RASCAL

Release 0.3.0

Josh Veitch-Michaelis, Marco Lam

Aug 04, 2021

INSTALLATION

Wavelength calibration is the proess of modelling a spectrometer such that every pixel can be mapped to a wavelength. This is normally perfomred by either amnuially matching spectral peaks to a line atlas, of using cross-correlation with a known lamp spectrum.

Manual calibration is tedious, particularly for arc lamps with many emission lines. Cross-correlation or template matching methods are often built for specific instruments and make assumptions about the long term stability of the calibration lamp.

RANSAC- **A**ssisted **S**pectral **CAL**ibration aims to produce a fit model **automatically** from an arc lamp spectrum with only **minimal prior information**. RASCAL is inspired by the method of Song (2018).

RASCAL is written in **Python 3** and has minimal dependencies. It has be tested for the ASPIRED pipeline and with other scientific and commercial spectra.

Note: How fast is it? RASCAL takes seconds to run.

How accurate do the initial conditions need to be? We usually assume 10-20% uncertainty in the dispersion and spectral range.

What sources does it work with? Anything. We have included the NIST lines by default.

What if I don't know the lamp? Run RASCAL multiple times with different lammp options and inspect the outputs with the lowest errors.

CHAPTER

ONE

BASIC USAGE

The bare minimum example code to to get a wavelength calibration:

```
import numpy as np
from scipy.signal import find_peaks
from astropy.io import fits

from rascal.calibrator import Calibrator
from rascal.util import refine_peaks

# Open the example file
spectrum2D = fits.open("filename.fits")[0].data

# Get the median along the spectral direction
spectrum = np.median(spectrum2D, axis=0)

# Get the spectral lines
peaks, _ = find_peaks(spectrum)

# Set up the Calibrator object
c = Calibrator(peaks)

# Load the Lines from library
c.add_atlas(["Xe"])

# Solve for the wavelength calibration
best_polyfit_coefficient, rms, residual, peak_utilisation = c.fit()

# Produce the diagnostic plot
c.plot_fit(spectrum, best_polyfit_coefficient)
```

Some more complete examples are available in the *Quickstart* tutorial.

CHAPTER

TWO

HOW TO USE THIS GUIDE

To start, you're probably going to need to follow the *Installation* guide to get RASCAL installed on your computer. After you finish that, you can probably learn most of what you need from the tutorials listed below (you might want to start with *Quickstart* and go from there). If you need more details about specific functionality, the User Guide below should have what you need.

We welcome bug reports, patches, feature requests, and other comments via the GitHub issue tracker.

CHAPTER

THREE

USER GUIDE

3.1 Installation

This package is registered on PyPI, you can always install the developement version directly from Github.

3.1.1 PyPI

pip install rascal

3.1.2 Github

Install from the stable/versioned main branch

pip install git+https://github.com/jveitchmichaelis/rascal.git

or from the development branch

pip install git+https://github.com/jveitchmichaelis/rascal.git@dev

3.1.3 Poetry

3.2 3-minute Summary

3.2.1 Working Principle

RASCAL begins by **enumerating all possible combinations of input peaks and emission lines** in a catalogue. Emission lines are initially filtered by prior knowledge of the system, for example, approximate dispersion and spectral range. At this point, we assume that any peak could match any catalogue line.

Most spectrometers are linear to first order. We can use the Hough Transform to find line fits to the enumerated peak/emission lines. Peaks in the **Hough Transform** accumulator correspond to lines in Cartesian space that pass through the most points. We constrain the line search given estimates of the system parameters.

The fitted lines effectively show a piece-wise linear fit to the non-linear dispersion of the spectrometer. This is also evident in the accumulator plot as two groups of local maxima. **We therefore choose the most common match for each peak, considering all of the top-N lines**.

The output of RASCAL is a set of matching peak locations (pixels) and lines (wavelengths), and a calibration model. Users are free to select their own fitting function.

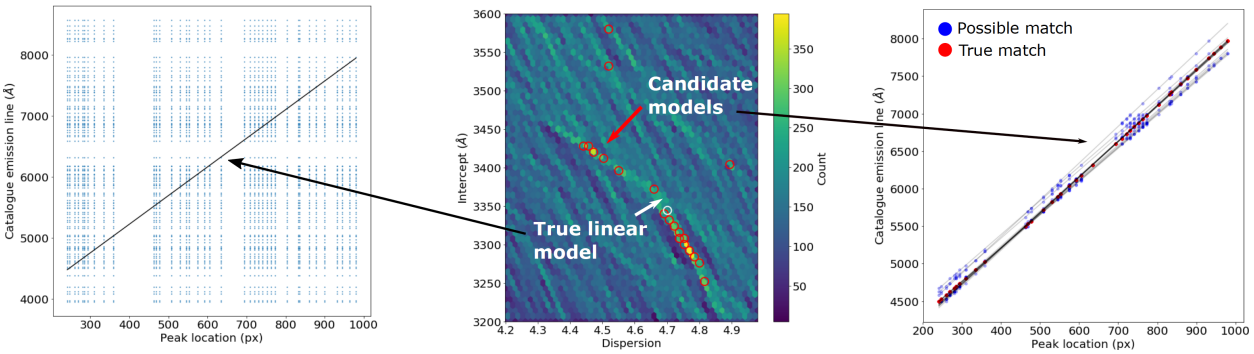


Fig. 1: *Left*: All possible combinations of arc lines (wavelengths) and peaks (pixels).
Middle: Peaks in the Hough Transform.
Right: The top 20 linear solutions and the peak/emission line pairs which are agreed to within 15A.

3.2.2 RANSAC & Hough transform

RANdom **SA**mple **C**onsensus is a popular algorithm from computer vision, used for performing robiust model fitting in the presence of large numbers of outliers. We use RANSAC to determine corresponding peaks and emission lines, posed as a noisy matching problem.

Hough Transform is a technique used for feature extraction in image processing and analysis. In our usage, the solution appears as overdensities in the Hough space.

Combining the two, RANSAC fits a (non-linear) model repeatedly from a minimum random sample of (Hough transformed) points. This process repeats many times, and we keep the best fit. Once RANSAC has produced a best fit solution, we can go through the detected peaks and choose the wavelength from the catalogue which matches it (to within a tolerance limit).

See *Hough Transform* and *RANSAC* in more details.

3.3 Hough Transform

Hough transform is a technique used for feature extraction in image processing and analysis. The general and more in-depth explanation can be found on Wikipedia. The following explains the specifics of its usage in RASCAL. We will examine the settings for Hough transform, also known as *hyperparameters* and how it is used to perform wavelength calibration. Please note that because for a given spectrum, each pixel has a unique wavelength, we are using the simple $y = m \times x + c$ parameterisation instead of using $D = r \times \theta$ which allow vertical lines in the hyperparameter space.

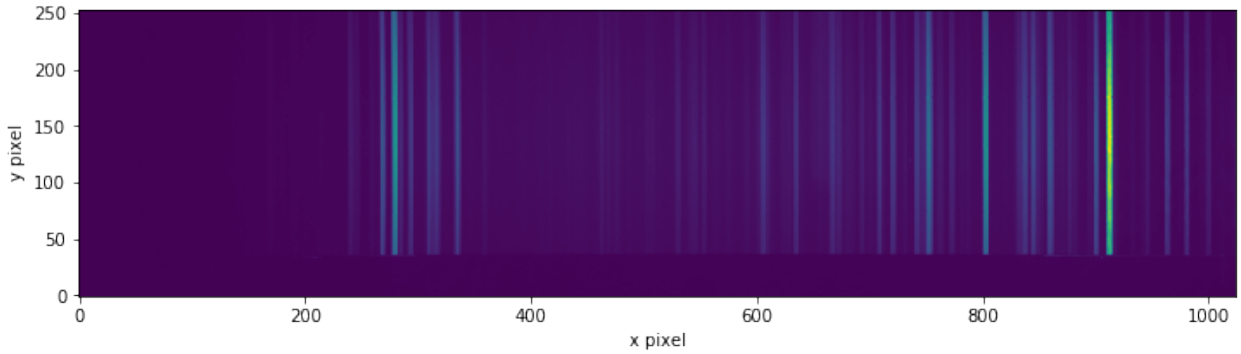
A version of this explanation with inline codes is available with the Jupyter Notebook at the GitHub repository.

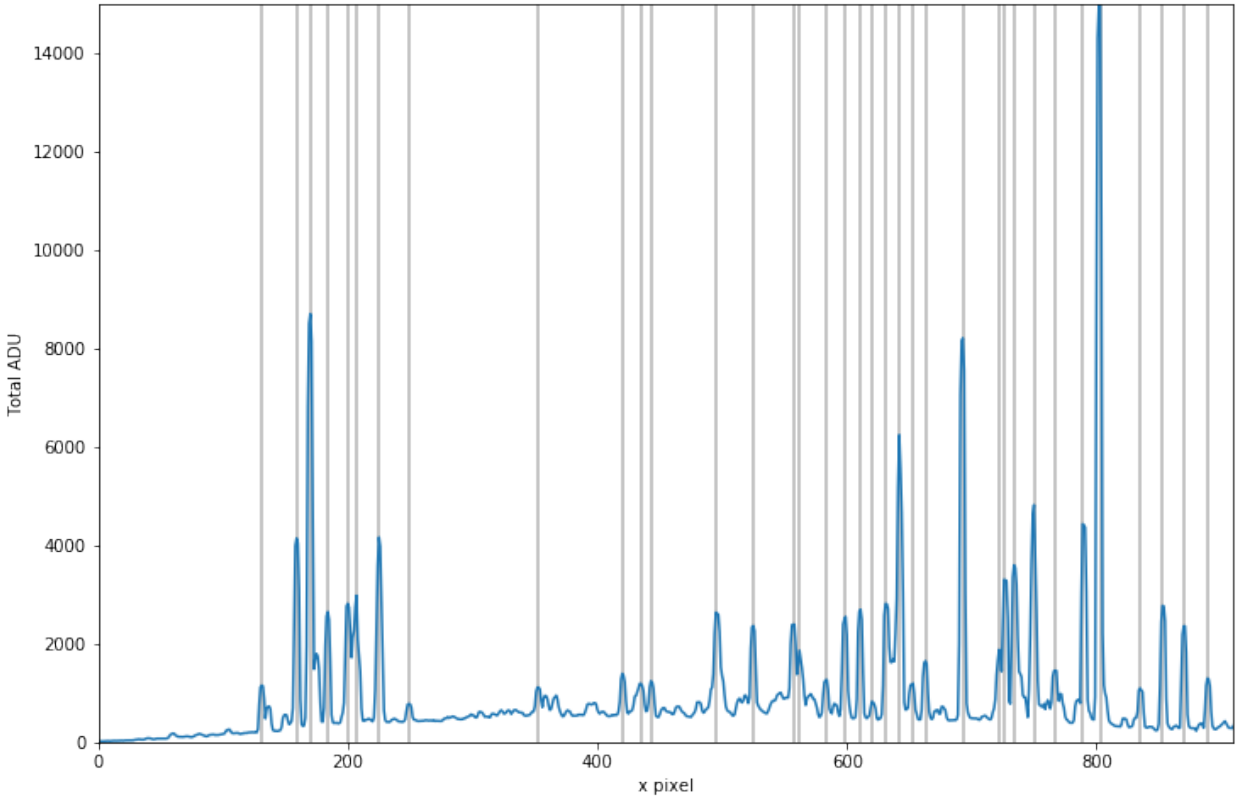
3.3.1 Spectrum and Lines

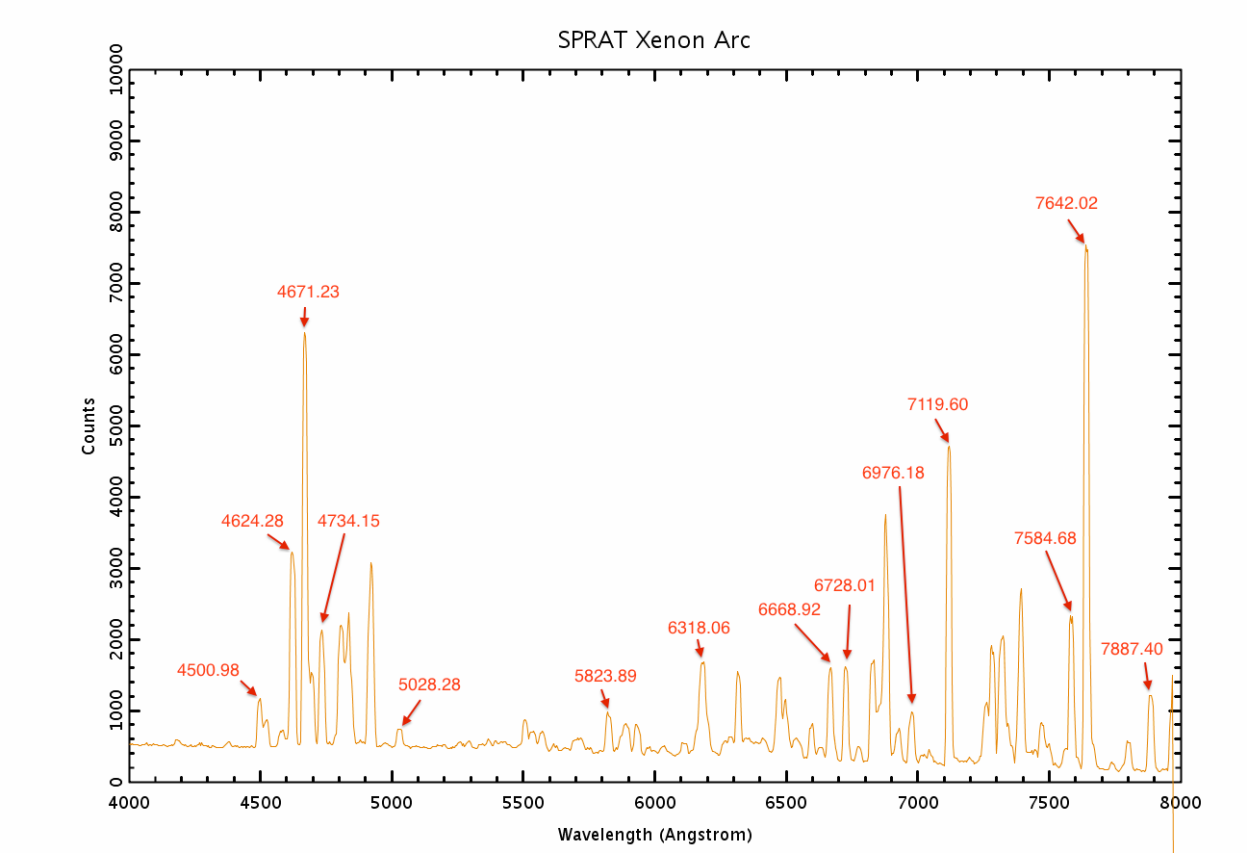
Before jumping into the transformation, we need some data. The image below is a 2D image of arc lines dispersed in the x-direction.

The first 110 pixels are trimmed because it is blank. Then, the median of along the y direction is taken to represent the arc spectrum and the locations of the peaks are identified for Hough transform.

This is what the arc should look like (taken from the Liverpool Telescope SPRAT instrument page). Note the first sizable peak on the left is at 4500A, and the three small but clear peaks to the far right are 7807A, 7887A and 7967A (partly truncated):


y pixel
x pixel
0
50
100
150
200
250
0
200
400
600
800
1000


Total ADU
x pixel
0
2000
4000
6000
8000
10000
12000
14000
0
200
400
600
800



3.3.2 Accumulator

Now we instantiate a `calibrator`. We can see from the above plot, and recognising some key lines that our range is around 4000 to 8000A. To visualise the process, first, we plot a few 2D Hough `accumulators` matrices with different values of the `num_slopes` – 10, 50, 100 and 500, to generate all the possible Hough pairs (gradient-interception pair). The `accumulators` look like the following for the 4 values of the slope resolution (clockwise from the top left). Each point on the plot corresponds to the gradient and the interception of the support line that goes through a wavelength-pixel pair within the given tolerance limit.

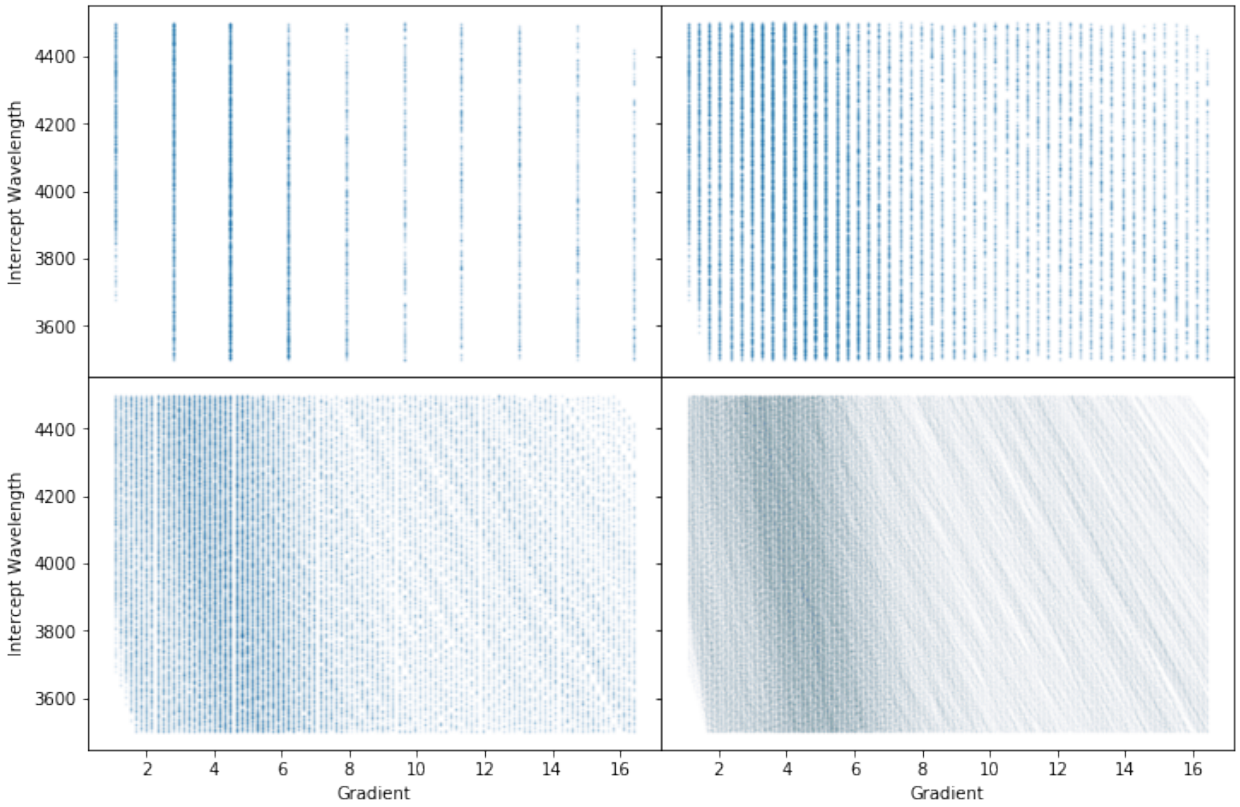
We have specified a spectral range (4000-8000A), and provided the number of pixels in the spectral direction (914 pix). Rascal guesses sensible values of the intercept depending on how confident your range guess is, e.g. the default of `range_tolerance` is 500A, giving the possible range of interception from 3500A to 4500A. The linearity in the pixel-wavelength relation has to be included to allow the gradual change in the gradient from one end to the other of the spectrum.

From this we know that the maximum gradient must be:

$$\frac{\text{Maximum Wavelength} - \text{Tolerance} - \text{Minimum Interception}}{\text{Number of Pixels} \,/\, \text{Linearity Tolerance Threshold}}$$

$$= \frac{(8000 + 500) - (4000 - 500)}{914/3}$$

$$\approx 16.41$$



and the minimum:

$$\begin{aligned}
&\frac{\text{Maximum Wavelength} + \text{Tolerance} - \text{Maximum Interception}}{\text{Number of Pixels} \times \text{Linearity Tolerance Threshold}} \\
&= \frac{(8000 - 500) - (4000 + 500)}{914 \times 3} \\
&\approx 1.09
\end{aligned}$$

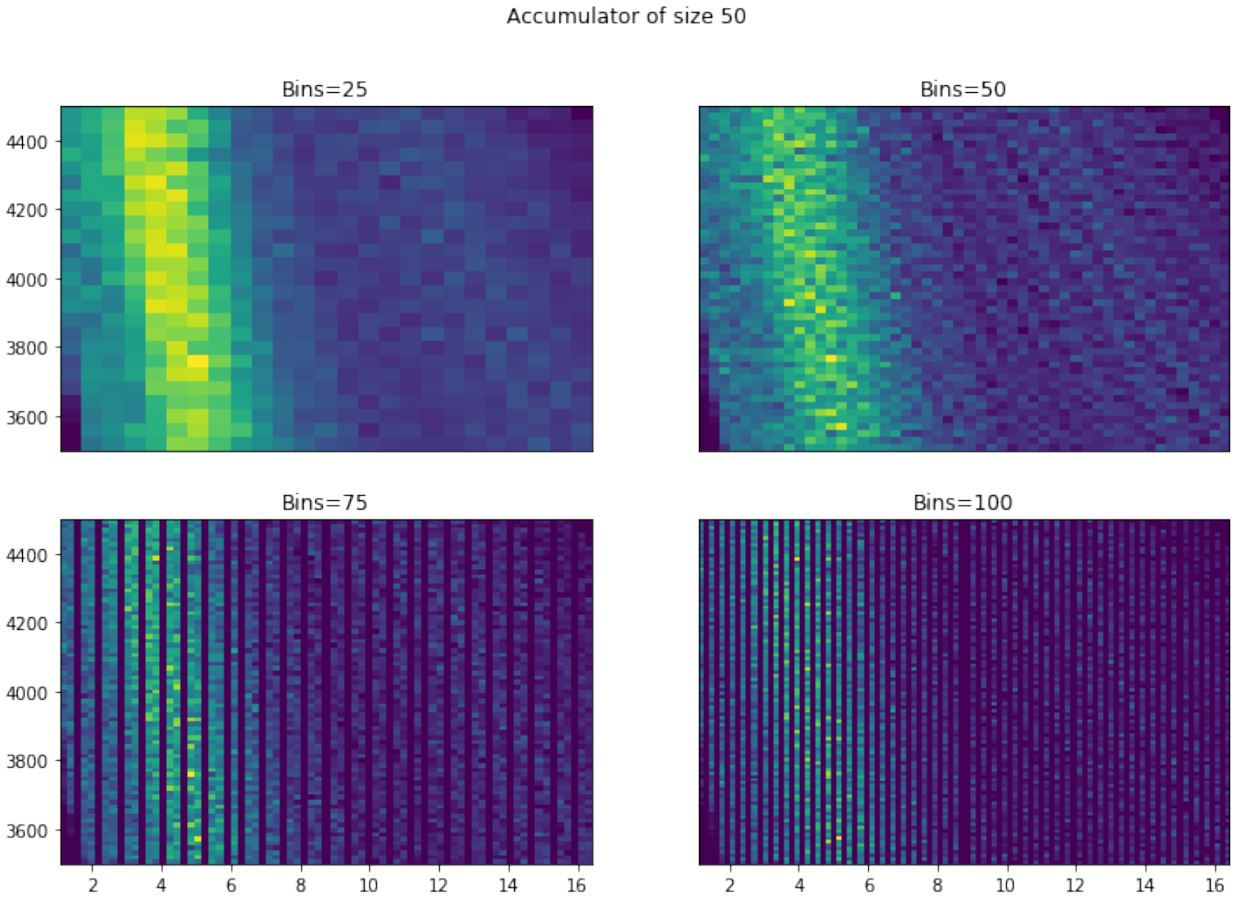
Note: We have chosen to enumerate over dispersion - the algorithm works by choosing a range of dispersion values to check, and finds the intercept values that supports them (i.e. we solve $y = m \times x + c$ for c given m). We could equivalently search over a range of initial wavelengths and calculate what dispersions would support that.

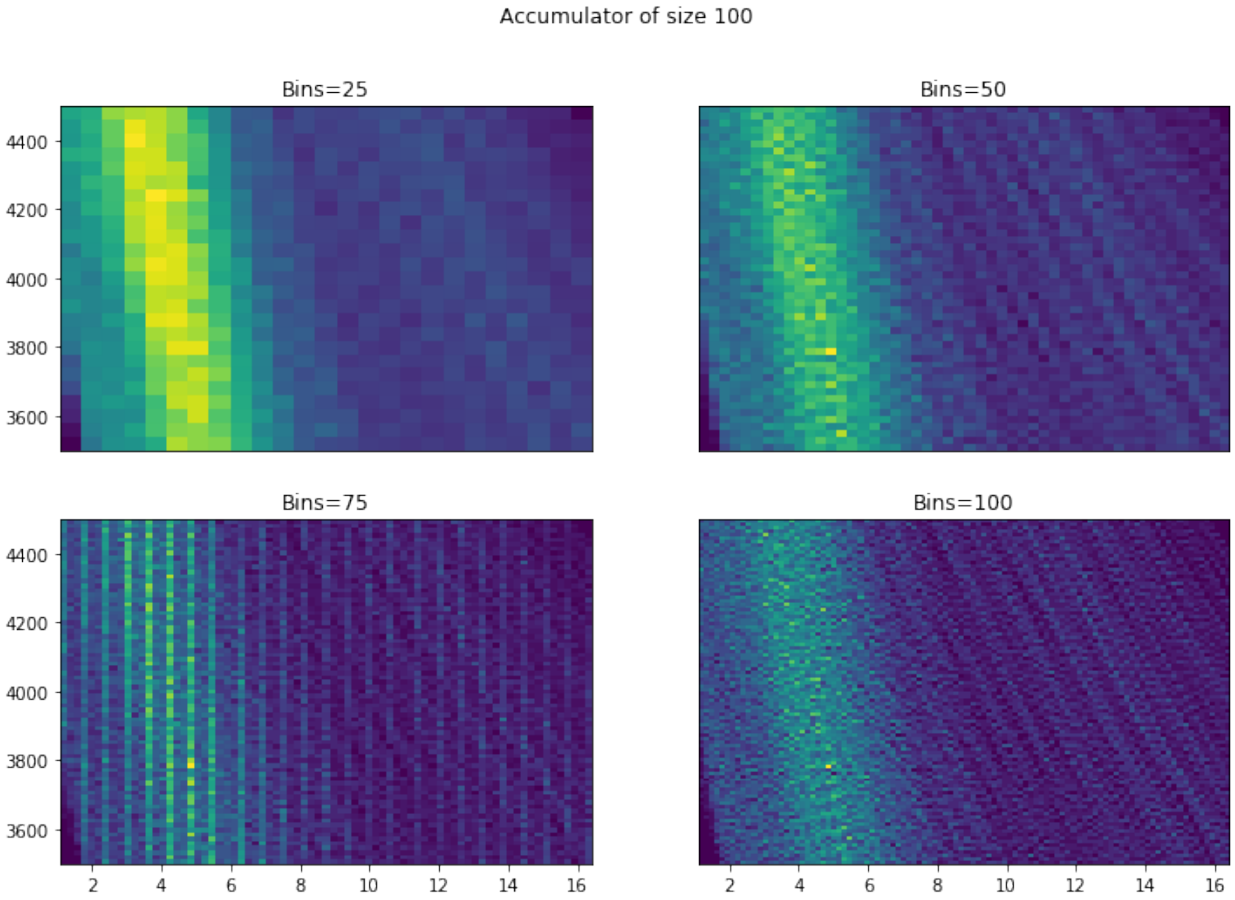
3.3.3 Binning

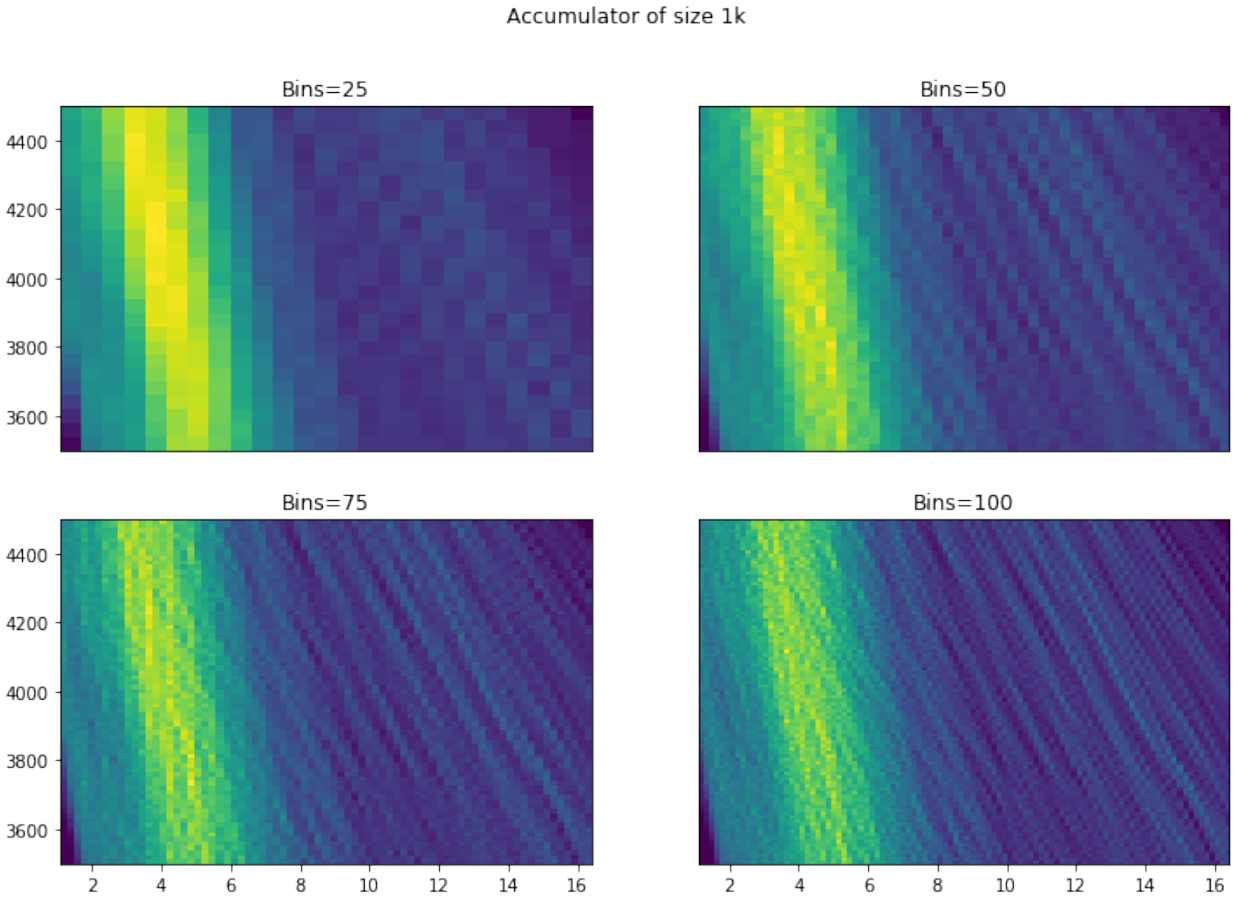
To better visualise these scattered data points, we can present them in 2D histogram of the accumulators at different resolution. The 4 figures below show from the lowest resolution Hough transform accumulator with 50 gradients to the highest with 10000 gradients:

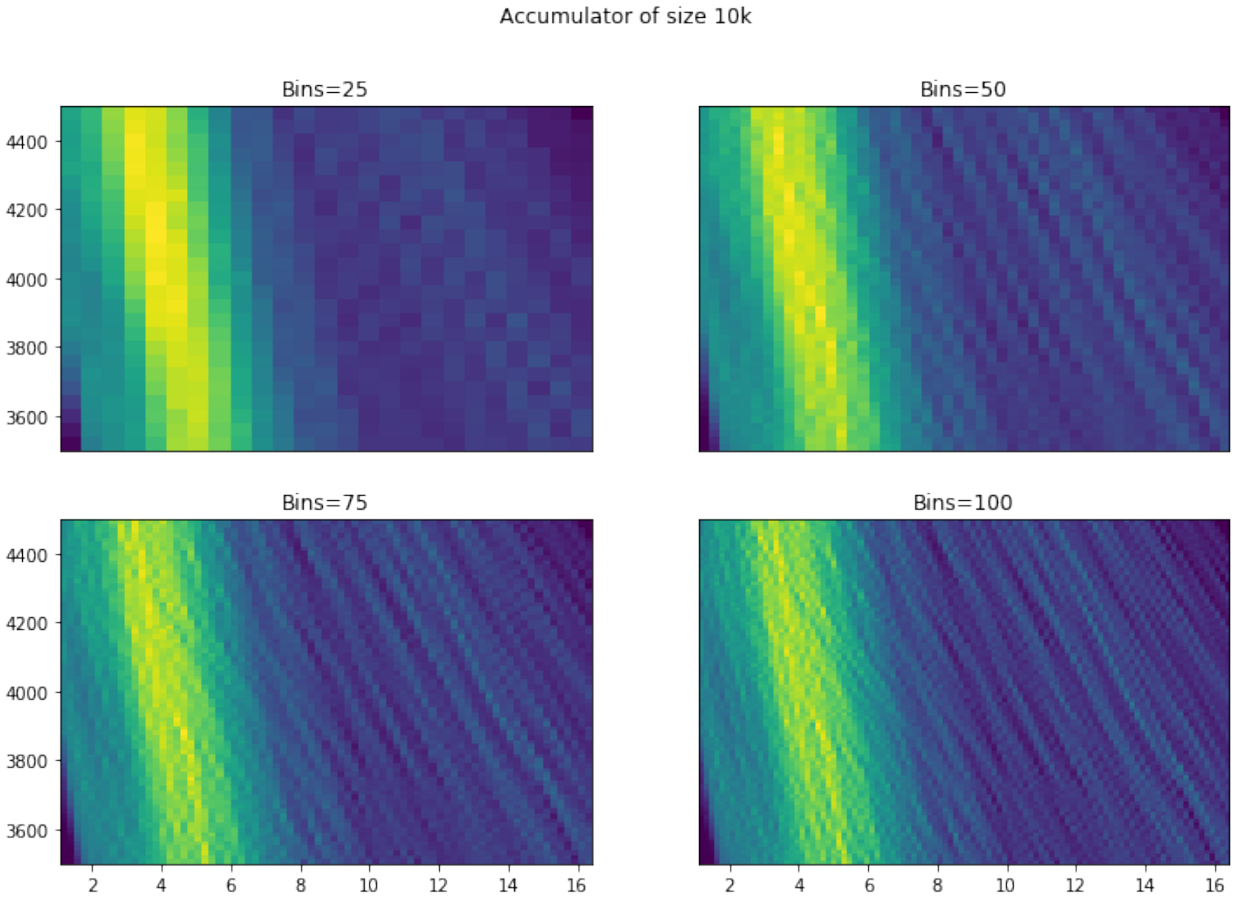
Two things stand out here. Firstly, it is important to have a high resolution Hough transform so that we do not miss possible solutions. Secondly, more bins are good - we can see from 50+ that we start to identify a possible solution (the small hotspot/peak). However, while there does seem to be an improvement going to around 75 bins, going to 100 isn't visually very different. Similarly there doesn't appear to be a huge advantage in choosing a very small bin

We can do a small grid search over the two variables: slope resolution (defined by the number of slope bins and wavelength range) and the histogram bin resolution.


Accumulator of size 50
Bins=25
Bins=50
Bins=75
Bins=100
4400
4200
4000
3800
3600
2
4
6
8
10
12
14
16

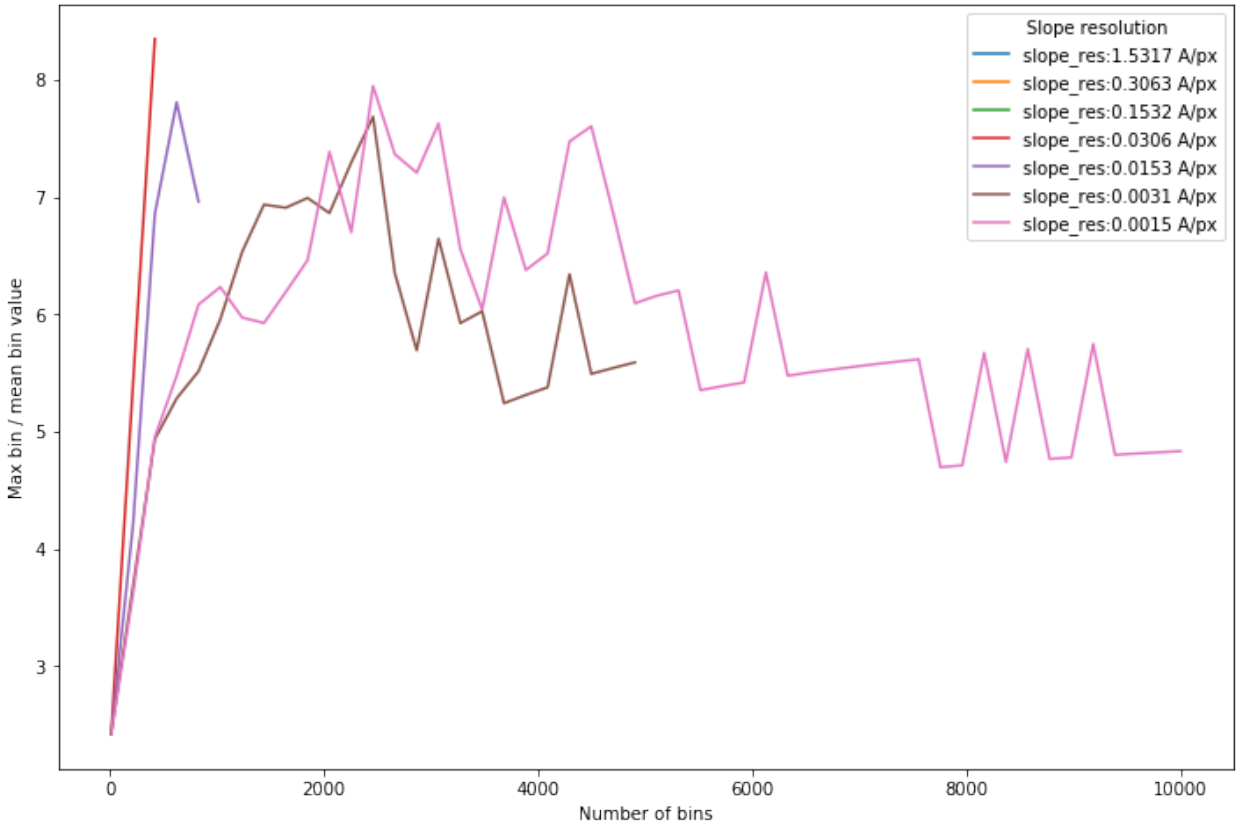

Accumulator of size 100
Bins=25
Bins=50
Bins=75
Bins=100
4400
4200
4000
3800
3600
2
4
6
8
10
12
14
16


Accumulator of size 1k
Bins=25
Bins=50
Bins=75
Bins=100
4400
4200
4000
3800
3600
2
4
6
8
10
12
14
16


Accumulator of size 10k
Bins=25
Bins=50
Bins=75
Bins=100
4400
4200
4000
3800
3600
2
4
6
8
10
12
14
16

3.3.4 Goodness-of-fit

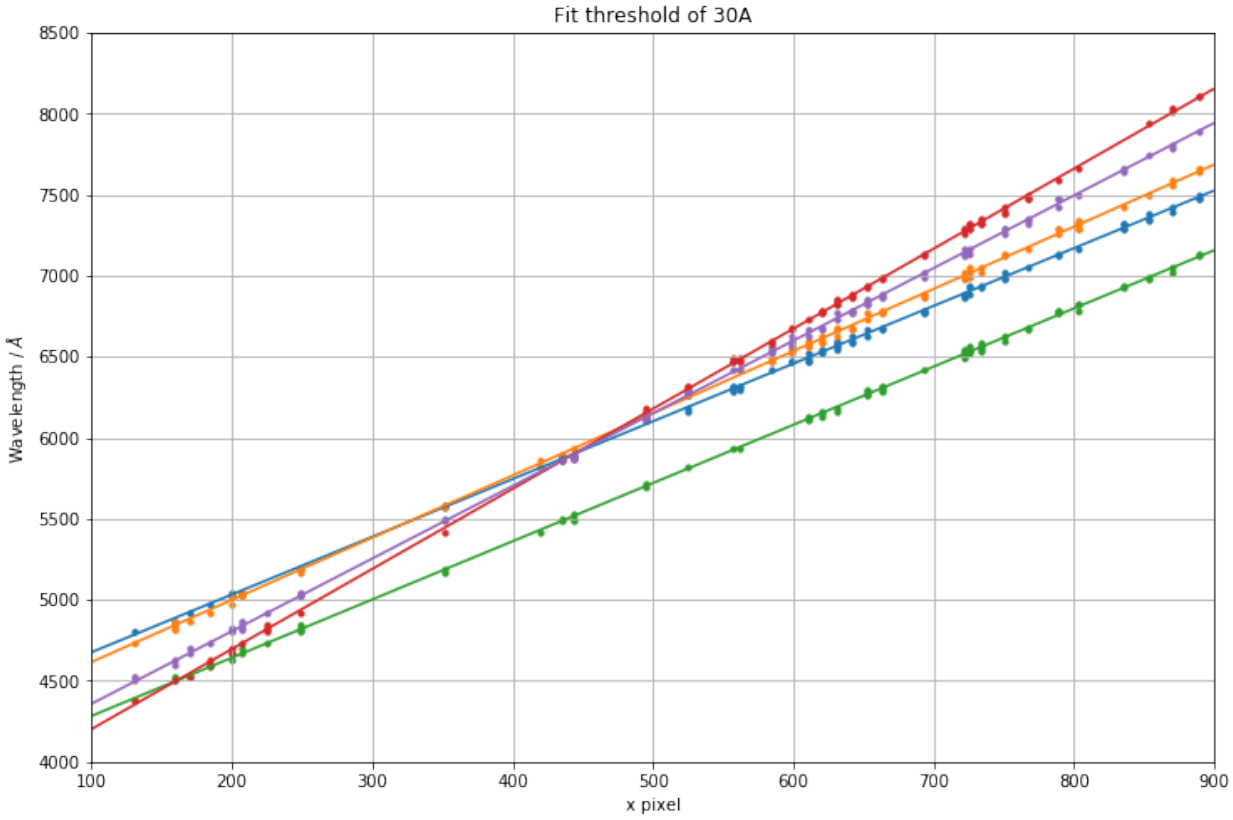
These curves are essentially showing how "good" the best line fit is. We don't want a histogram where there are lots of potential choices. It looks empirically that the number of bins should be around half the number of gradients to maximise contrast. A sensible default of 1000 slopes with 500 bins is appropriate (at least for this dispersion and range).

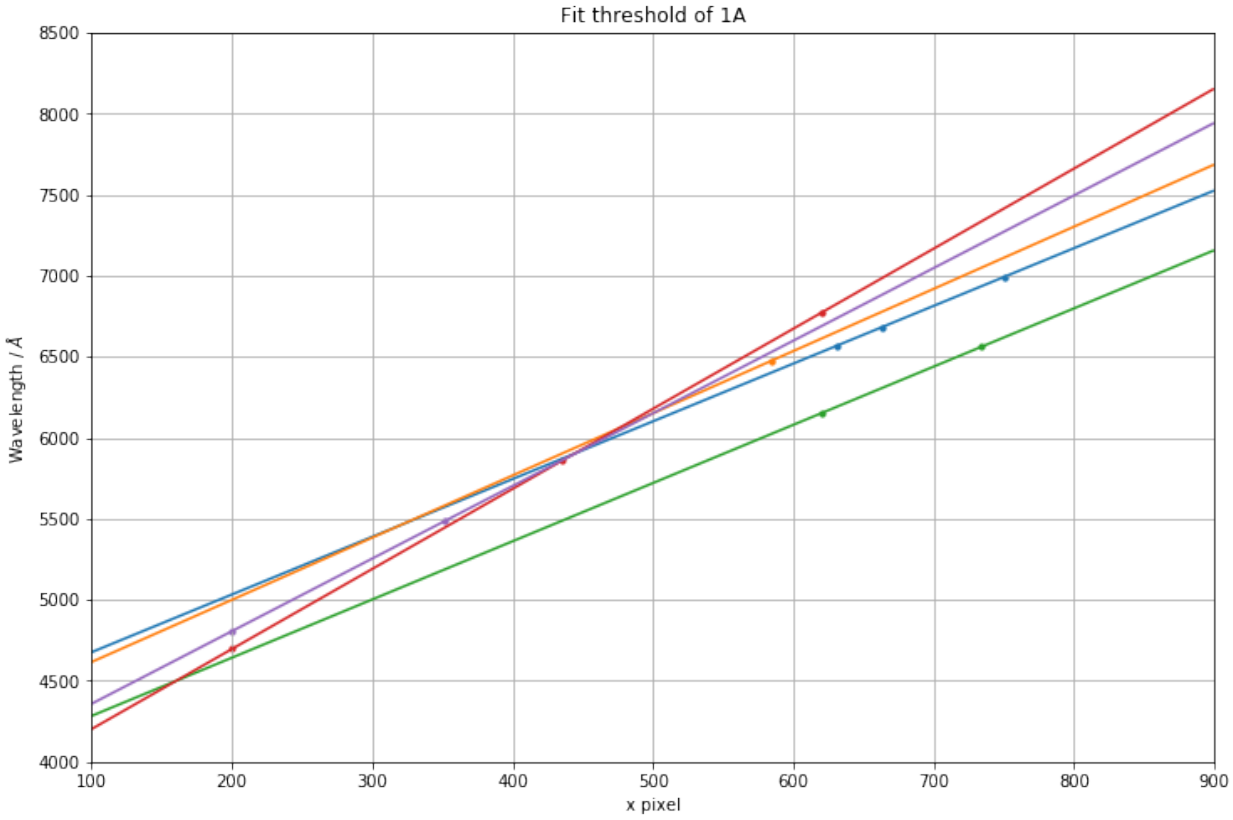


3.3.5 Threshold setting

We want to get a set of points with a high precision - that is, we want as many true positives as possible, at the risk that we have included a few false positives too. So it is important at this stage, that the acceptance threshold is quite high - as much as 50A or even 100A because this does not take into account for the non-linearity. Once we have some first guesses, the solutions can be fine tuned later quickly on a much reduced set of data.

Below illustrate how the threshold can completely throw good/potential solution out of the window.


Fit threshold of 30A
Wavelength / Å
x pixel
8500
8000
7500
7000
6500
6000
5500
5000
4500
4000
100
200
300
400
500
600
700
800
900


Fit threshold of 1A
Wavelength / Å
x pixel
8500
8000
7500
7000
6500
6000
5500
5000
4500
4000
100
200
300
400
500
600
700
800
900

3.4 Random sample consensus (RANSAC)

RANSAC is an algorithm for robust fitting, popularised by the computer vision community (Fischler and Bolles, 1981) .

3.4.1 Overview

Suppose we have a dataset that contains good points (inliers) and spurious points (outliers). If we randomly sample our dataset, eventually we will pick a set of points which only contain inliers. In that case, a model fit to this sample should also fit the majority of the rest of our data. Conversely if we sample an outlier by mistake, the fit will not agree with the rest of the dataset.

This is conceptually very simple, but works extremely well in practice. Typically the sample size is the minimum required to fit the model, so for a linear model we would draw 2 random points. If you know the percentage of outliers in your data it is possible to calculate some statistical estimate of how many iterations is required before an inlier-only sample is drawn (to some degree of confidence). In practice we never know the inlier:outlier ratio and we just try "a lot" of samples (e.g. 500).

We also need to decide how to score a particular sample fit. In the original version of RANSAC, the number if inliers corresponding to a fit is used. More inliers equals a better fit. In RASCAL we use a slight modification called M-SAC (Torr and Zisserman, 1996) which also weights inliers by the fit error. This is useful because it acts as a tie-breaker between two fits with the same number of inliers.

For more information, you can read the Wikipedia article .

3.4.2 User configurable options

If you find that you're not getting good fits and feel that more iterations would help, you can try that. Simply increase `max_tries` when you call `Calibrator.fit()`. As a first pass, try increasing by an order of magnitude to see if things improve, if that works then you can lower the number of iterations until you reliably get a good fit every time.

You can also adjust the threshold that RANSAC uses to define inliers (`ransac_tolerance`, in Angstrom). This should normally be set quite small, because for most science-grade instruments we expect a very good model fit. If you think there is likely to be a lot of jitter in your peak finding, or your spectrometer is very low resolution, then you might want to relax this.

Other settings can be twiddled in `Calibrator.set_ransac_properties()`, but for the most part you shouldn't need to adjust these and they are handled internally by the Calibrator.

3.5 Quickstart

To demonstrate a more custom-built general reduction, we are extending on the bare minimum example code on the homepage. Please bear in mind this does not include all the example configurability of RASCAL.

To begin, we need to import all the necessary libraries and load the spectrum:

```
import numpy as np
from scipy.signal import find_peaks
from astropy.io import fits

from rascal.calibrator import Calibrator
from rascal.util import refine_peaks
```

(continues on next page)

(continued from previous page)

```
# Open the example file
spectrum2D = fits.open("filename.fits")[0].data

# Get the median along the spectral direction
spectrum = np.median(spectrum2D, axis=0)
```

The emission lines from the arc lamp spectrum have to be idenified before wavelength calibration is possible. RASCAL does not contain a peaking finding function, in this example we are using the one available with SciPy which returns the peak accurate to the nearest pixel. For that reason RASCAL has a *refine_peaks* function to get the peak positions at subpixel level by fitting Gaussians over a few pixels on either side of the peaks.

```
# Get the spectral lines
peaks, _ = find_peaks(spectrum)
peaks_refined = refine_peaks(peaks)
```

To start the calibration, we need to initialise the calibrator by passing in the peak locations. The **calibrator properties** can be set altogether (it can be set later or modify by using *set_properties*). The *num_pix* or *pixel_list* are necessary for the calibration because the calibrator is abstracted from the properties of the detector array, however, for a detector plane with multiple detectors that are separated by a fixed number of pixels, the wavelength calibration will be completely unsable without taken into account of the step functions intoduced by the chip gaps. The *num_pix* is also important for checking the monotonicity of the entire range of the fitted pixel-to-wavelength function.

```
# Set up the Calibrator object
c = Calibrator(peaks_refined,
               spectrum)
c.set_calibrator_properties(num_pix=len(spectrum),
                            plotting_library='matplotlib',
                            log_level='info')
```

Once the Calibrator is provided with the peaks, and optionally the arc spectrum, a diagnostic plot for the arc spectrum can be plotted with

```
c.plot_arc()
```

To distinguish from the Hough transform and fitting from the calibrator, in manufacturing term, the calibrator is the factory, the Hough transform is the pre-processing before entering production line, while the fitter is the machine. Therefore, apart from setting the calibrator properties, we also need to set the **Hough transform properties**, and the **RANSAC properties**.

```
c.set_hough_properties(num_slopes=10000,
                       xbins=1000,
                       ybins=1000,
                       min_wavelength=3500.,
                       max_wavelength=9000.,
                       range_tolerance=500.,
                       linearity_tolerance=50)

c.set_ransac_properties(sample_size=5,
                        top_n_candidate=8,
                        filter_close=True)
```

The calibration still does not know what it is calibrating against, so we have to provide the arc lines or use the built-in library by providing the Chemical symbols.

```
# Load the Lines from library
c.add_atlas(["Xe"],
            min_intensity=10,
            min_distance=10,
            constrain_poly=True)
```

With everything set, we can perform the Hough transform on the pixel-wavelength pairs

```
c.do_hough_transform()
```

Finally, we can do the fitting, there are still a few more parameters that were not configured in the *set_ransac_properties*. The distinction is that, RANSAC properties concern the parameter space and the sampling of the fit, while the fitting function only concerns the properties of the polynomial.

```
# Solve for the wavelength calibration
(best_polyfit_coefficient, matched_peaks, matched_atlas, rms, residual,
 peak_utilisation, atlas_utilisation) = c.fit(max_tries=1000, polydeg=7)
```

Show the wavelength calibrated spectrum.

```
# Produce the diagnostic plot
c.plot_fit(best_polyfit_coefficient,
           spectrum,
           plot_atlas=True,
           log_spectrum=False,
           tolerance=5.)
```

Show the parameter space in which the solution searching was carried out.

```
c.plot_search_space()
```

3.6 Example - LT/SPRAT

The spectrogrphy SPRAT on the Liverpool Telescope is our primary testing instrument. It is a low-resolution high-throughput spectrograph employing a VPH grating and prism (grism). In the following, we explain the wavelength calibration procedure step-by-step.

1. Initialise the environment and the line list (see the other examples for using the NIST line list) for the data proecessing

```
import matplotlib.pyplot as plt
import numpy as np
from astropy.io import fits
from scipy.signal import find_peaks
from scipy.signal import resample

from rascal.calibrator import Calibrator
from rascal.util import refine_peaks

atlas = [
    4193.5, 4385.77, 4500.98, 4524.68, 4582.75, 4624.28, 4671.23, 4697.02,
    4734.15, 4807.02, 4921.48, 5028.28, 5618.88, 5823.89, 5893.29, 5934.17,
```

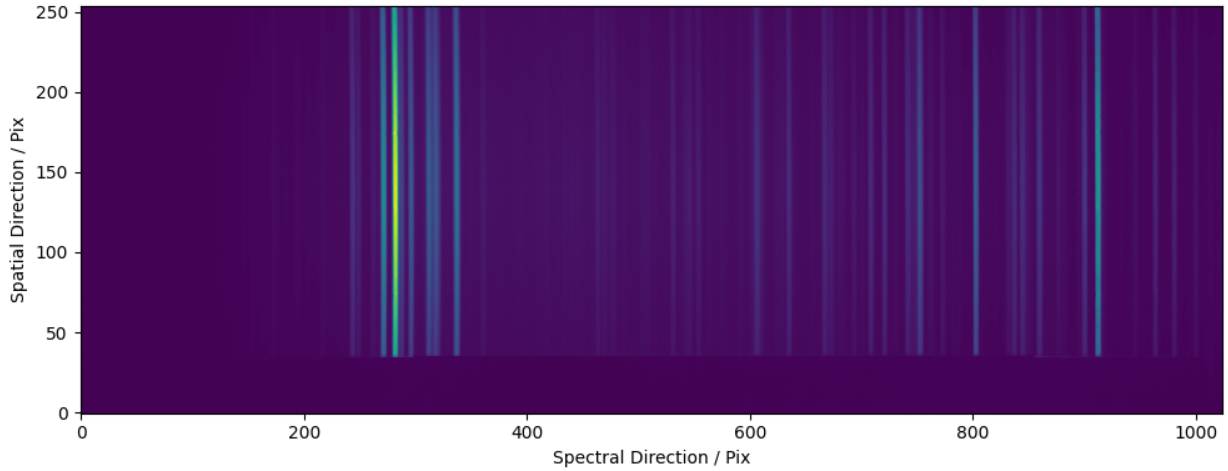
(continues on next page)

(continued from previous page)

```
    6182.42, 6318.06, 6472.841, 6595.56, 6668.92, 6728.01, 6827.32, 6976.18,
    7119.60, 7257.9, 7393.8, 7584.68, 7642.02, 7740.31, 7802.65, 7887.40,
    7967.34, 8057.258
]

element = ['Xe'] * len(atlas)
```

2. Load and inspect the arc image

```
data = fits.open('data_lt_sprat/v_a_20190516_57_1_0_1.fits')[0].data

plt.figure(1, figsize=(10, 4))
plt.imshow(np.log10(data.data), aspect='auto', origin='lower')
plt.tight_layout()
```



3. Normally you should be applying the trace from the spectral image onto the arc image, but in this example, we identify the arc lines in the middle of the frame.

```
spectrum = np.median(data[110:120], axis=0)

peaks, _ = find_peaks(spectrum, height=500, distance=5, threshold=None)
peaks_refined = refine_peaks(spectrum, peaks, window_width=5)
```

4. Initialise the calibrator and set the properties. There are three sets of properties: (1) the calibrator properties who concerns the highest level setting - e.g. logging and plotting; (2) the Hough transform properties which set the constraints in which the trasnform is performed; (3) the RANSAC properties control the sampling conditions.

```
c = Calibrator(peaks_refined, spectrum)

c.set_calibrator_properties(num_pix=len(spectrum),
                            plotting_library='matplotlib',
                            log_level='info')

c.set_hough_properties(num_slopes=5000,
                       xbins=100,
                       ybins=100,
```

(continues on next page)

(continued from previous page)

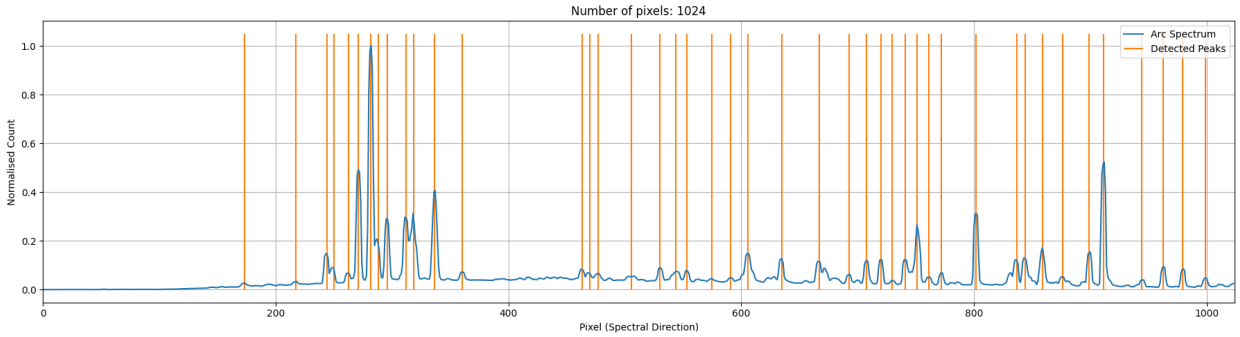
```
                        min_wavelength=3500.,
                        max_wavelength=8000.,
                        range_tolerance=500.,
                        linearity_tolerance=50)

c.set_ransac_properties(sample_size=5,
                        top_n_candidate=5,
                        filter_close=True,
                        ransac_tolerance=5,
                        candidate_weighted=True,
                        hough_weight=1.0)
```

The following *INFO* should be logged, where the first 3 lines are when the calibrator was initialised, and the last 3 lines are when the calibrator properties were set.

```
INFO:rascal.calibrator:num_pix is set to None.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
INFO:rascal.calibrator:num_pix is set to 1024.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
```

5. The extracted arc spectrum and the peaks identified can be plotted with the calibrator. Note that if only peaks are provided, only the orange lines will be plotted.

```
c.plot_arc()
```



6. Add the line list to the calibrator and perform the hough transform on the pixel-wavelength pairs that will be used by the RANSAC sampling and fitting.

```
c.add_user_atlas(elements=element,
                 wavelengths=atlas,
                 constrain_poly=True)
c.do_hough_transform()
```

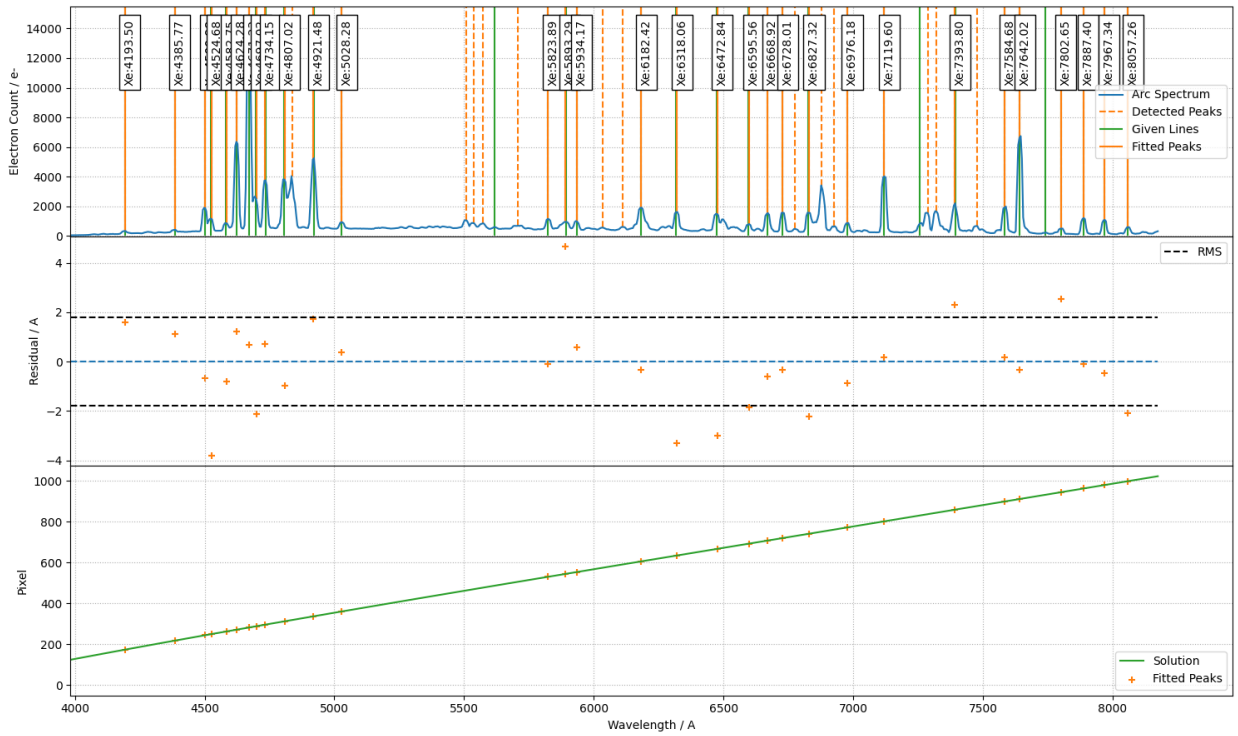
6. Perform polynomial fit on samples drawn from RANSAC, the deafult option is to fit with polynomial function.

```
(fit_coeff, matched_peaks, matched_atlas, rms, residual, peak_utilisation,
 atlas_utilisation) = c.fit(max_tries=1000)
c.plot_fit(fit_coeff,
           spectrum=spectrum,
```

(continues on next page)

(continued from previous page)

```
            plot_atlas=True,
            log_spectrum=False,
            tolerance=10.)
```



with some INFO output looking like this:

```
INFO:rascal.calibrator:Peak at: 4499.297026348797 A
INFO:rascal.calibrator:- matched to 4500.98 A
INFO:rascal.calibrator:Peak at: 4526.554911481822 A
INFO:rascal.calibrator:- matched to 4524.68 A
INFO:rascal.calibrator:Peak at: 4582.3986965959475 A
INFO:rascal.calibrator:- matched to 4582.75 A
INFO:rascal.calibrator:Peak at: 4622.359146063909 A
INFO:rascal.calibrator:- matched to 4624.28 A
INFO:rascal.calibrator:Peak at: 4670.358411620268 A
INFO:rascal.calibrator:- matched to 4671.23 A
INFO:rascal.calibrator:Peak at: 4699.197011259794 A
INFO:rascal.calibrator:- matched to 4697.02 A
INFO:rascal.calibrator:Peak at: 4733.787230028565 A
INFO:rascal.calibrator:- matched to 4734.15 A
INFO:rascal.calibrator:Peak at: 4771.916229880186 A
INFO:rascal.calibrator:Peak at: 4808.815218450723 A
INFO:rascal.calibrator:- matched to 4807.02 A
INFO:rascal.calibrator:Peak at: 4837.627044936143 A
INFO:rascal.calibrator:Peak at: 4921.104970950684 A
INFO:rascal.calibrator:- matched to 4921.48 A
INFO:rascal.calibrator:Peak at: 4972.668574650925 A
```

(continues on next page)

(continued from previous page)

```
INFO:rascal.calibrator:Peak at: 5029.378769376794 A
INFO:rascal.calibrator:- matched to 5028.28 A
INFO:rascal.calibrator:Peak at: 5075.231418755915 A
INFO:rascal.calibrator:Peak at: 5113.231338417058 A
INFO:rascal.calibrator:Peak at: 5194.825427316171 A
INFO:rascal.calibrator:Peak at: 5195.989773970972 A
INFO:rascal.calibrator:Peak at: 5258.353775628746 A
INFO:rascal.calibrator:Peak at: 5297.915852487534 A
INFO:rascal.calibrator:Peak at: 5343.780691305254 A
INFO:rascal.calibrator:Peak at: 5375.2984797500485 A
INFO:rascal.calibrator:Peak at: 5406.708402631906 A
INFO:rascal.calibrator:Peak at: 5508.132973975728 A
INFO:rascal.calibrator:Peak at: 5539.366089889084 A
INFO:rascal.calibrator:Peak at: 5572.823999338587 A
INFO:rascal.calibrator:Peak at: 5617.846667964729 A
INFO:rascal.calibrator:- matched to 5618.88 A
INFO:rascal.calibrator:Peak at: 5662.956860121537 A
INFO:rascal.calibrator:Peak at: 5701.816067831734 A
INFO:rascal.calibrator:Peak at: 5727.142609081708 A
INFO:rascal.calibrator:Peak at: 5754.74024326056 A
INFO:rascal.calibrator:Peak at: 5823.643795968694 A
INFO:rascal.calibrator:- matched to 5823.89 A
INFO:rascal.calibrator:Peak at: 5865.987482902671 A
INFO:rascal.calibrator:Peak at: 5891.953178770549 A
INFO:rascal.calibrator:- matched to 5893.29 A
INFO:rascal.calibrator:Peak at: 5932.939827113174 A
INFO:rascal.calibrator:- matched to 5934.17 A
INFO:rascal.calibrator:Peak at: 5980.090864138432 A
INFO:rascal.calibrator:Peak at: 6033.274764712287 A
INFO:rascal.calibrator:Peak at: 6109.999956392299 A
INFO:rascal.calibrator:Peak at: 6181.602224173326 A
INFO:rascal.calibrator:- matched to 6182.42 A
INFO:rascal.calibrator:Peak at: 6274.009092029612 A
INFO:rascal.calibrator:Peak at: 6304.457933473403 A
INFO:rascal.calibrator:Peak at: 6320.094607174053 A
INFO:rascal.calibrator:- matched to 6318.06 A
INFO:rascal.calibrator:Peak at: 6474.584829371679 A
INFO:rascal.calibrator:- matched to 6472.841 A
INFO:rascal.calibrator:Peak at: 6528.52442133471 A
INFO:rascal.calibrator:Peak at: 6596.24511848811 A
INFO:rascal.calibrator:- matched to 6595.56 A
INFO:rascal.calibrator:Peak at: 6668.468978515668 A
INFO:rascal.calibrator:- matched to 6668.92 A
INFO:rascal.calibrator:Peak at: 6727.387571039818 A
INFO:rascal.calibrator:- matched to 6728.01 A
INFO:rascal.calibrator:Peak at: 6828.796016028446 A
INFO:rascal.calibrator:- matched to 6827.32 A
INFO:rascal.calibrator:Peak at: 6975.731367185896 A
INFO:rascal.calibrator:- matched to 6976.18 A
INFO:rascal.calibrator:Peak at: 6881.94170925221 A
INFO:rascal.calibrator:Peak at: 6924.168896615056 A
INFO:rascal.calibrator:Peak at: 6976.673358856718 A
```

(continues on next page)

(continued from previous page)

```
INFO:rascal.calibrator:- matched to 6976.18 A
INFO:rascal.calibrator:Peak at: 7119.450515336577 A
INFO:rascal.calibrator:- matched to 7119.6 A
INFO:rascal.calibrator:Peak at: 7288.460358225626 A
INFO:rascal.calibrator:Peak at: 7320.807551289554 A
INFO:rascal.calibrator:Peak at: 7392.138488241831 A
INFO:rascal.calibrator:- matched to 7393.8 A
INFO:rascal.calibrator:Peak at: 7476.868102122697 A
INFO:rascal.calibrator:Peak at: 7585.306408966355 A
INFO:rascal.calibrator:- matched to 7584.68 A
INFO:rascal.calibrator:Peak at: 7643.122412222957 A
INFO:rascal.calibrator:- matched to 7642.02 A
INFO:rascal.calibrator:Peak at: 7800.5423234172285 A
INFO:rascal.calibrator:- matched to 7802.65 A
INFO:rascal.calibrator:Peak at: 7887.564519177092 A
INFO:rascal.calibrator:- matched to 7887.4 A
INFO:rascal.calibrator:Peak at: 7967.35925662364 A
INFO:rascal.calibrator:- matched to 7967.34 A
INFO:rascal.calibrator:Peak at: 8058.1295020285925 A
INFO:rascal.calibrator:- matched to 8057.258 A
```

7. Quantify the quality of fit

```
print("RMS: {}".format(rms))
print("Stdev error: {} A".format(np.abs(residual).std()))
print("Peaks utilisation rate: {}%".format(peak_utilisation*100))
print("Atlas utilisation rate: {}%".format(atlas_utilisation * 100))
```

8. We can also inspect the search space in the Hough parameter-space where the samples were drawn by running:

```
c.plot_search_space()
```

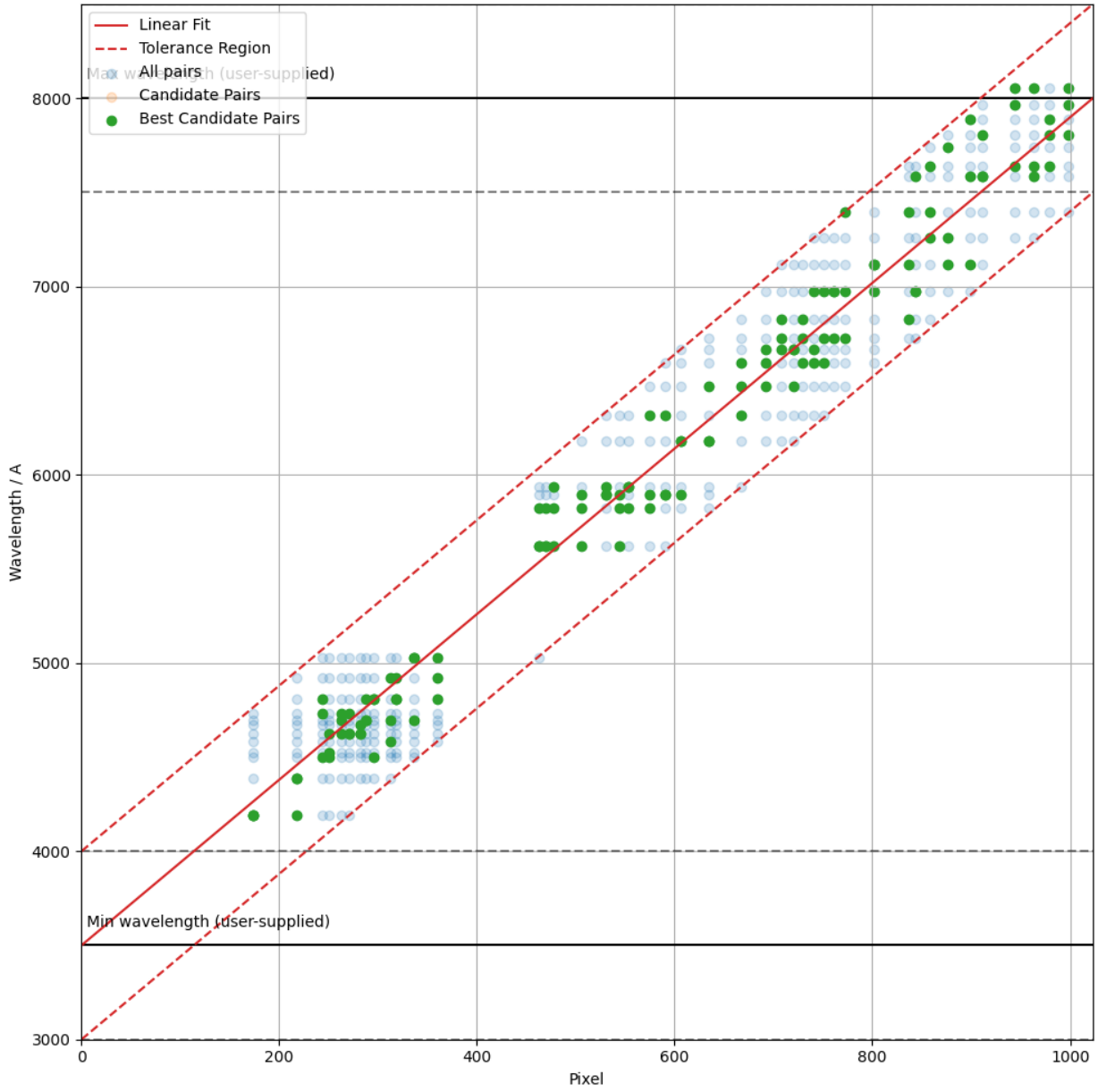
3.7 Example - Gemini/GMOSLS

The spectrogrphy GMOSLS on the Gemini Telescope is another important test instrument we focus on. The testing is performed with the medium/medium-low-resolution grating R400 in long-slit mode. This example also makes use of bhtomspec for flattening the arc image (which span over 12 FITS Image HDUs).

1. Initialise the environment and the line list (see the other examples for using the NIST line list) for the data proecessing

```
import numpy as np
from astropy.io import fits
from scipy.signal import find_peaks
from matplotlib import pyplot as plt
import os
from scipy import interpolate

from rascal.calibrator import Calibrator
from rascal import models
from rascal import util
```

(continues on next page)

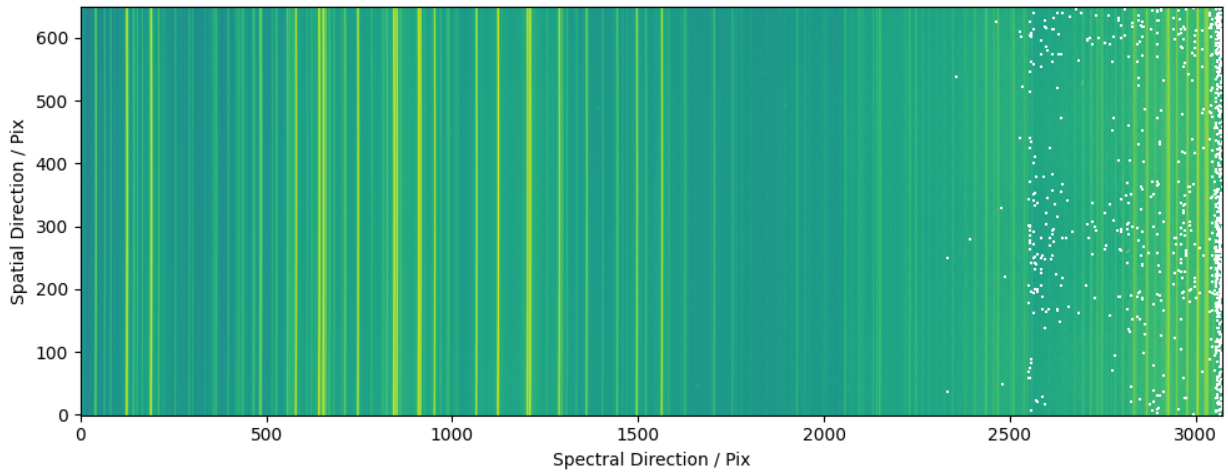

Linear Fit
Tolerance Region
All pairs
Candidate Pairs
Best Candidate Pairs
Min wavelength (user-supplied)
Wavelength / A
Pixel
8000
7000
6000
5000
4000
3000
0
200
400
600
800
1000

(continued from previous page)

```
import sys

atlas = [
    4703.632, 4728.19041, 4766.19677, 4807.36348, 4849.16386, 4881.22627,
    4890.40721, 4906.12088, 4934.58593, 4966.46490, 5018.56194, 5063.44827,
    5163.723, 5189.191, 5497.401, 5560.246, 5608.290, 5913.723, 6754.698,
    6873.185, 6967.352, 7032.190, 7069.167, 7149.012, 7274.940, 7386.014,
    7505.935, 7516.721, 7637.208, 7725.887, 7893.246, 7950.362, 8105.921,
    8117.542, 8266.794, 8410.521, 8426.963, 8523.783, 8670.325, 9125.471,
    9197.161, 9227.03, 9356.787, 9660.435, 9787.186
]

element = ['CuAr'] * len(atlas)
```

2. Import the *bhtomspec* to work with the GMOS data (*N20181115S0215_flattened.fits* is a flattened output image from bhtomspec) and get the pixel position that is corrected for the chip gaps for Gemini North, or flatten the image yourself and have the chip gaps adjusted, then continue with step 3.

```
sys.path.append('../../bhtomspec/GMOS')

from gmos_fieldflattening import create_pixel_array

pixels = create_pixel_array('north', 2)
rawpix_to_pix_itp = interpolate.interp1d(np.arange(len(pixels)), pixels)

# Load the data
base_dir = os.path.dirname('.')
spectrum2D = fits.open(
    os.path.join(base_dir,
                 'data_gemini_gmos/N20181115S0215_flattened.fits'))[0].data

# Collapse into 1D spectrum between row 110 and 120
spectrum = np.median(spectrum2D[300:310], axis=0)[::-1]
```



3. Load the data and identify the arc lines in the middle of the frame. Normally you should be applying the trace

from the spectral image onto the arc image

```
data = fits.open('data_lt_sprat/v_a_20190516_57_1_0_1.fits')[0].data

spectrum = np.median(data[110:120], axis=0)

peaks, _ = find_peaks(spectrum, height=500, distance=5, threshold=None)
peaks_refined = refine_peaks(spectrum, peaks, window_width=5)
```

4. Initialise the calibrator and set the properties. There are three sets of properties: (1) the calibrator properties who concerns the highest level setting - e.g. logging and plotting; (2) the Hough transform properties which set the constraints in which the trasnform is performed; (3) the RANSAC properties control the sampling conditions.

```
c = Calibrator(peaks_refined, spectrum)

c.set_calibrator_properties(num_pix=len(spectrum),
                            plotting_library='matplotlib',
                            log_level='info')

c.set_hough_properties(num_slopes=5000,
                       range_tolerance=500.,
                       xbins=200,
                       ybins=200,
                       min_wavelength=5000.,
                       max_wavelength=9500.)

c.set_ransac_properties(sample_size=5,
                        top_n_candidate=5)
```

The following *INFO* should be logged, where the first 3 lines are when the calibrator was initialised, and the last 3 lines are when the calibrator properties were set.

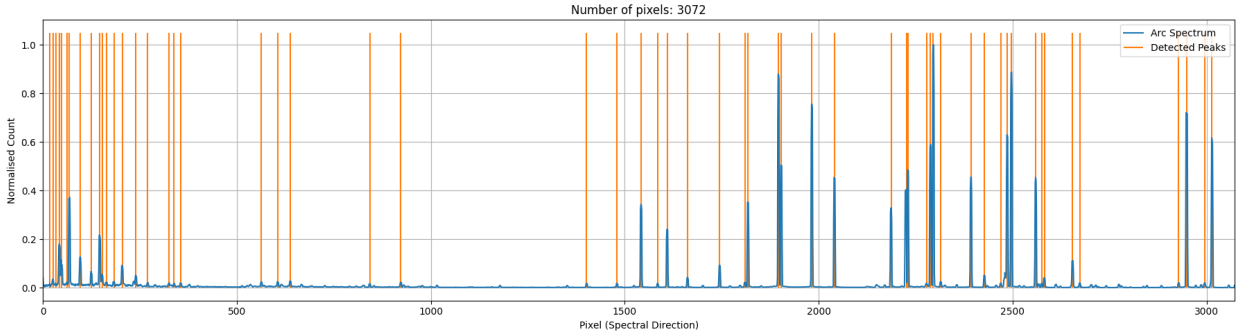
```
INFO:rascal.calibrator:num_pix is set to None.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
INFO:rascal.calibrator:num_pix is set to None.
INFO:rascal.calibrator:pixel_list is set to [0.000e+00 1.000e+00 2.000e+00 ... 3.136e+03␣
↪3.137e+03 3.138e+03].
INFO:rascal.calibrator:Plotting with matplotlib.
```

5. The extracted arc spectrum and the peaks identified can be plotted with the calibrator. Note that if only peaks are provided, only the orange lines will be plotted.

```
c.plot_arc()
```

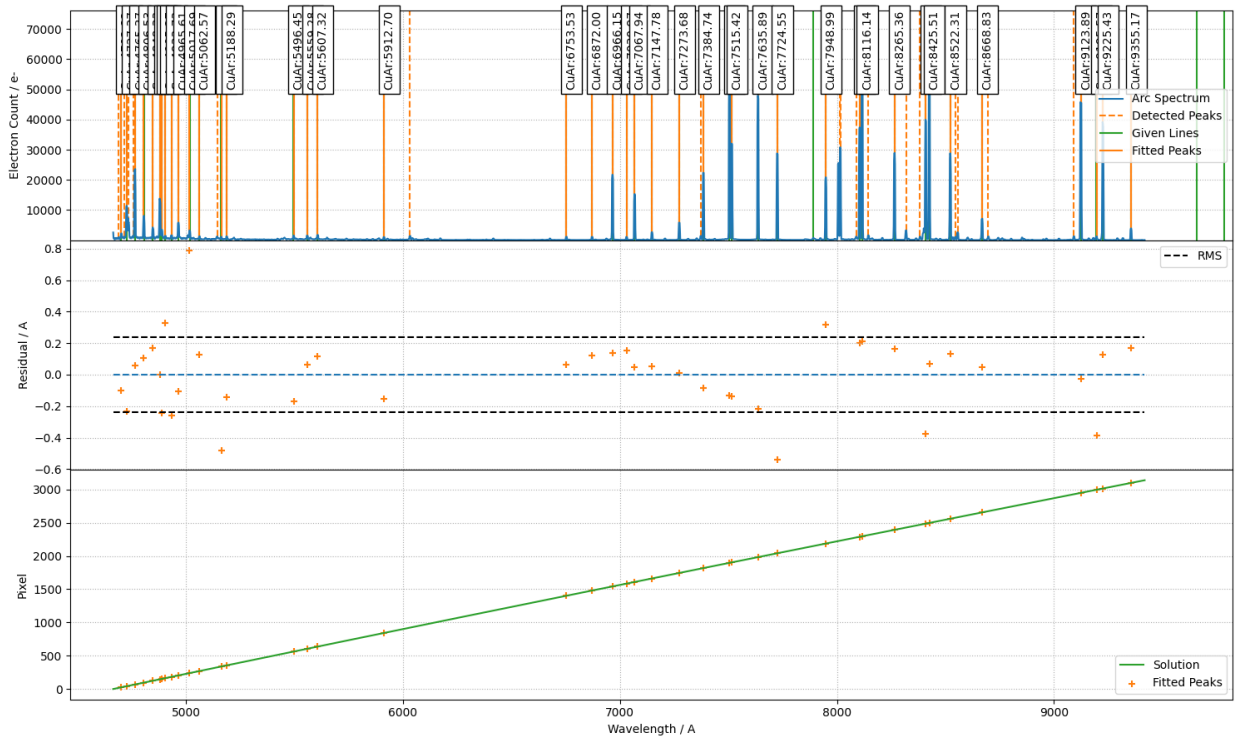
6. Add the line list to the calibrator and perform the hough transform on the pixel-wavelength pairs that will be used by the RANSAC sampling and fitting.

```
c.add_user_atlas(elements=element,
                 wavelengths=atlas,
                 vacuum=True,
                 pressure=61700.,
                 temperature=276.55,
                 relative_humidity=4.)
c.do_hough_transform()
```



6. Perform polynomial fit on samples drawn from RANSAC, the deafult option is to fit with polynomial function.

```
(fit_coeff, matched_peaks, matched_atlas, rms, residual, peak_utilisation,
 atlas_utilisation) = c.fit(max_tries=1000, fit_deg=4)

# Plot the solution
c.plot_fit(fit_coeff, spectrum, plot_atlas=True, log_spectrum=False, tolerance=5.)
```



with some INFO output looking like this:

```
INFO:rascal.calibrator:Peak at: 4690.432939431677 A
INFO:rascal.calibrator:Peak at: 4703.69698080062 A
INFO:rascal.calibrator:- matched to 4703.6319466224995 A
INFO:rascal.calibrator:Peak at: 4716.026520809748 A
INFO:rascal.calibrator:Peak at: 4728.383743644213 A
INFO:rascal.calibrator:- matched to 4728.190356343808 A
```

(continues on next page)

(continued from previous page)

```
INFO:rascal.calibrator:Peak at: 4737.097090466034 A
INFO:rascal.calibrator:Peak at: 4757.491217560396 A
INFO:rascal.calibrator:Peak at: 4766.093311886265 A
INFO:rascal.calibrator:- matched to 4766.196715912505 A
INFO:rascal.calibrator:Peak at: 4780.210922721128 A
INFO:rascal.calibrator:Peak at: 4807.206845286494 A
INFO:rascal.calibrator:- matched to 4807.36342544534 A
INFO:rascal.calibrator:Peak at: 4848.935535742634 A
INFO:rascal.calibrator:- matched to 4849.163804970983 A
INFO:rascal.calibrator:Peak at: 4866.990349869678 A
INFO:rascal.calibrator:Peak at: 4881.167498144336 A
INFO:rascal.calibrator:- matched to 4881.226214607134 A
INFO:rascal.calibrator:Peak at: 4890.591318605304 A
INFO:rascal.calibrator:- matched to 4890.407154502948 A
INFO:rascal.calibrator:Peak at: 4905.728124841728 A
INFO:rascal.calibrator:- matched to 4906.120824324627 A
INFO:rascal.calibrator:Peak at: 4934.778999493386 A
INFO:rascal.calibrator:- matched to 4934.585874001602 A
INFO:rascal.calibrator:Peak at: 4966.500342895015 A
INFO:rascal.calibrator:- matched to 4966.464843639834 A
INFO:rascal.calibrator:Peak at: 5011.146235531303 A
INFO:rascal.calibrator:Peak at: 5018.756785319058 A
INFO:rascal.calibrator:- matched to 5018.561883048629 A
INFO:rascal.calibrator:Peak at: 5063.247797723866 A
INFO:rascal.calibrator:- matched to 5063.448212539252 A
INFO:rascal.calibrator:Peak at: 5144.755132831136 A
INFO:rascal.calibrator:Peak at: 5164.128717766755 A
INFO:rascal.calibrator:- matched to 5163.722941401321 A
INFO:rascal.calibrator:Peak at: 5189.253818523486 A
INFO:rascal.calibrator:- matched to 5189.190941112306 A
INFO:rascal.calibrator:Peak at: 5497.49298195608 A
INFO:rascal.calibrator:- matched to 5497.400937614694 A
INFO:rascal.calibrator:Peak at: 5560.11042416126 A
INFO:rascal.calibrator:- matched to 5560.24593690152 A
INFO:rascal.calibrator:Peak at: 5608.100625609014 A
INFO:rascal.calibrator:- matched to 5608.289936356309 A
INFO:rascal.calibrator:Peak at: 5913.824836948544 A
INFO:rascal.calibrator:- matched to 5913.722932890211 A
INFO:rascal.calibrator:Peak at: 6033.807028259769 A
INFO:rascal.calibrator:Peak at: 6754.635766825111 A
INFO:rascal.calibrator:- matched to 6754.6979233467055 A
INFO:rascal.calibrator:Peak at: 6873.071461159328 A
INFO:rascal.calibrator:- matched to 6873.184922002098 A
INFO:rascal.calibrator:Peak at: 6967.222796782309 A
INFO:rascal.calibrator:- matched to 6967.351920933476 A
INFO:rascal.calibrator:Peak at: 7032.048764629298 A
INFO:rascal.calibrator:- matched to 7032.189920197685 A
INFO:rascal.calibrator:Peak at: 7069.13066654799 A
INFO:rascal.calibrator:- matched to 7069.166919778066 A
INFO:rascal.calibrator:Peak at: 7148.973051212137 A
INFO:rascal.calibrator:- matched to 7149.011918871972 A
INFO:rascal.calibrator:Peak at: 7274.942712866794 A
```

(continues on next page)

(continued from previous page)

```
INFO:rascal.calibrator:- matched to 7274.939917442923 A
INFO:rascal.calibrator:Peak at: 7374.290810404373 A
INFO:rascal.calibrator:Peak at: 7386.110457538571 A
INFO:rascal.calibrator:- matched to 7386.013916182439 A
INFO:rascal.calibrator:Peak at: 7506.080265224655 A
INFO:rascal.calibrator:- matched to 7505.934914821559 A
INFO:rascal.calibrator:Peak at: 7516.8721166291125 A
INFO:rascal.calibrator:- matched to 7516.720914699156 A
INFO:rascal.calibrator:Peak at: 7637.436474845539 A
INFO:rascal.calibrator:- matched to 7637.207913331853 A
INFO:rascal.calibrator:Peak at: 7726.433567729615 A
INFO:rascal.calibrator:- matched to 7725.886912325511 A
INFO:rascal.calibrator:Peak at: 7950.045808427023 A
INFO:rascal.calibrator:- matched to 7950.361909778136 A
INFO:rascal.calibrator:Peak at: 8012.828184407577 A
INFO:rascal.calibrator:Peak at: 8017.891550233657 A
INFO:rascal.calibrator:Peak at: 8090.176412302726 A
INFO:rascal.calibrator:Peak at: 8105.717761499685 A
INFO:rascal.calibrator:- matched to 8105.92090801283 A
INFO:rascal.calibrator:Peak at: 8117.323357964481 A
INFO:rascal.calibrator:- matched to 8117.541907880954 A
INFO:rascal.calibrator:Peak at: 8146.552760784422 A
INFO:rascal.calibrator:Peak at: 8266.617881247412 A
INFO:rascal.calibrator:- matched to 8266.79390618722 A
INFO:rascal.calibrator:Peak at: 8319.802992049212 A
INFO:rascal.calibrator:Peak at: 8384.554625340576 A
INFO:rascal.calibrator:Peak at: 8410.885574971622 A
INFO:rascal.calibrator:- matched to 8410.520904556186 A
INFO:rascal.calibrator:Peak at: 8426.881696730394 A
INFO:rascal.calibrator:- matched to 8426.962904369599 A
INFO:rascal.calibrator:Peak at: 8523.632359509633 A
INFO:rascal.calibrator:- matched to 8523.78290327087 A
INFO:rascal.calibrator:Peak at: 8547.188318167655 A
INFO:rascal.calibrator:Peak at: 8557.891662766575 A
INFO:rascal.calibrator:Peak at: 8670.25988469748 A
INFO:rascal.calibrator:- matched to 8670.324901607892 A
INFO:rascal.calibrator:Peak at: 8698.877173222663 A
INFO:rascal.calibrator:Peak at: 9093.532686785864 A
INFO:rascal.calibrator:Peak at: 9125.49128474281 A
INFO:rascal.calibrator:- matched to 9125.470896442828 A
INFO:rascal.calibrator:Peak at: 9197.54730560129 A
INFO:rascal.calibrator:- matched to 9197.16089562928 A
INFO:rascal.calibrator:Peak at: 9226.906860894214 A
INFO:rascal.calibrator:- matched to 9227.029895290323 A
INFO:rascal.calibrator:Peak at: 9356.635155506821 A
INFO:rascal.calibrator:- matched to 9356.786893817822 A
```

7. Quantify the quality of fit

```
print("RMS: {}".format(rms))
print("Stdev error: {} A".format(np.abs(residual).std()))
print("Peaks utilisation rate: {}%".format(peak_utilisation*100))
```

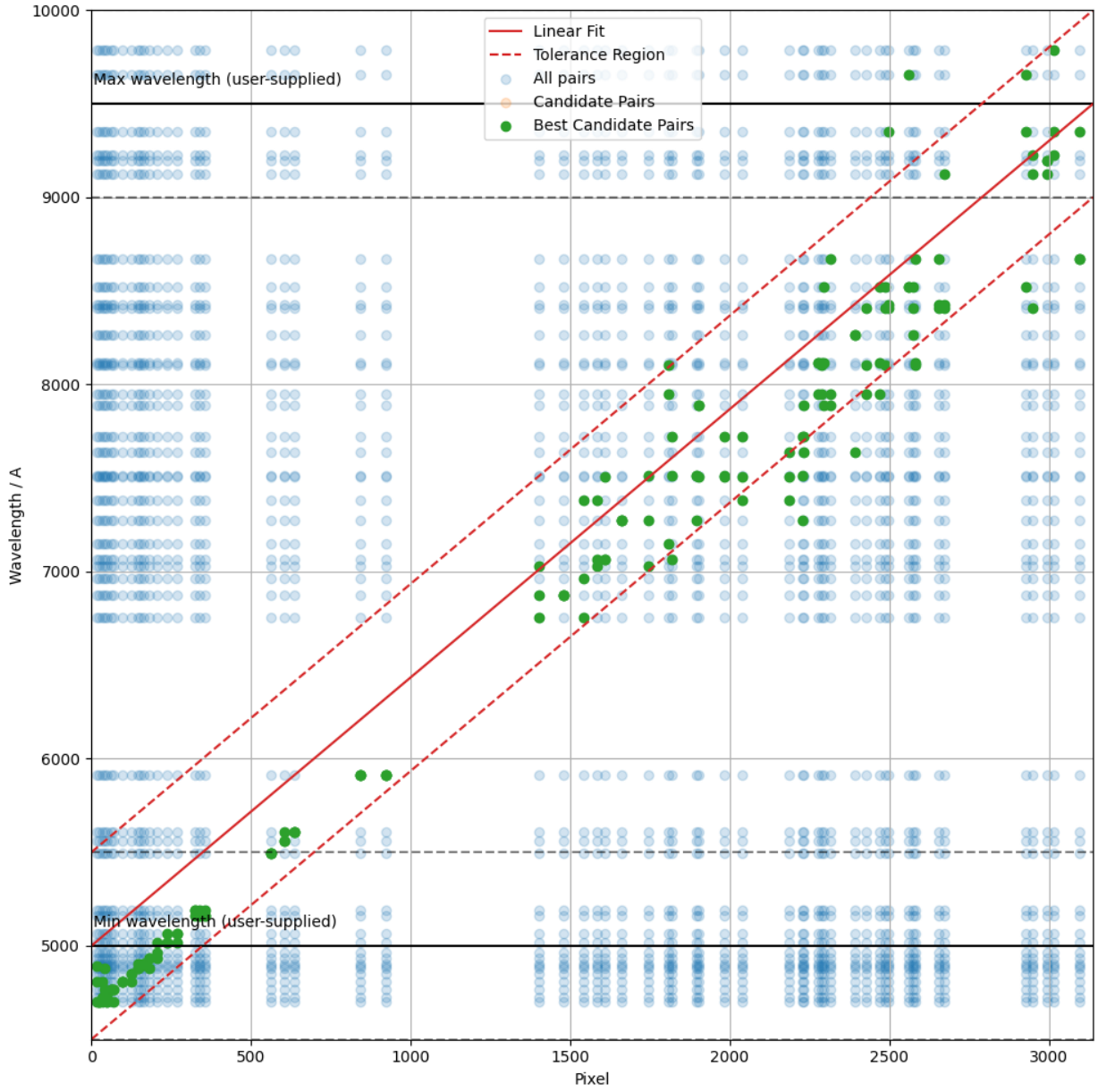
(continues on next page)

(continued from previous page)

```
print("Atlas utilisation rate: {}%".format(atlas_utilisation*100))
```

8. We can also inspect the search space in the Hough parameter-space where the samples were drawn by running:

```
c.plot_search_space()
```



3.8 Example - WHT/ISIS

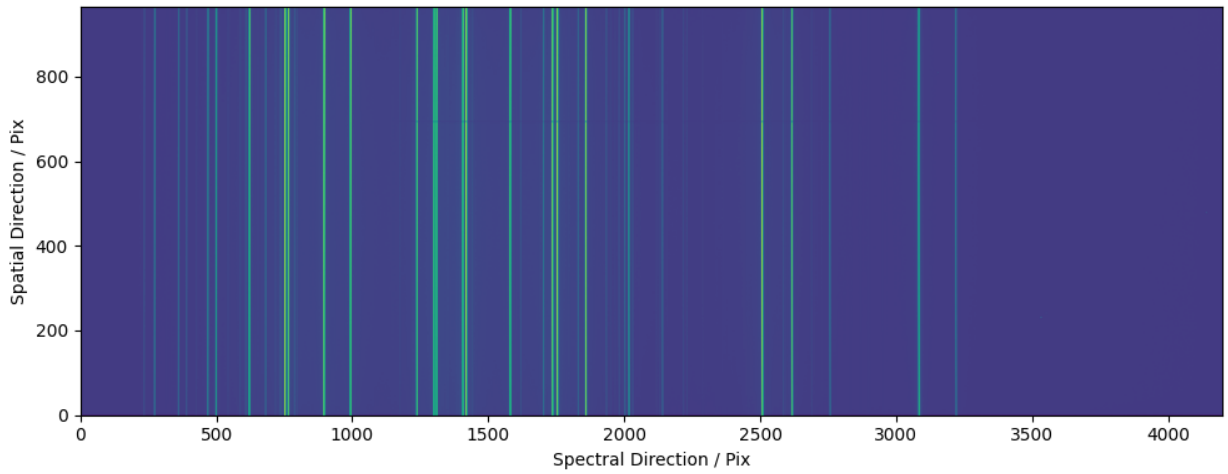
The Intermediate-dispersion Spectrograph and Imaging System (ISIS) is a Cassegrain instrument on the William Herschel Telescope. Like most instruments, it has many configurations, in this example, we are wavelength calibrating the R300R grating with the CuNe+CuAr lamp.

1. Initialise the environment and the line list (see the other examples for using the NIST line list) for the data proecessing

```
import matplotlib.pyplot as plt
import numpy as np
from astropy.io import fits
from scipy.signal import find_peaks
from scipy.signal import resample

from rascal.calibrator import Calibrator
from rascal.util import refine_peaks
```

2. Load and inspect the arc image

```
data = fits.open('data_wht_isis/r2701004_red_arc.fit')[1]

plt.figure(1, figsize=(10, 4))
plt.imshow(np.log10(data.data.T), aspect='auto', origin='lower')
plt.tight_layout()
```



3. Normally you should be applying the trace from the spectral image onto the arc image, but in this example, we identify the arc lines by summing over the entire frame in the spatial direction.

```
spectrum = np.median(data.data.T, axis=0)

peaks, _ = find_peaks(spectrum, prominence=80, distance=20, threshold=None)
peaks_refined = refine_peaks(spectrum, peaks, window_width=3)
```

4. Initialise the calibrator and set the properties. There are three sets of properties: (1) the calibrator properties who concerns the highest level setting - e.g. logging and plotting; (2) the Hough transform properties which set the constraints in which the trasnform is performed; (3) the RANSAC properties control the sampling conditions.

```
c = Calibrator(peaks_refined, spectrum)

c.set_calibrator_properties(num_pix=len(spectrum),
                            plotting_library='matplotlib',
                            log_level='info')

c.set_hough_properties(num_slopes=10000,
                       xbins=500,
                       ybins=500,
                       min_wavelength=7000.,
                       max_wavelength=10500.,
                       range_tolerance=500.,
                       linearity_tolerance=50)

c.add_atlas(["Ne", "Ar", "Cu"],
            min_atlas_wavelength=6000,
            max_atlas_wavelength=11000,
            min_intensity=10,
            min_distance=10,
            candidate_tolerance=10,
            constrain_poly=False,
            vacuum=False,
            pressure=101325.,
            temperature=273.15,
            relative_humidity=0.)

c.set_ransac_properties(sample_size=5,
                        top_n_candidate=5,
                        linear=True,
                        filter_close=True,
                        ransac_tolerance=5,
                        candidate_weighted=True,
                        hough_weight=1.0)

c.do_hough_transform()
```

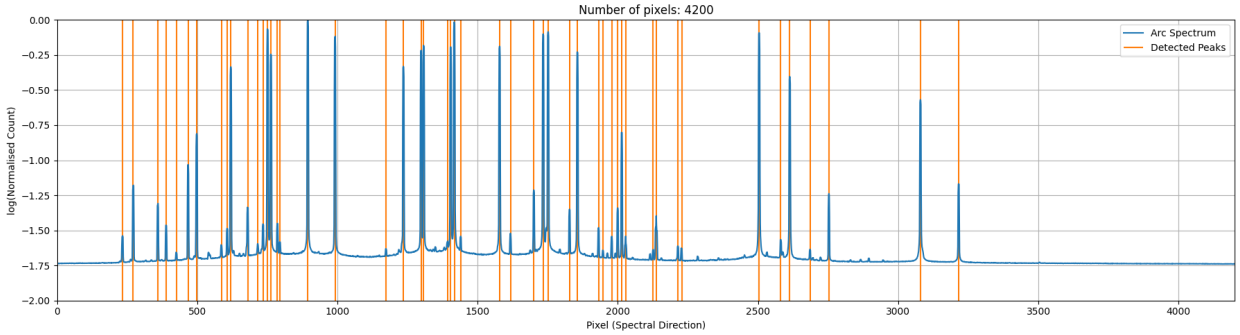
The following *INFO* should be logged, where the first 3 lines are when the calibrator was initialised, and the last 3 lines are when the calibrator properties were set.

```
INFO:rascal.calibrator:num_pix is set to None.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
INFO:rascal.calibrator:num_pix is set to 4200.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
```

5. The extracted arc spectrum and the peaks identified can be plotted with the calibrator. Note that if only peaks are provided, only the orange lines will be plotted.

```
c.plot_arc()
```

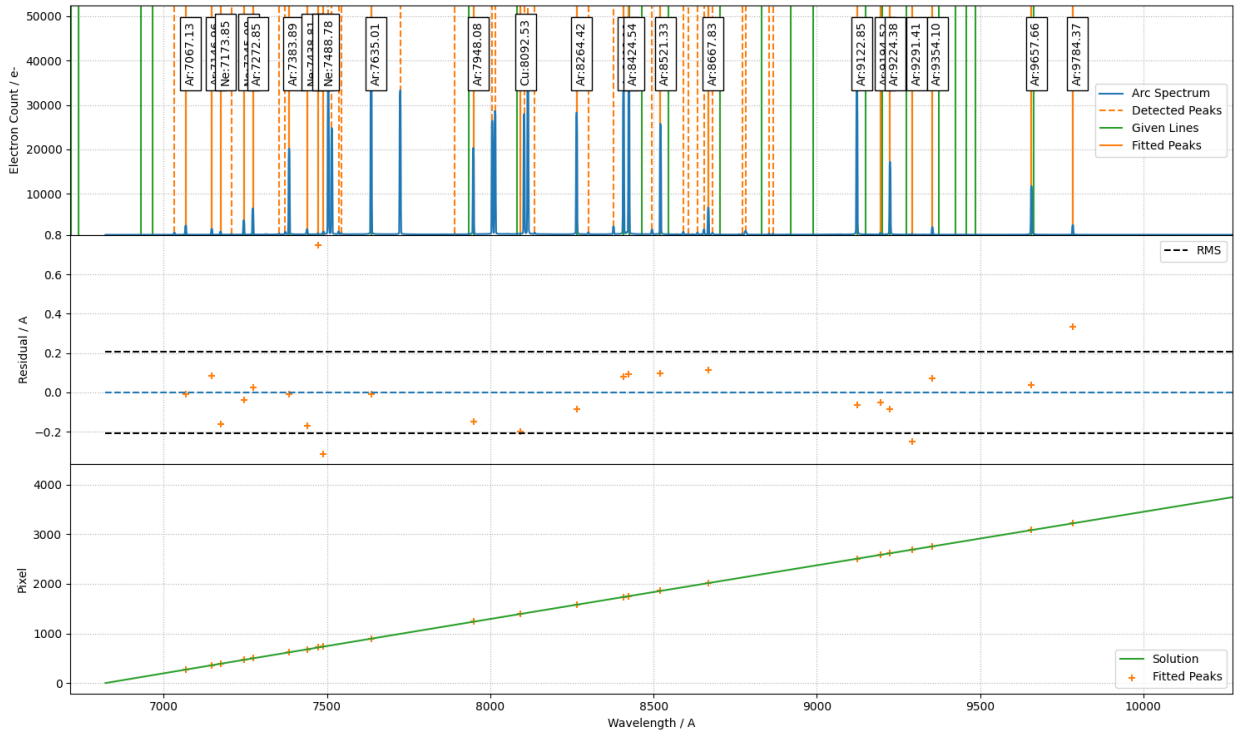
6. Add the line list to the calibrator and perform the hough transform on the pixel-wavelength pairs that will be used by the RANSAC sampling and fitting.




```
c.load_user_atlas(elements=element,
                  wavelengths=atlas,
                  constrain_poly=True)
c.do_hough_transform()
```

6. Perform polynomial fit on samples drawn from RANSAC, the deafult option is to fit with polynomial function.

```
fit_coeff, rms, residual, peak_utilisation = c.fit(max_tries=500)
c.plot_fit(fit_coeff,
           plot_atlas=True,
           log_spectrum=False,
           tolerance=10.)
```



with some INFO output looking like this:

```
INFO:rascal.calibrator:Peak at: 7032.03272057003 A
INFO:rascal.calibrator:- matched to 7030.16748046875 A
INFO:rascal.calibrator:Peak at: 7066.866238979784 A
INFO:rascal.calibrator:Peak at: 7146.706486468946 A
INFO:rascal.calibrator:- matched to 7146.95654296875 A
INFO:rascal.calibrator:Peak at: 7173.679578238163 A
INFO:rascal.calibrator:- matched to 7173.85205078125 A
INFO:rascal.calibrator:Peak at: 7206.739705323605 A
INFO:rascal.calibrator:- matched to 7206.892578125 A
INFO:rascal.calibrator:Peak at: 7245.089683763745 A
INFO:rascal.calibrator:- matched to 7245.0791015625 A
INFO:rascal.calibrator:Peak at: 7272.789393369358 A
INFO:rascal.calibrator:Peak at: 7311.850882354185 A
INFO:rascal.calibrator:- matched to 7304.75390625 A
INFO:rascal.calibrator:Peak at: 7353.675782032377 A
INFO:rascal.calibrator:- matched to 7353.20361328125 A
INFO:rascal.calibrator:Peak at: 7383.976473141233 A
INFO:rascal.calibrator:Peak at: 7439.086420302527 A
INFO:rascal.calibrator:- matched to 7438.80712890625 A
INFO:rascal.calibrator:Peak at: 7471.789387421437 A
INFO:rascal.calibrator:- matched to 7472.3466796875 A
INFO:rascal.calibrator:Peak at: 7504.0775689945185 A
INFO:rascal.calibrator:- matched to 7503.7763671875 A
INFO:rascal.calibrator:Peak at: 7535.9467914563065 A
INFO:rascal.calibrator:Peak at: 7635.527653296165 A
INFO:rascal.calibrator:- matched to 7635.01171875 A
INFO:rascal.calibrator:Peak at: 7724.2916173416315 A
INFO:rascal.calibrator:- matched to 7724.52685546875 A
INFO:rascal.calibrator:Peak at: 7891.547551265124 A
INFO:rascal.calibrator:- matched to 7890.97607421875 A
INFO:rascal.calibrator:Peak at: 7948.68404307157 A
INFO:rascal.calibrator:- matched to 7948.07568359375 A
INFO:rascal.calibrator:Peak at: 8015.266095970854 A
INFO:rascal.calibrator:Peak at: 8093.237023885084 A
INFO:rascal.calibrator:- matched to 8092.53125 A
INFO:rascal.calibrator:Peak at: 8115.647151051324 A
INFO:rascal.calibrator:- matched to 8115.20751953125 A
INFO:rascal.calibrator:Peak at: 8136.9823474983705 A
INFO:rascal.calibrator:Peak at: 8264.92957037352 A
INFO:rascal.calibrator:- matched to 8264.4169921875 A
INFO:rascal.calibrator:Peak at: 8300.622776221953 A
INFO:rascal.calibrator:Peak at: 8377.93242372805 A
INFO:rascal.calibrator:Peak at: 8424.969864843704 A
INFO:rascal.calibrator:- matched to 8424.5400390625 A
INFO:rascal.calibrator:Peak at: 8495.507558139581 A
INFO:rascal.calibrator:- matched to 8495.25 A
INFO:rascal.calibrator:Peak at: 8521.649655751076 A
INFO:rascal.calibrator:- matched to 8521.33203125 A
INFO:rascal.calibrator:Peak at: 8591.500778968331 A
INFO:rascal.calibrator:Peak at: 8634.822380309419 A
INFO:rascal.calibrator:Peak at: 8668.074138404148 A
INFO:rascal.calibrator:- matched to 8667.83203125 A
INFO:rascal.calibrator:Peak at: 8761.821578994182 A
```

(continues on next page)

(continued from previous page)

```
INFO:rascal.calibrator:Peak at: 8781.595744197106 A
INFO:rascal.calibrator:Peak at: 8853.730475790377 A
INFO:rascal.calibrator:- matched to 8853.7509765625 A
INFO:rascal.calibrator:Peak at: 9122.730835375478 A
INFO:rascal.calibrator:- matched to 9122.8466796875 A
INFO:rascal.calibrator:Peak at: 9194.447034742407 A
INFO:rascal.calibrator:- matched to 9194.5166015625 A
INFO:rascal.calibrator:Peak at: 9224.23302586336 A
INFO:rascal.calibrator:- matched to 9224.376953125 A
INFO:rascal.calibrator:Peak at: 9291.251217839615 A
INFO:rascal.calibrator:- matched to 9291.408203125 A
INFO:rascal.calibrator:Peak at: 9354.010724203863 A
INFO:rascal.calibrator:- matched to 9354.0966796875 A
INFO:rascal.calibrator:Peak at: 9657.622355120493 A
INFO:rascal.calibrator:- matched to 9657.65625 A
INFO:rascal.calibrator:Peak at: 9784.483780841349 A
INFO:rascal.calibrator:- matched to 9784.37109375 A
```

7. Quantify the quality of fit

```
print("RMS: {}".format(rms))
print("Stdev error: {} A".format(np.abs(residual).std()))
print("Peaks utilisation rate: {}%".format(peak_utilisation*100))
```

with these output

```
RMS: 0.6826146587462325
Stdev error: 0.5576730885022209 A
Peaks utilisation rate: 53.65853658536586%
```

8. We can also inspect the search space in the Hough parameter-space where the samples were drawn by running:

```
c.plot_search_space()
```

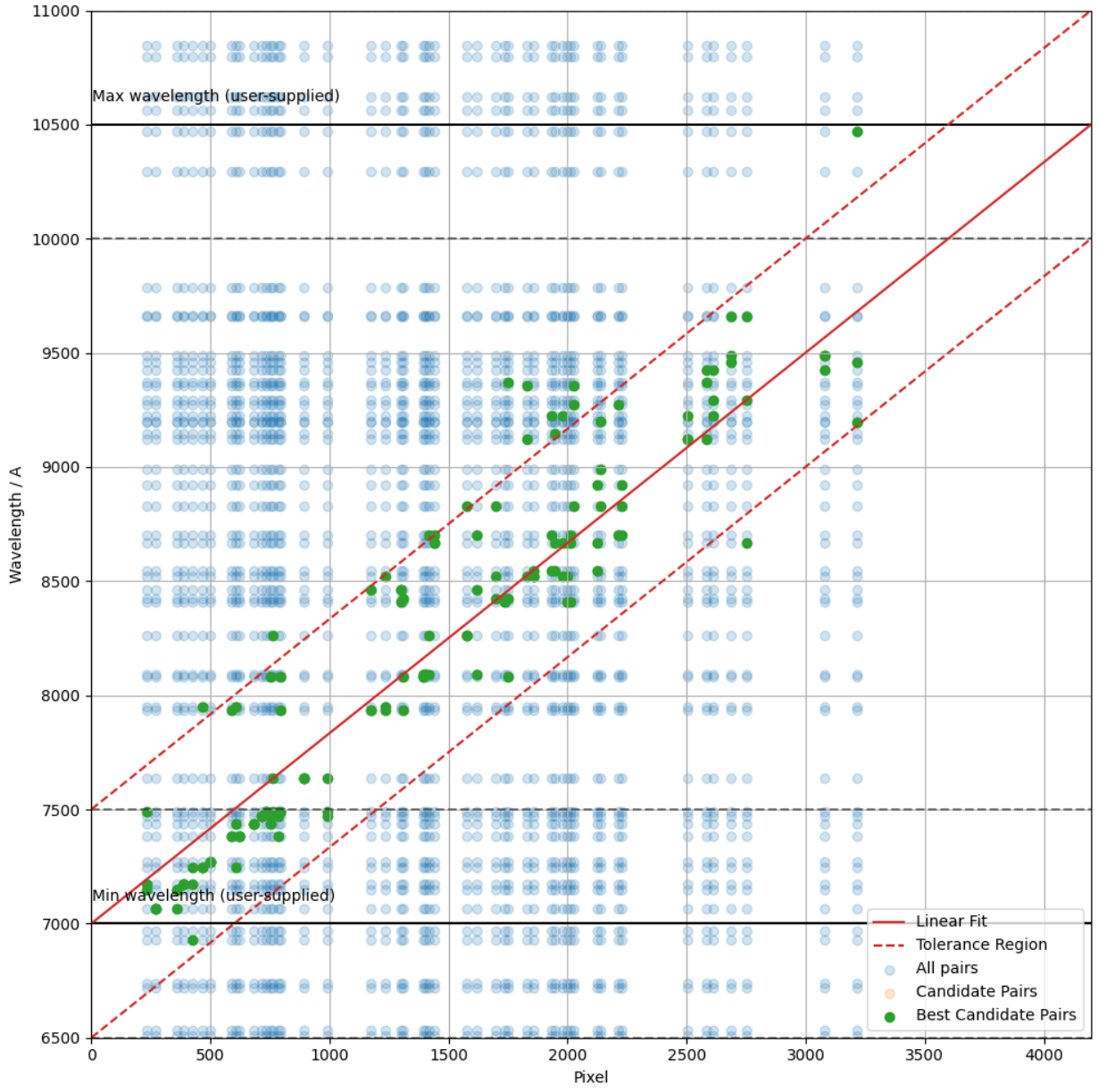
3.9 Example - Keck/DEIMOS

This example performs wavelength calibration on DEep Imaging Multi-Object Spectrograph (DEIMOS installed on the Keck Telescope) with the 830G grating, containing Ne, Ar abd Kr without providing a line list. The 1D arc spectrum is taken from pypeit on GitHub, as of 6 April 2020.

1. Initialise the environment and the line list (see the other examples for using the NIST line list) for the data proecessing

```
import json
import numpy as np
from scipy.signal import find_peaks

from rascal.calibrator import Calibrator
from rascal.util import refine_peaks
```

2. Load the arc spectrum and identify the arc lines


Max wavelength (user-supplied)
Min wavelength (user-supplied)
Wavelength / A
Pixel
Linear Fit
Tolerance Region
All pairs
Candidate Pairs
Best Candidate Pairs

```
spectrum_json = json.load(open('data_keck_deimos/keck_deimos_830g_l_PYPIT.json'))
spectrum = np.array(spectrum_json['spec'])

peaks, _ = find_peaks(spectrum, prominence=200, distance=10)
peaks_refined = refine_peaks(spectrum, peaks, window_width=3)
```

3. Initialise the calibrator and set the properties. There are three sets of properties: (1) the calibrator properties who concerns the highest level setting - e.g. logging and plotting; (2) the Hough transform properties which set the constraints in which the trasnform is performed; (3) the RANSAC properties control the sampling conditions.

```
c = Calibrator(peaks_refined, spectrum)

c.set_calibrator_properties(num_pix=len(spectrum),
                            plotting_library='matplotlib',
                            log_level='info')

c.set_hough_properties(num_slopes=10000,
                       xbins=200,
                       ybins=200,
                       min_wavelength=6500.,
                       max_wavelength=10500.,
                       range_tolerance=500.,
                       linearity_tolerance=50)

c.set_ransac_properties(sample_size=5,
                        top_n_candidate=5,
                        linear=True,
                        filter_close=True,
                        ransac_tolerance=5,
                        candidate_weighted=True,
                        hough_weight=1.0)

c.do_hough_transform()
```

The following *INFO* should be logged, where the first 3 lines are when the calibrator was initialised, and the last 3 lines are when the calibrator properties were set.

```
INFO:rascal.calibrator:num_pix is set to None.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
INFO:rascal.calibrator:num_pix is set to 4096.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
```

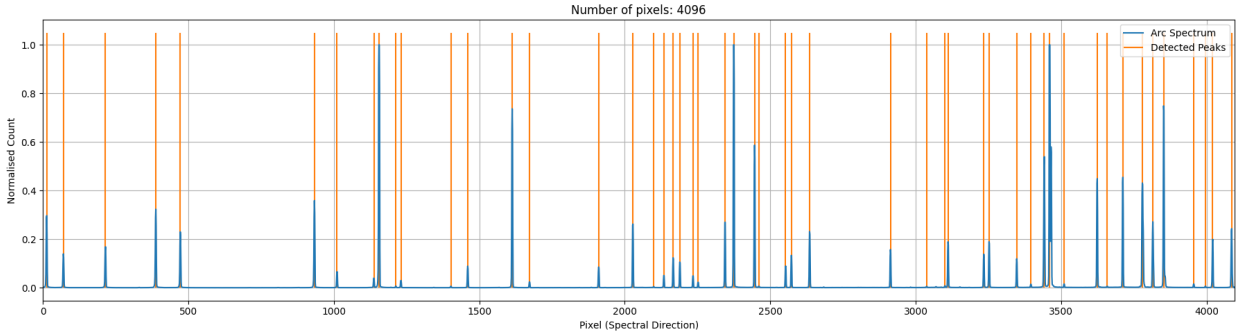
4. The extracted arc spectrum and the peaks identified can be plotted with the calibrator. Note that if only peaks are provided, only the orange lines will be plotted.

```
c.plot_arc()
```

5. Add the line list downloaded from NIST to the calibrator and perform the hough transform on the pixel-wavelength pairs that will be used by the RANSAC sampling and fitting.

```
c.add_atlas(["He", "Ar", "Kr"],
            min_intensity=15,
```

(continues on next page)

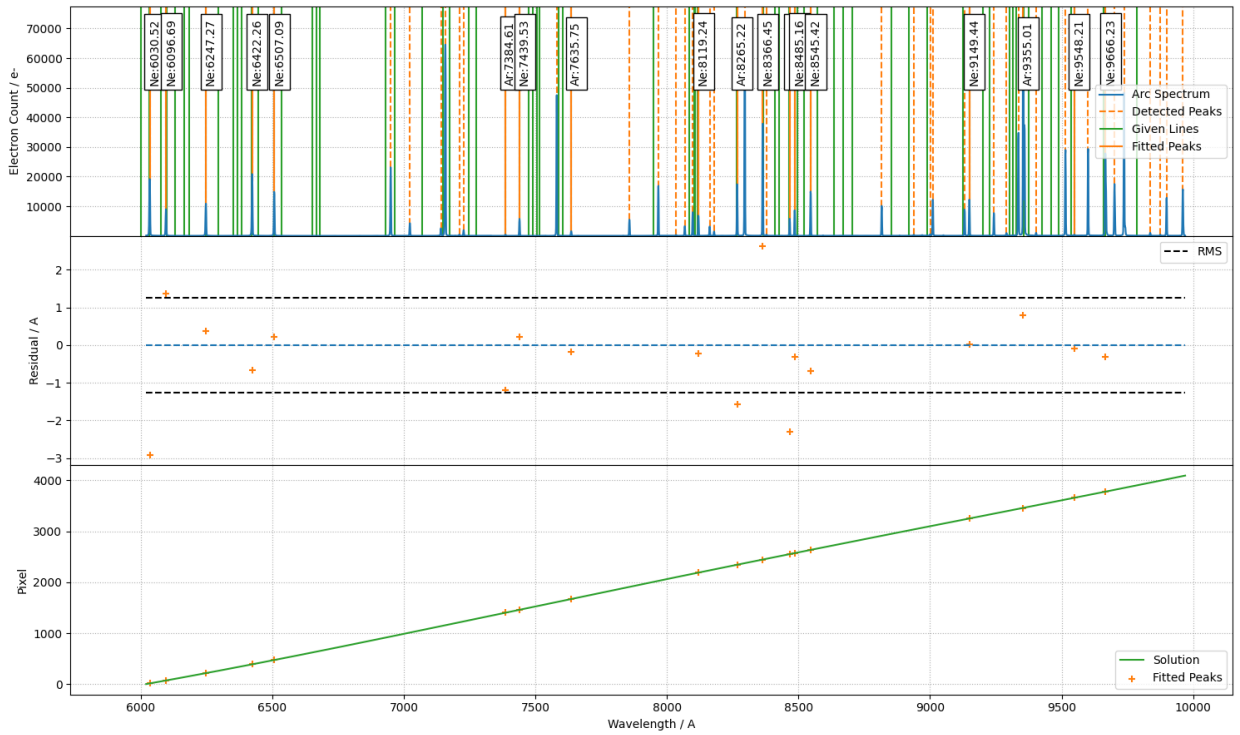


(continued from previous page)

```
            min_intensity=1000.,
            pressure=70000.,
            temperature=285.)
c.do_hough_transform()
```

6. Perform polynomial fit on samples drawn from RANSAC, the deafult option is to fit with polynomial function.

```
(fit_coeff, matched_peaks, matched_atlas, rms, residual, peak_utilisation,
 atlas_utilisation) = c.fit(max_tries=1000)
c.plot_fit(fit_coeff,
           spectrum=spectrum,
           plot_atlas=True,
           log_spectrum=False,
           tolerance=10.)
```



with some INFO output looking like this:

```
INFO:rascal.calibrator:Peak at: 6181.21116563306 A
INFO:rascal.calibrator:Peak at: 6256.333529966452 A
INFO:rascal.calibrator:Peak at: 6439.811724316869 A
INFO:rascal.calibrator:Peak at: 6652.09949066169 A
INFO:rascal.calibrator:- matched to 6652.158203125 A
INFO:rascal.calibrator:Peak at: 6753.108192719367 A
INFO:rascal.calibrator:- matched to 6752.755859375 A
INFO:rascal.calibrator:Peak at: 7284.4291622913315 A
INFO:rascal.calibrator:- matched to 7287.173828125 A
INFO:rascal.calibrator:Peak at: 7371.7744273814915 A
INFO:rascal.calibrator:- matched to 7372.0283203125 A
INFO:rascal.calibrator:Peak at: 7513.2846911492425 A
INFO:rascal.calibrator:- matched to 7514.5595703125 A
INFO:rascal.calibrator:Peak at: 7533.316917280944 A
INFO:rascal.calibrator:Peak at: 7597.325991944456 A
INFO:rascal.calibrator:- matched to 7601.45166015625 A
INFO:rascal.calibrator:Peak at: 7616.837505897713 A
INFO:rascal.calibrator:Peak at: 7807.543435151655 A
INFO:rascal.calibrator:- matched to 7806.42431640625 A
INFO:rascal.calibrator:Peak at: 7871.742919008845 A
INFO:rascal.calibrator:Peak at: 8041.538650167087 A
INFO:rascal.calibrator:Peak at: 8107.7513815373395 A
INFO:rascal.calibrator:- matched to 8103.58984375 A
INFO:rascal.calibrator:Peak at: 8373.173467243614 A
INFO:rascal.calibrator:Peak at: 8504.898771185366 A
INFO:rascal.calibrator:- matched to 8508.763671875 A
INFO:rascal.calibrator:Peak at: 8585.472074103302 A
INFO:rascal.calibrator:Peak at: 8625.137975870266 A
INFO:rascal.calibrator:Peak at: 8661.21940010808 A
INFO:rascal.calibrator:Peak at: 8687.226193001674 A
INFO:rascal.calibrator:Peak at: 8738.248959745788 A
INFO:rascal.calibrator:Peak at: 8758.17852455273 A
INFO:rascal.calibrator:- matched to 8755.0810546875 A
INFO:rascal.calibrator:Peak at: 8862.992736150627 A
INFO:rascal.calibrator:Peak at: 8897.362194686848 A
INFO:rascal.calibrator:Peak at: 8978.409139377678 A
INFO:rascal.calibrator:- matched to 8977.8681640625 A
INFO:rascal.calibrator:Peak at: 8995.427783056684 A
INFO:rascal.calibrator:- matched to 8999.0712890625 A
INFO:rascal.calibrator:Peak at: 9099.958999233066 A
INFO:rascal.calibrator:Peak at: 9122.555380323614 A
INFO:rascal.calibrator:- matched to 9122.3662109375 A
INFO:rascal.calibrator:Peak at: 9194.145054206449 A
INFO:rascal.calibrator:- matched to 9194.5166015625 A
INFO:rascal.calibrator:Peak at: 9249.269222476234 A
INFO:rascal.calibrator:Peak at: 9510.487993935229 A
INFO:rascal.calibrator:Peak at: 9588.910006488015 A
INFO:rascal.calibrator:Peak at: 9651.46472100934 A
INFO:rascal.calibrator:Peak at: 9687.974467875612 A
INFO:rascal.calibrator:Peak at: 9722.775033426266 A
INFO:rascal.calibrator:Peak at: 9734.57822981183 A
INFO:rascal.calibrator:Peak at: 9780.201795417905 A
INFO:rascal.calibrator:- matched to 9784.37109375 A
```

(continues on next page)

(continued from previous page)

```
INFO:rascal.calibrator:Peak at: 9872.315086684568 A
INFO:rascal.calibrator:Peak at: 9892.726457724415 A
INFO:rascal.calibrator:Peak at: 9997.754248989899 A
INFO:rascal.calibrator:Peak at: 10051.246817850328 A
INFO:rascal.calibrator:- matched to 10051.9287109375 A
INFO:rascal.calibrator:Peak at: 10101.391516626534 A
INFO:rascal.calibrator:Peak at: 10121.71692116084 A
INFO:rascal.calibrator:- matched to 10120.8193359375 A
INFO:rascal.calibrator:Peak at: 10175.549623735413 A
INFO:rascal.calibrator:Peak at: 10297.004370039842 A
INFO:rascal.calibrator:- matched to 10296.7880859375 A
INFO:rascal.calibrator:Peak at: 10333.286701702546 A
INFO:rascal.calibrator:- matched to 10332.5771484375 A
INFO:rascal.calibrator:Peak at: 10389.359445544913 A
INFO:rascal.calibrator:Peak at: 10459.030756555107 A
INFO:rascal.calibrator:Peak at: 10495.358328803779 A
INFO:rascal.calibrator:Peak at: 10533.23093013465 A
INFO:rascal.calibrator:Peak at: 10635.260454507756 A
INFO:rascal.calibrator:Peak at: 10674.011112594191 A
INFO:rascal.calibrator:- matched to 10673.4189453125 A
INFO:rascal.calibrator:Peak at: 10698.018361022285 A
INFO:rascal.calibrator:- matched to 10699.181640625 A
INFO:rascal.calibrator:Peak at: 10759.154132550711 A
INFO:rascal.calibrator:- matched to 10759.015625 A
```

7. Quantify the quality of fit

```
print("RMS: {}".format(rms))
print("Stdev error: {} A".format(np.abs(residual).std()))
print("Peaks utilisation rate: {}%".format(peak_utilisation*100))
print("Atlas utilisation rate: {}%".format(atlas_utilisation*100))
```

where the low numer of peaks utilisation suggests it may be a good fit by chance for the 1/3 of all the peaks, or confusion of the peaks causes the problem. These values alone cannot tell which is the case here and the diagnotic plot should be inspected to confirm the quality of fit.

8. We can also inspect the search space in the Hough parameter-space where the samples were drawn by running:

```
c.plot_search_space()
```

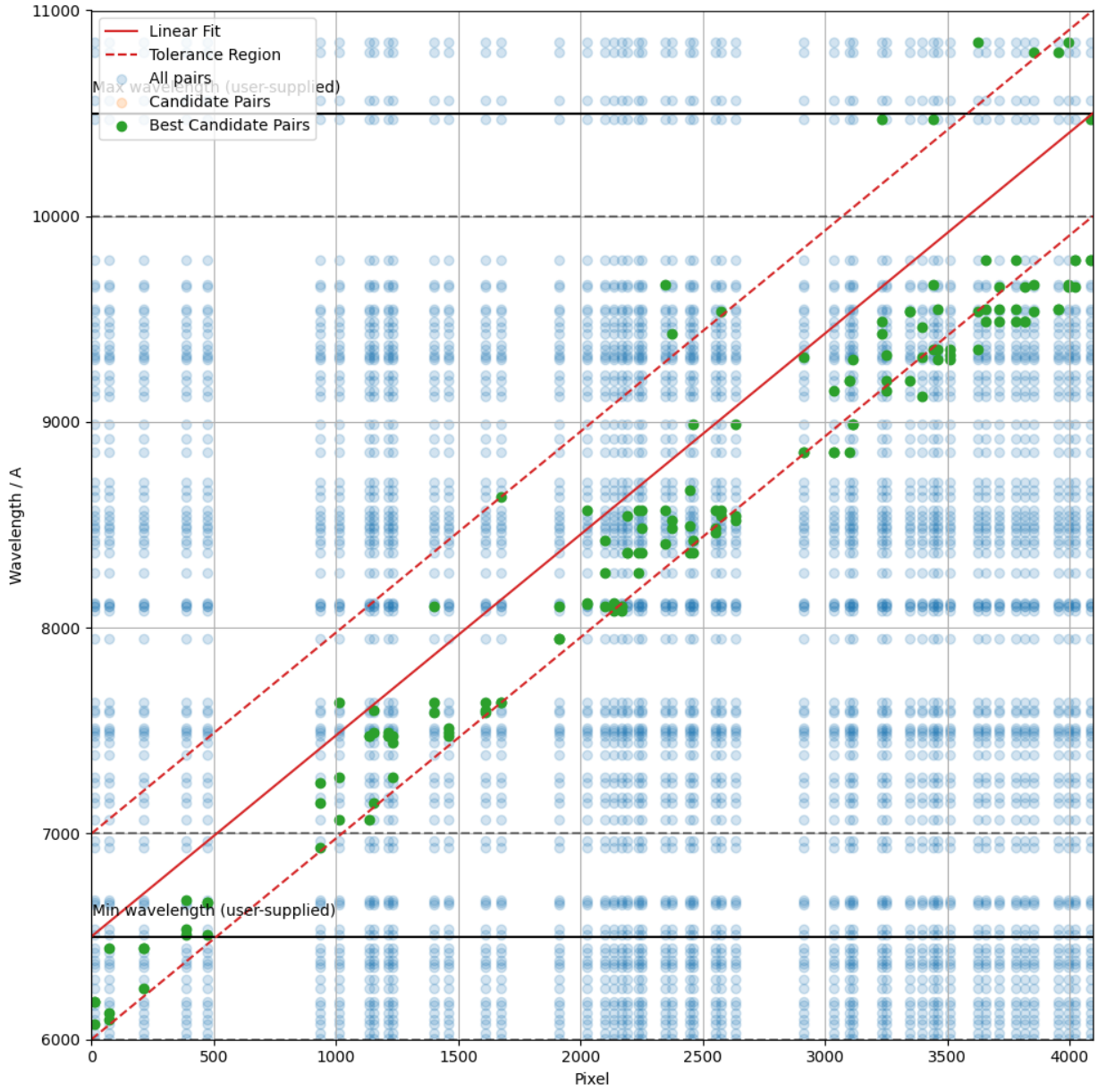
3.10 Example - GTC/OSIRIS

The spectrogrphy OSIRIS is one of the most used instrument on the Gran Telescopio Canarias (GTC). In this example, we wavelength calibration the R1000B HgAr arc.

1. Initialise the environment and the line list (see the other examples for using the NIST line list) for the data proecessing

```
from astropy.io import fits
import matplotlib.pyplot as plt
import numpy as np
```

(continues on next page)

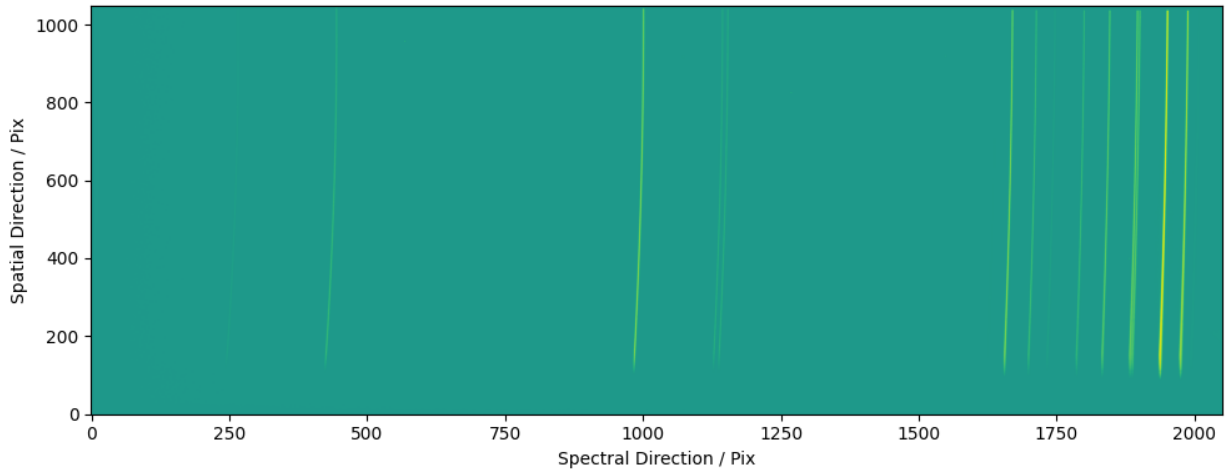

Linear Fit
Tolerance Region
All pairs
Candidate Pairs
Best Candidate Pairs
Max wavelength (user-supplied)
Min wavelength (user-supplied)
Wavelength / A
Pixel
11000
10000
9000
8000
7000
6000
0
500
1000
1500
2000
2500
3000
3500
4000

(continued from previous page)

```
from scipy.signal import find_peaks
from scipy.signal import resample

from rascal.calibrator import Calibrator
from rascal.util import refine_peaks

atlas = [
    3650.153, 4046.563, 4077.831, 4358.328, 5460.735, 5769.598, 5790.663,
    6682.960, 6752.834, 6871.289, 6965.431, 7030.251, 7067.218, 7147.042,
    7272.936, 7383.981, 7503.869, 7514.652, 7635.106, 7723.98
]
element = ['HgAr'] * len(atlas)
```

2. Load and inspect the arc image

```
data = fits.open('data_gtc_osiris/0002672523-20200911-OSIRIS-OsirisCalibrationLamp.fits
↪')[1]

plt.figure(1, figsize=(16,5))
plt.imshow(np.log(data.data), aspect='auto', origin='lower')
plt.tight_layout()
```



3. Normally you should be applying the trace from the spectral image onto the arc image, but in this example, we identify the arc lines in the middle of the frame.

```
spectrum = np.median(data.data.T[550:570], axis=0)

peaks, _ = find_peaks(spectrum, height=1250, prominence=20, distance=3, threshold=None)
peaks_refined = refine_peaks(spectrum, peaks, window_width=3)
```

4. Initialise the calibrator and set the properties. There are three sets of properties: (1) the calibrator properties who concerns the highest level setting - e.g. logging and plotting; (2) the Hough transform properties which set the constraints in which the trasnform is performed; (3) the RANSAC properties control the sampling conditions.

```
c = Calibrator(peaks_refined, spectrum)
```

(continues on next page)

(continued from previous page)

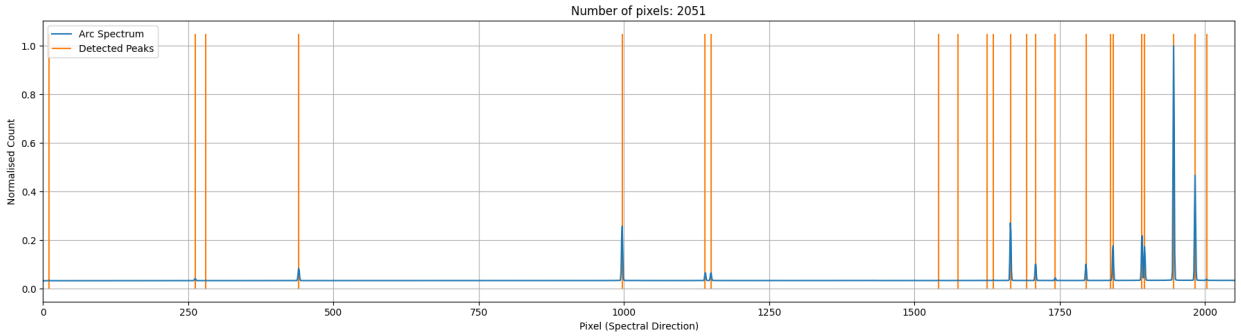
```
c.set_hough_properties(num_slopes=2000,
                       xbins=100,
                       ybins=100,
                       min_wavelength=3500.,
                       max_wavelength=8000.,
                       range_tolerance=500.,
                       linearity_tolerance=50)

c.add_user_atlas(elements=element,
                 wavelengths=atlas,
                 constrain_poly=True)

c.set_ransac_properties(sample_size=5,
                        top_n_candidate=5,
                        linear=True,
                        filter_close=True,
                        ransac_tolerance=5,
                        candidate_weighted=True,
                        hough_weight=1.0)

c.do_hough_transform()
```

The following *INFO* should be logged, where the first 3 lines are when the calibrator was initialised, and the last 3 lines are when the calibrator properties were set.

```
INFO:rascal.calibrator:num_pix is set to None.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
INFO:rascal.calibrator:num_pix is set to 2051.
INFO:rascal.calibrator:pixel_list is set to None.
INFO:rascal.calibrator:Plotting with matplotlib.
```

5. The extracted arc spectrum and the peaks identified can be plotted with the calibrator. Note that if only peaks are provided, only the orange lines will be plotted.

```
c.plot_arc()
```

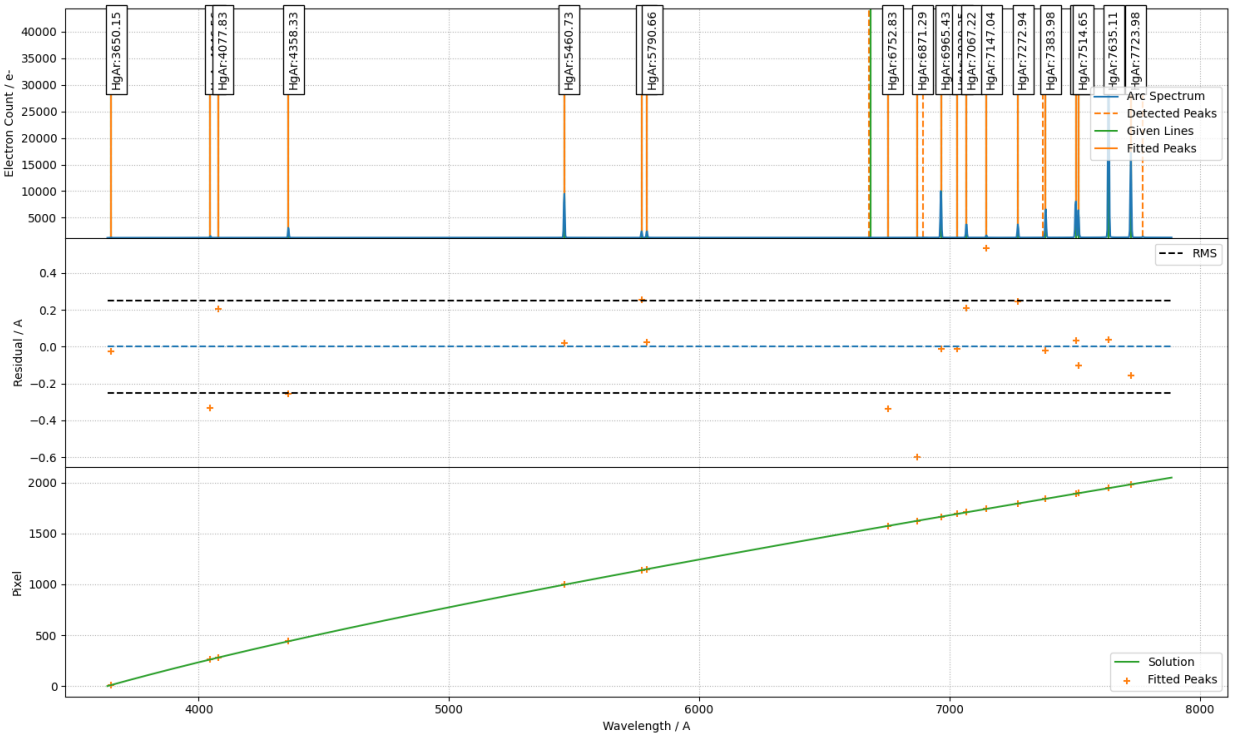


6. Add the line list to the calibrator and perform the hough transform on the pixel-wavelength pairs that will be used by the RANSAC sampling and fitting.

```
c.add_user_atlas(elements=element,
                 wavelengths=atlas,
                 constrain_poly=True)
c.do_hough_transform()
```

6. Perform polynomial fit on samples drawn from RANSAC, the deafult option is to fit with polynomial function.

```
(fit_coeff, matched_peaks, matched_atlas, rms, residual, peak_utilisation,
 atlas_utilisation) = c.fit(max_tries=200, fit_tolerance=10., fit_deg=4)

c.plot_fit(fit_coeff,
           plot_atlas=True,
           log_spectrum=False,
           tolerance=5.)
```



with some INFO output looking like this:

```
INFO:rascal.calibrator:Peak at: 3650.115449538473 A
INFO:rascal.calibrator:- matched to 3650.153 A
INFO:rascal.calibrator:Peak at: 4046.8620798203365 A
INFO:rascal.calibrator:- matched to 4046.563 A
INFO:rascal.calibrator:Peak at: 4077.595238720406 A
INFO:rascal.calibrator:- matched to 4077.831 A
INFO:rascal.calibrator:Peak at: 4358.576445488649 A
INFO:rascal.calibrator:- matched to 4358.328 A
INFO:rascal.calibrator:Peak at: 5460.7567985483165 A
INFO:rascal.calibrator:- matched to 5460.735 A
INFO:rascal.calibrator:Peak at: 5769.381771912414 A
```

(continues on next page)

(continued from previous page)

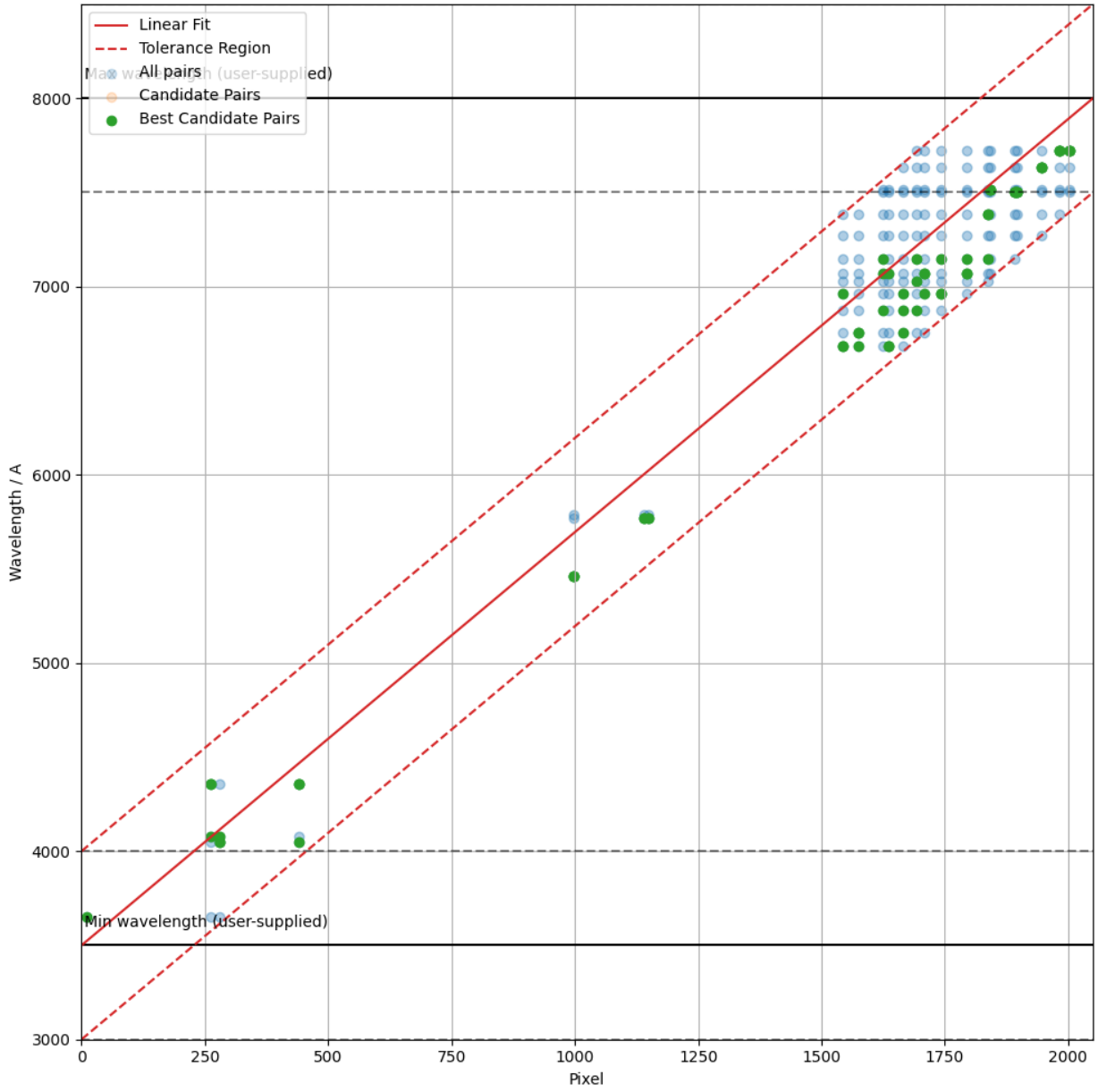
```
INFO:rascal.calibrator:- matched to 5769.598 A
INFO:rascal.calibrator:Peak at: 5790.674908711722 A
INFO:rascal.calibrator:- matched to 5790.663 A
INFO:rascal.calibrator:Peak at: 6677.195577604758 A
INFO:rascal.calibrator:Peak at: 6753.169173651684 A
INFO:rascal.calibrator:- matched to 6752.834 A
INFO:rascal.calibrator:Peak at: 6871.881456822988 A
INFO:rascal.calibrator:- matched to 6871.289 A
INFO:rascal.calibrator:Peak at: 6894.527274171375 A
INFO:rascal.calibrator:Peak at: 6965.436232273663 A
INFO:rascal.calibrator:- matched to 6965.431 A
INFO:rascal.calibrator:Peak at: 7030.256183107306 A
INFO:rascal.calibrator:- matched to 7030.251 A
INFO:rascal.calibrator:Peak at: 7067.0001611640355 A
INFO:rascal.calibrator:- matched to 7067.218 A
INFO:rascal.calibrator:Peak at: 7146.4973429914635 A
INFO:rascal.calibrator:- matched to 7147.042 A
INFO:rascal.calibrator:Peak at: 7272.679474233918 A
INFO:rascal.calibrator:- matched to 7272.936 A
INFO:rascal.calibrator:Peak at: 7373.953964433149 A
INFO:rascal.calibrator:Peak at: 7383.992970996112 A
INFO:rascal.calibrator:- matched to 7383.981 A
INFO:rascal.calibrator:Peak at: 7503.831035910258 A
INFO:rascal.calibrator:- matched to 7503.869 A
INFO:rascal.calibrator:Peak at: 7514.750835501081 A
INFO:rascal.calibrator:- matched to 7514.652 A
INFO:rascal.calibrator:Peak at: 7635.073536260965 A
INFO:rascal.calibrator:- matched to 7635.106 A
INFO:rascal.calibrator:Peak at: 7724.151294456838 A
INFO:rascal.calibrator:- matched to 7723.98 A
INFO:rascal.calibrator:Peak at: 7772.136274228791 A
```

7. Quantify the quality of fit

```
print("RMS: {}".format(rms))
print("Stdev error: {} A".format(np.abs(residual).std()))
print("Peaks utilisation rate: {}%".format(peak_utilisation*100))
print("Atlass utilisation rate: {}%".format(atlas_utilisation*100))
```

8. We can also inspect the search space in the Hough parameter-space where the samples were drawn by running:

```
c.plot_search_space()
```


Linear Fit
Tolerance Region
All pairs
Candidate Pairs
Best Candidate Pairs
Min wavelength (user-supplied)
Wavelength / A
Pixel
8000
7000
6000
5000
4000
3000
0
250
500
750
1000
1250
1500
1750
2000

3.11 Calibrator

class rascal.calibrator.**HoughTransform**

This handles the hough transform operations on the pixel-wavelength space.

add_hough_points(*hp*)

Extending the Hough pairs with an externally supplied HoughTransform object. This can be useful if the arc lines are very concentrated in some wavelength ranges while nothing in available in another part.

Parameters **hp** (*numpy.ndarray with 2 columns or HoughTransform object*) – An externally supplied HoughTransform object that contains hough_points.

bin_hough_points(*xbins*, *ybins*)

Bin up data by using a 2D histogram method.

Parameters

- **xbins** (*int*) – The number of bins in the pixel direction.
- **ybins** (*int*) – The number of bins in the wavelength direction.

generate_hough_points(*x*, *y*, *num_slopes*)

Calculate the Hough transform for a set of input points and returns the 2D Hough hough_points matrix.

Parameters

- **x** (*1D numpy array*) – The x-axis represents peaks (pixel).
- **y** (*1D numpy array*) – The y-axis represents lines (wavelength). Vertical lines (i.e. infinite gradient) are not accommodated.
- **num_slopes** (*int*) – The number of slopes to be generated.

generate_hough_points_brute_force(*x*, *y*)

Calculate the Hough transform for a set of input points and returns the 2D Hough hough_points matrix.

Parameters

- **x** (*1D numpy array*) – The x-axis represents peaks (pixel).
- **y** (*1D numpy array*) – The y-axis represents lines (wavelength). Vertical lines (i.e. infinite gradient) are not accommodated.
- **num_slopes** (*int*) – The number of slopes to be generated.

load(*filename='hough_transform'*, *filetype='npy'*)

Store the binned Hough space and/or the raw Hough pairs.

Parameters

- **filename** (*str (default: 'hough_transform')*) – The filename of the output, not used if to_disk is False. It will be appended with the content type.
- **filetype** (*str (default: 'npy')*) – The file type of the saved hough transform. Choose from 'npy' and 'json'.

save(*filename='hough_transform'*, *fileformat='npy'*, *delimiter='+'*, *to_disk=True*)

Store the binned Hough space and/or the raw Hough pairs.

Parameters

- **filename** (*str*) – The filename of the output, not used if to_disk is False. It will be appended with the content type.
- **format** (*str (default: 'npy')*) – Choose from 'npy' and json'

- **delimiter** (*str (default: '+')*) – Delimiter for format and content types
- **to_disk** (*boolean*) – Set to True to save to disk, else return a numpy array object

Returns **hp_hough_points** – only return if to_disk is False.

Return type numpy.ndarray

set_constraints(*min_slope*, *max_slope*, *min_intercept*, *max_intercept*)

Define the minimum and maximum of the intercepts (wavelength) and gradients (wavelength/pixel) that Hough pairs will be generated.

Parameters

- **min_slope** (*int or float*) – Minimum gradient for wavelength/pixel
- **max_slope** (*int or float*) – Maximum gradient for wavelength/pixel
- **min_intercept** (*int/float*) – Minimum interception point of the Hough line
- **max_intercept** (*int/float*) – Maximum interception point of the Hough line

class rascal.calibrator.**Calibrator**(*peaks*, *spectrum=None*)

Initialise the calibrator object.

Parameters

- **peaks** (*list*) – List of identified arc line pixel values.
- **spectrum** (*list*) – The spectral intensity as a function of pixel.

add_atlas(*elements*, *min_atlas_wavelength=None*, *max_atlas_wavelength=None*, *min_intensity=10.0*, *min_distance=10.0*, *candidate_tolerance=10.0*, *constrain_poly=False*, *vacuum=False*, *pressure=101325.0*, *temperature=273.15*, *relative_humidity=0.0*)

Adds an atlas of arc lines to the calibrator, given an element.

Arc lines are taken from a general list of NIST lines and can be filtered using the minimum relative intensity (note this may not be accurate due to instrumental effects such as detector response, dichroics, etc) and minimum line separation.

Lines are filtered first by relative intensity, then by separation. This is to improve robustness in the case where there is a strong line very close to a weak line (which is within the separation limit).

The vacuum to air wavelength conversion is deafult to False because observatories usually provide the line lists in the respective air wavelength, as the corrections from temperature and humidity are small. See https://emtoolbox.nist.gov/Wavelength/Documentation.asp

Parameters

- **elements** (*string or list of strings*) – Chemical symbol, case insensitive
- **min_atlas_wavelength** (*float (default: None)*) – Minimum wavelength of the arc lines.
- **max_atlas_wavelength** (*float (default: None)*) – Maximum wavelength of the arc lines.
- **min_intensity** (*float (default: None)*) – Minimum intensity of the arc lines. Refer to NIST for the intensity.
- **min_distance** (*float (default: None)*) – Minimum separation between neighbouring arc lines.
- **candidate_tolerance** (*float (default: 10)*) – toleranceold (Angstroms) for considering a point to be an inlier during candidate peak/line selection. This should be reasonable small as we want to search for candidate points which are *locally* linear.

- **constrain_poly** (*boolean*) – Apply a polygonal constraint on possible peak/atlas pairs
- **vacuum** (*boolean*) – Set to True if the light path from the arc lamb to the detector plane is entirely in vacuum.
- **pressure** (*float*) – Pressure when the observation took place, in Pascal. If it is not known, we suggest you to assume 10% decrement per 1000 meter altitude.
- **temperature** (*float*) – Temperature when the observation took place, in Kelvin.
- **relative_humidity** (*float*) – In percentage.

add_pix_wave_pair(*pix*, *wave*)

Adding extra pixel-wavelength pair to the Calibrator for refitting. This DOES NOT work before the Calibrator having fit for a solution yet: use set_known_pairs() for that purpose.

Parameters

- **pix** (*float*) – pixel position
- **wave** (*float*) – wavelength

add_user_atlas(*elements*, *wavelengths*, *intensities=None*, *candidate_tolerance=10.0*, *constrain_poly=False*, *vacuum=False*, *pressure=101325.0*, *temperature=273.15*, *relative_humidity=0.0*)

Add a single or list of arc lines. Each arc line should have an element label associated with it. It is recommended that you use a standard periodic table abbreviation (e.g. 'Hg'), but it makes no difference to the fitting process.

The vacuum to air wavelength conversion is deafult to False because observatories usually provide the line lists in the respective air wavelength, as the corrections from temperature and humidity are small. See https://emtoolbox.nist.gov/Wavelength/Documentation.asp

Parameters

- **elements** (*list/str*) – Elements (required). Preferably a standard (i.e. periodic table) name for convenience with built-in atlases
- **wavelengths** (*list/float*) – Wavelengths to add (Angstrom)
- **intensities** (*list/float*) – Relative line intensities (NIST value)
- **candidate_tolerance** (*float (default: 15)*) – toleranceold (Angstroms) for considering a point to be an inlier during candidate peak/line selection. This should be reasonable small as we want to search for candidate points which are *locally* linear.
- **constrain_poly** (*boolean*) – Apply a polygonal constraint on possible peak/atlas pairs
- **vacuum** (*boolean*) – Set to true to convert the input wavelength to air-wavelengths based on the given pressure, temperature and humidity.
- **pressure** (*float*) – Pressure when the observation took place, in Pascal. If it is not known, assume 10% decrement per 1000 meter altitude
- **temperature** (*float*) – Temperature when the observation took place, in Kelvin.
- **relative_humidity** (*float*) – In percentage.

clear_atlas()

Remove all the lines loaded to the Calibrator.

fit(*max_tries=500*, *fit_deg=4*, *fit_coeff=None*, *fit_tolerance=5.0*, *fit_type='poly'*, *candidate_tolerance=2.0*, *brute_force=False*, *progress=True*)

Solve for the wavelength calibration polynomial by getting the most likely solution with RANSAC.

Parameters

- **max_tries** (*int (default: 5000)*) – Maximum number of iteration.
- **fit_deg** (*int (default: 4)*) – The degree of the polynomial to be fitted.
- **fit_coeff** (*list (default: None)*) – Set the baseline of the least square fit. If no fits outform this set of polynomial coefficients, this will be used as the best fit.
- **fit_tolerance** (*float (default: 5.0)*) – Sets a tolerance on whether a fit found by RANSAC is considered acceptable
- **fit_type** (*string (default: 'poly')*) – One of 'poly', 'legendre' or 'chebyshev'
- **candidate_tolerance** (*float (default: 2.0)*) – toleranceold (Angstroms) for considering a point to be an inlier
- **brute_force** (*boolean (default: False)*) – Set to True to try all possible combination in the given parameter space
- **progress** (*boolean (default: True)*) – True to show progress with tdqm. It is overrid if tdqm cannot be imported.

Returns

- **fit_coeff** (*list*) – List of best fit polynomial fit_coefficient.
- **rms** (*float*) – The root-mean-squared of the residuals
- **residual** (*float*) – Residual from the best fit
- **peak_utilisation** (*float*) – Fraction of detected peaks (pixel) used for calibration [0-1].
- **atlas_utilisation** (*float*) – Fraction of supplied arc lines (wavelength) used for calibration [0-1].

get_pix_wave_pairs()

Return the list of matched_peaks and matched_atlas with their position in the array.

Returns **pw_pairs** – List of tuples each containing the array position, peak (pixel) and atlas (wavelength).

Return type list

list_atlas()

List all the lines loaded to the Calibrator.

load_hough_transform(*filename='hough_transform'*, *filetype='npy'*)

Store the binned Hough space and/or the raw Hough pairs.

Parameters

- **filename** (*str (default: 'hough_transform')*) – The filename of the output, not used if to_disk is False. It will be appended with the content type.
- **filetype** (*str (default: 'npy')*) – The file type of the saved hough transform. Choose from 'npy' and 'json'.

manual_refit(*matched_peaks=None*, *matched_atlas=None*, *degree=None*, *x0=None*)

Perform a refinement of the matched peaks and atlas lines.

This function takes lists of matched peaks and atlases, along with user-specified lists of lines to add/remove from the lists.

Any given peaks or atlas lines to remove are selected within a user-specified tolerance, by default 1 pixel and 5 atlas Angstrom.

The final set of matching peaks/lines is then matched using a robust polyfit of the desired degree. Optionally, an initial fit x0 can be provided to condition the optimiser.

The parameters are identical in the format in the fit() and match_peaks() functions, however, with manual changes to the lists of peaks and atlas, peak_utilisation and atlas_utilisation are meaningless so this function does not return in the same format.

Parameters

- **matched_peaks** (*list*) – List of matched peaks
- **matched_atlas** (*list*) – List of matched atlas lines
- **degree** (*int*) – Polynomial fit degree (Only used if x0 is None)
- **x0** (*list*) – Initial fit coefficients

Returns

- **fit_coeff** (*list*) – List of best fit polynomial coefficients
- **matched_peaks** (*list*) – List of matched peaks
- **matched_atlas** (*list*) – List of matched atlas lines
- **rms** (*float*) – The root-mean-squared of the residuals
- **residuals** (*numpy 1D array*) – Residual match error per-peak

match_peaks(*fit_coeff=None*, *n_delta=None*, *refine=False*, *tolerance=10.0*, *method='Nelder-Mead'*, *convergence=1e-06*, *min_frac=0.5*, *robust_refit=True*, *fit_deg=None*)

** refine option is EXPERIMENTAL, use with caution **

Refine the polynomial fit fit_coefficients. Recommended to use in it multiple calls to first refine the lowest order and gradually increase the order of fit_coefficients to be included for refinement. This is be achieved by providing delta in the length matching the number of the lowest degrees to be refined.

Set refine to True to improve on the polynomial solution.

Set robust_refit to True to fit all the detected peaks with the given polynomial solution for a fit using maximal information, with the degree of polynomial = fit_deg.

Set both refine and robust_refit to False will return the list of arc lines are well fitted by the current solution within the tolerance limit provided.

Parameters

- **fit_coeff** (*list (default: None)*) – List of polynomial fit fit_coefficients.
- **n_delta** (*int (default: None)*) – The number of the lowest polynomial order to be adjusted
- **refine** (*boolean (default: True)*) – Set to True to refine solution.
- **tolerance** (*float (default: 10.)*) – Absolute difference between fit and model in the unit of nm.
- **method** (*string (default: 'Nelder-Mead')*) – scipy.optimize.minimize method.
- **convergence** (*float (default: 1e-6)*) – scipy.optimize.minimize tol.
- **min_frac** (*float (default: 0.5)*) – Minimum fractionof peaks to be refitted.
- **robust_refit** (*boolean (default: True)*) – Set to True to fit all the detected peaks with the given polynomial solution.

- **fit_deg** (*int (default: length of the input fit_coefficients)*) – Order of polynomial fit with all the detected peaks.

Returns

- **fit_coeff** (*list*) – List of best fit polynomial fit_coefficient.
- **peak_match** (*numpy 1D array*) – Matched peaks
- **atlas_match** (*numpy 1D array*) – Corresponding atlas matches
- **rms** (*float*) – The root-mean-squared of the residuals
- **residual** (*numpy 1D array*) – The difference (NOT absolute) between the data and the best-fit solution. * EXPERIMENTAL *
- **peak_utilisation** (*float*) – Fraction of detected peaks (pixel) used for calibration [0-1].
- **atlas_utilisation** (*float*) – Fraction of supplied arc lines (wavelength) used for calibration [0-1].

plot_arc(*pixel_list=None*, *log_spectrum=False*, *save_fig=False*, *fig_type='png'*, *filename=None*, *return_jsonstring=False*, *renderer='default'*, *display=True*)

Plots the 1D spectrum of the extracted arc.

Parameters

- **pixel_list** (*array (default: None)*) – pixel value of the of the spectrum, this is only needed if the spectrum spans multiple detector arrays.
- **log_spectrum** (*boolean (default: False)*) – Set to true to display the wavelength calibrated arc spectrum in logarithmic space.
- **save_fig** (*boolean (default: False)*) – Save an image if set to True. matplotlib uses the pyplot.save_fig() while the plotly uses the pio.write_html() or pio.write_image(). The support format types should be provided in fig_type.
- **fig_type** (*string (default: 'png')*) – Image type to be saved, choose from: jpg, png, svg, pdf and iframe. Delimiter is '+'.
- **filename** (*string (default: None)*) – Provide a filename or full path. If the extension is not provided it is defaulted to png.
- **return_jsonstring** (*boolean (default: False)*) – Set to True to return json strings if using plotly as the plotting library.
- **renderer** (*string (default: 'default')*) – Indicate the Plotly renderer. Nothing gets displayed if json is set to True.
- **display** (*boolean (Default: False)*) – Set to True to display disgnostic plot.

Returns

- *Return json strings if using plotly as the plotting library and json*
- *is True.*

plot_fit(*fit_coeff*, *spectrum=None*, *tolerance=5.0*, *plot_atlas=True*, *log_spectrum=False*, *save_fig=False*, *fig_type='png'*, *filename=None*, *return_jsonstring=False*, *renderer='default'*, *display=True*)

Plots of the wavelength calibrated arc spectrum, the residual and the pixel-to-wavelength solution.

Parameters

- **fit_coeff** (*1D numpy array or list*) – Best fit polynomial fit_coefficients
- **spectrum** (*1D numpy array (N)*) – Array of length N pixels

- **tolerance** (*float (default: 5)*) – Absolute difference between model and fitted wavelengths in unit of angstrom.
- **plot_atlas** (*boolean (default: True)*) – Display all the relavent lines available in the atlas library.
- **log_spectrum** (*boolean (default: False)*) – Display the arc in log-space if set to True.
- **save_fig** (*boolean (default: False)*) – Save an image if set to True. matplotlib uses the pyplot.save_fig() while the plotly uses the pio.write_html() or pio.write_image(). The support format types should be provided in fig_type.
- **fig_type** (*string (default: 'png')*) – Image type to be saved, choose from: jpg, png, svg, pdf and iframe. Delimiter is '+'.
- **filename** (*string (default: None)*) – Provide a filename or full path. If the extension is not provided it is defaulted to png.
- **return_jsonstring** (*boolean (default: False)*) – Set to True to return json strings if using plotly as the plotting library.
- **renderer** (*string (default: 'default')*) – Indicate the Plotly renderer. Nothing gets displayed if json is set to True.
- **display** (*boolean (Default: False)*) – Set to True to display disgnostic plot.

Returns

- *Return json strings if using plotly as the plotting library and json*
- *is True.*

plot_search_space(*fit_coeff=None*, *top_n_candidate=3*, *weighted=True*, *save_fig=False*, *fig_type='png'*, *filename=None*, *return_jsonstring=False*, *renderer='default'*, *display=True*)

Plots the peak/arc line pairs that are considered as potential match candidates.

If fit fit_coefficients are provided, the model solution will be overplotted.

Parameters

- **fit_coeff** (*list (default: None)*) – List of best polynomial fit_coefficients
- **top_n_candidate** (*int (default: 3)*) – Top ranked lines to be fitted.
- **weighted** (*(default: True)*) – Draw sample based on the distance from the matched known wavelength of the atlas.
- **save_fig** (*boolean (default: False)*) – Save an image if set to True. matplotlib uses the pyplot.save_fig() while the plotly uses the pio.write_html() or pio.write_image(). The support format types should be provided in fig_type.
- **fig_type** (*string (default: 'png')*) – Image type to be saved, choose from: jpg, png, svg, pdf and iframe. Delimiter is '+'.
- **filename** (*(default: None)*) – The destination to save the image.
- **return_jsonstring** (*(default: False)*) – Set to True to save the plotly figure as json string. Ignored if matplotlib is used.
- **renderer** (*(default: 'default')*) – Set the rendered for the plotly display. Ignored if matplotlib is used.
- **display** (*boolean (Default: False)*) – Set to True to display disgnostic plot.

Returns

Return type json object if json is True.

remove_atlas_lines_range(*wavelength*, *tolerance=10*)
Remove arc lines within a certain wavelength range.

Parameters

- **wavelength** (*float*) – Wavelength to remove (Angstrom)
- **tolerance** (*float*) – Tolerance around this wavelength where atlas lines will be removed

remove_pix_wave_pair(*arg*)
Remove fitted pixel-wavelength pair from the Calibrator for refitting. The positions can be found from get_pix_wave_pairs(). One at a time.

Parameters **arg** (*int*) – The position of the pairs in the arrays.

save_hough_transform(*filename='hough_transform'*, *fileformat='npy'*, *delimiter='+'*, *to_disk=True*)
Save the HoughTransform object to memory or to disk.

Parameters

- **filename** (*str*) – The filename of the output, not used if to_disk is False. It will be appended with the content type.
- **format** (*str (default: 'npy')*) – Choose from 'npy' and json'
- **delimiter** (*str (default: '+')*) – Delimiter for format and content types
- **to_disk** (*boolean*) – Set to True to save to disk, else return a numpy array object

Returns **hp_hough_points** – only return if to_disk is False.

Return type numpy.ndarray

set_calibrator_properties(*num_pix=None*, *pixel_list=None*, *plotting_library=None*, *seed=None*, *logger_name='Calibrator'*, *log_level='warning'*)
Initialise the calibrator object.

Parameters

- **num_pix** (*int*) – Number of pixels in the spectral axis.
- **pixel_list** (*list*) – pixel value of the of the spectrum, this is only needed if the spectrum spans multiple detector arrays.
- **plotting_library** (*string (default: 'matplotlib')*) – Choose between matplotlib and plotly.
- **seed** (*int*) – Set an optional seed for random number generators. If used, this parameter must be set prior to calling RANSAC. Useful for deterministic debugging.
- **logger_name** (*string (default: 'Calibrator')*) – The name of the logger. It can use an existing logger if a matching name is provided.
- **log_level** (*string (default: 'info')*) – Choose {critical, error, warning, info, debug, notset}.

set_hough_properties(*num_slopes=None*, *xbins=None*, *ybins=None*, *min_wavelength=None*, *max_wavelength=None*, *range_tolerance=None*, *linearity_tolerance=None*)

Parameters

- **num_slopes** (*int (default: 1000)*) – Number of slopes to consider during Hough transform

- **xbins** (*int (default: 50)*) – Number of bins for Hough accumulation
- **ybins** (*int (default: 50)*) – Number of bins for Hough accumulation
- **min_wavelength** (*float (default: 3000)*) – Minimum wavelength of the spectrum.
- **max_wavelength** (*float (default: 9000)*) – Maximum wavelength of the spectrum.
- **range_tolerance** (*float (default: 500)*) – Estimation of the error on the provided spectral range e.g. 3000-5000 with tolerance 500 will search for solutions that may satisfy 2500-5500
- **linearity_tolerance** (*float (default: 100)*) – A toleranceold (Ansgtroms) which defines some padding around the range tolerance to allow for non-linearity. This should be the maximum expected excursion from linearity.

set_known_pairs(*pix=()*, *wave=()*)

Provide manual pixel-wavelength pair(s), they will be appended to the list of pixel-wavelength pairs after the random sample being drawn from the RANSAC step, i.e. they are ALWAYS PRESENT in the fitting step. Use with caution because it can skew or bias the fit significantly, make sure the pixel value is accurate to at least 1/10 of a pixel. We do not recommend supplying more than a coupld of known pairs unless you are very confident with the solution and intend to skew with the known pairs.

This can be used for example for low intensity lines at the edge of the spectrum. Or saturated lines where peaks cannot be well positioned.

Parameters

- **pix** (*numeric value, list or numpy 1D array (N) (default: ())*) – Any pixel value, can be outside the detector chip and serve purely as anchor points.
- **wave** (*numeric value, list or numpy 1D array (N) (default: ())*) – The matching wavelength for each of the pix.

set_ransac_properties(*sample_size=None*, *top_n_candidate=None*, *linear=None*, *filter_close=None*, *ransac_tolerance=None*, *candidate_weighted=None*, *hough_weight=None*, *minimum_matches=None*, *minimum_peak_utilisation=None*, *minimum_fit_error=None*)

Configure the Calibrator. This may require some manual twiddling before the calibrator can work efficiently. However, in theory, a large max_tries in fit() should provide a good solution in the expense of performance (minutes instead of seconds).

Parameters

- **sample_size** (*int (default: 5)*) – Number of samples used for fitting, this is automatically set to the polynomial degree + 1, but a larger value can be specified here.
- **top_n_candidate** (*int (default: 5)*) – Top ranked lines to be fitted.
- **linear** (*boolean (default: True)*) – True to use the hough transformed gradient, otherwise, use the known polynomial.
- **filter_close** (*boolean (default: False)*) – Remove the pairs that are out of bounds in the hough space.
- **ransac_tolerance** (*float (default: 1)*) – The distance criteria (Angstroms) to be considered an inlier to a fit. This should be close to the size of the expected residuals on the final fit (e.g. 1A is typical)
- **candidate_weighted** (*boolean (default: True)*) – Set to True to down-weight pairs that are far from the fit.

- **hough_weight** (*float or None (default: 1.0)*) – Set to use the hough space to weigh the fit. The theoretical optimal weighting is unclear. The larger the value, the heavily it relies on the overdensity in the hough space for a good fit.
- **minimum_matches** (*int or None (default: 0)*) – Set to only accept fit solutions with a minimum number of matches. Setting this will prevent the fitting function from accepting spurious low-error fits.
- **minimum_peak_utilisation** (*int or None (default: 0)*) – Set to only accept fit solutions with a fraction of matches. This option is convenient if you don't want to specify an absolute number of atlas lines. Range is 0 - 1 inclusive.
- **minimum_fit_error** (*float or None (default: 1e-4)*) – Set to only accept fits with a minimum error. This avoids accepting "perfect" fit solutions with zero errors. However if you have an extremely good system, you may want to set this tolerance lower.

use_matplotlib()
Call to switch to matplotlib.

use_plotly()
Call to switch to plotly.

which_plotting_library()
Call to show if the Calibrator is using matplotlib or plotly library (or neither).

3.12 Models

rascal.models.**normalise_input**(*x*, *y*)
Transforms inputs to have unit variance

rascal.models.**poly_cost_function**(*a*, *x*, *y*, *degree*)
Polynomial cost function. Returns the absolute difference between the target value and predicted values.

Parameters

- **a** (*list*) – Polynomial coefficients
- **x** (*list*) – Values to evaluate polynomial at
- **y** (*list*) – Target values for each x
- **degree** (*int*) – Polynomial degree

Returns **residual** – y - f(x)

Return type list

rascal.models.**polynomial**(*a*, *degree=3*)
Returns a lambda function which computes an nth order polynormal:

f(x, a) = sum_i (a[degree-i] * x**i)

rascal.models.**robust_polyfit**(*x*, *y*, *degree=3*, *x0=None*)
Perform a robust polyfit given a set of values (x,y).

Specifically this function performs a least squares fit to the given data points using the robust Huber loss. Inputs are normalised prior to fitting.

Parameters

- **x** (*list*) – Data points
- **y** (*list*) – Target data to fit

- **degree** (*int*) – Polynomial degree to fit
- **x0** (*list or None*) – Initial coefficients

Returns **p** – Polynomial coefficients

Return type list

3.13 Synthetic

class rascal.synthetic.**SyntheticSpectrum**(*coefficients=None*, *min_wavelength=200.0*, *max_wavelength=1200.0*)

Creates a synthetic spectrum generator which, given a suitable model, outputs the expected pixel locations of input wavelengths. It is expected that this will be used mainly for model testing.

Parameters **coefficients** (*list*) – coefficients for the model

get_pixels(*wavelengths*)
Returns a list of pixel locations for the wavelengths provided

set_model(*coefficients*)
Set the model to fit

set_wavelength_limit(*min_wavelength=None*, *max_wavelength=None*)
Set a wavelength filter for the 'get_pixels' function.

3.14 Utility

rascal.util.**edlen_refraction**(*wavelengths*, *temperature*, *pressure*, *vapour_partial_pressure*)
Appendix A.IV of https://emtoolbox.nist.gov/Wavelength/Documentation.asp

rascal.util.**filter_intensity**(*lines*, *min_intensity=0*)
Filters a line list by an intensity threshold

Parameters

- **lines** (*list[tuple (str, float, float)]*) – A list of input lines where the 2nd parameter is intensity
- **min_intensity** (*int*) – Intensity threshold

Returns **lines** – Filtered line list

Return type list

rascal.util.**filter_separation**(*wavelengths*, *min_separation=0*)
Filters a wavelength list by a separation threshold.

Parameters

- **wavelengths** (*list*) – List of input wavelengths
- **min_separation** (*int*) – Separation threshold, Ansgtrom

Returns **distance_mask** – Mask of values which satisfy the separation criteria

Return type list

rascal.util.**filter_wavelengths**(*lines*, *min_atlas_wavelength*, *max_atlas_wavelength*)
Filters a wavelength list to a minimum and maximum range.

Parameters

- **lines** (*list*) – List of input wavelengths
- **min_atlas_wavelength** (*int*) – Min wavelength, Ansgtrom
- **max_atlas_wavelength** (*int*) – Max wavelength, Angstrom

Returns **lines** – Filtered wavelengths within specified range limit

Return type list

rascal.util.**gauss**(*x*, *a*, *x0*, *sigma*)
1D Gaussian

Parameters

- **x** – value or values to evaluate the Gaussian at
- **a** (*float*) – Magnitude
- **x0** (*float*) – Gaussian centre
- **sigma** (*float*) – Standard deviation (spread)

Returns **out** – The Gaussian function evaluated at provided x

Return type list

rascal.util.**get_vapour_partial_pressure**(*relative_humidity*, *vapour_pressure*)
Appendix A.II of https://emtoolbox.nist.gov/Wavelength/Documentation.asp

rascal.util.**get_vapour_pressure**(*temperature*)
Appendix A.I of https://emtoolbox.nist.gov/Wavelength/Documentation.asp

rascal.util.**load_calibration_lines**(*elements=[]*, *min_atlas_wavelength=3000*, *max_atlas_wavelength=15000*, *min_intensity=10*, *min_distance=10*, *vacuum=False*, *pressure=101325.0*, *temperature=273.15*, *relative_humidity=0.0*)
Load calibration lines from the standard NIST atlas. Rascal provides a cleaned set of NIST lines that can be used for general purpose calibration. It is recommended however that for repeated and robust calibration, the user should specify an instrument-specific atlas.

Provide a wavelength range suitable to your calibration source. You can also specify a minimum intensity that corresponds to the values listed in the NIST tables.

If you want air wavelengths (default), you can provide atmospheric conditions for your system. In most cases the default values of standard temperature and pressure should be sufficient.

Parameters

- **elements** (*list*) – List of short element names, e.g. He as per NIST
- **min_atlas_wavelength** (*int*) – Minimum wavelength to search, Angstrom
- **max_atlas_wavelength** (*int*) – Maximum wavelength to search, Angstrom
- **min_intensity** (*int*) – Minimum intensity to search, per NIST
- **max_intensity** (*int*) – Maximum intensity to search, per NIST
- **vacuum** (*bool*) – Return vacuum wavelengths
- **pressure** (*float*) – Atmospheric pressure, Pascal
- **temperature** (*float*) – Temperature in Kelvin, default room temp
- **relative_humidity** (*float*) – Relative humidity, percent

Returns **out** – Emission lines corresponding to the parameters specified

Return type list

rascal.util.**refine_peaks**(*spectrum*, *peaks*, *window_width=10*, *distance=None*)

Refine peak locations in a spectrum from a set of initial estimates.

This function attempts to fit a Gaussian to each peak in the provided list. It returns a list of sub-pixel refined peaks. If two peaks are very close, they can be refined to the same location. In this case only one of the peaks will be returned - i.e. this function will return a unique set of peak locations.

Parameters

- **spectrum** (*list*) – Input spectrum (list of intensities)
- **peaks** (*list*) – A list of peak locations in pixels
- **window_width** (*int*) – Size of window to consider in fit either side of initial peak location

Returns **refined_peaks** – A list of refined peak locations

Return type list

rascal.util.**vacuum_to_air_wavelength**(*wavelengths*, *temperature=273.15*, *pressure=101325*, *relative_humidity=0*)

The conversion follows the Modified Edlén Equations

https://emtoolbox.nist.gov/Wavelength/Documentation.asp

pressure drops by ~10% per 1000m above sea level temperature depends heavily on the location relative humidity is between 0-100, depends heavily on the location

Parameters

- **wavelengths** (*float or numpy.array*) – Wavelengths in vacuum
- **temperature** (*float*) – In unit of Kelvin
- **pressure** (*float*) – In unit of Pa
- **relative_humidity** (*float*) – Unitless in percentage (i.e. 0 - 100)

Returns **air wavelengths** – The wavelengths in air given the condition

Return type float or numpy.array

3.15 Change log

3.15.1 Version 0.3.0

Date 28 July 2021

- @jveitchmichaelis:
  - Unit testing of SyntheticSpectrum (#43)
  - Automate PyPI publishing (#40)
  - fit() sometimes get stuck (#38, #35)
  - Automate PyPI publishing (#37)
  - Manual removal of pixel-wavelength pairs after fitting (#30)
  - Manual removal of atlas lines (#29)

  - Renamed the primary branch to main (#28)

- @cylammarco:
  - Match fit() and match_peaks() output format (#44)
  - refine_peaks() handles nan (#42)
  - Include version log (#41)
  - refine_peaks() filters duplicated peaks (#36)
  - Fixed plot_fit() issue with matplotlib (#32)
  - Allow merging of HoughTranform objects (#31)

CHAPTER

FOUR

LICENSE & ATTRIBUTION

If you make use of RASCAL in your work, please cite our paper (arXiv, ADS, BibTeX).

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

S

U

V

W