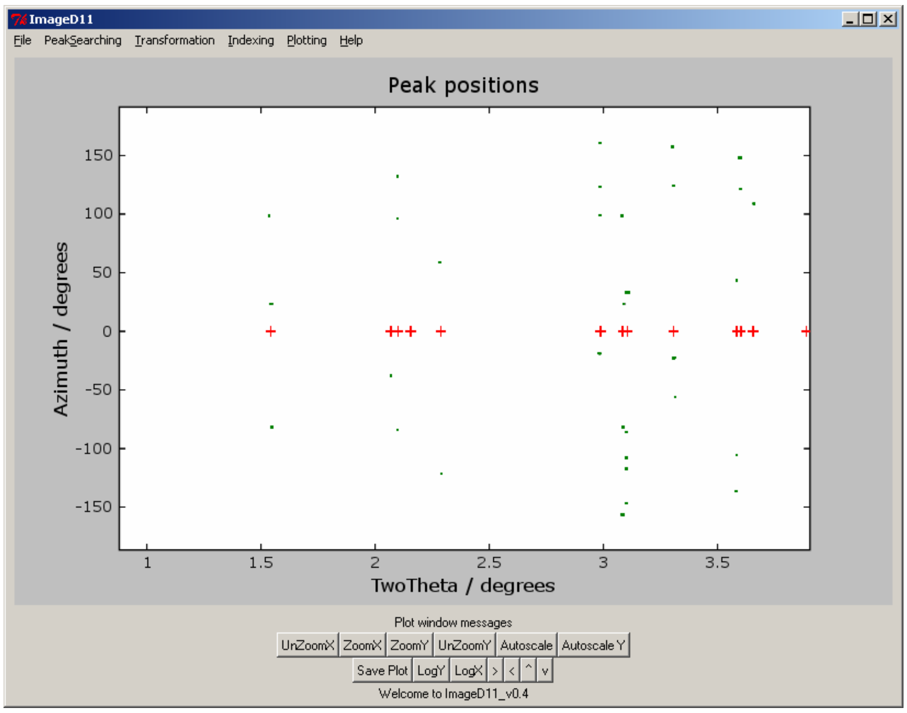

Detector parameters

cell__a	4.8192
cell__b	7.6983
cell__c	8.5156
cell_alpha	90
cell_beta	90
cell_gamma	90
cell_lattice_[P,A,B,C,I,F]	I
distance	302.287142
fit_tolerance	0.02
tilt-y	-0.002322
tilt-z	0.001618
wavelength	0.153684
y-center	502.618176
y-size	96.14045
z-center	502.192120
z-size	93.54977

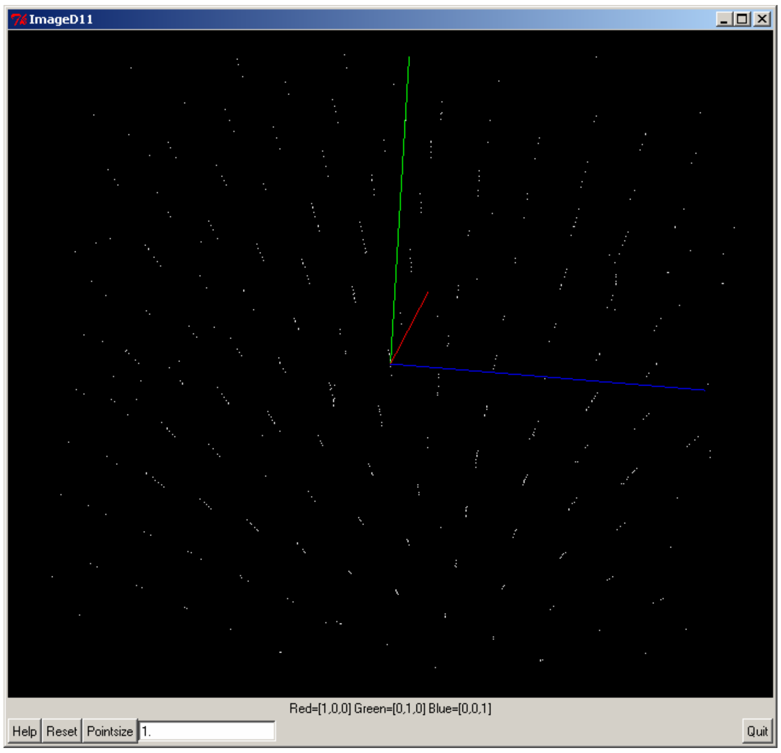
These are pretty good for this data. Now, to check what is happening go to “plotting -> clear plot” and then transformation “plot tth-eta” to see the peak positions transformed into two theta (Bragg angle) and eta (angle around the azimuth of the powder ring). (see next page).

If all is OK then the peak positions compute (marked by crosses) will line up with the data in two theta.

Now compute the g-vectors (transformation menu, compute g-vectors) and then save them (transformation menu, save g-vectors). You might want also to save the parameters you used for the transformation…



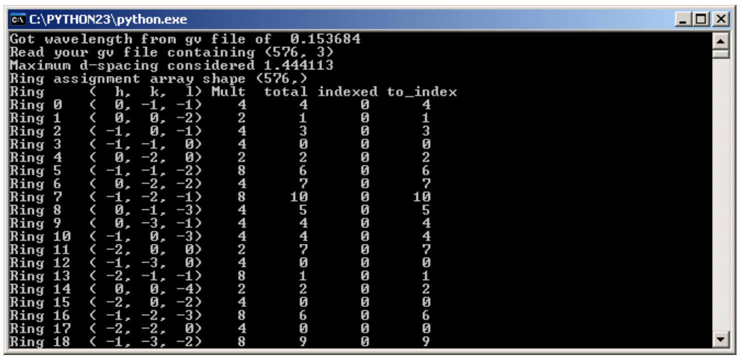
Now start on the indexing menu.
Load the g-vectors in.
Optionally plot them in 3D to see if you like them (indexing ->plot xyz):



These ones look good (obviously a 3D lattice which will be indexing by one unit cell).

The .gv file contained the unit cell and wavelength as well as the peak positions.

Now assign the peaks to powder rings: “indexing ->assign to powder rings”

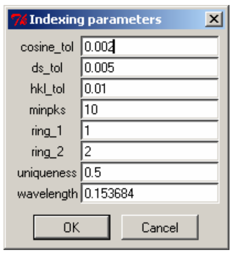


```
Got wavelength from gv file of  0.153684
Read your gv file containing (576, 3)
Maximum d-spacing considered 1.444113
Ring assignment array shape (576,)
Ring      (  h,  k,  l) Mult  total indexed to_index
Ring 0    (  0, -1, -1)    4      4       0      4
Ring 1    (  0,  0, -2)    2      1       0      1
Ring 2    ( -1,  0, -1)    4      3       0      3
Ring 3    ( -1, -1,  0)    4      0       0      0
Ring 4    (  0, -2,  0)    2      2       0      2
Ring 5    ( -1, -1, -2)    8      6       0      6
Ring 6    (  0, -2, -2)    4      7       0      7
Ring 7    ( -1, -2, -1)    8     10       0     10
Ring 8    (  0, -1, -3)    4      5       0      5
Ring 9    (  0, -3, -1)    4      4       0      4
Ring 10   ( -1,  0, -3)    4      4       0      4
Ring 11   ( -2,  0,  0)    2      7       0      7
Ring 12   ( -1, -3,  0)    4      0       0      0
Ring 13   ( -2, -1, -1)    8      1       0      1
Ring 14   (  0,  0, -4)    2      2       0      2
Ring 15   ( -2,  0, -2)    4      0       0      0
Ring 16   ( -1, -2, -3)    8      6       0      6
Ring 17   ( -2, -2,  0)    4      0       0      0
Ring 18   ( -1, -3, -2)    8      9       0      9
```

The black window which came up with the gui should show output like that above. You might need to click in the top left corner, select “properties”, then the layout tab, and then make the “screen buffer” larger, so you can scroll back up this screen.

The program is telling you what the powder rings it finds are for the unit cell, what is the multiplicity of the ring, how many g-vectors were assigned to that ring, and how it is going for indexing the peaks.

Now go to “edit parameters” in the indexing menu



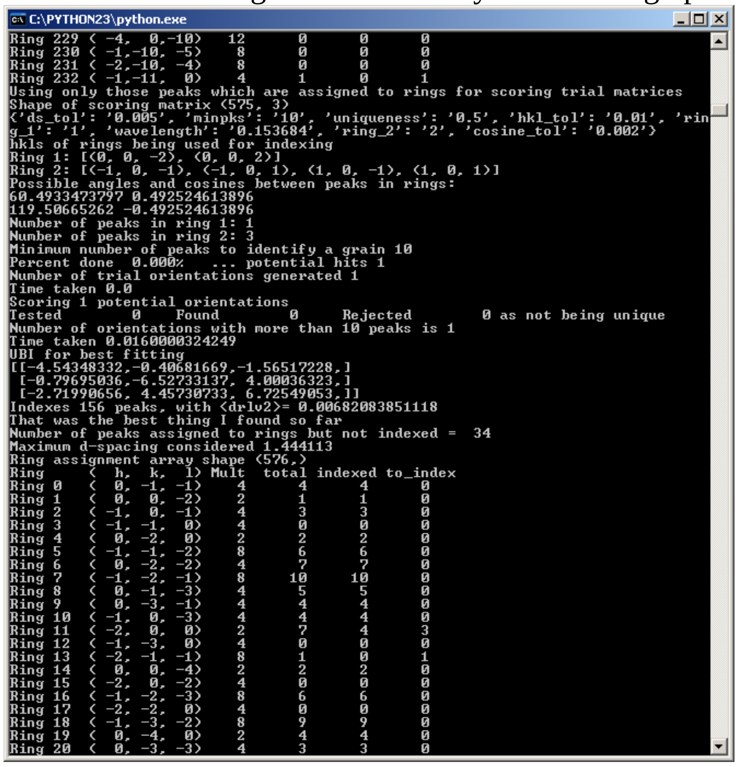
cosine_tol	0.002
ds_tol	0.005
hkl_tol	0.01
minpks	10
ring_1	1
ring_2	2
uniqueness	0.5
wavelength	0.153684

Translation:
cosine_tol = used to decide if two observed peaks that are assigned to powder rings are related to each other. For example, a pair (100),(010) peaks have a 90 degree angle between them. If the cosine of the angle observed between two peaks is within cosine_tol of the value predicted from the unit cell, then a trial orientation matrix can be generated.
ds_tol = Tolerance in d* (=1/d in angstroms) for assigning peaks to powder rings
hkl_tol = Tolerance of hkl being integers to make a peak belong to a grain (eg, compute the hkl of a peak from the orientation matrix, if they come out as integer, the peak is indexed)
minpks = Minimum number of peaks to keep an orientation matrix
ring_1 and **ring_2** = the powder rings to use for generating orientations. The program will pair all of the peaks in ring 1 with all of the peaks in ring 2 to generate trial orientations. Runtime depends on which are chosen.
uniqueness = fraction of peaks a new grain indexes to be accepted. If an orientation matrix indexes n peaks and m of these are previously indexed, then the orientation is accepted if (n-m)/n > uniqueness
wavelength = the wavelength used in the experiment. Used to generate angles for calculated peaks (two theta and eta).

For this example, the defaults ought to be OK.

Now select indexing -> generate trial orientations
Then indexing -> score trial orientations

It should index the grain. The check you can “assign peaks to powder rings” again:



```
Ring 229 ( -4,  0,-10)   12       0       0       0
Ring 230 ( -1,-10, -5)    8       0       0       0
Ring 231 ( -2,-10, -4)    8       0       0       0
Ring 232 ( -1,-11,  0)    4       1       0       1
Using only those peaks which are assigned to rings for scoring trial matrices
Shape of scoring matrix (575, 3)
{'ds_tol': '0.005', 'minpks': '10', 'uniqueness': '0.5', 'hkl_tol': '0.01', 'rin
g_1': '1', 'wavelength': '0.153684', 'ring_2': '2', 'cosine_tol': '0.002'}
hkls of rings being used for indexing
Ring 1: [(0, 0, -2), (0, 0, 2)]
Ring 2: [(-1, 0, -1), (-1, 0, 1), (1, 0, -1), (1, 0, 1)]
Possible angles and cosines between peaks in rings:
60.4933473797 0.492524613896
119.50665262 -0.492524613896
Number of peaks in ring 1: 1
Number of peaks in ring 2: 3
Minimum number of peaks to identify a grain 10
Percent done  0.000%   ... potential hits 1
Number of trial orientations generated 1
Time taken 0.0
Scoring 1 potential orientations
Tested         0    Found         0     Rejected        0 as not being unique
Number of orientations with more than 10 peaks is 1
Time taken 0.0160000324249
UBI for best fitting
[[-4.54348332,-0.40681669,-1.56517228,]
 [-0.79695036,-6.52733137, 4.00036323,]
 [-2.71990656, 4.45730733, 6.72549053,]]
Indexes 156 peaks, with <drlv2>= 0.00682083851118
That was the best thing I found so far
Number of peaks assigned to rings but not indexed =  34
Maximum d-spacing considered 1.444113
Ring assignment array shape (576,)
Ring     (  h,  k,  l) Mult  total indexed to_index
Ring 0   (  0, -1, -1)    4       4       4       0
Ring 1   (  0,  0, -2)    2       1       1       0
Ring 2   ( -1,  0, -1)    4       3       3       0
Ring 3   ( -1, -1,  0)    4       0       0       0
Ring 4   (  0, -2,  0)    2       2       2       0
Ring 5   ( -1, -1, -2)    8       6       6       0
Ring 6   (  0, -2, -2)    4       7       7       0
Ring 7   ( -1, -2, -1)    8      10      10       0
Ring 8   (  0, -1, -3)    4       5       5       0
Ring 9   (  0, -3, -1)    4       4       4       0
Ring 10  ( -1,  0, -3)    4       4       4       0
Ring 11  ( -2,  0,  0)    2       7       4       3
Ring 12  ( -1, -3,  0)    4       0       0       0
Ring 13  ( -2, -1, -1)    8       1       0       1
Ring 14  (  0,  0, -4)    2       2       2       0
Ring 15  ( -2,  0, -2)    4       0       0       0
Ring 16  ( -1, -2, -3)    8       6       6       0
Ring 17  ( -2, -2,  0)    4       0       0       0
Ring 18  ( -1, -3, -2)    8       9       9       0
Ring 19  (  0, -4,  0)    2       4       4       0
Ring 20  (  0, -3, -3)    4       3       3       0
```

So now write out your results (indexing -> write out indexed peaks)
The output file should contain:

```
Grain: 0   Npeaks=103   <drlv>=0.006960
UBI:
[[-4.53719628,-0.40523341,-1.57299856,]
 [-0.80373851,-6.52817142, 4.00010064,]
 [-2.72909869, 4.45604726, 6.72392047,]]
Peak    (  h       k       l       )   drlv          Omega_obs Omega_calc   Eta_obs Eta_calc    tth_obs tth_calc
0       ( -0.0001  0.9941 -0.9973 )   0.00651868      -8.8000   -8.7645       98.3668   98.2839      1.5351    1.5420
1       ( -0.0049  0.9989  1.0023 )   0.00547953      44.2000   44.5180       23.4639   23.7875      1.5426    1.5420
2       ( -0.0046  0.9993  1.0058 )   0.00742354      44.4000   44.5180       23.4270   23.7875      1.5454    1.5420
3       (  0.0002 -1.0050  1.0036 )   0.00613971      -7.2000   -7.2062      -81.7558  -81.7161      1.5487    1.5420
4       (  0.0000  0.0000  1.9999 )   0.00011819     -29.8000  -29.7999      -37.8399  -37.8398      2.0681    2.0682
.
.
.
450     (  2.0040  3.0026 -6.9957 )   0.00646625       6.5682    6.5003      125.6953  125.7195      8.8139    8.8133
500     (  2.0030  1.9972 -7.9944 )   0.00693679       2.4000    2.3354      134.6800  134.6668      9.3349    9.3387
550     ( -0.0016 -0.0003  10.0043 )   0.00463522     -23.0000  -23.0290      -37.5411  -37.5488     10.3590   10.3545


And now listing via peaks which were assigned to rings

Peak= 0     Ring= 0     gv=[  0.0241 -0.1708 -0.0254 ]   omega=   -8.8000   eta=   98.3668   tth=    1.5351
Grain 0     (  0,  1, -1)  ( -0.0001  0.9941 -0.9973 )   omega=   -8.7645   eta=   98.2839   tth=    1.5420
.
.
.

Peak= 9     Ring= 4     gv=[  0.0266  0.2207 -0.1349 ]   omega=    8.2000   eta= -121.2544   tth=    2.2900
Grain 0     (  0, -2,  0)  (  0.0022 -2.0019  0.0041 )   omega=    8.3577   eta= -121.3129   tth=    2.2878

Peak= 10    Ring= 5     gv=[  0.1065  0.0312  0.3197 ]   omega=   78.2000   eta=  -19.0899   tth=    2.9801
Peak not assigned, closest=[ -0.9987  0.9892  1.9981 ] for grain 0
.
.
.
Peak= 575   Ring= 232   gv=[ -0.7719  0.6437  1.0369 ]   omega=  -41.0000   eta=  -43.7382   tth=   12.7423
Peak not assigned, closest=[  1.6104  0.5660  11.9474 ] for grain 0


Total number of peaks was 576
Peaks assigned to grains 103
Peaks assigned to rings but remaining unindexed 472
Peaks not assigned to rings at all 1
```

Unindexed peaks (eg: number 10) show that the hkl tolerance was too low for that peak, but for peak 575, this orientation matrix is no good (it is probably a spurious peak).