

automar

- step by step -

A very basic introductory tutorial

(Version Oct.2008)

By Dr. Klaus S. Bartels
Klaus[at]marresearch.com

marresearch

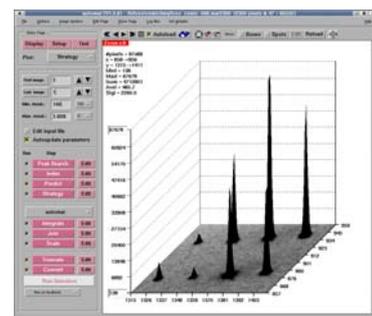
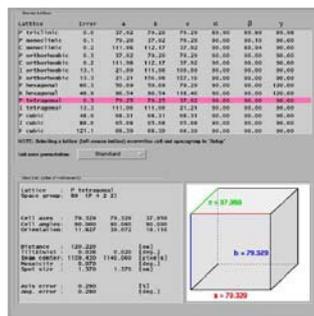
Marresearch GmbH
Hans-Böckler-Ring 17
D-22851 Norderstedt
Germany

More detailed information is available in the related set of manuals which consists of:

The *automar GUI* (graphics user interface - user's guide)

The *marIndex*-manual (how to auto-index diffraction patterns)

The *automar*-manual (how to process rotation/oscillation images)



Introduction

automar is designed to make processing of crystal diffraction patterns as easy, as quick, and as automatic as conceivable. You have to judge whether we have succeeded to your satisfaction.

Comments, criticisms, suggestions and questions are always welcome.

You are merely assumed to have very basic knowledge of computers, i.e. you ought to know:

- what directories and data files are;
- how to use keyboard and mouse.

You should be familiar with crystallography to the extent that you have an idea of

- crystal cell and lattice symmetry;
- X-ray wave-length, and possibly some other beam properties;
- what integrated reflexion intensities are good for.

You are probably in the situation that diffraction images of your crystal have been, or are being, collected (i.e. “exposed”).

If this is done on any *mar*-detector mounted on a *mar*-base (*dtb*), you need not bother about any other experimental parameters (like crystal-to-detector distance, pixel size, etc.), because these are automatically stored in, and then read from, the image file *header*.

On a synchrotron beam-line, the wave-length need be supplied.

All *mar*-programs, *automar* as well as *marView* and other utilities, can deal with images from any of the *mar*-detectors:

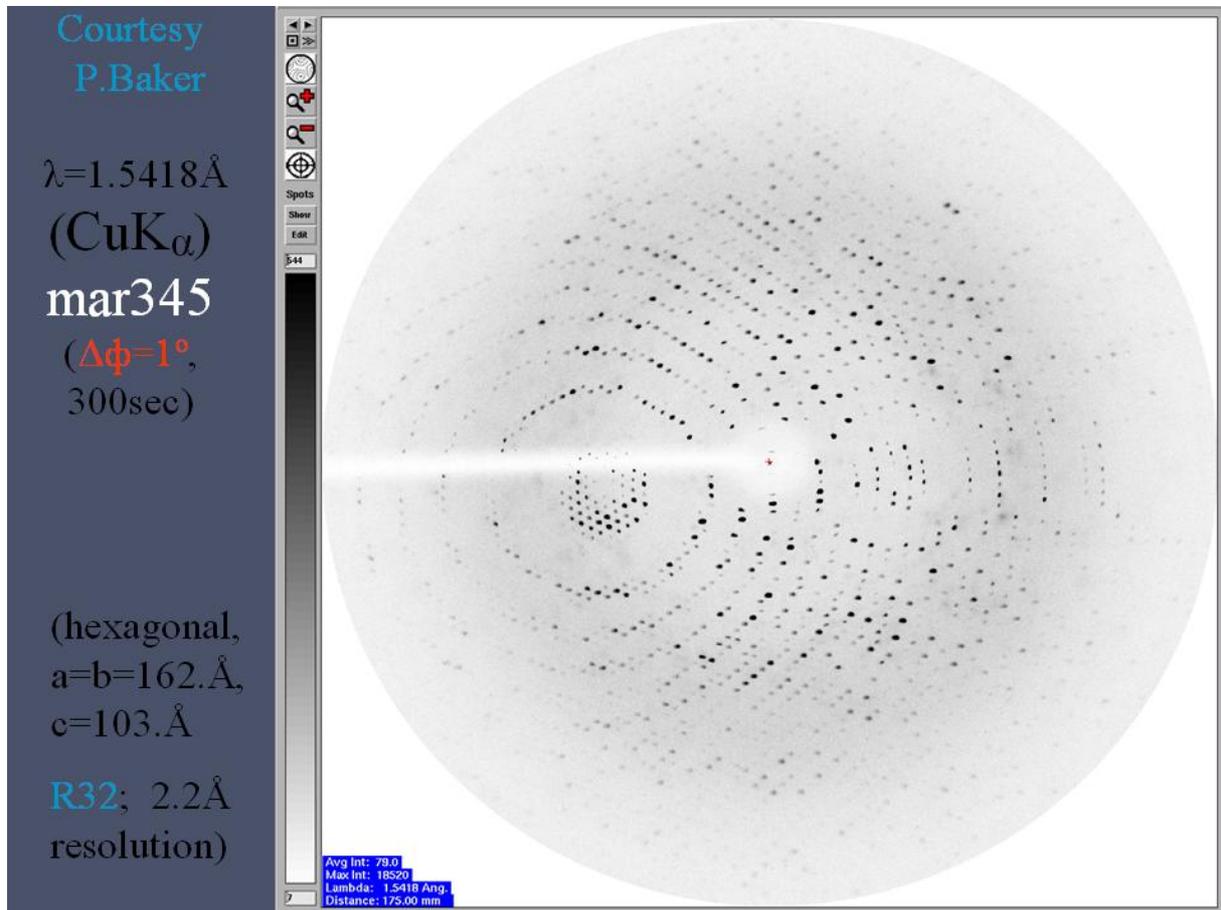
- Image-plate scanners (mar180, mar300, mar345)
- CCD (mar165, mar-mosaic 3x3 or 4x4)
- Flat Panel (mar555)

Your images are now stored on the computer in some directory, which (hopefully) is descriptive of your project.

Let's call it your “image-directory”.

The individual image files are named according to a standard name convention: “root-prefix_###_suffix”, where ### denote the running number (usually 3 or 4 digits); “_suffix” may be missing (e.g. CCD).

Probably you will have had a first look at your image(s) with *marView*, either from within the data collection program (e.g. mar345dtb, mar555dtb), or explicitly invoked:



(fig.1)

Remark: Intentionally, I have chosen this old-fashioned home-lab example, because it exhibits some features that may be helpful for an introduction. If none of these hints is needed in your case – all the better.

> *Hint 1:* Note the beam stop holder shadow in this image. Although more or less parallel to the rotation axis (PHI-axis), we will have to exclude this shadowed region from the processing, because its partly shielded spots are false information. We will see how this is done in *automar*.

Now we are set.

A few mouse clicks in the automar GUI will usually take you to the integrated and scaled data set. The main section of this tutorial takes appreciably longer to read than to execute...

The few possible failures are dealt with in the [appendices](#).

The Mouse-Click-*automar* Approach

You might start *automar* from the image-directory.

It is helpful, though, to keep the processing output files apart, instead of messing them up with the images.

So let's create a "working-directory", which may be a parallel- or a sub-directory (as seen from the image-directory), like this:

```
klaus% mkdir ../a, ... or: ./a
```

(this way you can later have several *automar* runs, using different parameter sets, in `./a1`, `./a2`, etc.). Or it may be located on a different disk (in particular when your image data reside on a CD or DVD); or even on another network-connected computer.

Remark: In the example of this tutorial, the space group was known to be R32; therefore there were two sets of images exposed, `A_###` and `B_###`, 70 images at 1.deg. each, 180.deg. apart, for complete anomalous Bijvoet pairs. Thus it was intuitive to have two working-directories: `../A` and `../B`

Make this "working-directory" your current one, and invoke *automar*:

```
klaus% cd ../B
```

```
klaus% automar
```

(this form of the command assumes that the *automar*-executable is in your *path*, otherwise you have to type it with its directory, of course).

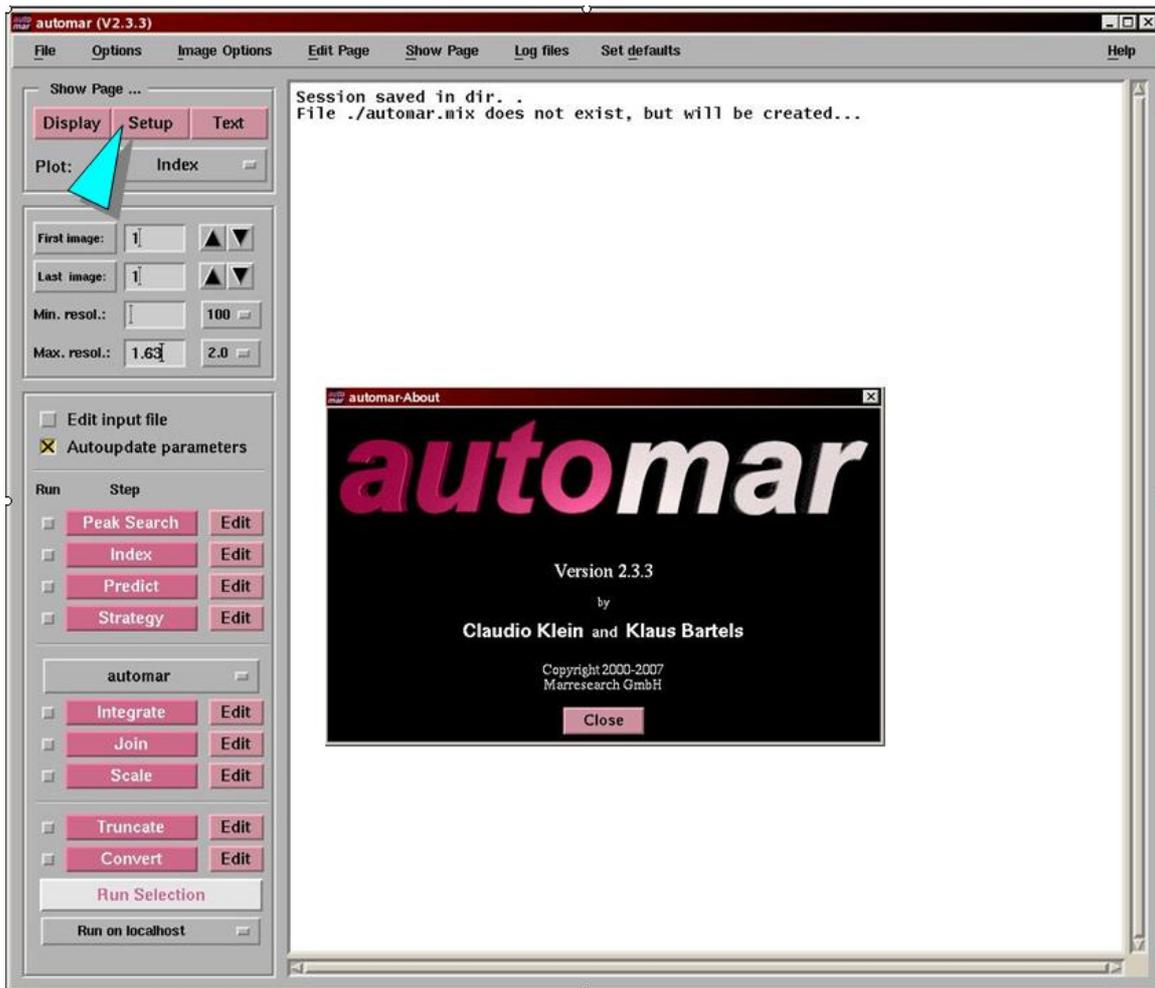
Now the *automar* window pops up – see *fig.2*; the small "about" inset window with the version number disappears after a few seconds.

The *automar* window consists of a frame with permanently available menu options, and a large square which may display a variety of contents, selectable by menu buttons of the frame:

At any time you may choose between

- image display (with/without superimposed calculated patterns),
- several parameter masks,
- graphic representation of results,
- text window with log-output,

in many cases with additional options.

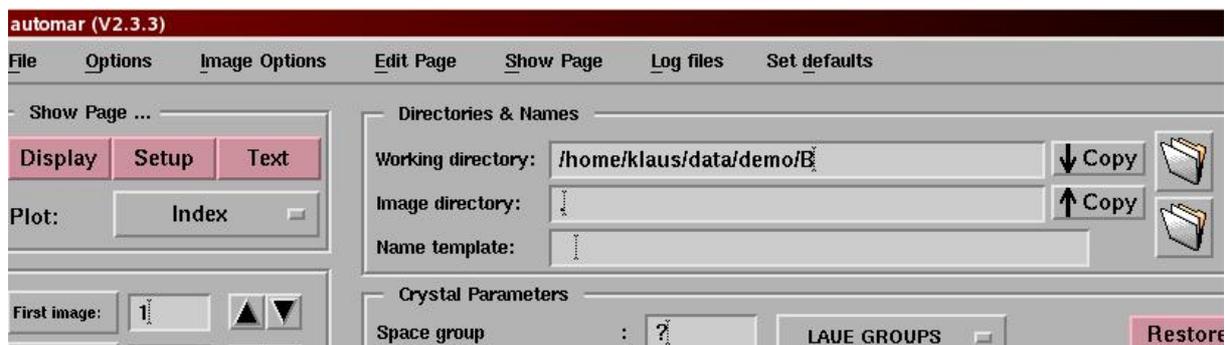


(fig.2)

(The words in this text-window differ according to the mode of *automar* installation, see [appendix 1.](#))

1st mouse-click: SETUP (see blue-ish arrow in fig.2)
 (click with mouse-button 1 = MB1)

For the time being, concentrate on the upper right part: Directories and file names; and forget about the (possibly) overwhelming number of other parameters, until we are prepared for them:



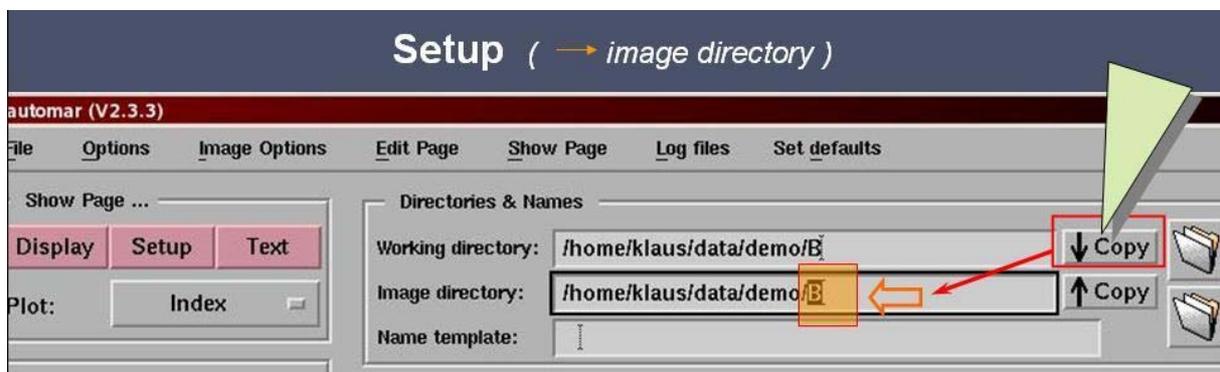
(fig.3)

The current directory (from which *automar* was started) is taken as the “working-directory” by default.

2nd mouse-click: Define the “image-directory”

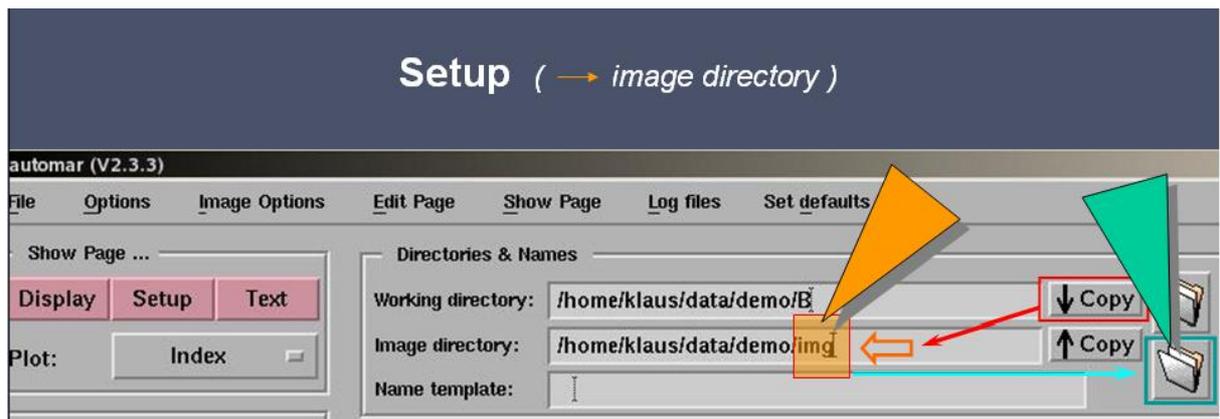
Click into the text field (2nd line) and type the directory path in full. (This applies, for instance, if your data reside on a DVD.)

If you defined the “working-directory” as above (next to the “image-directory”), there is a short-cut for the definition:



(fig.4)

The “Copy” button takes the 1st to the 2nd line; now you need only type the changes (orange arrow). *Hint*: Anywhere in the GUI, text marked (with MB1 pressed) will be replaced by any new typing.

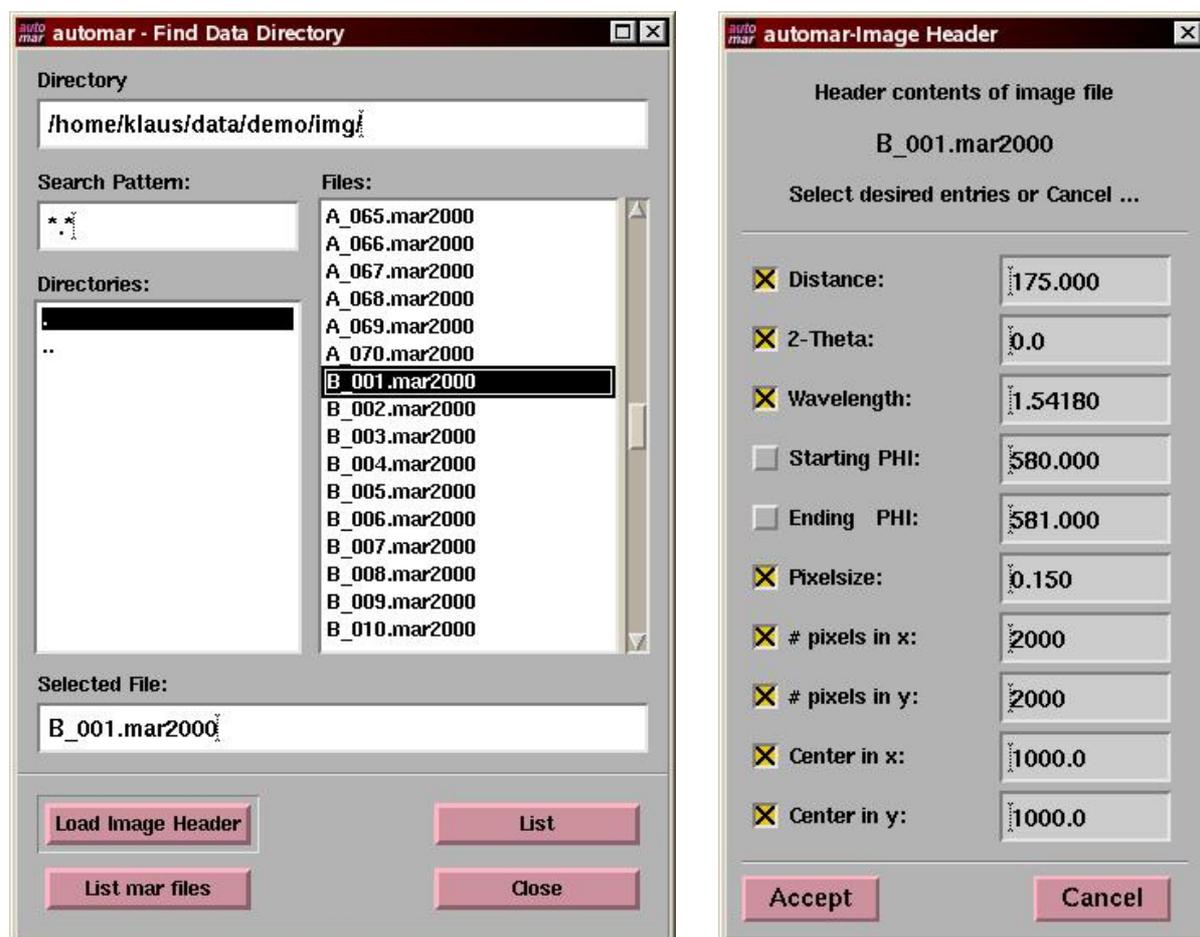


(fig.5)

3rd mouse-click: Image file name (“template”)

The ‘folder’ button to the right (see green arrow in fig.5) opens an image-directory window, with the current image directory in the top

field, its directory tree to the left, and the files contained to the right (fig.6a):



(fig.6a & b)

Click 4 on the image name you want to start with (usually the first; in this example, outlined above, we start with the first image of the second block ‘B’ of images.) The selected file name is copied to the field below. Then click on “Load image header”. (Or as a short-cut: double-click on the file name in the listing.)

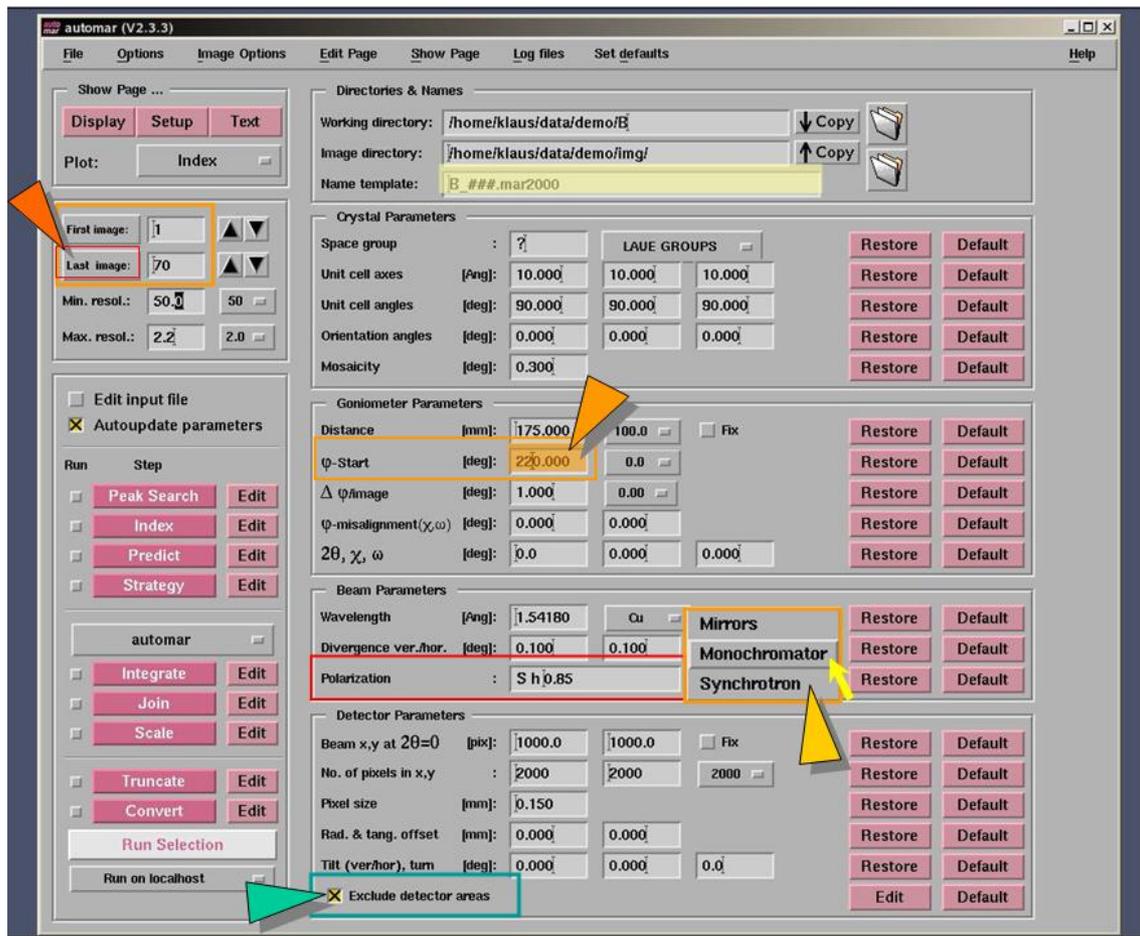
The image-header parameter window pops up, as shown in fig.6b.

In the best of all cases, clicking on “Accept” should be all that is needed.

In our example, the PHI readings are a bit awkward, so we disselect them (remove the yellow box crosses) with MB1, and input the values later.

Warning: You cannot *change* any parameters in this window.

“Accept” will now load all selected header parameters into the SETUP mask:



(fig.7)

Note that the selected image file name (fig.6a) has been converted to the image “Name template” in line 3 (below “Image directory”, light yellow shade in fig.7)

Very few numbers may need some attention at this stage:

- Upper left margin (dark orange arrow): Clicking the button for “Last image” (possibly also “First image” if needed) will automatically fill in these numbers from the directory listing.
- (!) Please check also the resolution limits just below; since these numbers, due to a still unresolved program bug, are not always correct, or even un-set, in the default.
Modify (= *mark with MBI & retype*) any number if needed.
- ϕ -start & $\Delta\phi/\text{image}$ (light orange arrow): In our example, we replace the weird PHI=580.deg.-reading by 220.; instead, you might fill in some pseudo- ϕ value (e.g. 180.deg. for the B-range, if you feel that a start of $\phi=0$. for the A-range is appropriate).

- Polarization (yellow arrow): This keyword implicitly defines the beam by the type of monochromatization. The default “Mirrors” is appropriate for the majority of home lab installations. For “Synchrotron”, see the *automar-manual* for input options, and check with your beam-line representative.
- Excluded area(s): If shadows of the beam-stop holder and/or of the cooling device shield part of the diffraction pattern, the area(s) should be excluded, to avoid false spots along the edges, or inappropriate zero-intensities (usually with very low sigma giving them a strong weight).

I am working on automatic pattern recognition of shadows; whether or not such shadows shield part of the pattern will still have to be chosen (or confirmed) by the user.

For the time being, manual definition is required.

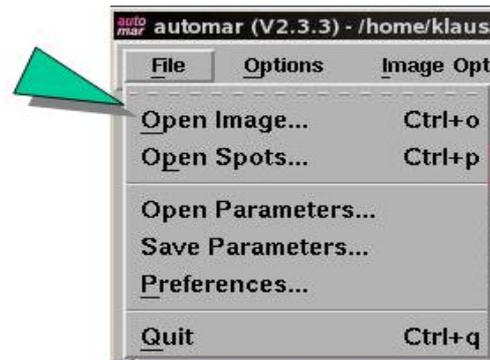
Check the yellow square (lower margin, green arrow) *and* click the “Edit” button to its right to bring up its window.

Let’s come back to the last point in a moment.

First we need the image in question in the “Display”.

Two choices:

- The long (but more versatile) way is explicitly loading a file:
Top-margin menu: ► “File” –



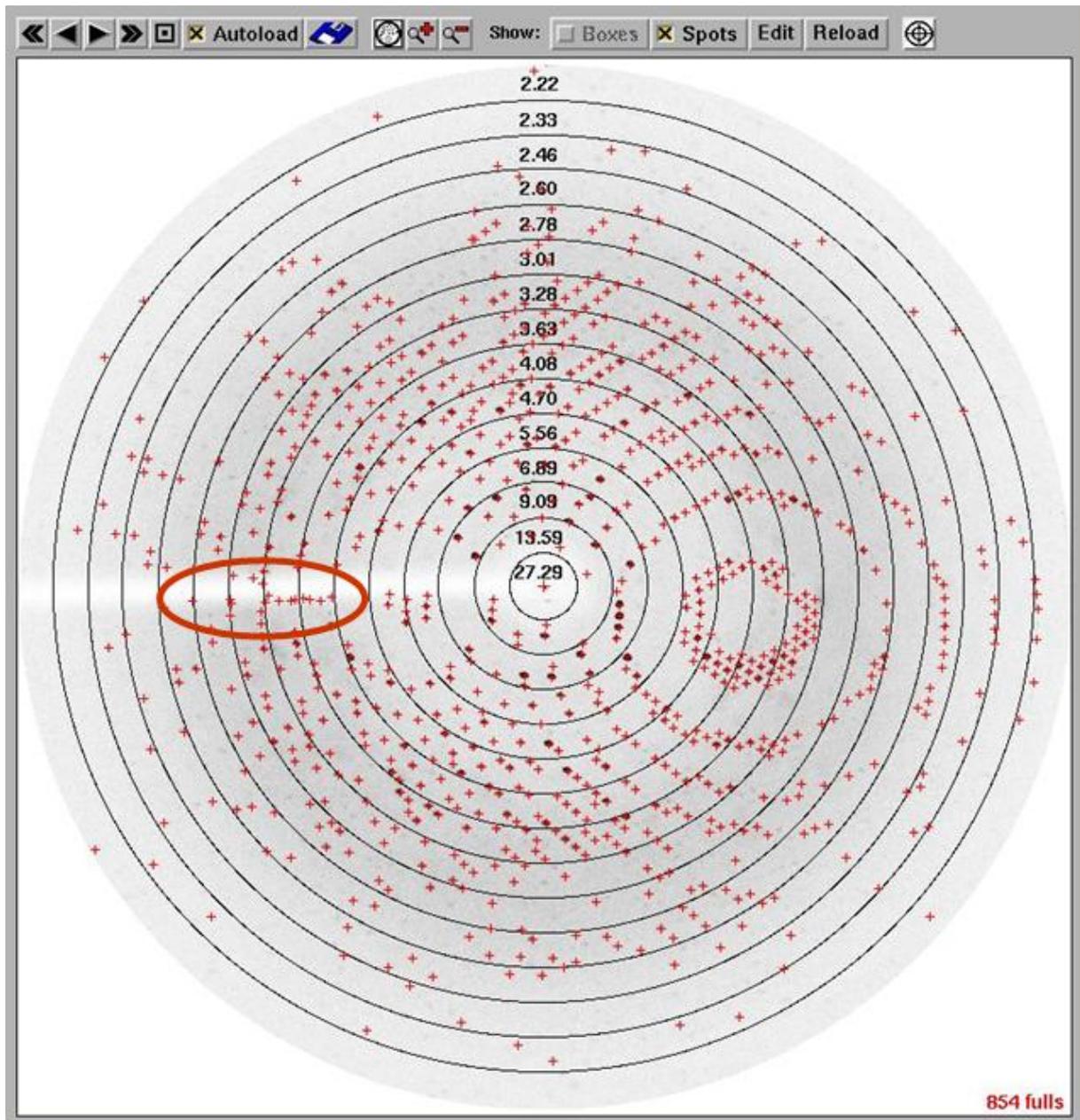
► “Open Image”

(fig.7b)

Next, a window pops up, similar to fig.6a, where you can choose any image file from any directory.

- Clicking left-margin coloured program buttons “PeakSearch” (or also “Predict” – which needs crystal parameters, though) will execute these programs on first image, automatically load the image display, and superimpose the calculated spot pattern.

Try the second mode, clicking “PeakSearch” and see what happens:



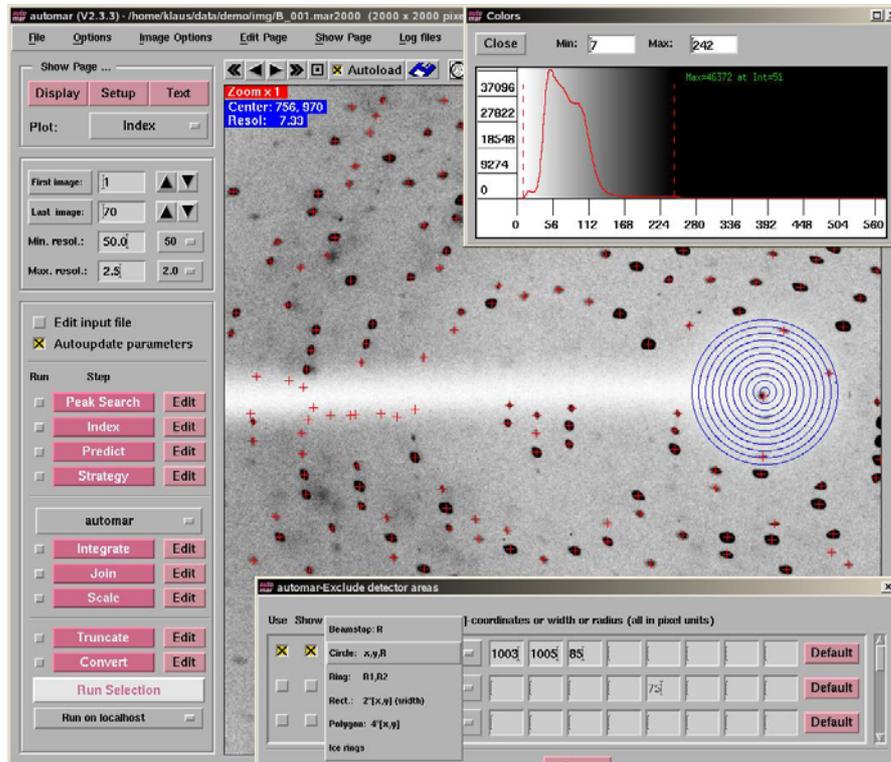
(fig.8)

In most cases, the result is reasonable, and we could go on with the [auto-indexing step](#) immediately ([see p.17](#)).

Sometimes, however, false spots, or too few spots may deteriorate the *marIndex* result, or even cause it to [fail](#). False spots may show up along edges, like those encircled in the above picture. (In fact, *these are not* detrimental.)

Definitely, no reflexions ought to be predicted and integrated in the shadow. Let's learn how to exclude such areas.

Excursion: Definition of areas to be excluded



(fig.9a)

(Short-cut: **F10** function key; or: “ShowPage” ► lowest menue entry)



(fig.9b)

If the central beam stop shadow is well centred, its exclusion can easily be achieved by appropriate definition of “Min.resol.“, otherwise it is usually a circular shadow, defined by x,y,R

This is a good point to introduce several GUI manipulation elements:

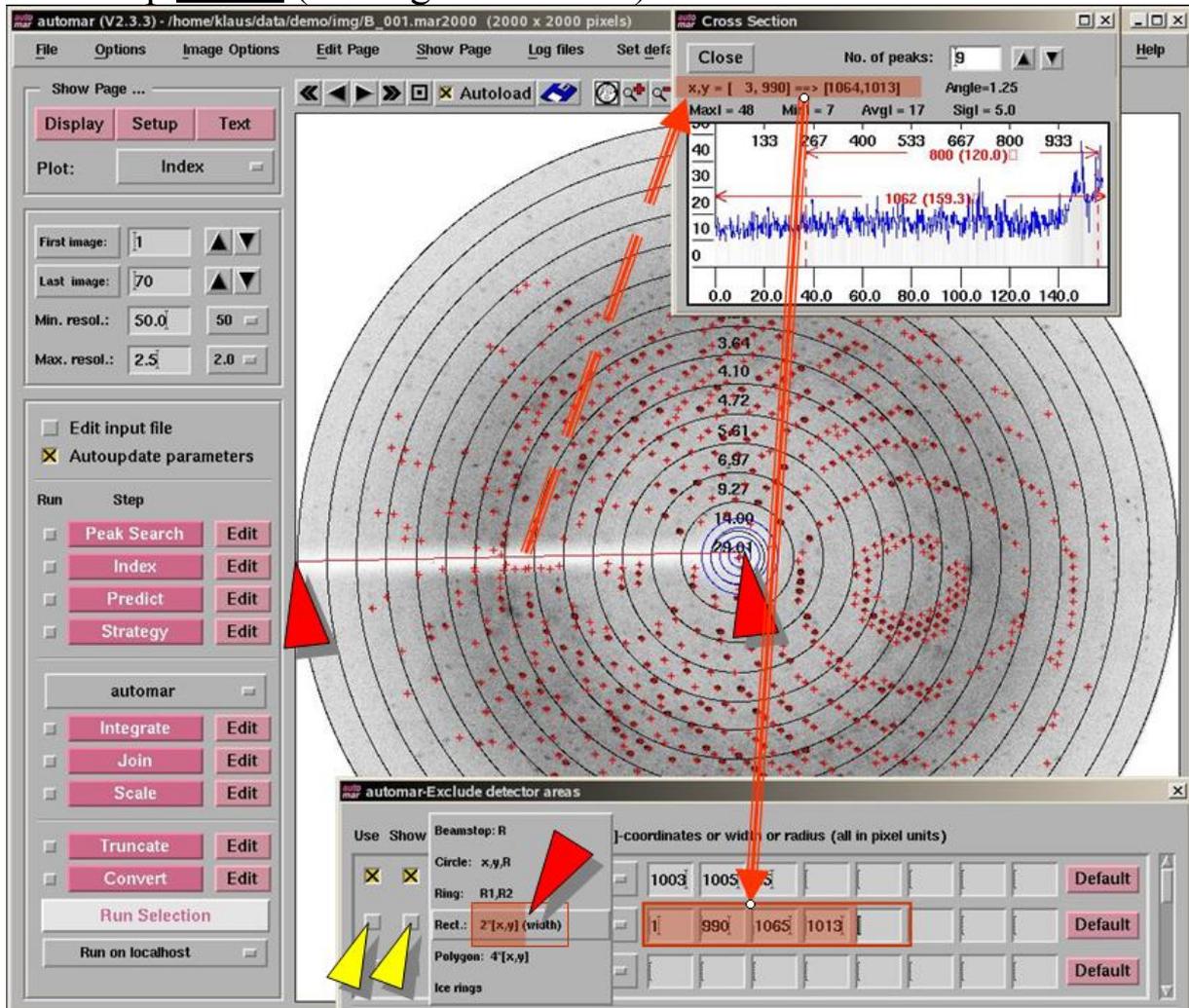
- Click middle mouse button (MB2) anywhere in the display:
 - ▶ The pixel value and its x,y coordinates are shown in the lower or upper left corner
- Draw a line with left mouse button (MB1) pressed:
 - ▶ A new window “Cross section” pops up, which shows the intensity profile along this line, along with x,y coordinates of start and end point
 - ▶ You may click into this profile with MB1 and MB3 to mark left and right limits of any sub-section; the distance between them is displayed.
 - ▶ (This window can do more: count peaks along the line and calculate lattice constants from them; but this is for later.)
- Click on  above the display to zoom in [+] / out [-], return to full [O]
- Open a ‘rubber band’ box, with MB3 (right button) pressed, to zoom in
- Click MB3 at any pixel to ‘pan’ (make this clicked pixel the new centre of the zoomed image section)
- The “ring” tool (uppermost right) displays 2 circles along with a new descriptive window to adjust and read centre and radius

- The colour scale window (top margin menu “Image Options” – “Image Colours”) lets you adjust the white and black limits of the grey scale (again with left and right mouse button) to optimize the contrast to the intensity range of interest.
Note: <Ctrl/C> as a short-cut does not work when the keypad numbers are locked (i.e. <NumLock> must be OFF)

Now we are going to use these features in the manual definition of areas to be excluded.

Refer to *fig.9a* once more: We had chosen the centre of the circular beam stop shadow by eye (MB2) and drawn a line across the shadow to read 2R from its length. “Show” (excluded area) hashes the circle.

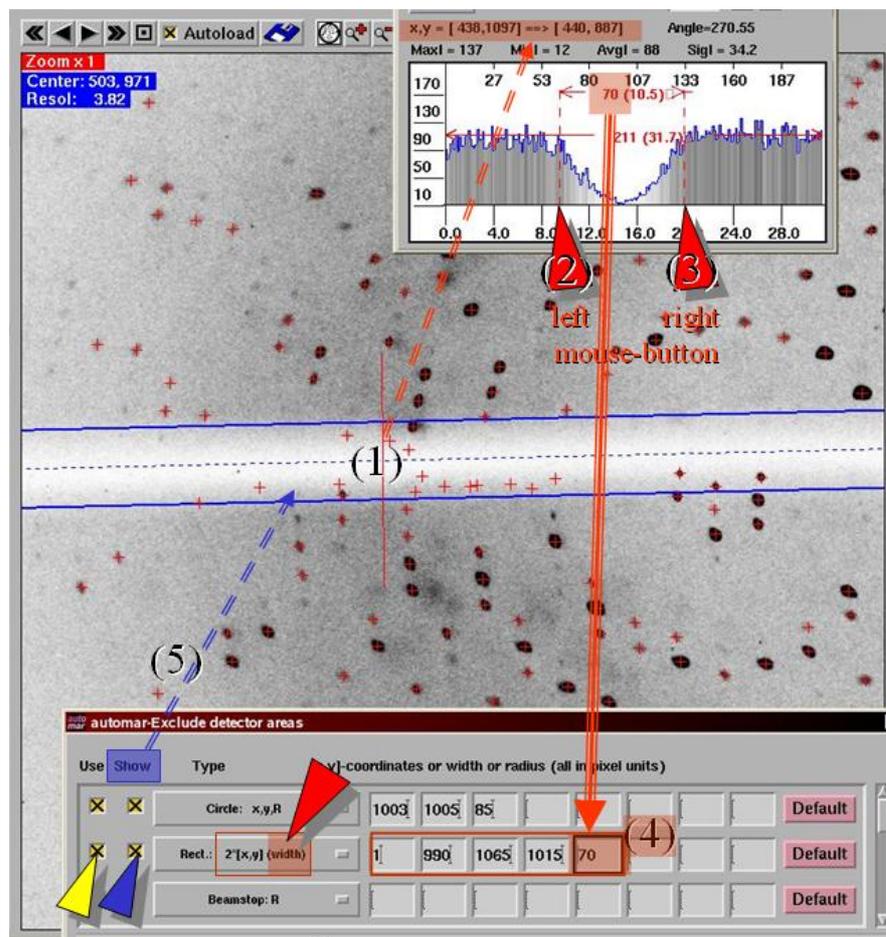
Now let’s do these operations *with illustrating figures* to define the beam stop holder (rectangular shadow):



(fig.9c)

- (1) (red arrow  in “Excluded areas” window) Click on the next (i.e. 2nd here) definition-line in the “Excluded areas” window, and select “Rect.: 2*[x,y] (width)”
- (2) To define the ‘backbone’ of the rectangular shadow, draw a line (with MB1 pressed) from the left edge to the centre.
- (3) Read x,y start (left) and end (right) from the top line in the “Cross section” window, type these values (one by one...) into the corresponding fields of the “Excluded areas” window.
- (4) The width of the rectangle is still missing. Do the following:

(For clarity, I have zoomed in, but this is not necessary.)



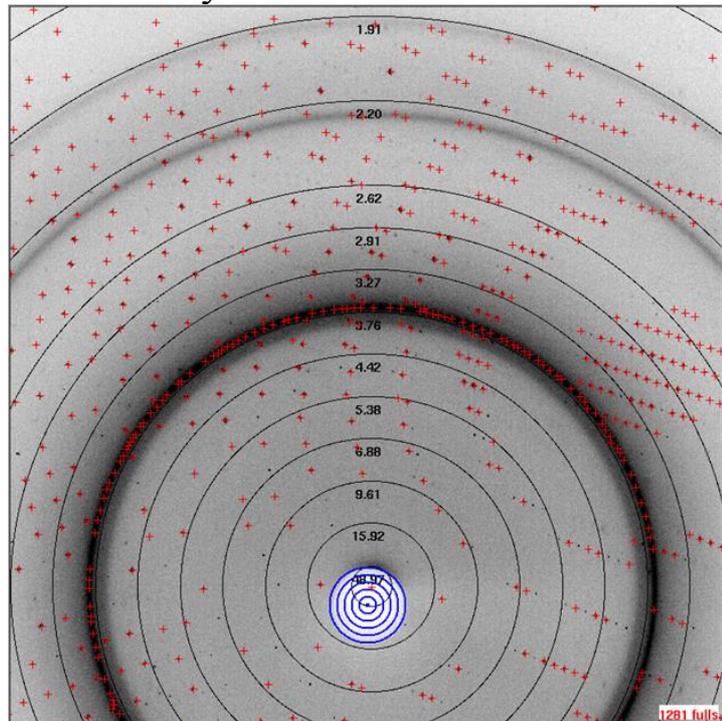
(fig.9d)

- (4₁) Draw a (short) line across the shadow; for optimal contrast this is best done in the water-ring.
- (4_{2,3}) Mark the left and right edges in the “Cross section” window (MB1 and MB3), read the width of the “dip”, and
- (4₄) type it into the 5th field of the rectangle definition.
- (4₅) Check box for “Show” (blue arrow); see whether the (blue) outlines properly enclose the shadow, otherwise modify the defining coordinates accordingly.
Check the box for “Use” (otherwise the programs will ignore this area.)

The next example will demonstrate, why this difference in “Use” and “Show” is useful.

Like shadows (and their delineating edges), superimposed powder rings (and in particular ice rings) cause trouble. Their exclusion has different aspects, though.

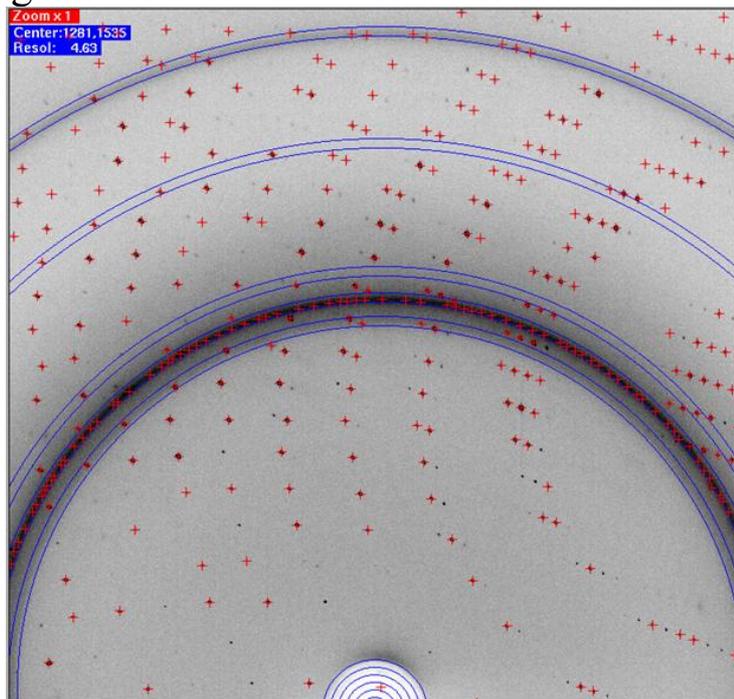
marPeaks is likely to find many false peaks on strong ice rings, this way making life for *marIndex* extremely difficult:



(fig.9e)

You might try to reduce the “Max.resol.” to, say, 4.Å – this may help in some cases, but the number of spots ‘inside’ may not be sufficient for auto-indexing.

The “Excluded areas” window has separate menu entries for general “ring” definition (which may also be useful for ring-shaped shadows), and for the (“inner”) “ice rings”.

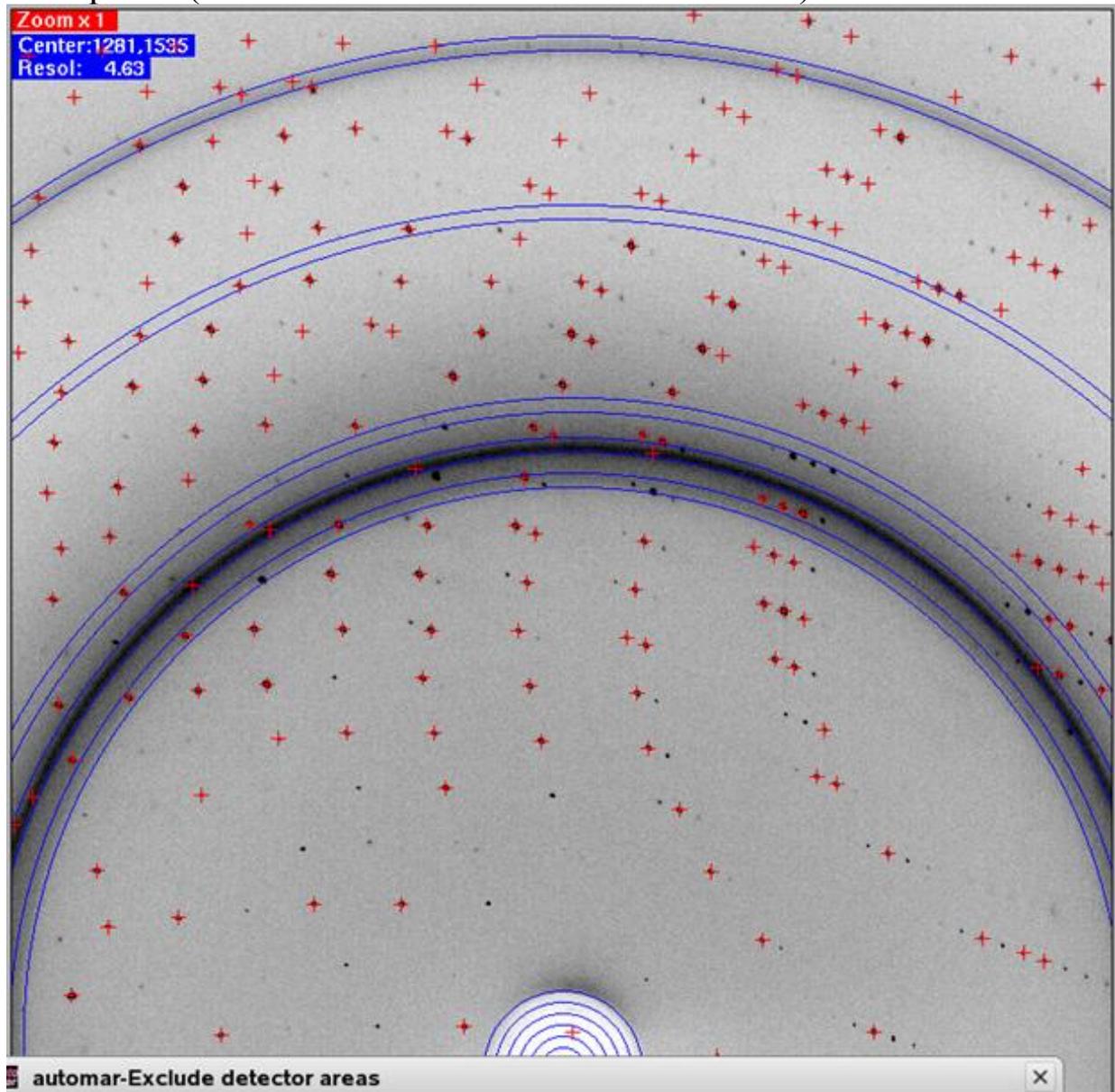


(fig.9f)

“Show” ice rings will clarify whether the rings are ice-rings indeed.

(*Additional hint*: This tool displays where ice-rings *would be expected*; this can be helpful to adjust experimental parameters, like distance, beam centre, possibly wave-length, until ice-rings existent in your image are matched.)

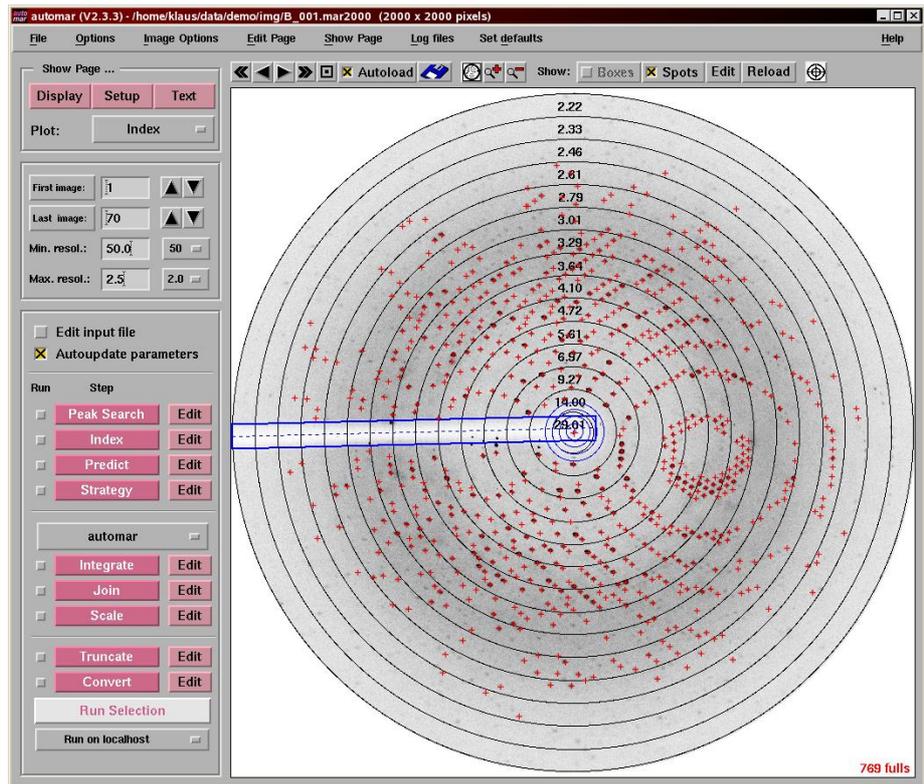
“Use” ice rings – correct experimental parameters given – will exclude false peaks (either in *marPeaks* and/or in *marIndex*):



(fig.9g)

For spot integration (*marProcess*) later on, however, it would be a loss to exclude *all* reflexions on, or very close to, such ice rings. Instead it is preferable to reject bad intensities in the scaling process.

Therefore: Do not “Use” ice rings beyond *marPeaks/marIndex*!

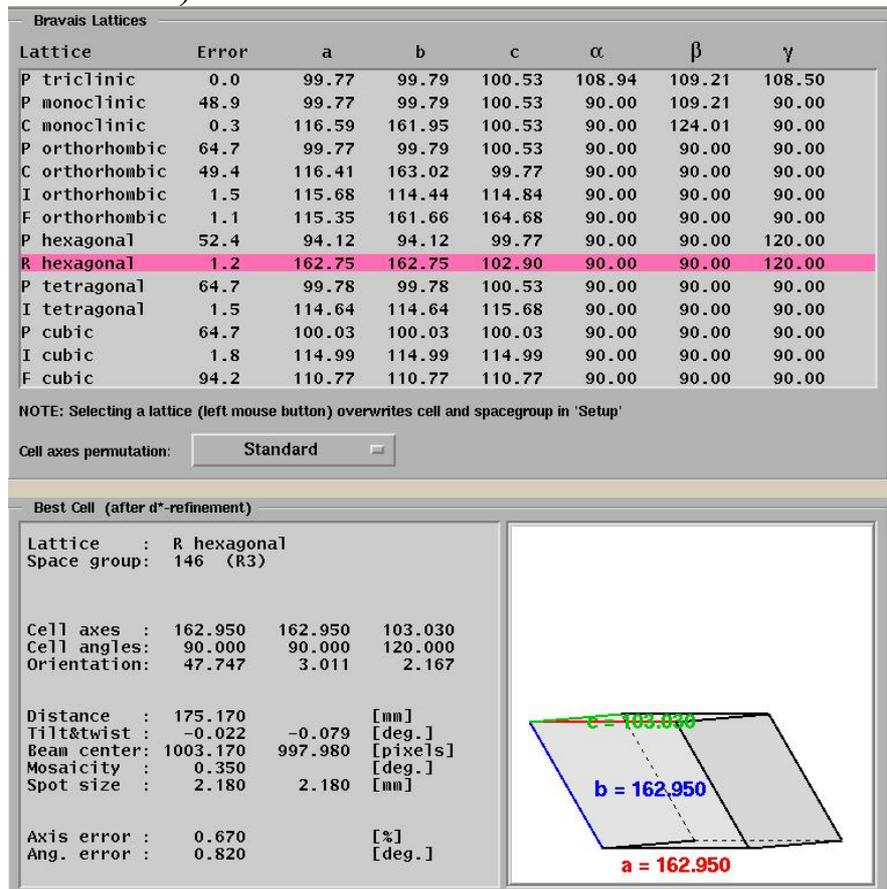


O.K.

(fig.8 modified)

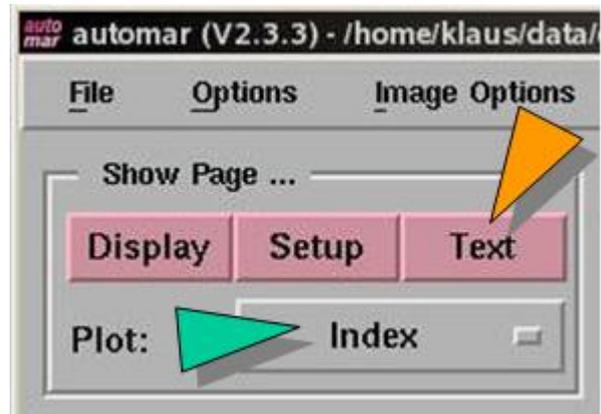
Having eliminated false spots from the peaks list, we are ready for

► “Index” (i.e. run *marIndex*)



(fig.10a)

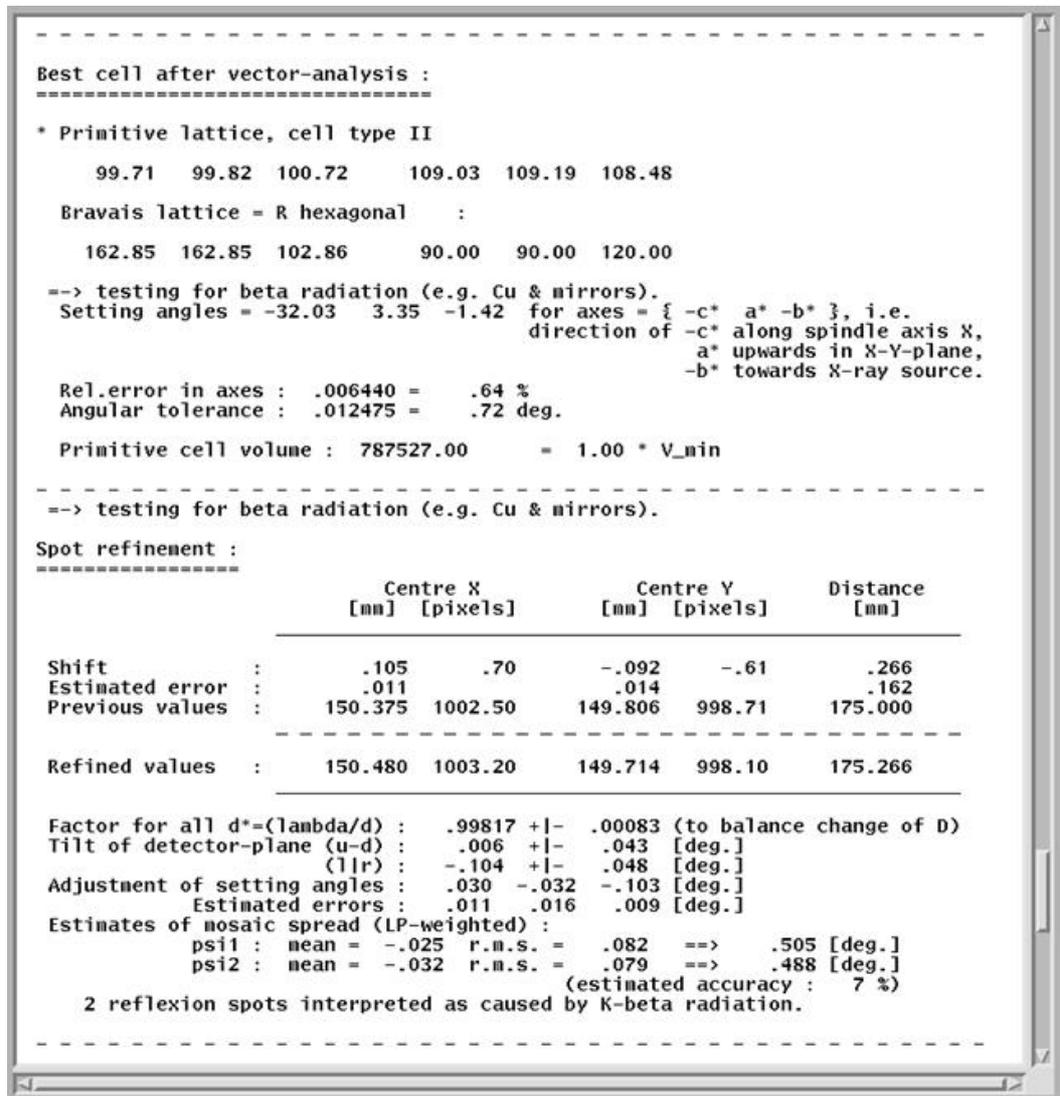
Fig.10a is the graphical display of auto-indexing results, which you can read in more detail (but less compact) in the text window (or even more detailed in the log-file)



Change to “Text” (orange arrow)

(fig.10b)

Scroll backwards with slider:



(fig.10c)

Return to graphics window (green arrow in fig.10b).

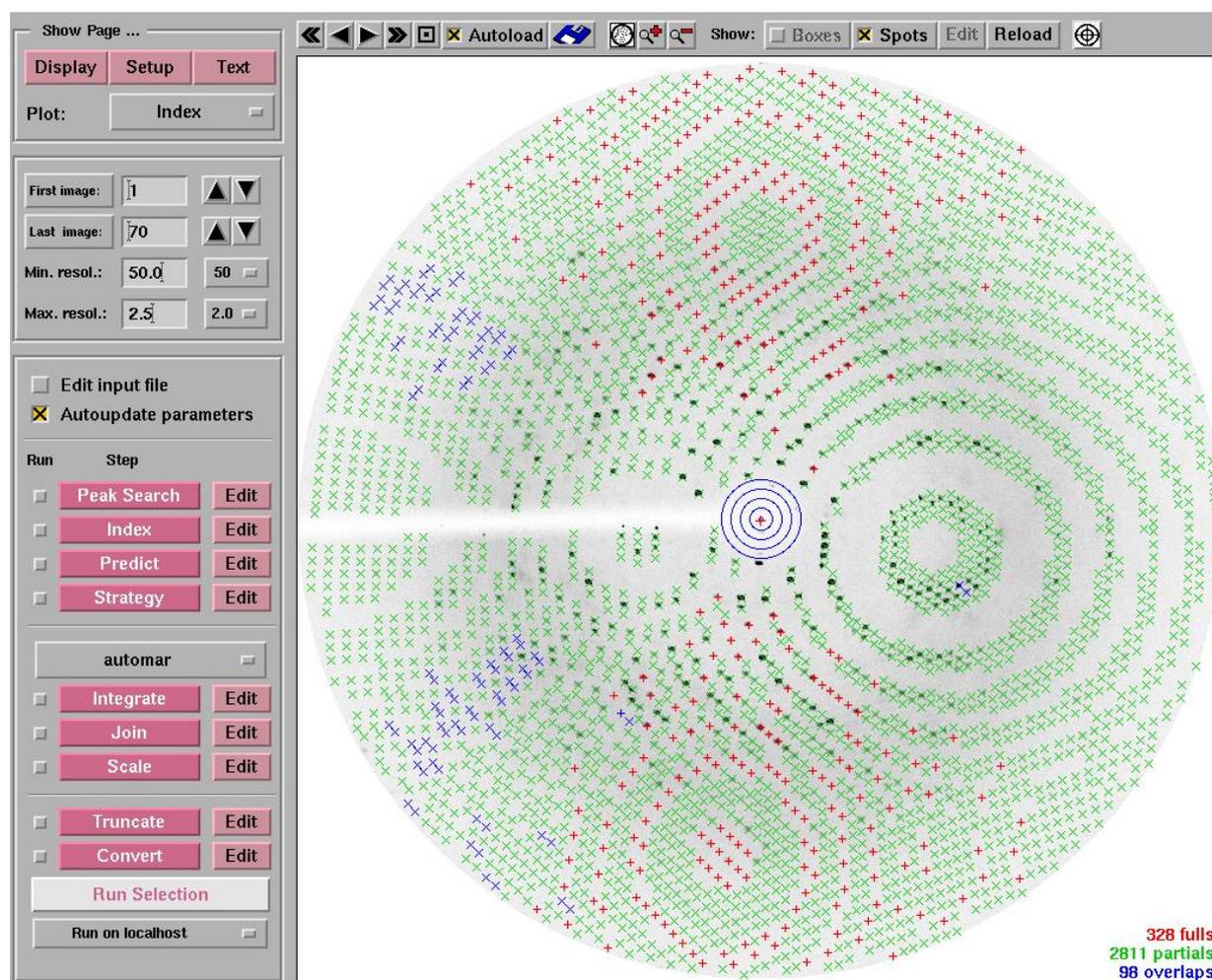
The last 2 lines in the “Plot” (graphics window) give a first clue to whether indexing has worked:

The relative error of axes, and the error of angles, are usually rather similar numbers (by pure coincidence).

If they are $< 1\%$ and $< 1.\text{deg.}$, you are probably alright.

The easiest way to test for success is by clicking on

► **“Predict”** (i.e. run *marPredict* with current Setup parameters):



(fig.11)

Red crosses are expected fully recorded, green partials, blue possibly overlapped. (Crosses as symbols may be replaced by ringlets etc. with “File”, cf. fig.7b, - “Preferences...” – bottom line of the menu.)

Note that for clarity of the display, I have un-set the “Show” excluded rectangle. The consequence of “Use” excluded rectangle is obvious.

If the calculated pattern matches the spot pattern – more or less, not necessarily to the last spot – you can just go on with processing your full image set (Integration). Parameters are updated in due course.

► “Integrate” (i.e. run *marProcess*):

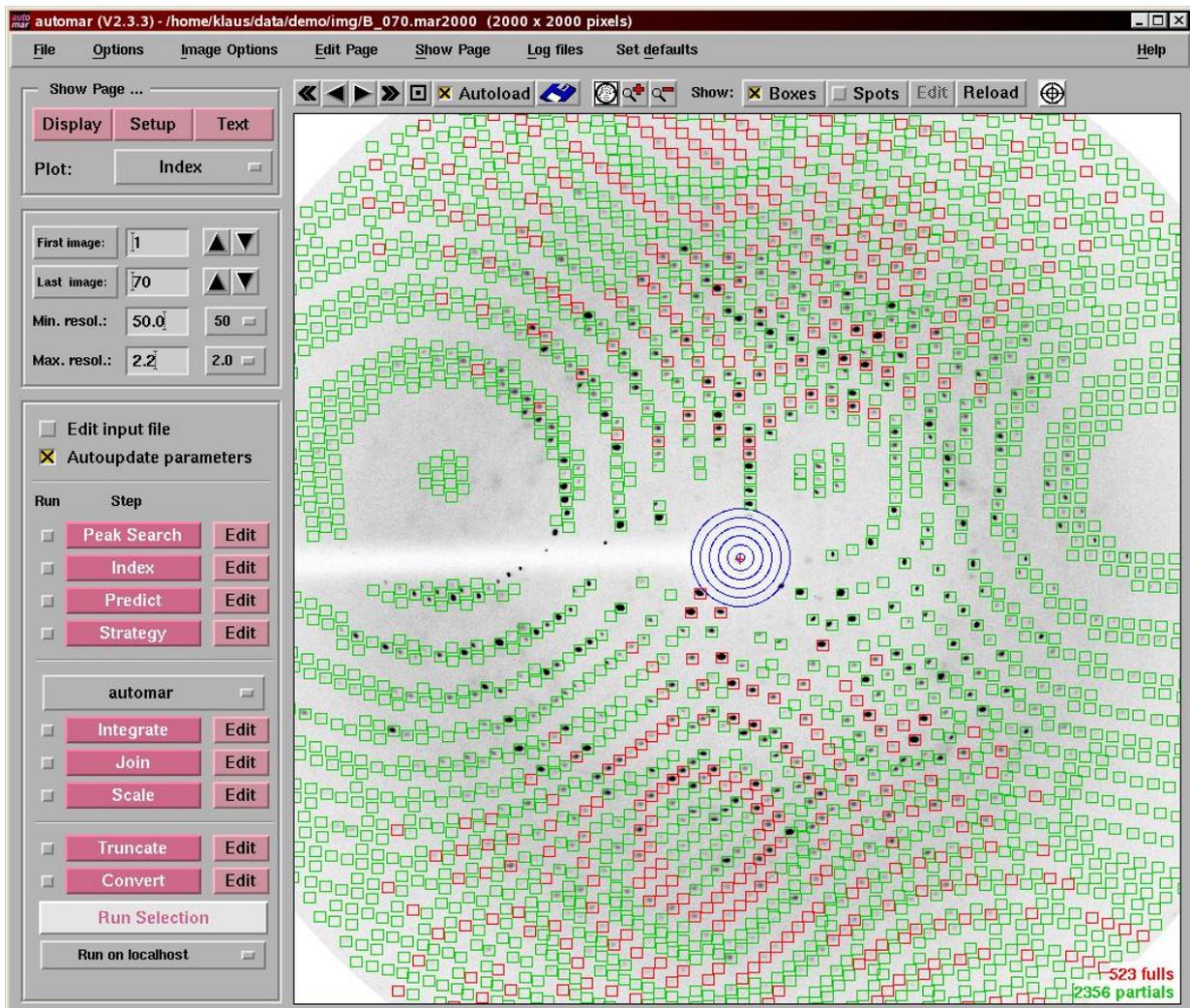
Watch the images as they are processed, with possible restart(s) after significant parameter updates. You can toggle between the “Text” (protocol) and the “Display” windows. In the latter, make sure that “Boxes” are set, “Spots” are un-set **Show: Boxes Spots** (otherwise the diffraction spots are covered by the crosses, and you have no visual control).

“Text” protocol

```

----- Short protocol of evaluation -----
Image      setting angles      vert.  hor. widths  good  total  <I/s>
#          [deg.]              width  (mos. div.) (I>3s) pred.  total beyond
--->      (1) ... (2) ... (3)  [deg.] [deg.]
          Spot-size 15 x 15
001  47.588  2.990  2.195      0.360 0.100  1686  2813   17.0  5.3
001  47.565  2.990  2.197      0.360 0.101  1675  2810   16.6  5.3
002  47.565  2.977  2.197      0.360 0.100  1714  2843   17.1  5.2
>adapt: spot-size 15 x 13 asym. . . . .
001  47.556  2.986  2.205      0.359 0.098  1696  2823   17.1  5.3
002  47.556  2.965  2.205      0.360 0.100  1716  2838   17.4  5.2
003  47.556  2.987  2.205      0.360 0.100  1765  2830   18.5  5.4
004  47.570  2.966  2.205      0.360 0.100  1604  2811   16.1  5.2
  
```

Processing stops with the “Display” window of the last image:



A number of checks have been implemented in *marProcess*, to make sure, and let you get a feeling for, whether experiment and processing have run successfully; and it is worth while reading the *automar-manual* to find out:

- Quality of high resolution reflexions (crystal decay)
- Background analysis (including stability of primary beam)
- Intensity profile of reflexions (both in XY and in PHI)
- Mechanical stability / reliability of goniometer and shutter
- ...

But these details are beyond the scope of this introduction.

It should be noted here, that the GUI parameters are updated after each step in the program suite, see “Setup” window, unless you have dis-selected the **X Autoupdate** option (left margin, half height). Modified parameters are marked by coloured background of their field.

Make a note of the adapted current parameter values at this stage, for later comparison after post-refinement, to test parameter stability.

To finish off, click

► “**Join**” (i.e. run *marPost* - which stands both for post-processing and post-refinement.)

Post-processing consists of summation of partials, including an analysis of completeness of measurements for each hkl and the profile in PHI.

Then output *all* measured integrated intensities (“Fs_q”) into one single file.

Post-refinement uses reflexion profiles and centroids to refine the crystal cell, setting angles, mosaicity and beam divergence, and misalignment of the rotation axis.

If the changes in cell parameters or alignment angles are severe, you might consider to re-run **Integrate** (*marProcess*), possibly in a new working directory such as to keep the previous results (cf. [appendix 2](#)).

You might as well run **Join and Scale** without intervention: check the little boxes left of both program buttons, and then click **Run Selection**.

► “Scale”

marScale reads the *marPost* output file. (It can digest a variety of other formats, or more than one Fsq-file as well. But this tutorial is meant for the “mouse-click approach”...)

Like for *marPost* (“Join”), *Scale* stays with the “Text” window and presents tables with the scaling statistics:

- statistics of measured hkl (input file),
- overall (linear) R-factors, unscaled and scaled,
- multiplicity and completeness of different hkl sub-sets,
- anomalous R-factors, centric reflexions,
- I/sigma (again for different hkl sub-sets),
- different types of R-factors (PCV, R_{rim} , R_{pim} , R_{squ} , R_{lin}), both unweighted and sigma-weighted,

in (equal-volume) resolution shells.

Many graphical display options are available in the “Plot” window. More detailed tables, scale- and B-factors, preliminary Wilson-plot, etc., can be found in the log-file, which is named *AUTOMAR.marScale* (if run from the GUI), or anyway

‘root’_’1st’-‘last’.marScale

where ‘root’ is your image name prefix, and ‘1st’ and ‘last’ are the image numbers (3 or 4 digits) of your *marProcess* (“Integrate”) run.

Several options for the scaling can be chosen in the “Scale” – “Edit” menu:

- resolution limits,
- number of resolution shells for the statistics tables,
- normal or anomalous scaling,
- B-factors ON/OFF
- Different output file types, both for merged-mean and unmerged intensities, including *scalepack*-emulation and *shelx* hkl4, in addition to genuine *automar* formats.

In the end you may

► **Truncate** (according to French & Wilson), and

► **Convert** to mtz-format

for subsequent structure solution programs. Good luck!

==== **The END of *automar*** ====

Appendix 1: Installation

The authors differ as to their preferences for installation of the *automar* suite. So you, too, may have your own opinion.

- (1) The *default* installation from the distribution kit creates the “automar tree” with environment names for standard directories, to which the GUI writes backup-files (“automar.mix”) and certain types of log-files.
- (2) *Manual* installation evades those environment directories. Usually all programs will reside in your ~/bin/ or /usr/local/bin/ directory or the like. Fig.2 illustrates this situation: both the “automar.mix” (i.e. parameter backup) file, and all log-files are created and stored in the current working directory. This way, every run in a newly created working directory starts with a ‘clean slate’ – unless you copy some existing automar.mix file to it.

The disadvantage of the one is the advantage of the other:

- (1) bears the danger of old, inappropriate parameters inferred from one crystal, or even project, to another completely different one.
- (2) requires that you re-define every re-occurring / permanent experimental condition, time and again (e.g. beam-stop shadows, beam divergence, spindle axis mis-alignment, etc.)

Appendix 2: Running and re-running *automar* programs

automar is a suite of individual executables.

Each program reads a control file with input parameters, and the header parameters that are stored in the data file(s). Control file parameters override any header parameters which in turn override default values of the programs.

The executables can either be invoked by clicking the program buttons of the GUI (left margin lower half); or be called explicitly from the command line. (Command line parameters have the highest priority, overriding even control file parameters.)

The GUI collects the current applicable values of the “Setup”, writes a control file in standardized format, possibly lets you **Edit input file**, and calls the program with this file. After termination of the program, the control file is *not* deleted.

This way you can edit it off-line, varying or adding/deleting any parameters, and then re-run the program from the command line (with or without optional command line parameters).

Mind, however, that there is no auto-update of “Setup” then.

Also mind that the control file that you edited off-line is overwritten by the next run from the GUI (so re-name it if you want to keep it for comparison or re-use).

If unsure which parameters are most adequate for the processing of a particular set of images, create new directories, copy the first run’s *automar.mix* to all of them, and modify whichever parameter may be in doubt.

Appendix 3: Failures of Indexing

We have already discussed false spots and how to [eliminate](#) them: [ice](#), and [edges of shadows](#).

There is one more cause for false spots: *salt* crystal satellites. Usually, you can easily spot them by inspection (they do not ‘belong’, they are pretty strong and have tails or anyway weird shapes). Remove them from the “peaks” list (i.e. from the *index.pks* file) with the “Edit” option (top margin):



This will bring up a new little menu with entries “Add”, “Delete”, [“More”, “Fewer”] and “Done”. Click on “Delete” (MB1), then click on crosses to be deleted with MB2. If the GUI can unambiguously locate which spot in the list you meant, the associated terminal window displays the text: “Spot #546 at [1216.7, 1726.3] removed”. (The cross may not disappear until the image display is renewed, in particular when you click “Done”.)

Surprisingly enough, one reason for failure may be *too high resolution* (“less is more”). Advice in short:

Reduce “Max.resol.” to 4/5 (possibly down to 2/3) of the image radius.

To understand, let’s talk about the first step, a basic principle, of auto-indexing methods of rotation photographs:

Detector coordinates of reflexion spots are converted to reciprocal lattice vectors by projection onto the Ewald-sphere (this conversion from 2-*d* to 3-*d* depends on more or less correct values for the beam

centre, wave-length, and crystal-to-detector distance). These 3d-vectors have a spatial uncertainty (more or less perpendicular to the surface of the sphere) due to the deltaPHI-rotation. This uncertainty increases proportional to the distance from the rotation axis and can, for large deltaPHI, come close to, or even exceed, the lattice constants such that the spot indices become ambiguous and thus cause indexing failure.

Split (or twinned) crystals.

- If you are lucky, one of the two patterns is considerably stronger than the other(s), such that the ‘peaks’ are mainly representative of one crystal lattice only. Indexing may succeed, but with unusually large error bars. Spot integration will suffer from overlapping spots of the companion, resulting in bad R-factors.
- Indexing finds a super-lattice which tries to ‘harmonize’ both sets of spots. This failure is immediately recognizable by huge cell edges, unreasonably large for the pattern.
- More likely, the program will end with some sort of message of total failure.

While I am working on a procedure that can index, and then process and refine, two superimposed lattices in one go, there is (to my knowledge) no program as yet to integrate twin patterns on rotation photographs.

Meanwhile, sorry - try to grow good crystals...

Appendix 4:

In *marScale*, due to reasons unknown and/or not understood, the matrix inversion library routine (with singular or eigen-value filtering) does not converge in rare cases. It prints the corresponding eigen-vector, and does a poor job for the scale factors – not completely wrong, nor does it crash, but results are poor (e.g. S-factors too close to 1.0).

Sometimes it helps to split the data in two, scale the parts separately and then finally scale the two halves. Or just throw out part of the images that you think you could do without...

As yet I have not given up hope to find a work-around for this routine.