AtomCRad Strain Analysis Code Overview

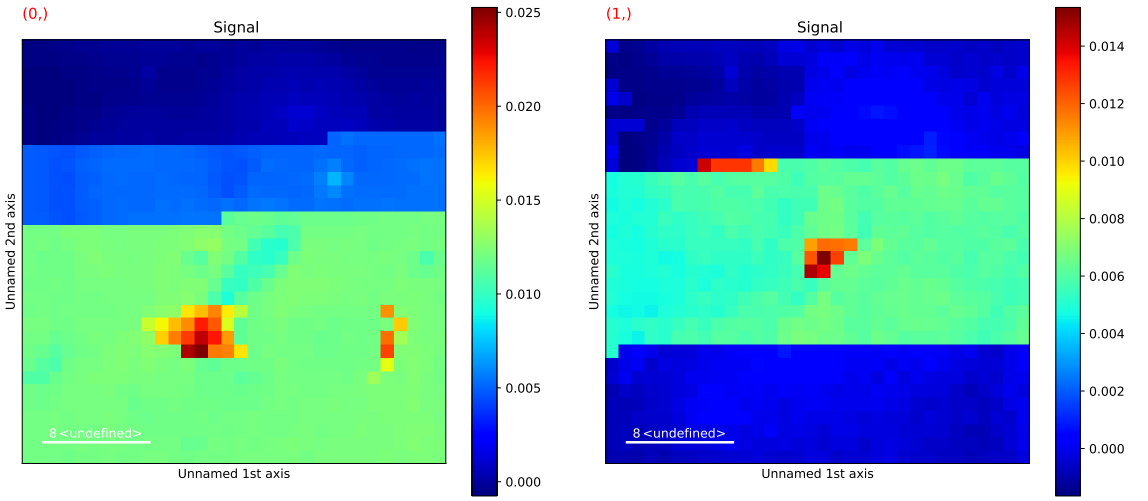
Sanket Gadgil

November 2020

1 Key Parts

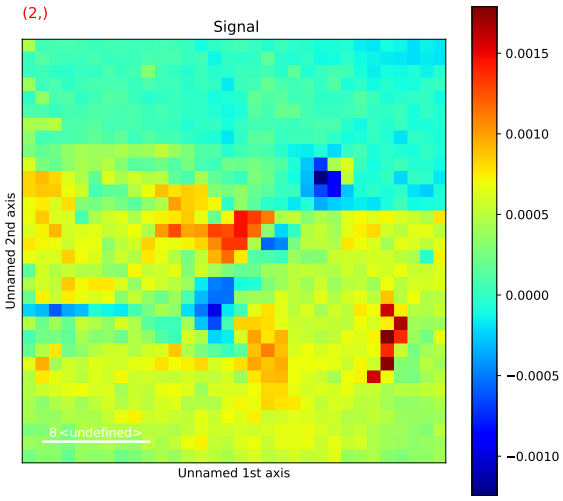
- Functionality implemented with `hyperspy` and `pyxem` for test data.
- Firstly the coupling between pixels, regarding the current pixel using the result from the previous pixel as an initial guess, was removed.
- The main addition is a parent-child algorithm which uses binned data at successively higher resolutions and takes the result from lower resolution data as a starting guess for higher resolution data.
- A bounded fit function was tested and implemented for the back-end of the parent-child algorithm.

2 Strain Maps using original fitting

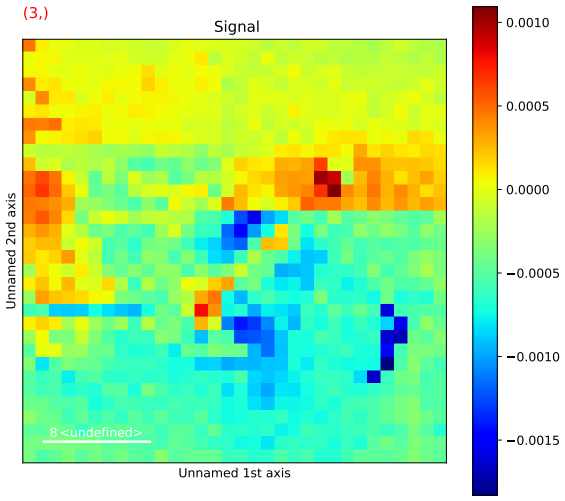


ϵ_{xx}

ϵ_{yy}



ϵ_{xy}



ϵ_{θ}

The iterator here is set to "flyback", this means that it goes about fitting by progressing from pixel to pixel such that it finish one row and goes to the start of the next row. As a result of relying on the results of the previous pixel, the fitting is prone to finding itself stuck in a local minimum and hence the vast regions invalid strain measurements.

3 Parent-Child algorithm

The current implementation of strain map analysis using `hyperspy` and `pyxem` as outlined in the demo files for `pyxem`[1] estimates the values of a certain set of free parameters for each

pixel,
(d_{11}, d_{12}, d_{21}, d_{22}, t_1, t_2), these define the elements of an affine transformation which is applied to the reference image in order to get it closely matching a given pixel's diffraction disk image. The first four values control the scaling, stretching and shearing of the image whilst the last two values control the translation of the image (the offset from centre). For completeness, the affine transform is defined as:

$$T_a = \begin{bmatrix} d_{11} & d_{12} & t_1 \\ d_{21} & d_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

The default initial guess for these parameters is $(d_{11}, d_{12}, d_{21}, d_{22}, t_1, t_2) = (1, 0, 0, 1, 0, 0)$. As an example, take a micro-graph of 256x256 pixels. Each pixel has it's own 128x128 resolution diffraction disk image. If one now bins the original micro-graph such that it now produces a lower resolution 2x2 image then each of the 4 pixels will have a diffraction disk image which is an average of four tiles which are averages of 64x64 pixel regions. The reference image (zero-strain) that each of those 4 diffraction disk patterns is compared to is itself a binned version of the original micro-graph. In the case of the reference image, it is a 1x1 pixel image which has one 128x128 diffraction disk image which is an average of the entire 256x256 region. There are now 4 initial guesses of the set of 6 parameters stored in `hyperspy model` object that will be applied to a component of the model called the `ScalableReferencePattern`, which is created by using the 1x1 pixel reference image. The fitting function modifies these 6 parameters for each of the 4 pixels from the binned data. These fitted parameters are stored in the model in a 2x2 arrangement with each element being a list of parameters. If now we wish to compare a higher resolution version of the micrograph, for example 4x4 pixels, then to pass the previous 2x2 parameter grid it is necessary to modify it such that it fits into a 4x4 grid. To demonstrate, it's useful to observe an example, in this case looking at the d_{11} parameter, denoted as x with subscripts corresponding to the grid position:

$$\begin{bmatrix} x_{0,0} & x_{0,1} \\ x_{1,0} & x_{1,1} \end{bmatrix} \rightarrow \begin{bmatrix} x_{0,0} & x_{0,0} & x_{0,1} & x_{0,1} \\ x_{0,0} & x_{0,0} & x_{0,1} & x_{0,1} \\ x_{1,0} & x_{1,0} & x_{1,1} & x_{1,1} \\ x_{1,0} & x_{1,0} & x_{1,1} & x_{1,1} \end{bmatrix} \xrightarrow{\text{fitting}} \begin{bmatrix} x^*_{0,0} & x^*_{0,1} & x^*_{0,2} & x^*_{0,3} \\ x^*_{1,0} & x^*_{1,1} & x^*_{1,2} & x^*_{1,3} \\ x^*_{2,0} & x^*_{2,1} & x^*_{2,2} & x^*_{2,3} \\ x^*_{3,0} & x^*_{3,1} & x^*_{3,2} & x^*_{3,3} \end{bmatrix}$$

As can be seen it is necessary to repeat the elements of the original grid such as $x_{0,0}$ in a 2x2 tile as is evident in the subsequent 4x4 grid. This is done for all of the parameters and used as the initial guess for the higher resolution fitting. The results from fitting the 4x4 grid of parameters is then used as the initial guess for the next level up in resolution, 8x8 pixels. This gives a good initial guess for each level which is closer to the ground truth answer than the default guess and allows more accurate results of the fitting algorithm.

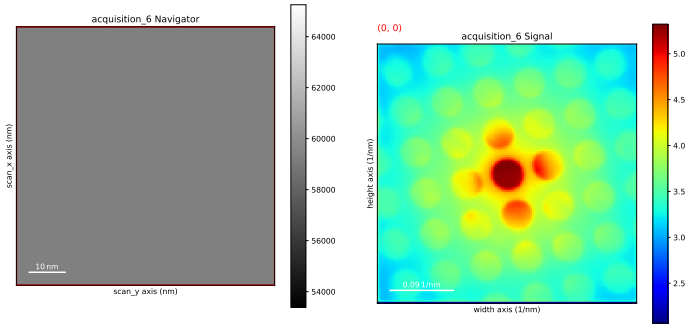
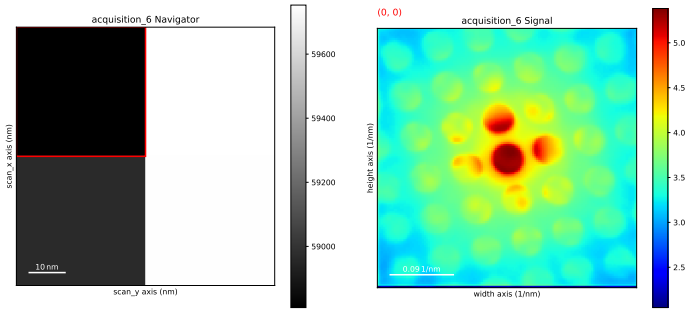
4 Code Modifications & Additions

To implement this algorithm, some changes to `model.fit(...)` and `model.multifit(...)` from `hyperspy` were needed. In order to make these changes, maintain compatibility and understand the treatment of the data, simplified replicas of the functions were constructed.

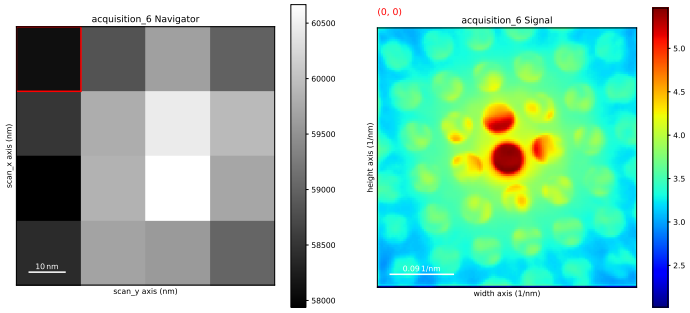
The main differences in these replicas are meant to be in how the parameters are handled before and after fitting. Originally the fitting algorithm uses the fitted parameters from the previous pixel(as it iterates through the micro-graph) as the initial guess for the next pixel. The first thing to do was to decouple this and have the fitting for every pixel start with the default parameters (shown above) as the initial guess. Once this was done and it was evident how to control the initial guess given to the fitting function. The replica `model.fit(...)` had an argument added to its definition which allowed the passing of an initial guess which itself would be passed to the `scipy.optimize.leastsq(...)` function. The bigger changes were made to the `model.multifit(...)` replica which was modified so that it could handle both the initial fitting step (with the 2x2 binned micrograph) and the subsequent higher resolution steps as well.

Furthermore a `jupyter` notebook has been written to perform strain analysis using these functions. The cells each represent a distinct portion of the workflow and for the most part contain individual functions. Starting with data import and pre-processing → data binning → model creation → model fitting → reference data determination → execution of the parent-child algorithm and plotting of results.

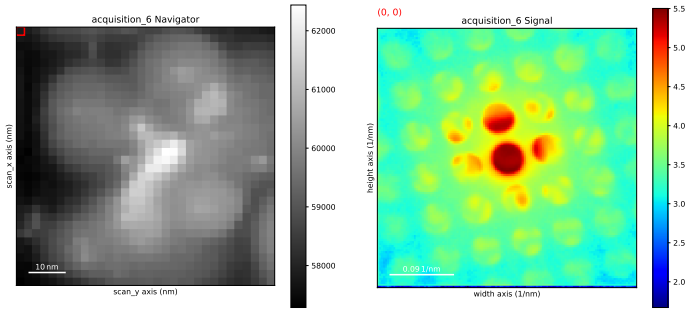
5 "Acquisition 6" dataset results

	Micrograph	Diffraction disk pattern at (0,0)
Reference data		
2x2 binned data		

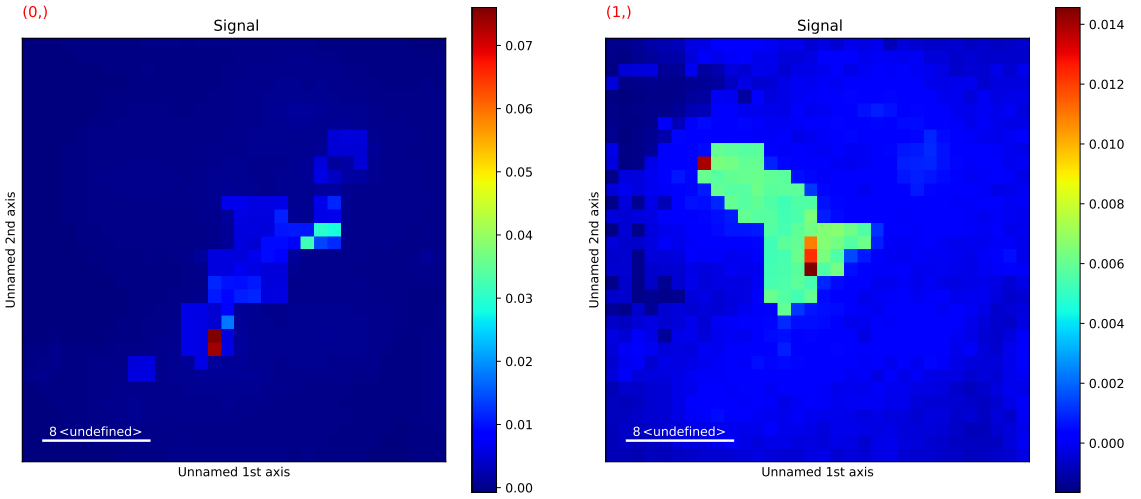
4x4 binned data



32x32 binned data

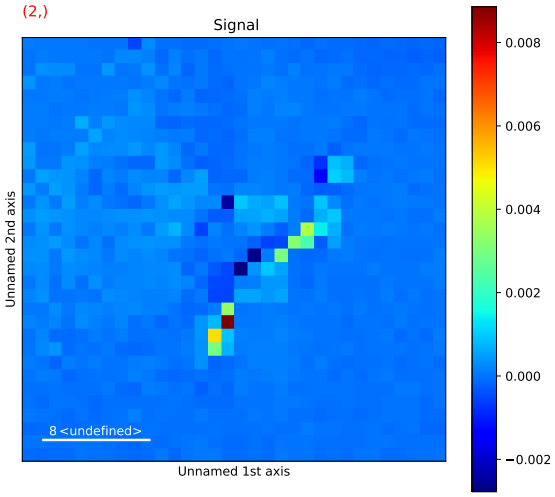


6 Strain Maps (32x32 binned data)

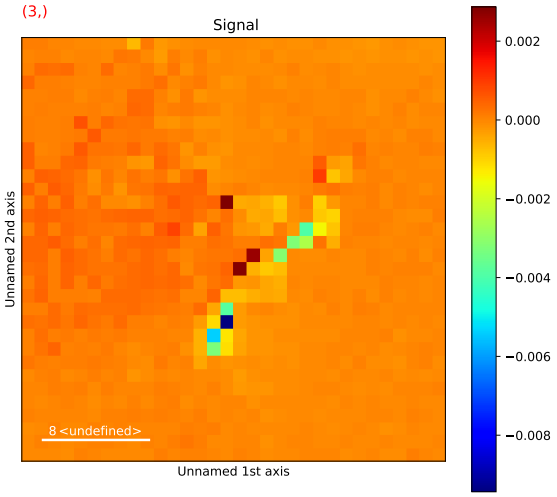


ϵ_{xx}

ϵ_{yy}



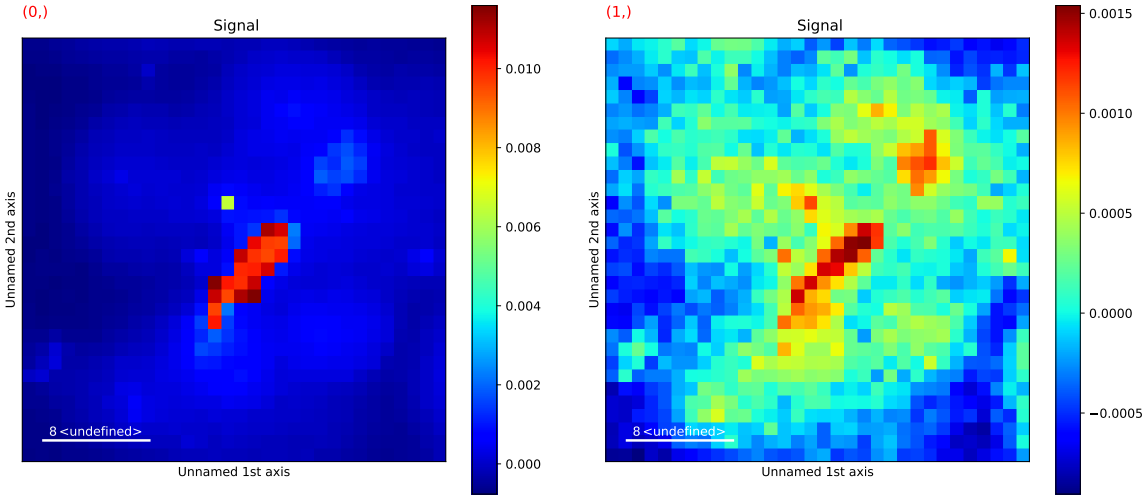
ϵ_{xy}



ϵ_{θ}

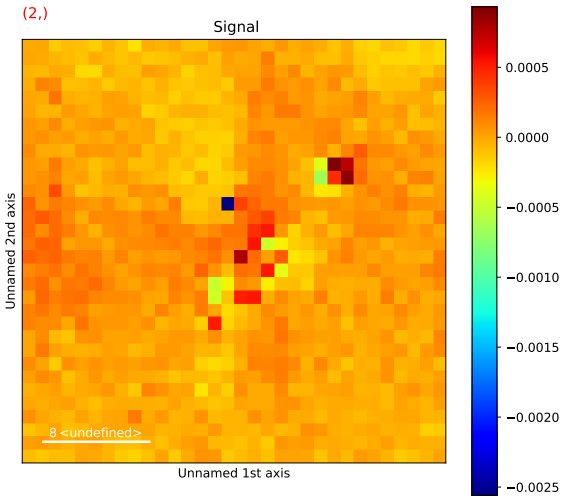
As can be seen from the strain maps, whilst there are no longer massive regions of invalid strain, there are still a few abnormal pixels (with strains greater than 5%) which skew the range of strain values. In order to fix this, the same parent-child algorithm was used with a bounded fit function. This places max/min constraints on the affine transformation parameters which in turn places a limit on the strains that are recovered.

7 Strain Maps (Bounded fit, 32x32 binned data)

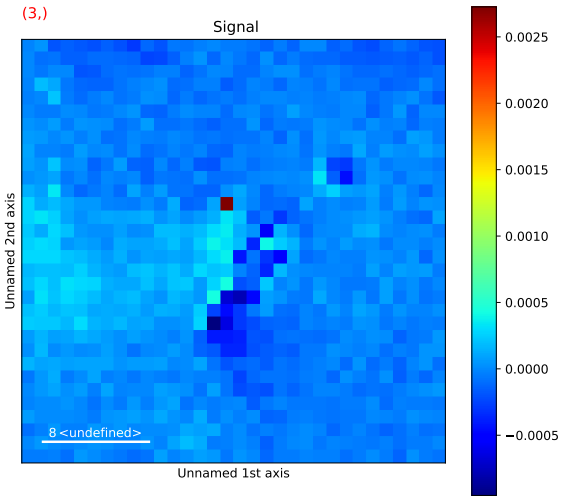


ϵ_{xx}

ϵ_{yy}



ϵ_{xy}



ϵ_{θ}

There are still artifacts of the lower resolution levels in these results but it does still show agreement with the main features of the 32x32 binned micrograph image.

8 Future Work

There are still many additions and refinements that can be made to this process but the basic framework seems to be valid. For example, whilst the θ strain data is outputted by the fitting algorithm, this is not a free parameter and cannot be set beforehand. Adding this to the list of fitting parameters would help with using this fitting method with a synthetic

reference image with zero rotation as well. Furthermore the background intensity of the diffraction patterns changes depending on a given pixel, this could also be added to the list of parameters. Alternatively, a synthetic reference pattern with it's own unique list of parameters could also be used since the fitting algorithm is more or less agnostic to the form of the `ScalableReferencePattern`.

9 Hyperspy Buglist

- Strain map plots do not have labelling for axes or scales. Daniel mentioned that the labelling for the scales in the diffraction pattern images may also be incorrect.
- The loading of TEM data using `.xml` files is a bit faulty at the moment. The diffraction patterns associated with each pixel of the micrograph are being imported incorrectly. The diffraction patterns are stored as 128 columns by 130 rows of pixels in the raw data. When importing, `hyperspy` is meant to discard the last two rows of the data (whose values are deliberately set to be near zero), instead `hyperspy.load(...)` is discarding the first two rows of the data. So there are two rows at the bottom of the imported diffraction patterns which are incorrect.
- Not a bug so much as a limitation, when faced with a particularly difficult fitting task the `leastsq(...)` function tends to struggle and chooses a solution that is completely unrealistic(strains $> 70\%$). This should be less of a problem now that the fit for each pixel is independent of another but regardless when it occurs it is easily spotted and should be noted.
- One other item of note with regards to the parent-child algorithm is that there can be issues due to the coupling between layers. These manifest as blocks where the fitting algorithm refuses to shift significantly from a solution it finds in one of the previous layers. This is demonstrated as such, where there is a "fingerprint" of the original 2x2 base level in the 32x32 strain map:

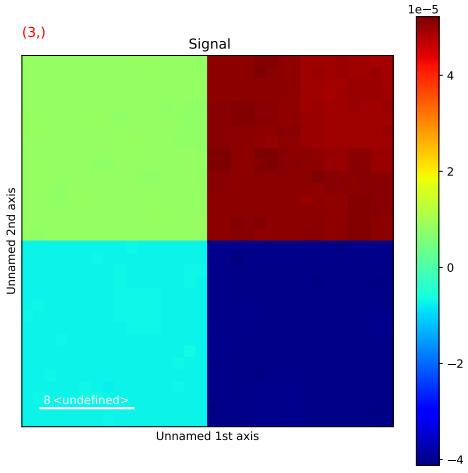


Figure 1: This is actually a 32x32 pixel strain map for θ taken from processing `acquisition_4` in the dataset. Observe the top right quadrant for evidence of the true resolution.

References

[1] https://github.com/pyxem/pyxem-demos/blob/master/05%20Simulate%20Data%20-%20Strain%20Mapping.ipynb