Meteor spectra calibration with Python, user manual

Martin Dubs, FMA

Summary

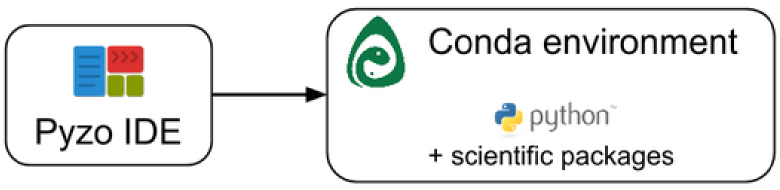
This procedure replaces the calibration of meteor spectra with laser calibration images using IRIS for the determination of laser point coordinates and EXCEL for the fitting of parameters for the distortion model. It is required for the change from image coordinates to the orthographic projection, which is used to linearize meteor spectra.
The goal is to simplify the analysis of meteor spectra. In a further step, also the analysis of meteor spectra with IRIS, ImageTools and SpectroTools will be converted to Python, so that only one language will be used for the complete analysis.
Python was chosen, because

- it contains all the necessary tools to do the analysis
- it finds widespread use in the astronomy community
- it is free
- it runs on different platforms

I was inspired to use Python by Giovanni Leidi, who was giving a talk about the use of Python for the analysis of spectra in the spectroscopy workshop at OHP 2018 (https://www.shelyak.com/ohp-spectro-star-party-2018/).
Note of caution: I am new to Python, so the script presented here may not be the best solution. Some things have been done in a complicated way, copying examples from different sources and trying to make it work. It certainly should be improved for clarity and safety of operation. Therefore I hope you will suggest improvements.

Installation of Python

Python comes in different versions. Apart from the base version it requires additional packages, in particular:
Matplotlib
Numpy
Astropy
Etc.
Some versions are not quite compatible. For that reason it is highly recommended to install a pre-packaged version, such as Anaconda, unless you are familiar with Python installing packages.
For running and editing Python scripts you need a development environment (IDE). I use PYZO, as recommended by Giovanni. It is installed from http://www.pyzo.org/start.html.



To get started with Pyzo, you need to install the Pyzo IDE (in which you write your code) and a Python environment (in which you run your code).

Download and install it. Pyzo recommends installing Miniconda afterwards. I had some problems with Miniconda and other packages; therefore I recommend using Anaconda instead. The installation of Anaconda is described next. You are free to use whatever version of Python you like, but do not complain if it does not work **(do not use Python 2.7, it does not work with my script).**

Installation of Anaconda

Go to
https://www.anaconda.com/download/
Install the 3.6 version (64-bit in my case):

Python 3.6 version *
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (506 MB)

Follow the tips for installation
https://docs.anaconda.com/anaconda/install/windows
Install Anaconda in C:\, otherwise you may have problems finding it on your computer (makes it easier to write path to anaconda, but you are free to install it elsewhere).
At this point, if everything has gone well the Pyzo IDE should recognize without telling him the environment Anaconda you chose and you can start writing in python. To install packages simply type in the shell (you will find it on the right in Pyzo): install numpy, install scipy, install astropy and so on

Load libraries

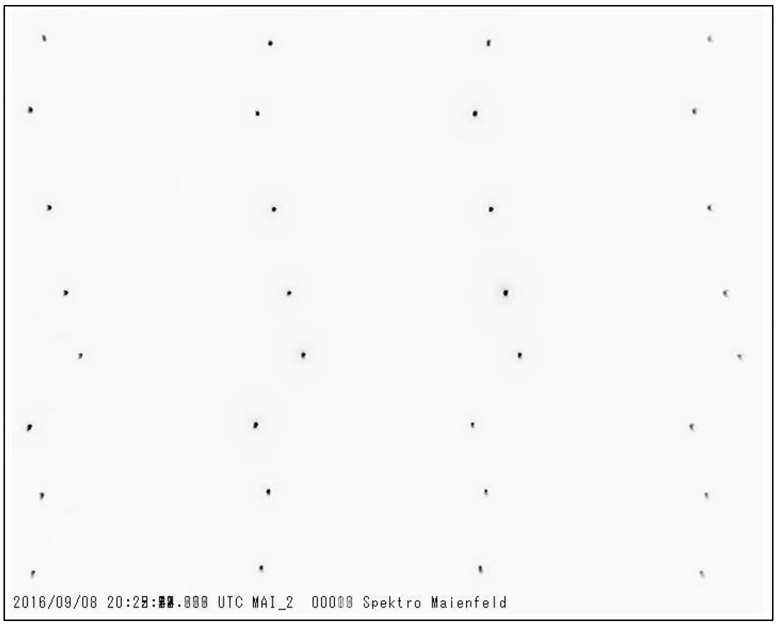
If the scripts do not run, it may be that a library is missing. You have to load libraries, which are not contained in your Python distribution.
With Anaconda the command is, e.g. for the installation of the least square fit routine:

```
>>> conda install -c conda-forge lmfit
```

You find the installation command by going to the Anaconda cloud https://anaconda.org/ and search for "lmfit".
The installation checks for the compatibility with the different already installed packages and makes the necessary updates.

Laser calibration of meteor spectra

For the calibration of meteor spectra you need some calibration images, containing laser spectra with different orders in different positions of the image area. Ideally they cover most of the image area, in order to avoid extrapolation of the fitting functions. Details are described in the old manual
https://meteorspectroscopy.files.wordpress.com/2018/01/processing-meteor-spectra-v151.pdf
and are not repeated here. The theory is described in the following paper:
A practical method for the analysis of meteor spectra
Martin Dubs, Peter Schlatter (WGN Journal, Ausgabe 43-4, 2015):
Meteor_Spectroscopy_WGN43-4_2015
We assume that the spectra are in a single image, obtained with ADD_MAX2 in IRIS (get peak values from several images):
Example: addb.fit:



(B/w inverted)

On each horizontal row 1 spectrum is displayed, with 4 laser lines in different orders, for 8 spectra in total. This was recorded with

WATEC 902 H2 ultimate image size: 720x576 pixels

Tamron 12VG412ASIR ½", f: 4-12mm f/1.2, used at f: 8 mm

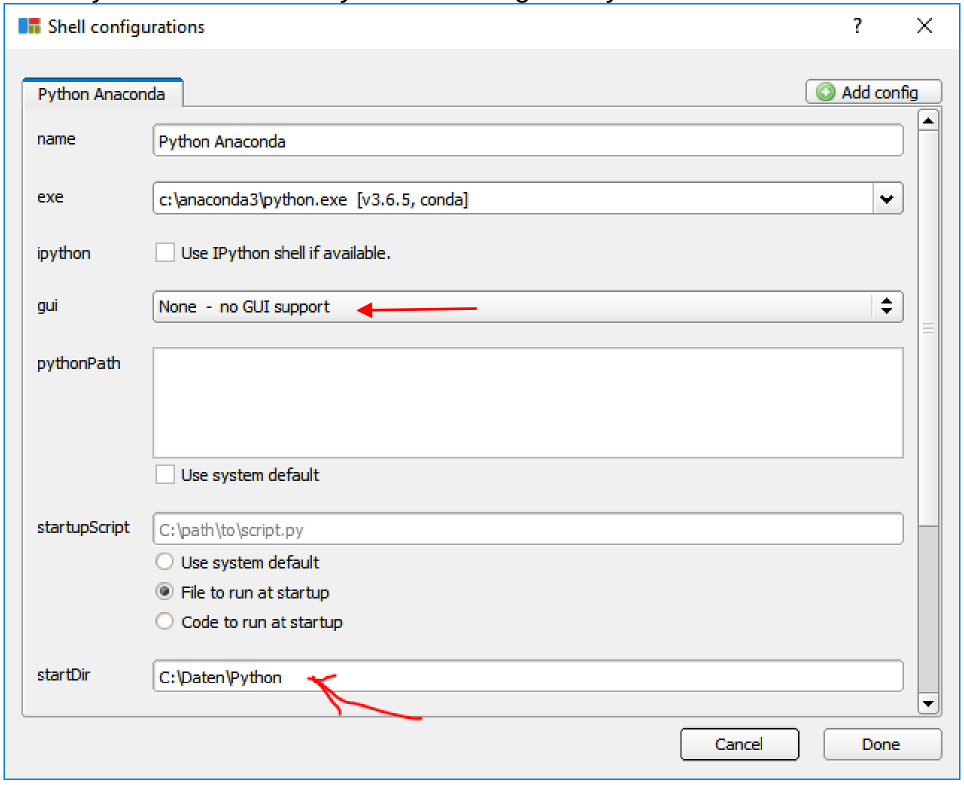
Grating: 50x50 600 lines/mm Thorlabs

Calibration laser: 405 nm wavelength from eBay:

http://www.ebay.com/itm/Diode-Laser-405nm-20mW-Violet-Purple-Laser-Dot-Module-w-cable-13x42mm-3-5-5V-/191174143732?ssPageName=ADME:L:OC:CH:3160

Run the test example

Put the files of the example into the start Directory of Pyzo or make the startDir the directory where your files are. This you can change in Pyzo – Shell – edit shell configuration:



name: Python Anaconda

exe: c:\anaconda3\python.exe [v3.6.5, conda]

ipython: Use IPython shell if available. [ ]

gui: None - no GUI support

pythonPath:

Use system default [ ]

startupScript: C:\path\to\script.py

Use system default ( )

File to run at startup [✓]

Code to run at startup ( )

startDir: C:\Daten\Python

Note also that PYZO uses the Anaconda version of Python (3.6.5)

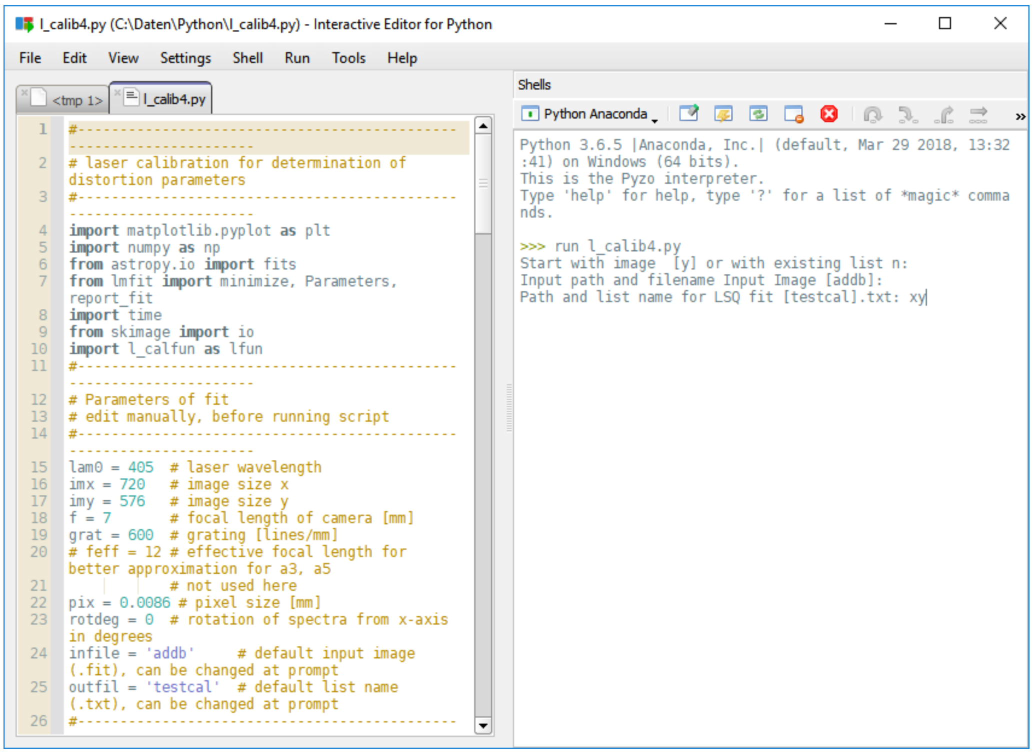
Next run the script, either by loading it into the editor, where you can adjust some default values of the parameters (not needed when you run the test example) and select Run – Run file as script.
As an alternative you can type

```
>>> run l_calib5.py
```

after the prompt in the shell window and ENTER
After some delay to compile the script the following lines of the script appear:

```
Start with image [y] or with existing list n:
Input path and filename Input Image [addb]:
Path and list name for LSQ fit [testcal].txt: xy
```

Each line expects an input, the [default value] can be selected with ENTER, for changing it you have to enter a new value. The list file testcal already exists, so I chose another name xy (.txt), where the selected laser lines will be written.



```
#---------------------------------------------
---------------------
# laser calibration for determination of
distortion parameters
#---------------------------------------------
---------------------
import matplotlib.pyplot as plt
import numpy as np
from astropy.io import fits
from lmfit import minimize, Parameters,
report_fit
import time
from skimage import io
import l_calfun as lfun
#---------------------------------------------
---------------------
# Parameters of fit
# edit manually, before running script
#---------------------------------------------
---------------------
lam0 = 405  # laser wavelength
imx = 720   # image size x
imy = 576   # image size y
f = 7       # focal length of camera [mm]
grat = 600  # grating [lines/mm]
# feff = 12 # effective focal length for
better approximation for a3, a5
            # not used here
pix = 0.0086 # pixel size [mm]
rotdeg = 0  # rotation of spectra from x-axis
in degrees
infile = 'addb'     # default input image
(.fit), can be changed at prompt
outfil = 'testcal'  # default list name
(.txt), can be changed at prompt
#---------------------------------------------
```

```
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32
:41) on Windows (64 bits).
This is the Pyzo interpreter.
Type 'help' for help, type '?' for a list of *magic* comma
nds.

>>> run l_calib4.py
Start with image  [y] or with existing list n:
Input path and filename Input Image [addb]:
Path and list name for LSQ fit [testcal].txt: xy
```

Notice than when you hit ENTER again, a window with the image will appear BEHIND! the editor window (this is a Windows - Python problem). Bring it to the front by clicking on the Windows menu image plot symbol :

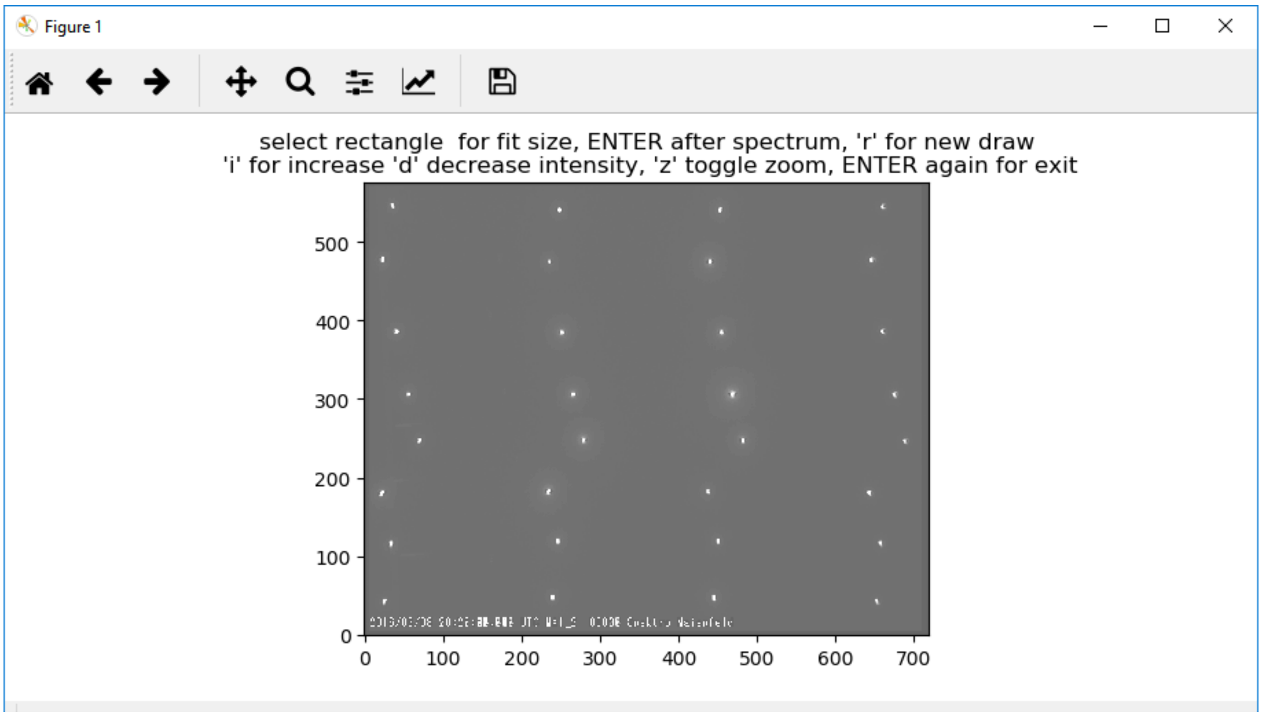
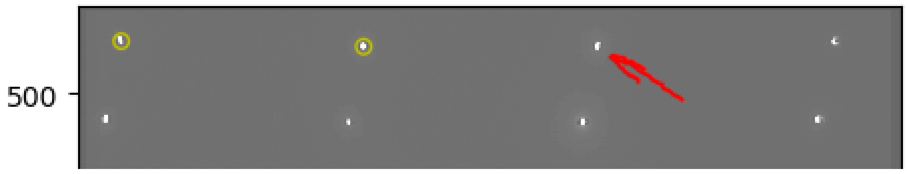


Figure 1
select rectangle for fit size, ENTER after spectrum, 'r' for new draw
'i' for increase 'd' decrease intensity, 'z' toggle zoom, ENTER again for exit
500
400
300
200
100
0
0
100
200
300
400
500
600
700

Now you are ready to selct the different orders of the laser spectra.

Select points of the different orders

- Start with the lowest order at the left and click with the mouse on the spot you would like to select:



  The program calculates a Gaussian fit of the spectrum and saves the result. A yellow circle shows the selected point. Continue with the next point.
- If you have selected all orders of one spectrum, hit ENTER, the coordinates and widths of the different orders will be saved to the text file.
- If you made a mistake, type ‘r’ on the keyboard, to start with the same or another spectrum
- With ‘i’ and ‘d’ you can adjust the image contrast
- With ‘z’ a plot of the Gaussian fit is displayed. Use it only for testing, you have to close each image before you can continue. Type ‘z’ again to turn these images off. Type ‘r’ to clear the data before you continue with a new spectrum.
- Hit ENTER again after the last spectrum

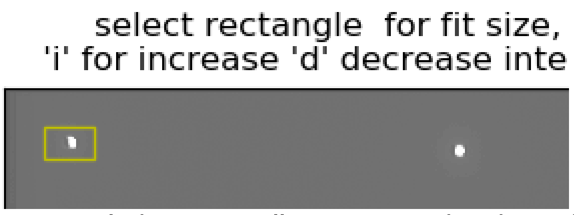
In the shell window the following text appears:

```
Finished, saved xy.txt
continue [y] or edit textfile n:
```

If you hit ENTER the program will continue with a least square fit of the parameters (corresponds to “Solve” in EXCEL). Press ‘n’ to view the file first.

Note:

If you have problems with the selection of points, you may change the size of the rectangle, in which the Gaussian peak is searched and fitted (default is a rectangle with corners x0, y0 +/- 10 pixels. You may draw a rectangle in the image by selecting one corner and draw with the mouse down to the opposite corner and release the button there:


select rectangle for fit size,
'i' for increase 'd' decrease inte

If the rectangle is too small, you may miss the point when selecting it, if the rectangle is too large, the Gaussian fit may fail. You will get some error messages in the shell, try with another size!
Notice, this is still a test version.

Inspection of the calibration text file

It should look like this:
(testcal.txt)

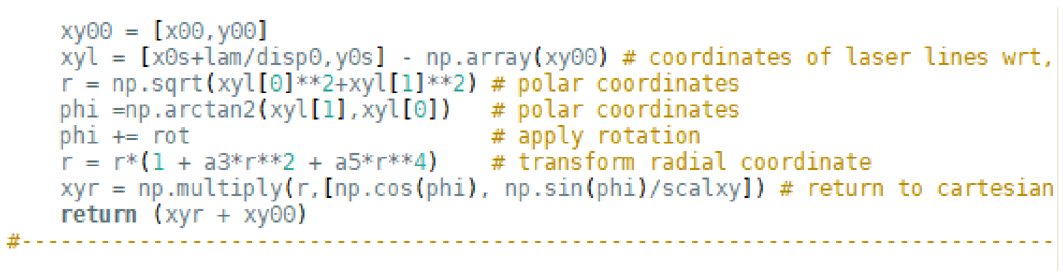
```
# 1
  36.71  545.19    2.78    4.77
 248.88  540.42    3.17    3.57
 453.57  540.66    2.53    4.83
 660.90  544.51    3.51    4.96
   0.00    0.00    0.00    0.00
# 2
  23.93  477.95    3.36    4.45
 236.88  474.54    2.70    3.54
 440.96  474.46    3.28    4.34
 646.60  476.59    3.09    4.33
   0.00    0.00    0.00    0.00
…
# 8
  26.28   43.31    2.88    4.25
 240.39   47.97    2.68    3.80
 445.94   47.09    2.64    5.44
 653.47   43.05    3.00    4.82
   0.00    0.00    0.00    0.00
# 9
   0.00    0.00    0.00    0.00
```

- The columns denote x-coordinate, y-coordinate, FWHM(x) and FWHM(y) of the spots, as determined by the Gaussian fit
- The lines with # are comments and ignored for the fit.
- Each spectrum ends with a line of zeros as separator
- The end of the file is indicated by a second line of zeros
- Additional lines of zeros in between are ignored, also spectra containing only one line (there is nothing to fit there)
- If you forgot one or more spectra, you can run the script again and add the missing spectra
- If you recorded one spectrum twice or entered wrong data points, you may erase these in the text file manually and save the file again for the following step.

Least square fit

For the determination of the distortion parameters, a linear dispersion law is postulated in the transformed image
x‘ = x0 + i*lam0*disp0, I = 0, 1, 2, …
y’ = y0

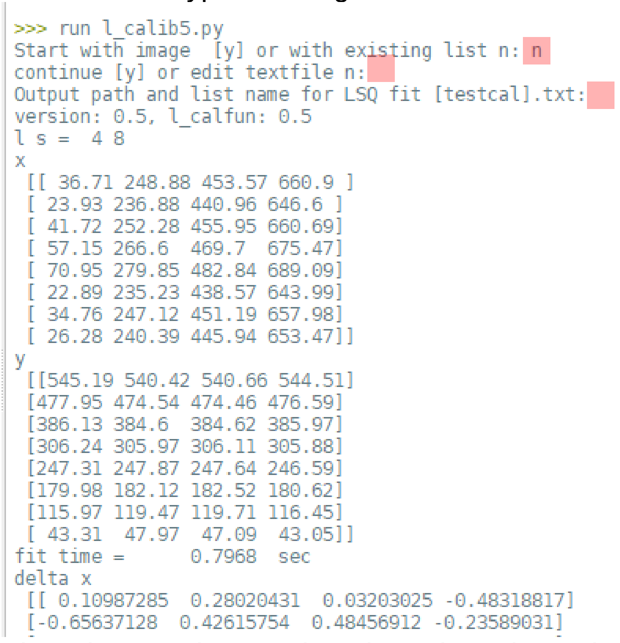
from these coordinates the transformation to the image coordinates (as required in the coordinate transformation of the meteor images) is calculated by a transformation to polar coordinates around the image axis at coordinates (x00, y00), rotation of the image by an angle rotdeg, transforming the radial coordinate by
$r = r' * (1 + a3*r'^2 + a5*r'^4)$ (plus higher terms in the future),
transforming back to Cartesian coordinates with the axis remaining at (x00, y00)
For those interested in the details, here is the corresponding code from the Python function. (x0, y0s) correspond to (x', y').
xyr corresponds to (x,y) in the calibration image, to be fitted to the coordinates in the calibration file (testcal.txt in the example)



```
    xy00 = [x00,y00]
    xyl = [x0s+lam/disp0,y0s] - np.array(xy00) # coordinates of laser lines wrt, optical axis
    r = np.sqrt(xyl[0]**2+xyl[1]**2) # polar coordinates
    phi =np.arctan2(xyl[1],xyl[0])   # polar coordinates
    phi += rot                       # apply rotation
    r = r*(1 + a3*r**2 + a5*r**4)    # transform radial coordinate
    xyr = np.multiply(r,[np.cos(phi), np.sin(phi)/scalxy]) # return to cartesian coordinates
    return (xyr + xy00)
#-------------------------------------------------------------------------------
```

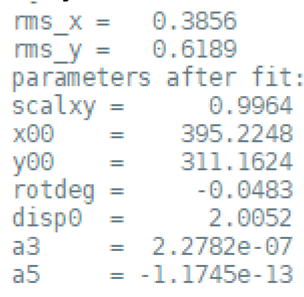
Run the fit for the example testcal.txt with l_calib5.py included in the zip file

Let us assume that you have edited the file testcal.txt and corrected all the errors (wrong points, wrong order, etc.) and you would like to start the least square fit with these data. You run the script l_calib5.py. When asked for image, type n (you want to skip that step), then type enter to continue and type enter again to confirm the choice of testcal.txt and the fit should start:



```
>>> run l_calib5.py
Start with image  [y] or with existing list n: n
continue [y] or edit textfile n:
Output path and list name for LSQ fit [testcal].txt:
version: 0.5, l_calfun: 0.5
l s =  4 8
x
 [[ 36.71 248.88 453.57 660.9 ]
 [ 23.93 236.88 440.96 646.6 ]
 [ 41.72 252.28 455.95 660.69]
 [ 57.15 266.6  469.7  675.47]
 [ 70.95 279.85 482.84 689.09]
 [ 22.89 235.23 438.57 643.99]
 [ 34.76 247.12 451.19 657.98]
 [ 26.28 240.39 445.94 653.47]]
y
 [[545.19 540.42 540.66 544.51]
 [477.95 474.54 474.46 476.59]
 [386.13 384.6  384.62 385.97]
 [306.24 305.97 306.11 305.88]
 [247.31 247.87 247.64 246.59]
 [179.98 182.12 182.52 180.62]
 [115.97 119.47 119.71 116.45]
 [ 43.31  47.97  47.09  43.05]]
fit time =      0.7968  sec
delta x
 [[ 0.10987285  0.28020431  0.03203025 -0.48318817]
 [-0.65637128  0.42615754  0.48456912 -0.23589031]
```

There is some information about the written, then the data points are given in matrix form and finally the results of the least square fit:

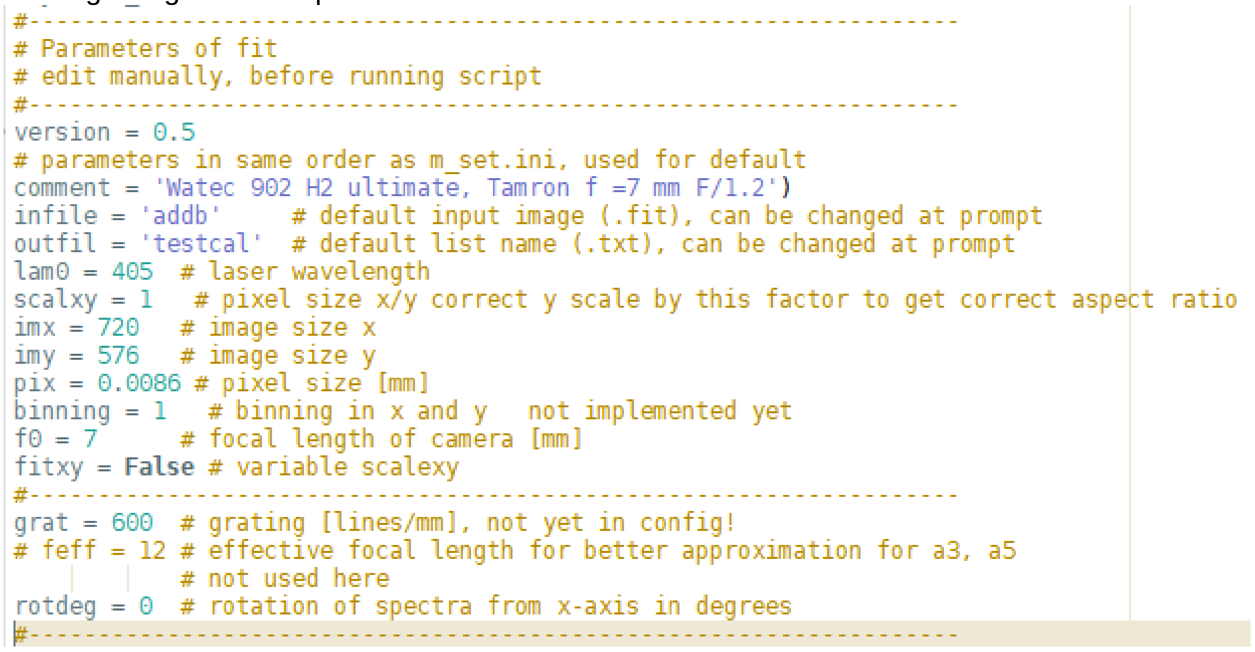


```
rms_x =    0.3856
rms_y =    0.6189
parameters after fit:
scalxy =      0.9964
x00    =    395.2248
y00    =    311.1624
rotdeg =     -0.0483
disp0  =      2.0052
a3     =  2.2782e-07
a5     = -1.1745e-13
```

These parameters you may use to transform your meteor spectra to the orthographic projection. Notice that the fit runs quite fast, the Levenberg Marquard algorithm is very efficient for this kind of problem.

Run the fit for different camera, lens, grating

For the fit some additional (fixed) parameters are required. At the moment these are given at the beginning of the script:



```
#---------------------------------------------------------------------
# Parameters of fit
# edit manually, before running script
#---------------------------------------------------------------------
version = 0.5
# parameters in same order as m_set.ini, used for default
comment = 'Watec 902 H2 ultimate, Tamron f =7 mm F/1.2')
infile = 'addb'     # default input image (.fit), can be changed at prompt
outfil = 'testcal'  # default list name (.txt), can be changed at prompt
lam0 = 405  # laser wavelength
scalxy = 1   # pixel size x/y correct y scale by this factor to get correct aspect ratio
imx = 720   # image size x
imy = 576   # image size y
pix = 0.0086 # pixel size [mm]
binning = 1   # binning in x and y   not implemented yet
f0 = 7      # focal length of camera [mm]
fitxy = False # variable scalexy
#---------------------------------------------------------------------
grat = 600  # grating [lines/mm], not yet in config!
# feff = 12 # effective focal length for better approximation for a3, a5
            # not used here
rotdeg = 0  # rotation of spectra from x-axis in degrees
#---------------------------------------------------------------------
```

As indicated, these can be changed manually for other cameras, gratings and laser wavelength. The focal length, pixel size and grating constant are used for the initial value of the dispersion and the coefficients a3 and a5 (they are not critical).

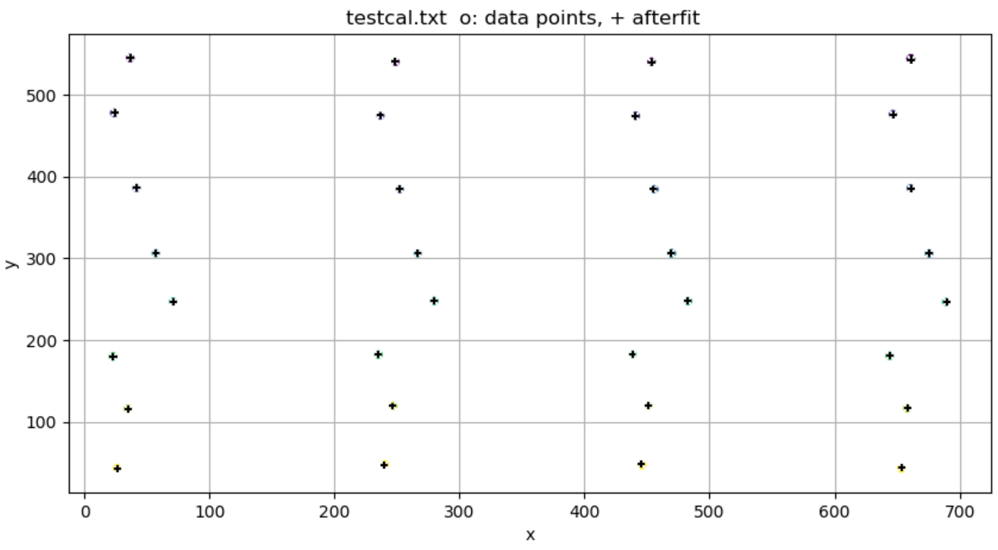
Results

If the fit runs correctly, the rms error of the pixel coordinates and the values for the parameters are output on the shell. For comparison in the last two columns the parameters obtained from a fit with EXCEL, using the same image for calibration. In the last column data obtained with IRIS for the spectral line positions with the original EXCEL fit. In this case x00 and y00 have been corrected by 1 pixel to account for the difference in pixel numbering in IRIS (starting with (1, 1) in the bottom left corner).

Comparison with EXCEL fit

	Python	EXCEL	EXCEL IRIS
rms_x	0.3856	0.386	0.388
rms_y	0.619	0.619	0.621
parameters after fit:			
scalxy	0.9964	0.9964	0.9926
x00	395.22	395.23	395.7
y00	311.16	311.16	312.6
rotdeg	-0.0483	-0.0483	-0.0489
disp0	2.0052	2.0052	2.0040
a3	2.28E-07	2.28E-07	2.25E-07
a5	-1.17E-13	-1.17E-13	-1.12E-13

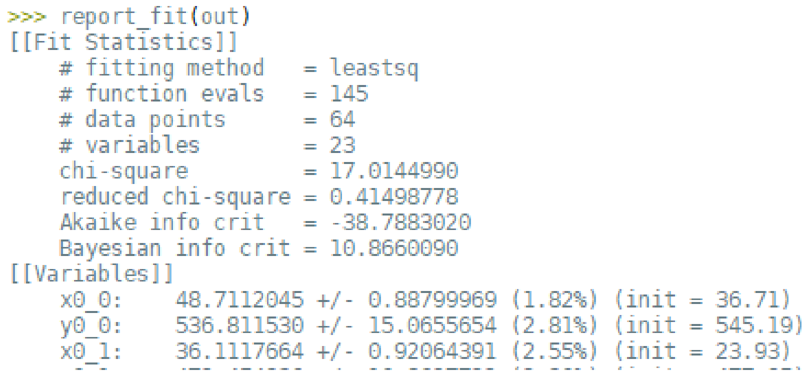
(The results of the fit are written to the file ‘m_set.ini’ for future use in the calibration of meteor spectra in the script l_calib6.py described below).



The plot of the image coordinates and the fit shows bad points (you may zoom into the image with . The plot of data points is saved as testcal_lsfit.png.
You may also inspect the arrays deltax and deltay in the shell window.

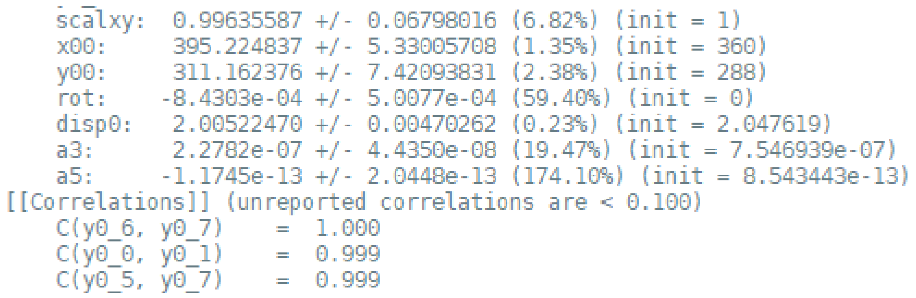
Results of Least Square Fit

A detailed view of the results of the fit is given with the command “**report_fit(out)**” (commented out in the script), which shows statistical errors, correlations etc. for the interested reader. For practical purposes the average rms error of the fit in x- and y-coordinates rms_x and rms_y are shown. With a well corrected and focused lens these values should be < 1 pixel. With noise free and aberration free synthetic data produced with ImageTools, rms-errors were < 0.05 pixel.



```
>>> report_fit(out)
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 145
    # data points      = 64
    # variables        = 23
    chi-square         = 17.0144990
    reduced chi-square = 0.41498778
    Akaike info crit   = -38.7883020
    Bayesian info crit = 10.8660090
[[Variables]]
    x0_0:    48.7112045 +/- 0.88799969 (1.82%) (init = 36.71)
    y0_0:    536.811530 +/- 15.0655654 (2.81%) (init = 545.19)
    x0_1:    36.1117664 +/- 0.92064391 (2.55%) (init = 23.93)
```

…



```
    scalxy:  0.99635587 +/- 0.06798016 (6.82%) (init = 1)
    x00:     395.224837 +/- 5.33005708 (1.35%) (init = 360)
    y00:     311.162376 +/- 7.42093831 (2.38%) (init = 288)
    rot:    -8.4303e-04 +/- 5.0077e-04 (59.40%) (init = 0)
    disp0:   2.00522470 +/- 0.00470262 (0.23%) (init = 2.047619)
    a3:      2.2782e-07 +/- 4.4350e-08 (19.47%) (init = 7.546939e-07)
    a5:     -1.1745e-13 +/- 2.0448e-13 (174.10%) (init = 8.543443e-13)
[[Correlations]] (unreported correlations are < 0.100)
    C(y0_6, y0_7)    =  1.000
    C(y0_0, y0_1)    =  0.999
    C(y0_5, y0_7)    =  0.999
```

Notice, that the scalexy is within error equal to 1 and that the coefficient a5 is not well defined by the chosen data points (4 points in one spectrum cannot define the 5th power of a polynomial.

Configuration file

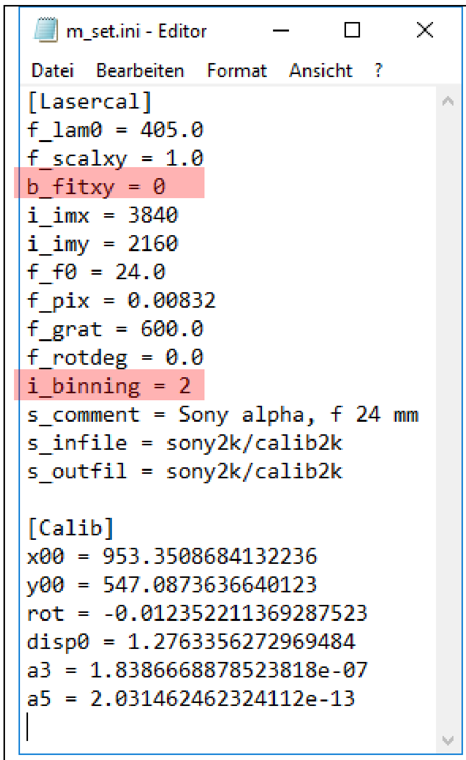
The program l_calib6.py was modified to read and write parameters from a configuration file **m_set.ini**. It requires the import of ConfigParser. Since the installation did not work with Anaconda, the source ConfigParser.py has to be included in the same directory as l_calib6.py. If m_set.ini is not found, the default values from L_calib.py are used.
If you have problems running l_calib6.py use l_calib5.py and edit the parameters manually at the beginning of the file.
At the start, l_calib6.py asks for the directory where the configuration file is located:

```
>>> run l_calpath.py
version: 0.6, l_config: 0.2, config: m_set.ini
Set Path for m_set.ini []: sony
dir sony
Configuration sony\m_set.ini read
--> ['Lasercal', 'Calib']
[Lasercal]
- [f_lam0] = 405.0
…
```

Python has the possibility to enter subdirectories with forward / or backward \ slash. Both are accepted here:

```
Set Path for m_set.ini []: 0207/nnn
dir 0207\nnn
```

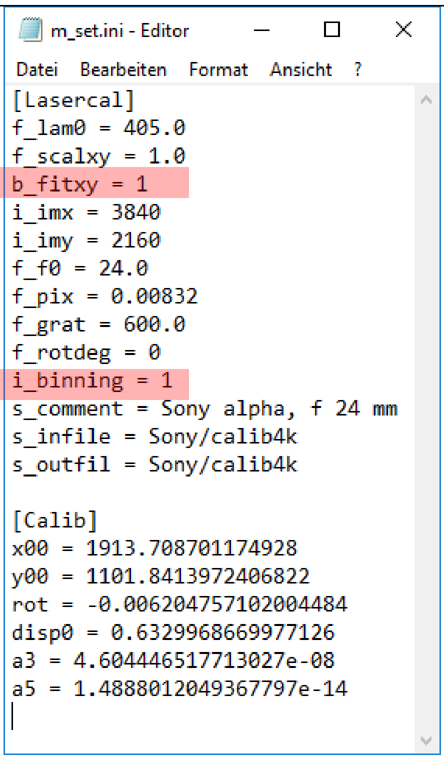
The default (working) directory you select with “Enter”



```
[Lasercal]
f_lam0 = 405.0
f_scalxy = 1.0
b_fitxy = 0
i_imx = 3840
i_imy = 2160
f_f0 = 24.0
f_pix = 0.00832
f_grat = 600.0
f_rotdeg = 0.0
i_binning = 2
s_comment = Sony alpha, f 24 mm
s_infile = sony2k/calib2k
s_outfil = sony2k/calib2k

[Calib]
x00 = 953.3508684132236
y00 = 547.0873636640123
rot = -0.012352211369287523
disp0 = 1.2763356272969484
a3 = 1.8386668878523818e-07
a5 = 2.031462462324112e-13
```

```
Sony alpha, f 24 mm
sony2k/calib2k.txt
rms_x = 0.2861
rms_y = 0.4682
parameters after fit:
…
Save config in directory []: sony2k
\sony2k\m_set.ini saved
```



```
[Lasercal]
f_lam0 = 405.0
f_scalxy = 1.0
b_fitxy = 1
i_imx = 3840
i_imy = 2160
f_f0 = 24.0
f_pix = 0.00832
f_grat = 600.0
f_rotdeg = 0
i_binning = 1
s_comment = Sony alpha, f 24 mm
s_infile = Sony/calib4k
s_outfil = Sony/calib4k

[Calib]
x00 = 1913.708701174928
y00 = 1101.8413972406822
rot = -0.006204757102004484
disp0 = 0.6329968669977126
a3 = 4.604446517713027e-08
a5 = 1.4888012049367797e-14
```

```
Sony alpha, f 24 mm
Sony/calib4k.txt
rms_x = 1.8611 ☹
rms_y = 3.5570 ☹
parameters after fit:
…
Save config in directory []: sony
\sony\m_set.ini saved
```

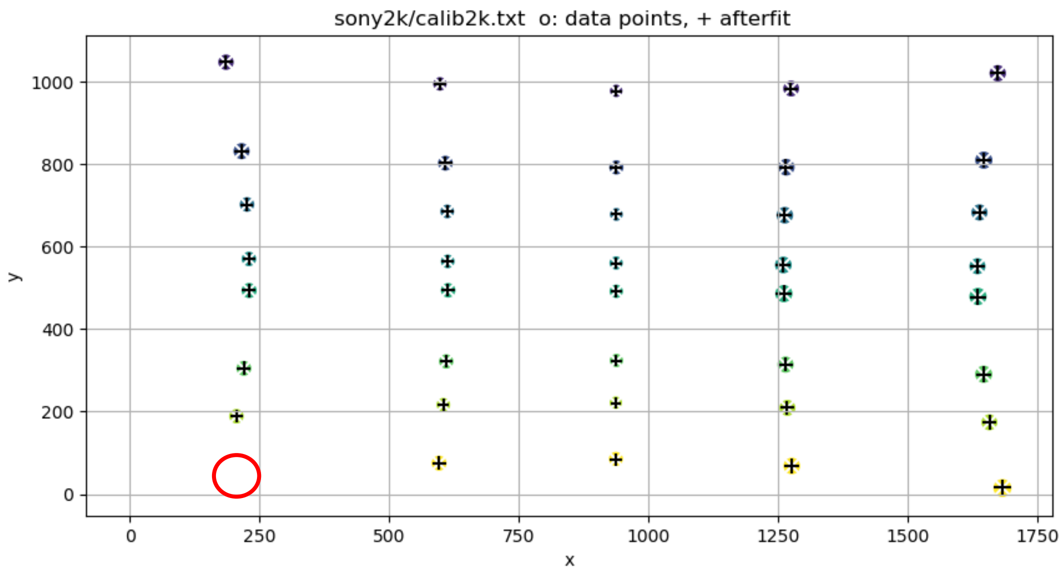
Note:

The parameter names in the configuration are preceded by a letter indicating the type of the variable. This simplifies the conversion. f_ is for floating point parameters, i_ for integers, s_ for strings. b_ is for boolean variables, **True** replaced by 1 and **False** by 0.

After finishing a calculation, the relevant parameters are stored in the file m_set.ini. It corresponds to a Windows-style *.INI and contains the input parameters under the heading [lasercal] and the results of the least square fit under the header [Calib]. The input and output directories you can change during the running of the program l_calib.py. Other changes have to be done manually by editing the file m_set.ini with a text editor such as Notepad.
Next time you run l_calib6.py the new values will be used.

Binning, directory structure

If you have different cameras, lenses, image formats you may need more than one configuration. In this case you can use different subdirectories for each case, in this example there is a directory "sony" for 4k images and a directory "sony2k" for binned 2k images. The only difference in the configuration is the binning = 2 for the 2k images, as compared to binning = 1 for the 4k images.Of course the results for [Calib] are different for the different binning factors; the values of x00 and y00 are 2x larger for the unbinned images, the values of a3 and a5 4x respectively 16x smaller for the unbinned images. Notice also that the errors are much smaller in the 2k images. This is because the first line in the lowest laser spectrum was omitted in the 2k analysis (this is permitted for the first or the last order, but you may not skip intermediate orders), since it overlapped partly with the image caption:



The results are included in the configuration file, because they will be used in the analysis of the meteor spectra, for the calculation of the distortion to transform to orthographic projection. This will follow in the next step.

Conclusion

The program l_calib5 or l_calib6.py allows calibration of meteor spectra with the same results as by the earlier EXCEL method. It has the following advantages:

- It runs on different platforms
- It is free (no license needed)
- It is much faster than EXCEL (about a factor 10 and less critical dependent on start values)
- It does not use language dependent macros, which made the EXCEL version unusable in different countries (Japan, Ukraine, China)

- Once finished, it allows the treatment of meteor spectra without different software (IRIS, ImageTools, ISIS, speeding up the treatment of meteor spectra.

Important note

The present versions are preliminary, only tested by myself. They probably contain many bugs. Comments about bugs or improvements are highly appreciated.
Please report to: martin-dubs@bluewin.ch

Content