Ultramicroscopy 237 (2022) 113517

Contents lists available at ScienceDirect

Ultramicroscopy

journal homepage: www.elsevier.com/locate/ultramic

# Free, flexible and fast: Orientation mapping using the multi-core and GPU-accelerated template matching capabilities in the Python-based open source 4D-STEM analysis toolbox Pyxem

Niels Cautaerts [a], Phillip Crout [b], Håkon W. Ånes [c], Eric Prestat [d], Jiwon Jeong [a], Gerhard Dehm [a], Christian H. Liebscher [a,*]

[a] Max-Planck-Institut für Eisenforschung GmbH, Max-Planck-Straße 1, 40237, Düsseldorf, Germany
[b] Department of Materials Science & Metallurgy, University of Cambridge, 27 Charles Babbage Road, Cambridge, United Kingdom
[c] Department of Materials Science and Engineering, NTNU, Alfred Getz vei 2, N-7491, Trondheim, Norway
[d] Department of Materials, University of Manchester, The Mill E2, M13 9PL, Manchester, United Kingdom

ARTICLE INFO

*Keywords:*
Precession electron diffraction
Orientation mapping
Scanning/transmission electron microscopy
Open source
GPU acceleration
Template matching

ABSTRACT

This work presents the new template matching capabilities implemented in Pyxem, an open source Python library for analyzing four-dimensional scanning transmission electron microscopy (4D-STEM) data. Template matching is a brute force approach for deriving local crystal orientations. It works by comparing a library of simulated diffraction patterns to experimental patterns collected with nano-beam and precession electron diffraction (NBED and PED). This is a computationally demanding task, therefore the implementation combines efficiency and scalability by utilizing multiple CPU cores or a graphical processing unit (GPU). The code is built on top of the scientific Python ecosystem, and is designed to support custom and reproducible workflows that combine the image processing, template library generation, indexation and visualization all in one environment. The tools are agnostic to file size and format, which is significant in light of the increased adoption of pixelated detectors from different manufacturers. This paper details the implementation and validation of the method. The method is illustrated by calculating orientation maps of nanocrystalline materials and precipitates embedded in a crystalline matrix. The combination of speed and flexibility opens the door for automated parameter studies and real-time on-line orientation mapping inside the TEM.

## 1. Introduction

Scanning nanobeam electron diffraction (NBED) and precession electron diffraction (PED) have been applied for over two decades inside transmission electron microscopes (TEM) for orientation mapping in nano-crystalline materials [1,2]. These methods rely on scanning an electron probe with a small convergence angle on the order of 1 mrad across an electron transparent sample while capturing a nanobeam diffraction pattern image at each scan point. A common approach to extracting orientations from the diffraction patterns is to use template matching, whereby the pattern is compared to a large library of pre-computed templates of simulated diffraction patterns [1,2]. For sufficiently thin samples and a well aligned electron probe a spatial resolution of around 1 nm can be achieved [3]. However, the angular resolution of the technique tends to be limited to about 1° [4]. These limitations notwithstanding, the technique enjoys widespread use as a fast and convenient method for nano-scale orientation mapping and has been extensively utilized to determine grain orientations in nanocrystalline materials [5–7] even combined with in-situ straining [8].

Recently, interest in the field of 4-dimensional scanning transmission electron microscopy (4D-STEM) has been reignited thanks to the emergence of fast and direct electron detectors [9]. With direct electron detectors, high quality diffraction patterns can be collected in a fraction of the time compared to the camera systems that are currently most used. Additionally, as some of the authors of this manuscript have shown, NBED orientation mapping can also benefit from the improved signal-to-noise characteristics of pixelated detectors [3,10]. Improved detector technology results in datasets that are larger and more complex, requiring customizable analysis frameworks. As the complexity of the analysis pipelines increases it becomes increasingly important for them to be traceable and reproducible.

* Corresponding author.
*E-mail address:* c.liebscher@mpie.de (C.H. Liebscher).

https://doi.org/10.1016/j.ultramic.2022.113517
Received 14 November 2021; Received in revised form 7 March 2022; Accepted 18 March 2022
Available online 29 March 2022

For these reasons, a fast, scaleable, and flexible template matching workflow was implemented in the open source Pyxem library. The implementation makes use of the just-in-time (JIT) Numba compiler [11] for compiling performance critical parts of the code, Dask [12] for parallelizing the workload and processing datasets larger than memory, and CuPy [13] for performing the calculations on the GPU. Pyxem is a free and open source Python library that offers a large number of 4D-STEM data analysis routines [14]. It builds on the open source HyperSpy library [15] for microscopy data analysis. HyperSpy includes a large number of file readers, making workflows in Pyxem file format agnostic. With Pyxem and HyperSpy, analysis pipelines are built up with a minimal amount of Python code in Jupyter notebooks [16] or scripts; these are transparent and can be made completely reproducible. As the entire code base is written with Python syntax, development times are rapid and the code can be easily contributed to by non-professional programmers like scientists in the field. Pyxem is also bundled with two sub-libraries that are essential for practical orientation mapping: diffsims [17] for simulating the template libraries and orix [18,19] for analyzing and visualizing orientation data.

A few alternative template matching implementations have appeared in open source software packages recently, like Problematic [20], diffractem [21] and py4DSTEM [22]. Problematic and diffractem are aimed at analyzing data from serial diffraction and diffraction tomography. These datasets are usually a lot smaller than 4D-STEM datasets so different constraints may apply. py4DSTEM is specifically aimed at 4D-STEM data, but uses a very different template matching approach compared to the commercial solution which is very robust but can be slow for large template libraries and datasets.

This paper discusses the details of this new template matching implementation in Pyxem, demonstrates its performance, compares the orientation mapping results to those obtained with a commercial solution, and illustrates with case studies.

## 2. The algorithm and implementation

### 2.1. Simulation of diffraction pattern templates using diffsims

Indexation of orientations via template matching relies on a library of pre-computed diffraction pattern templates. The library is calculated with diffsims, a Pyxem sub-library [17]. Each template is stored as a list of Bragg reflections with reciprocal space coordinates $(k_x, k_y)_i$ and associated intensities $I_i$. The template library consists of a list of crystal orientations that each have an associated template. Sampling all of orientation space (the $SO(3)$ group) at small increments requires a very large template library, but the number of templates can be drastically reduced by observing that many orientations produce templates related by a rotation in the imaging plane. If orientations are represented by Euler angles $(\phi_1, \Phi, \phi_2)$ according to the Bunge convention [23], then all templates that share the same $\Phi$ and $\phi_2$ are related by in-plane rotation. Therefore, the list of orientations making up the library must only sample $(\Phi, \phi_2)$ or equivalently the surface of the unit sphere $(S^2)$; the in-plane angle $\phi_1$ is set to zero for all the templates and is fit to the image during the indexation process. The list of orientations is further reduced by crystal symmetry to a subsection of $S^2$ representing unique beam directions; all Laue groups are supported in the software.

The first step in creating a template library is to form a list of candidate orientations represented by points on the unit sphere surface. Different sampling schemes were implemented in diffsims as shown in Fig. 1(a); for the rest of this paper the *Spherified cube 2* sampling is used. When the sampling density is sufficiently high, i.e. when the sampling interval is smaller than the precision of the method, the type of sampling does not have a big influence on the results.

In the second step, a kinematical diffraction pattern is simulated for each orientation on the grid. A reciprocal space grid within a limiting radius is constructed, rotated by $(0, \Phi, \phi_2)$, and the distance of each grid point to the surface of the Ewald sphere is calculated. This geometric excitation error is used in a shape factor function, representing the relrods centered on the diffraction spots, to determine the spot intensity. Multiple shape factor functions are implemented and the user can provide custom functions if desired. The reflection intensities are determined by the structure factor $F$ through the well known relation $I = FF^*$; finally $I$ is multiplied by the shape factor to obtain the spot intensity. Structure factors are calculated using a user supplied crystal structure and atomic scattering factors calculated from parametrizations described in literature [24,25]. An alternative approach is to set atomic scattering factors to 1, which may be preferred for template matching [26] since it increases the weight of faint distant reflections in calculating the best matching template. The final step to obtaining the 2D diffraction pattern consists of projecting the spots close to the Ewald sphere onto the $k_x - k_y$ plane and discarding reflections below a minimum intensity threshold.

This process is illustrated in Fig. 1(b), where the large yellow surface represents part of the Ewald sphere and the points represent the reciprocal space grid of face centered cubic (FCC) Fe (serving as a simple prototype for austenitic steel). Points far away from the Ewald sphere are not excited and are represented with small blue spheres. Reflections that are close by and intense are represented as large spheres going from green, to yellow, to red when getting closer to the Ewald sphere. The resulting projected pattern is shown in Fig. 1(c); the size and color of the reflections serve to represent their intensity.

Diffsims can also take beam precession into consideration for simulating the patterns. A full treatment of precession requires a numerical integration of the relrods over the distance traced by the Ewald sphere. This calculation is possible in diffsims but is computationally demanding and impractical for simulating large template libraries. To simulate large libraries, the approximations described in Ref. [27] were implemented whereby the non-precessed intensity profile as a function of geometric excitation distance is assumed to follow a Lorentzian distribution. In this analytic approximation, the intensity of a diffraction spot under precession $I_{prec}$ is given by

$$I_{prec} = I_{tbint} \frac{\sigma}{\pi} \sqrt{2 \frac{u+z}{z^2}} \tag{1}$$

where $u = \sigma^2(g^2\phi^2 - s^2) + 1$ and $z = \sqrt{u^2 + 4\sigma^2 s^2}$. Here, $I_{tbint}$ is the integrated intensity of the relrod from negative to positive infinity, $\sigma$ is equal to $\pi t$ with $t$ the thickness of the sample, $g$ is the length of the reciprocal lattice vector, $\phi$ is the precession angle in radians, and $s$ is the geometric excitation error without beam precession.

Currently diffsims does not take dynamic diffraction or multiple diffraction into consideration. For simple crystal systems, considering dynamic effects in the simulation does not substantially improve the accuracy of the method. Dynamic effects mainly influence relative spot intensities, but accurate spot intensities are not critical for obtaining a good template matching result as was shown in Refs. [3,26] and was confirmed in this work. In more complex crystal systems, dynamic diffraction may produce kinematically forbidden reflections. These reflections would be missing in the templates, and since the template matching algorithm cannot distinguish between kinematical and dynamical reflections it may produce incorrect indexation results. For most materials, the issue of dynamical diffraction can be largely overcome by using beam precession during the experiment, since the Bragg conditions at which dynamic reflections are active are more restricted. In future versions of diffsims multiple diffraction may become supported.

### 2.2. Indexation workflow

Each image in the 4D-STEM dataset must be compared to each template in the library. Simultaneously, the in-plane angle $\phi_1$ that produces the best match between image and template must be found.

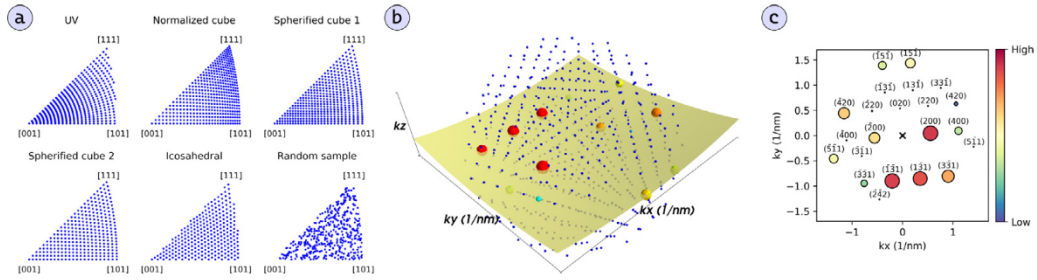


**Fig. 1.** Illustration of the template library procedure in diffsims. (a) Various orientation grids for the cubic crystal structure generated with different meshing algorithms. The orientation is represented by the beam directions plot in the stereographic projection. (b) The reciprocal space grid of FCC Fe rotated to $\Phi = 15°$ and $\phi_2 = 5°$. The Ewald sphere is represented by the yellow surface. Spots closer to the surface are more intense, represented with larger and more intense colors. (c) The projected diffraction pattern with indexed spots. Spots below a minimum intensity threshold were filtered out.

Following the method described in Ref. [28], the measure of best fit is taken to be the correlation index as described by Eq. (2):

$$Q = \frac{\sum_i P(x_i, y_i) T(x_i, y_i)}{\sqrt{\sum_i P(x_i, y_i)^2} \sqrt{\sum_i T(x_i, y_i)^2}}, \tag{2}$$

where $P(x, y)$ represent coordinates in the image and $T(x, y)$ coordinates in the template for all pixels $i$. In practice, the template is very sparse and only non-zero at coordinates corresponding to diffraction spots, so the sum in the numerator is reduced to a sum over all the template diffraction spots. The denominator serves the purpose of optionally normalizing $P$ and $T$; these values can be pre-computed and applied to images and templates before indexation, thereby reducing the equation for $Q$ to a dot product.

In the preparation of this manuscript alternative quality metrics such as Pearson's correlation coefficient were considered. This was motivated by perceived limitations of $Q$, including the fact that weak high-index reflections, which may be more sensitive to small changes in orientation than strong low-index reflections, do not substantially contribute to the dot product. In addition, erroneous reflections in the template that do not match with a reflection in the image are not penalized, because in calculating $Q$ their intensity is multiplied by the background intensity that is close to zero. However, alternative quality metrics did not noticeably improve the reliability of the indexation result compared to when $Q$ was used, but they were significantly more computationally expensive. In practice, carefully tailored image processing combined with $Q$ produced the best results. Typically a gamma correction with $\gamma < 1$ is applied, which raises each pixel value in the image to the power $\gamma$. This has the effect of increasing the relative intensity of weak reflections. Secondly, subtracting a constant from all the pixels in the image helps to overcome the second limitation of $Q$, as erroneous positive reflections in the template are multiplied by a negative value from the background, thereby penalizing $Q$.

In summary, image processing and template simulation conditions must be found such that:

1. the number of reflections in the image that are accounted for by reflections in the template are maximized
2. the number of reflections in the template that are not accounted for in the image are minimized

These metrics are easily used to qualitatively assess a well matching template, but they are difficult to translate directly to a quantitative optimization criterion. Realizing when the indexation algorithm has issues with either of these metrics and adjusting image processing and simulation conditions accordingly remains a manual effort of multiple iterations.

Different approaches can be used to optimize the in-plane rotation angle $\phi_1$ when comparing a template to an image. Currently Pyxem implements a direct approach which is illustrated in Fig. 2(a)–(e). The image (a) and template (b) are both converted to polar coordinates, (c) and (d) respectively. This is computationally efficient, but it is important that the center of the direct beam is used as the origin for the transformation. The optimal $\phi_1$ is found by sliding the template across the image, with wraparound, along the azimuthal axis $\phi$ and calculating $Q$ at each shifted position as shown in Fig. 2(e). The diffraction spot coordinates of the templates $(r_i, \phi_i')$ are rounded to the nearest integer so that they correspond directly to coordinates of pixels $P'(r_i, \phi_i')$ that are used to evaluate Eq. (2). The step size in $\phi$, and thus the resolution of the in-plane angle, is determined by the angular sampling used during the polar transform. It corresponds to 360° divided by the number of pixel columns in the polar image. In the example of Fig. 2(c), the polar image is 360 pixels wide so $\Delta\phi_1 = 1°$. The maximum correlation index and the angle at which it is achieved are stored, and the process is repeated for all templates in the library. The best correlation index for each template can be plotted on the stereographic projection as shown in Fig. 2(f). Note that the region in the stereographic projection is twice as large as the original stereographic projection representing the templates, since mirror images of all templates also represent unique orientations and must be considered. Mirrored templates are generated by reflecting spots across the $y = 0$ line. The experimental pattern is then indexed by selecting the template and corresponding in-plane angle with the highest correlation index as shown in Fig. 2(f) and (g). Alternatively, at the cost of some additional computation, the list of correlations can be sorted and the top $N$ best matching templates can be queried instead of only the best one.

The time complexity of the direct approach to match one template to one image is $O(S\phi)$ with $S$ the number of reflections in the template and $\phi$ the width of the polar image. If $S$ becomes very large and the template approaches a dense image, i.e. $S \approx r\phi$ with $r$ the height of the polar image, then the complexity becomes $O(r\phi^2)$. In this case a fast Fourier transform (FFT) based cross-correlation method with time complexity $O(r\phi \log \phi)$ may be preferred [29]. This method no longer depends on the number of diffraction spots in a template, but benchmarks showed that a few hundred diffraction spots per template are necessary before the cross-correlation approach outperforms the direct approach. An FFT based approach is used in the template matching implementation of py4DSTEM [22].

While the direct approach implementation is reasonably fast, it may still take half an hour to a few hours to index a dataset of a few gigabytes in size on a common laptop with a limited number of CPU cores. Therefore, an option was added to pre-filter the template library as described in Ref. [30]. In this method, the image and templates are integrated over the azimuthal direction, and the correlation between the integrated image profile and the integrated template profile library is calculated by a single matrix–vector product. Only the templates with the highest correlation values are passed onto the full indexation procedure. In this way the library for full indexation can be significantly reduced, and typically only a tenth of the templates must be preserved to arrive at the same optimum [30]. However, there is no guarantee that the optimal template will always be included in the pre-filtered set, and trials indicated that the method becomes less reliable for larger template libraries and templates containing more spots. An evaluation

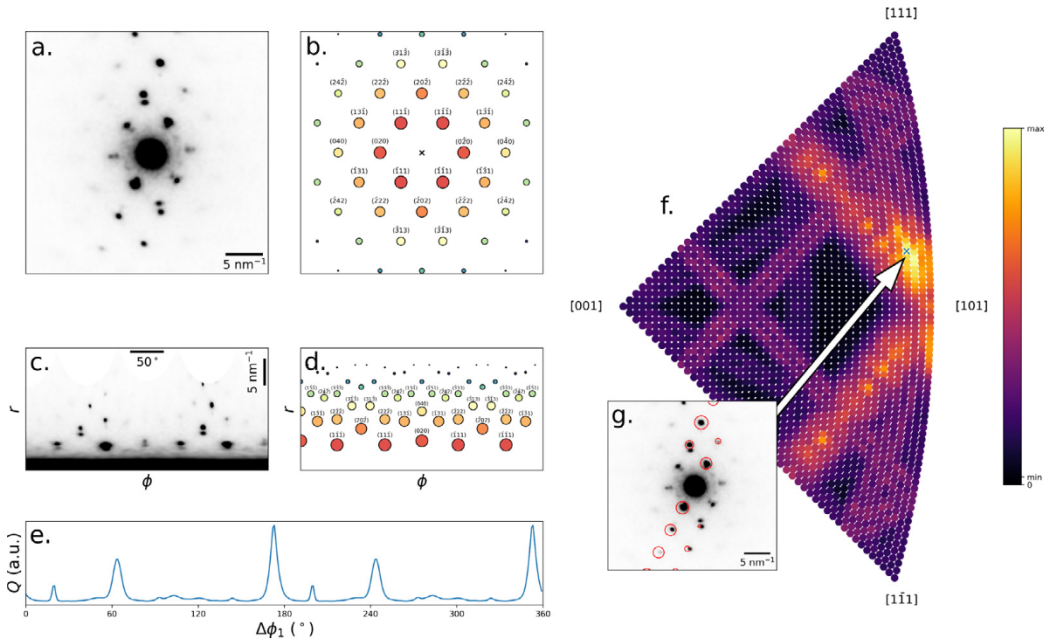


**Fig. 2.** Illustration of the indexation procedure. (a) A diffraction pattern and (b) a template are converted to polar coordinates as shown in (c) and (d) respectively. (e) The template is shifted across the polar image and the correlation index $Q$ is calculated at each position. The process is repeated for all templates and the mirror images of each template and the maximum correlation index and the angle at which it occurs is recorded for each. (f) The best correlation index for each template is mapped to each orientation and plotted on the stereographic projection. Better matching templates are brighter, and the template with the highest correlation is taken to be the solution. The minimum and maximum correlation values are indicated on the color bar; precise values of $Q$ are arbitrary and depend on image preprocessing and simulation parameters. (g) The reflections in the best matching template is represented with red circles on top of the experimental pattern.

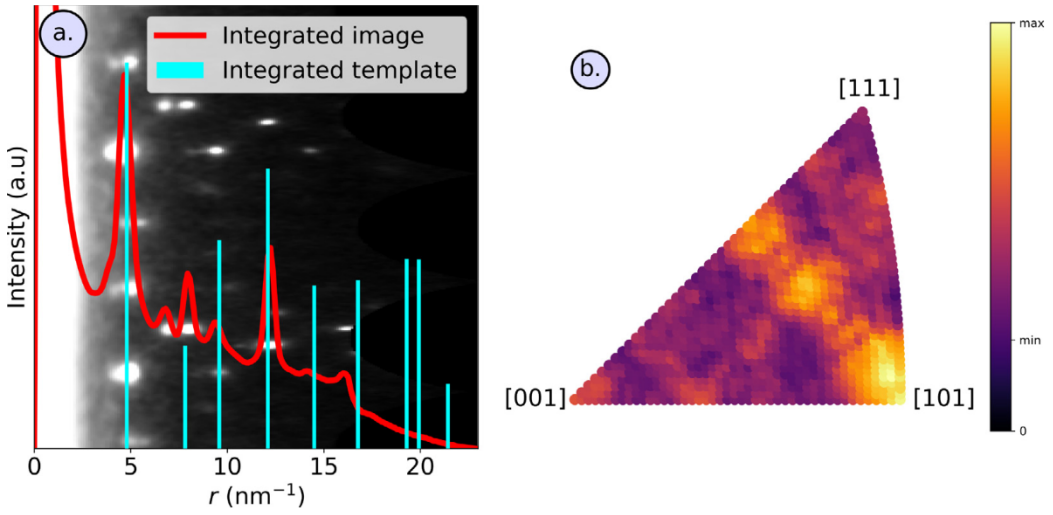


**Fig. 3.** Illustration of the fast matching between integrated images and templates on the same image and template library as in Fig. 2. (a) The polar image with the azimuthally integrated pattern plotted on top, as well as the azimuthally integrated best-fit template. (b) The correlation index between azimuthally integrated images and templates plotted on the stereographic projection. Features look comparable to the results obtained from the full matching procedure in Fig. 2(f).

of the effect of pre-filtering on the orientation mapping result is given in the supplementary materials; it is not used in any of the analyses in this paper. The integrated matching procedure is illustrated in Fig. 3.

To index a large dataset, the indexation procedure is simply repeated for each image. This is parallelized and scheduled as optimally as possible using Dask [12]. Dask also makes it possible to process datasets larger than computer memory by only loading parts of the dataset from storage when they need to be processed.

## 3. Materials, methods and datasets

Two datasets were collected inside a JEOL 2200FS TEM operating at 200 kV with a 4 K TemCAM-XF416 pixelated CMOS detector (TVIPS). The first dataset was collected on a sample of Cu-Ag alloy described in Ref. [31]. The dataset was thoroughly analyzed in Ref. [3]. The microscope conditions were a beam convergence angle of around 2 mrad, a camera length of 15 cm, a precession angle of 0.3° and precession frequency of 100 Hz. Images were collected at 2k×2k pixel resolution with a dwell time of 50 ms, but were binned to 512 × 512 pixels for analysis. The second dataset was collected on a sample of ion irradiated Ti-stabilized stainless steel containing nano-sized precipitates. The dataset is described in Ref. [10]. The convergence angle was around 0.5 mrad and the camera length 80 cm; the beam was not precessed. For both datasets, scan points were about 2 nm apart. Data was collected in the .tvips file format and converted to the .BLO and .HSPY formats using in house tools that have been made freely available [32]. All

analyses in this paper using Pyxem were run on a desktop computer with a 16 core AMD Ryzen 9 3950x CPU, an NVIDIA RTX 3080 GPU, and 64 GB of RAM running the Arch Linux operating system.

## 4. Results

### 4.1. Evaluation of reliability

To verify the reliability of the implementation, the Cu-Ag dataset was indexed both in Pyxem and using the commercial ASTAR software (NanoMegas). Indexation results from ASTAR are widely accepted in the community and thus form a suitable benchmark for comparison to the Pyxem method. In both softwares, the image pre-processing and template simulation parameters were optimized iteratively such that the indexation of randomly chosen diffraction patterns looked reasonable by visual inspection as per the two rules in Section 2.2. In Pyxem, the direct beam in each image was centered, the diffuse background in the images was removed using a difference-of-Gaussians procedure, and the resulting image was smoothed by Gaussian blur. Thresholding was used to set low intensity pixels to zero and gamma correction with exponent 0.5 was applied to enhance the weaker reflections. In both ASTAR and Pyxem a template library, derived from the FCC Cu structure, of around 11000 templates was used corresponding to a maximum angular deviation between neighboring orientations of 0.3°. In Pyxem atomic scattering was ignored in simulating the templates and a linear shape function was used for the relrods. Indexation was performed using the "full indexation" setting in ASTAR and without pre-filtering in Pyxem. Since ASTAR requires the images to be in 8-bit format, the original 16-bit data was truncated at pixel values of 1000 (the maximum pixel value was around 4000 in the direct beam) and the range 0–1000 was scaled to the 8-bit 0–255 range. The full Pyxem analysis workflow with parameters for the preceding steps is provided as a Jupyter notebook in a publicly accessible git repository [33]. The original data can be downloaded as a Zenodo dataset [34].

The inverse pole figure (IPF) maps for the Z, Y and X direction are shown in Fig. 4(a), (d) and (g) for Pyxem and Fig. 4(b), (e), and (h) for ASTAR respectively. To highlight the differences between the maps, the angular deviation between the ASTAR and Pyxem results are shown in Fig. 4(c), (f), and (i). This is calculated as

$$\theta = \left| \cos^{-1}\left( \frac{v_A \cdot v_P}{|v_A||v_P|} \right) \right|, \tag{3}$$

with $v_A$ and $v_P$ the vector represented in each pixel of the ASTAR and Pyxem maps.

Fig. 4(j) and (k) show the correlation index maps (Eq. (2)) for Pyxem and ASTAR respectively. These two $Q$ maps cannot be compared on a pixel by pixel basis because the simulation of templates and normalization are done differently, resulting in different linear scaling.

Correspondence between the two methods is good; for most points in the grain interiors the deviation in grain orientation is less than 3°. The deviation is largest in some pixels near grain boundaries, and in locations where $Q$ is low. In these regions, there may be ambiguity in the diffraction patterns when multiple crystals contribute to the signal. Each template in the library can only represent a rotated single crystal, so ambiguous diffraction patterns can be indexed differently depending on small differences between image preprocessing parameters.

In some locations in the grain interior the two maps differ substantially, as shown for one diffraction pattern in Fig. 5(a) and (b). The inset shows the location on the map where the pattern was extracted. Pixels where the maps differed substantially were associated with ambiguous diffraction patterns containing contributions from overlapping crystals. The template identified by Pyxem accounts for most of the bright reflections but fails to capture the reflections marked with white arrows that originate from another grain. The ASTAR indexation does account for these reflections, but ASTAR misses the bright systematic row. Clearly this is a region where the green grain on the left and the orange grain on the right overlap, resulting in ambiguity. Indexing overlapping grains is possible in specific cases by using masking techniques [35] but they were not implemented for this example. Fig. 5(c) shows the correlation map on the stereographic projection as calculated by Pyxem, with the best orientations according to Pyxem and ASTAR indicated. Both represent local maxima, showing that a minor change in simulation parameters or image processing conditions could change their relative height.

In regions with small deviations between the ASTAR and Pyxem solutions there were no clear metrics to favor one solution over the other, suggesting that for this dataset and method the angular resolution is limited to about 3° although this might be improved by additional image processing [3].

### 4.2. Performance

The problem of comparing one image to all templates in the library in the current implementation has a time complexity of $O\left(\frac{NS}{\Delta\phi_1}\right)$, where $N$ is the number of templates in the library, $S$ is the average number of reflections in each template, and $\Delta\phi_1$ is the in-plane angular sampling. This is demonstrated in Fig. 6(a), which shows the time required to index a single pattern for various combinations of $N$, $S$, and $\Delta\phi_1$. Each point represents the mean of 5 independent measurements, the error bar represents ± the standard deviation. The linear relationship between time and the problem size metric $\frac{NS}{\Delta\phi_1}$ is apparent both for the CPU and GPU implementations. On the hardware tested here (see Section 3), the slope of the line of best fit was approximately $5.5 \times 10^{-8}$ s° on 16 CPU cores, and $1.1 \times 10^{-8}$ s° on the GPU. To provide a fair basis for comparison, the CPU implementation was also constrained to a single core, which yielded a slope of $8.0 \times 10^{-7}$ s°. This shows that the speed scales linearly with CPU cores. The GPU is 5 times faster than 16 cores for large problems, so 80 CPU cores would be necessary to achieve similar performance as the GPU. Performance depends on additional hardware parameters like clock cycles, cache sizes, number of GPU cores, etc. so the values reported here are only indicative.

All timings include the overhead of converting the image to polar coordinates. The GPU time also includes the time to send the data to the GPU and to retrieve the result. Fig. 6(a) shows that for large problem sizes this overhead is negligible given that the lines of best fit have a y-intercept very close to 0. For small problem sizes, constant overhead can dominate, as shown in Fig. 6(b) where the same data from (a) was plotted on log–log axes. The data is segmented by $\Delta\phi_1$, which shows that on the CPU the overhead of converting images to polar coordinates is substantially slowed down as the polar image size is increased (smaller $\Delta\phi_1$); this trend is not apparent in the GPU implementation. For very small template libraries, the times for a single core and multiple cores converge, showing that the polar conversion is performed on a single CPU core. The constant overhead on the GPU can be attributed mainly to sending and receiving image data from the GPU, and the plot shows this is not so significant: only for very small problems with coarse $\Delta\phi_1$, and thus very small constant overhead, can the CPU outperform the GPU. Hence, the acceleration of the polar transform on the GPU also provides substantial benefit. For both CPU and GPU implementations, the constant overhead will be influenced by the size of the experimental images; in this case a 512 by 512 pixel image was tested.

The time necessary to index a dataset of multiple images can be roughly estimated by multiplying the value from Fig. 6 with the total number of images. For example, indexing the entire dataset of 8400 images from Fig. 4 with $N = 11476$, $S = 57$, and $\Delta\phi_1 = 1°$, took 5 min 9 s on the CPU (37 ms per pattern) and 1 min 33 s on the GPU (11 ms per pattern). This is close to the values predicted from the lines of best fit: 42 ms/pattern and 8 ms/pattern on the CPU and GPU respectively. Note that to achieve these times the entire dataset was loaded into RAM and all images were already pre-processed. Including the time to

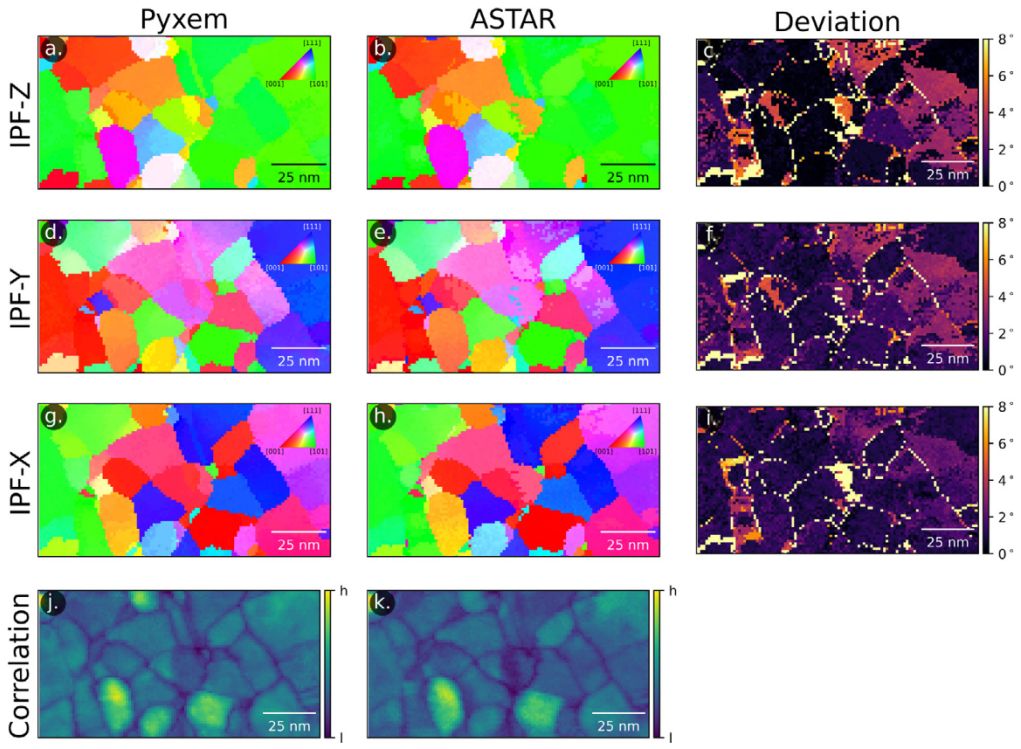


**Fig. 4.** Comparison of the indexation result for Pyxem and ASTAR. (a) and (b) represent the Pyxem and ASTAR IPF-Z maps respectively with (c) the angle between the vectors in each pixel of (a) and (b). Analogous maps are plotted for IPF-Y in (d-f) and IPF-X in (g-i). (j) and (k) show the correlation index maps for Pyxem and ASTAR. Due to differences in intensity range, these maps should not be compared on a pixel-by-pixel basis; only relative intensities within the same map are meaningful.

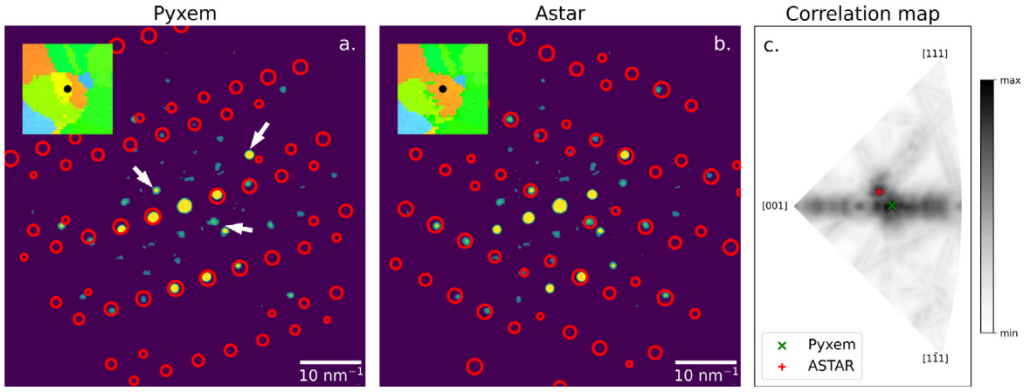


**Fig. 5.** Comparison of the indexation result in (a) Pyxem and (b) ASTAR for a single pattern where there is substantial disagreement. The inset in both images shows the location of the pattern on the IPF-Z map. (c) The correlation map for the pattern obtained with Pyxem, which represents the correlation of each template in the library with the image on the stereographic projection, shows that the Pyxem and ASTAR solutions are local maxima.

load data from disk and perform image preprocessing triples the time to 15 min on the CPU and 3 min on the GPU.

To assess the performance of the Pyxem implementation relative to other open source template matching codes, benchmarking results for indexing a single image in Problematic [20] and py4DSTEM [22] were also plotted in Fig. 4. There is only one point for Problematic, derived from the example provided by the developer, since $\Delta\phi_1$ is hard-coded and $N$ cannot be freely varied. For similar problem sizes, the 1-core CPU implementation in Pyxem is about 12 times faster compared to py4DSTEM and about 30% faster than Problematic. Problematic implements template matching in a similar way as Pyxem and the performance critical parts of the code are written in Cython so performance is good. py4DSTEM uses a very different template matching methodology, and likely trades performance for robustness and ease of use. In addition, since they use the FFT matching approach, performance likely does not depend on $S$, and relative performance may improve for more complex patterns. Nevertheless, compared to the tested open source alternatives, Pyxem is fast on a single core. In addition, it can take full advantage of additional cores or a GPU.

The performance of ASTAR is not shown on the figure since it was not available on the same hardware; only one license was available for an installation on a Windows 7 workstation from 2012 with 4 CPU cores and 8 GB of RAM. On this workstation, ASTAR was very performant for the problem sizes tested. A dataset of 52000 images could be indexed with a library of $N = 1081$ templates, $S = 16$, and $\Delta\phi_1 = 1°$ in 5 min 10 s on 3 CPU threads, which means on average 6 ms to index one image. This is faster than Pyxem-CPU with 16 cores for a comparable problem size. ASTAR was created at a time when available computational power was limited, hence using the available resource as efficiently as possible was likely a primary concern in the development. A number of design choices reflect the focus on maximizing performance, such as the fact that only 8-bit images can

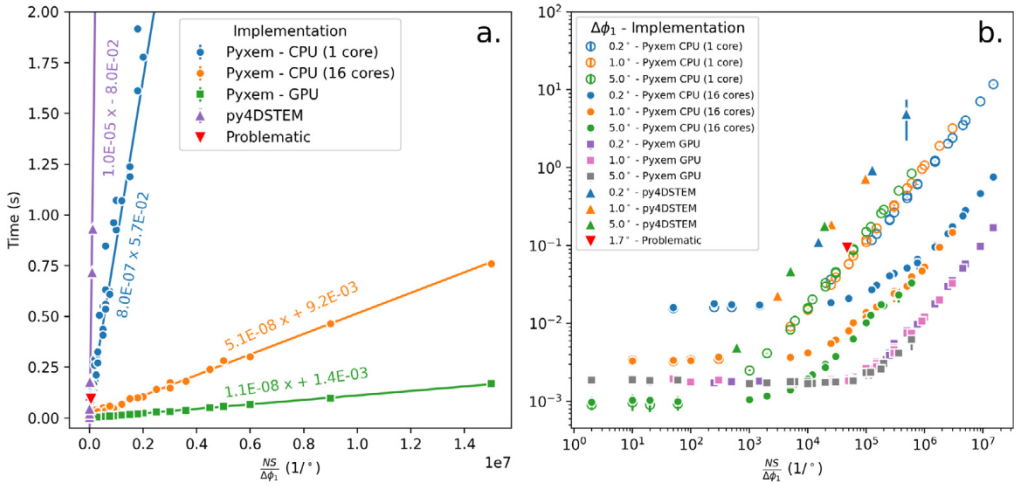


**Fig. 6.** Indexation time for a single diffraction pattern image for various combinations of in-plane angular increment $\Delta\phi_1$, number of templates in the library $N$, and number of spots per template $S$ on (a) linear axes and (b) log–log axes. In addition to the CPU and GPU implementations for Pyxem presented in this paper, benchmarking results from py4DSTEM [22] and Problematic [20] are also plotted. All benchmarks were performed on a desktop PC with an AMD Ryzen 16 core CPU and an NVIDIA RTX 3080 GPU. The CPU implementation was also timed on a single core.

be processed and template intensities are represented using single byte integers. However, this speed comes at the cost of reduced flexibility. It would also appear that ASTAR makes use of additional optimizations under the hood, since the authors' benchmarks showed that indexation times were not a linear function of the number of templates in the library.

The performance of Pyxem was also measured on different hardware, the details of which are given in the supplementary materials.

### 4.3. Custom indexation procedures

Thanks to the flexibility of Pyxem, customized workflows can easily be constructed. This section illustrates a two-stage indexing procedure for identifying orientations of overlapping phases, specifically for the case of nano-sized precipitates embedded in a crystalline matrix. Since the reflections of the matrix phase always dominate over the signal of the precipitates, a reliable indexation of the precipitates requires a subtraction of the matrix signal from the images. Strategies to achieve this are described in Refs. [28,35] and the tools were successfully applied to reveal orientations of nanoprecipitates in irradiated steel [36]. The technique consists of three stages: first the matrix is indexed, then the dominant matrix contribution in the experimental patterns is masked, and finally the precipitate fraction is indexed on the masked dataset. This is illustrated for one pattern in the irradiated steel dataset from Ref. [36] in Fig. 7(a), (b) and (c). The tools that were used to analyze this data in Ref. [36] are proprietary, and as of yet not widely accessible.

Here, a similar, fully reproducible workflow was constructed using Pyxem to analyze the same dataset. The complete data processing pipeline is provided in a Jupyter notebook accessible through a public git repository [33]; the dataset is accessible on Zenodo [37].

As with the Cu-Ag dataset, the patterns were centered and background-subtracted. To correct for astigmatism in the projector system an affine transformation was applied to the images, the parameters of which were found by comparing the austenite matrix reflections to the ⟨110⟩ template. This method was reliable, since after this transformation was applied, the correspondence between the G-phase signal and templates was also improved. However, it is recommended to derive image distortion parameters from the diffraction pattern rings in a sample like Au nanoparticles on a carbon support, especially if no good reference phase is available. Such a calibration was not performed prior to the collection of this dataset.

The patterns were subsequently indexed using a library of about 1000 templates (maximum deviation of 1° between templates) from FCC Fe. The result of the indexation is shown in Fig. 7(d) as an IPF-Z map, showing that the data was collected in a single grain close to a ⟨110⟩ zone. Fig. 7(e) and (f) show the result from Ref. [36] and its angular deviation from the Pyxem result respectively. Deviations up to about 4° can be observed.

To remove the intense matrix reflections, a mask was created by calculating the average of all the diffraction patterns and thresholding the resulting image. A number of erosion and dilation operations were performed to remove small noise-like features in the mask, and to ensure the removal of tails from intense diffraction peaks. The final mask that was applied to the images is shown in Fig. 7(b).

The masked dataset was subsequently indexed using a library of about 4000 templates (maximum deviation of 0.5° between templates) from $Mn_6Ni_{16}Si_7$ phase (G-phase). Due to the larger lattice parameter of the G-phase compared to the matrix phase (austenite), templates contain many more reflections and small changes in orientation can drastically change the patterns. For this reason, a larger template library was simulated using smaller orientation increments. Additionally, a very small relrod width was chosen, such that only the reflections very close to the Ewald sphere would be included.

The parameters of the indexation procedure were optimized iteratively on individual patterns containing signal from the precipitates. In order to meet the requirements discussed in Section 2.2 two processing steps had to be introduced: firstly, the intensity of all diffraction spots in the template library were set to unity (ignoring both the structure factor and relrod shape factor), and secondly a small constant (about 10% of the maximum pixel intensity) was subtracted from all pixels in the experimental images. Without these steps most patterns were erroneously indexed, because the templates contained a few intense reflections which dominated in the evaluation of $Q$ over many very weak reflections. The many weak reflections that matched with the background were also not penalized. After the gamma correction and masking of the matrix phase, the intensities of the already weak G-phase reflections in the experimental images were almost equal. Hence, by giving each reflection in the templates equal weight in the calculation of $Q$ and by setting the background to a small negative value, the optimal template is one with the highest number of matching spots and the fewest spots corresponding to background. Due to the large lattice parameter of the G-phase, reflections are closely spaced and most patterns contain reflections from higher order Laue zones that show the curvature of the Ewald sphere (see Fig. 7(c)). These features are

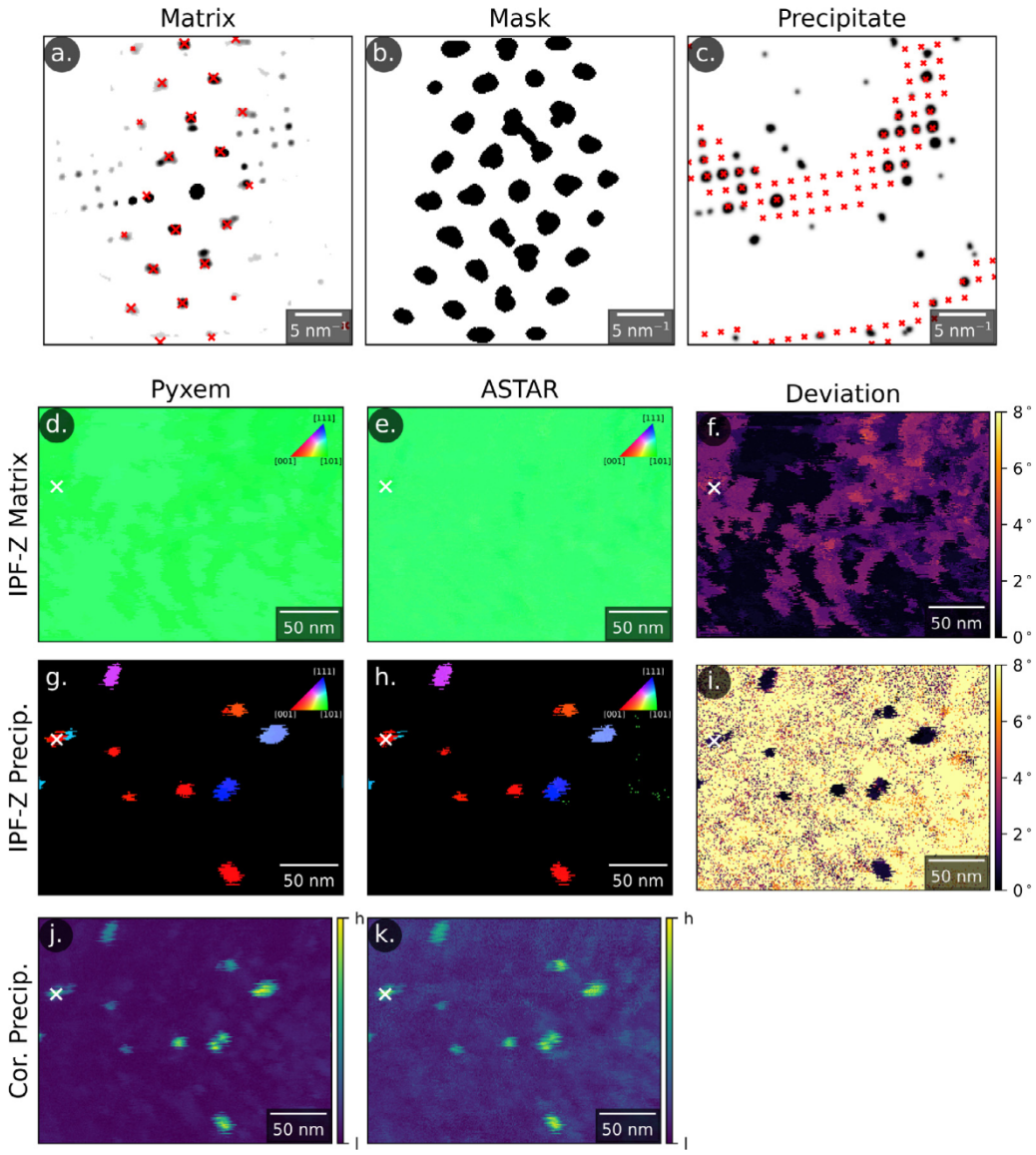


**Fig. 7.** Illustration of the two-stage indexation procedure. The dataset is first indexed with the matrix phase as shown in (a). (b) A mask is created to index the templates by averaging all the images and thresholding. (c) The masked dataset is indexed using the template library from the precipitates. (d), (e), and (f) show the IPF-Z maps obtained from indexing the matrix phase with Pyxem, ASTAR, and the angular deviation between them, respectively. (g) (h) and (i) show analogous maps for the precipitates (low correlation index pixels were set to black). (j) and (k) are the Pyxem and ASTAR correlation index maps for the precipitate indexation.

very sensitive to small changes in orientation, so their intensity is not relevant.

Fig. 7(g) shows the indexed result from the masked dataset as an IPF-Z map; pixels with low correlation index were set to black to reveal the precipitates. The correlation index map is shown in Fig. 7(j). The orientation and correlation maps from Ref. [36] are shown in Fig. 7(h) and Fig. 7(k) respectively. The angular deviation between the Pyxem and proprietary solutions is given in Fig. 7(i). Within the particles, deviation between the solutions is less than 1°, outside the particles the indexation is arbitrary since there is only noise in these patterns. Because the indexation only relies on the presence of spots and not their intensity, and because all templates in the library are very different from each other due to the small relrod width, the orientation of the precipitates is determined with very high precision. In the template library of the matrix phase, many templates are very similar and only differ in spot intensity, which results in a lower precision in the indexation. A smaller camera length, such that spots from higher order Laue zones would also be captured in the image, may improve the indexing precision of the matrix.

Alternative approaches to indexing the dataset were also attempted. For example, instead of using a single mask, the best matching template from each pattern in the first indexation step can be converted to a mask by placing circles at the coordinates of each reflection. For this particular dataset this proved to be an inferior strategy because there were features in the diffraction pattern that did not correspond to either of the phases, instead originating from defects or surface contamination and oxidation. These features were not masked with the template mask approach and subsequently interfered with the indexation of the precipitates. However, this type of approach may be necessary for datasets of polycrystalline materials containing multiple phases, since the required mask is different in every grain. Finally, it may also be possible to separate contributions from different phases in the diffraction patterns using unsupervised learning techniques, such as principal component analysis (PCA) and non-negative matrix factorization (NMF) [38]. These techniques are also implemented in Pyxem and may serve as a helpful preprocessing step before template matching.

## 5. Discussion

This paper demonstrated the template matching capabilities in Pyxem for extracting orientation information from NBED and PED datasets. Two use cases were provided as examples, and compared to the commercial solution ASTAR. Similar results could be obtained; deviations between the solutions could be attributed to ambiguity in the diffraction patterns from overlapping crystals or limited differentiability among templates in the library. Under these circumstances, small differences in image pre-processing or library simulation conditions could have an outsized effect on the best matching template for ambiguous patterns. With the integrated but flexible workflow offered by Pyxem, it is relatively easy to investigate the effect of these pre-processing parameters and their relative importance to the result. From this analysis, the maximum precision of the indexation result can be estimated. While 1° is the often quoted maximum precision of NBED/PED orientation mapping [4,39], as was shown in this paper the precision can vary strongly depending on experimental parameters such as camera length, local orientation and lattice parameters of the phases involved. Conversely, this paper also showed that differences in spot intensity are unreliable for indexation, so accurate simulations of spot intensities or pixel depth do not have a big influence.

In order to optimize data acquisition conditions for obtaining the highest precision in orientation, a quantitative relation to the various experimental parameters would be desirable. The highest uncertainty in determining the orientation is in determining the correct beam direction and associated angles $\Phi$ and $\phi_2$; $\phi_1$ can be determined to nearly arbitrary precision. Here beam direction should be interpreted as the crystal's real space vector that is parallel to the electron beam. To estimate the uncertainty in the beam direction, as a first order approximation the rel-rods that correspond to reflections in the diffraction pattern can be imagined as pinning points that constrain the surface of the Ewald sphere. Since the reflection is present in the pattern, the Ewald sphere must intersect the rel-rod. If it is assumed that each rel-rod has the same length equal to twice the maximum excitation error $s_{max}$, and if it is assumed the reflection intensity is not significant, then the Ewald sphere is most constrained by the reflections furthest away from the reciprocal space origin. In this approximation, all beam directions that correspond to an Ewald sphere that intersects with the rel-rods of the most distant diffraction spots are valid beam directions. In one dimension and for one reflection, the uncertainty interval for the beam direction angle can be expressed as

$$\pm \Delta\theta_{max} = \arctan\left(\frac{s_{max}}{|g|_{max}}\right) \tag{4}$$

where $|g|_{max}$ is the length of the reciprocal lattice vector corresponding to the most distant reflection. The beam direction has two degrees of freedom, so the uncertainty in the beam direction can be represented as a cone with an irregularly shaped base. In directions with fewer spots, the uncertainty in orientation is higher, meaning the cone of uncertainty is elongated in that direction. For example, in the matrix diffraction pattern in Fig. 2(a) there are few reflections along the $(020)-(0\bar{2}0)$ row of reflections. This means there is a higher uncertainty in the orientation component determined by the rotation around $(20\bar{2})$. The effect can be seen in Fig. 2(f): the patch of beam directions with high correlation is extended in the direction from [101] to [111], which is the path that is traced by the beam when rotating away from [101] over $(20\bar{2})$. This also illustrates that correlograms like Fig. 2(f) offer a practical way to visualize the cone of uncertainty, and that optimizing image processing and template simulation parameters should aim to make this peak as sharp as possible to minimize uncertainty in the orientation.

For the example in Fig. 2(a), using $|g|_{max} = 0.55$ Å$^{-1}$ for $g = (020)$ and $s_{max} = 0.1$ Å$^{-1}$ as was used in the template simulation library Eq. (4) predicts an uncertainty of about ±10°, which roughly agrees with the longest dimension of the bright peak in Fig. 2(f); the value is approximately twice as large as the maximum deviation from the ASTAR results in Fig. 7(f).

For the Cu-Ag dataset, the camera length was shorter and $|g|_{max} = 2.6$ Å$^{-}1$ and $s_{max} = 0.1$ Å$^{-}1$, which results in an uncertainty of ±2°. This analysis shows that in order to maximize angular resolution, the camera length should be chosen such that the most distant visible reflections are situated at the edge of the detector. The visibility of diffraction spots depends on the atomic scattering factor, which is approximately inversely proportional to beam energy. Hence a lower beam energy should increase $|g|_{max}$. A lower beam energy also increases the Ewald sphere curvature, making it easier to capture higher order Laue zone reflections in the image. Resolution also benefits from more sensitive detectors that can capture weaker reflections at higher scattering angles, as was shown in Ref. [3].

Eq. (4) can also help to estimate the necessary angular sampling for the template library. The maximum excitation error is generally unknown, and is typically estimated from the sample thickness. However, for generating template libraries, a much larger value is usually selected to include more diffraction spots in the templates [26]. An upper bound is half the distance between consecutive planes that make up the reciprocal space grid. The most widely spaced planes in reciprocal space lie at a distance $1/a$ from each other, with $a$ the lattice parameter, which would imply $s_{max} = \frac{1}{2a}$. For non-cubic crystal systems, $V^{1/3}$ could be used as a substitute for $a$ with $V$ the unit cell volume. This upper bound implies there is no gap between reflections from different Laue zones; if $s_{max}$ is larger, reflections from higher order Laue zones will overlap in the patterns. A better estimate of $s_{max}$ can be achieved by calibrating it based on the width of the empty bands between consecutive Laue zones. For the Cu and austenite template libraries, the upper boundary estimate of 0.14 Å$^{-1}$ is not far off from the value of 0.1 Å$^{-1}$ that was used to generate the template libraries. Using a smaller value for $s_{max}$ in these simple crystals usually makes the patterns very sparse with few reflections and this makes it difficult to converge to a good solution. Hence for these small crystals, a better resolution than about 2° will not be achieved and this resolution is sufficient for generating the template library. For the G-phase ($a = 11.3$ Å) on the other hand, $s_{max}$ should not exceed 0.04 Å$^{-1}$. For this phase, higher order Laue zones were visible in some of the diffraction patterns and 0.01 Å$^{-1}$ was found to match the experiments well. This yields a maximum resolution of ±0.3°, therefore a template library of approximately this resolution was used to index the G-phase. These ideas could be validated and refined by employing template matching on datasets where the orientation of the crystal is known a priori, for example by simulating the diffraction patterns using the multi-slice technique implemented in packages like abTEM [40].

Different requirements in template library resolution and differences in template complexity also influence performance. The relations above and the fact that indexing a pattern has time complexity $O\left(\frac{NS}{\Delta\phi_1}\right)$ can be used to make an educated guess about performance based on the phases present in the material. The number of templates $N$ is roughly proportional to $\frac{1}{(\Delta\theta)^2}$ with $\Delta\theta$ the spacing between templates. Through Eq. (4) and $s_{max} \propto a^{-1}$, it follows that $N \propto a^2$, i.e. a larger unit cell requires a much larger template library. In addition, the number of spots in a template $S$ is proportional to $a^2$. $\Delta\phi_1$ is not dependent on the crystal structure. Hence the time to index patterns scales roughly with $a^4$ or $V^{\frac{4}{3}}$. Actual performance will depend on the Laue groups of the phase, the chosen camera lengths, $\Delta\phi_1$, and the available hardware.

Workflow flexibility, openness, speed, and reproducibility are becoming increasingly important in light of the growing popularity of fast pixelated detectors which are producing ever more complex datasets. Due to the tight integration with HyperSpy, file formats do not impose any barriers for analyzing data with Pyxem. The open source nature of the code means that it can be freely adapted and extended, for example towards real-time indexing. A single diffraction pattern can be reliably indexed within a few to tens of milliseconds, which is on the same order as the acquisition time for a pattern on a CMOS or CCD based

detector. Hence these processes could easily run in parallel meaning that routine orientation mapping of single phase materials with NBED could become automated like electron backscatter diffraction (EBSD) in the scanning electron microscope. A prototype along these lines is being worked out by integrating with the LiberTEM library [41,42]. For the extremely fast direct electron detectors, such as the EMPAD detector, the code would not be fast enough to perform real time indexing. Multiple optimizations could still be explored, like improving the algorithm for optimizing the in-plane angle $\phi_1$ and parallelizing the image preprocessing steps; the authors welcome all contributions that improve the current implementation.

## 6. Conclusion

In this paper, the new template matching capabilities implemented in Pyxem for analyzing NBED and precession diffraction data were discussed, from implementation, through performance, to applications. The code enabled reliable and fast orientation mapping in nanostructured materials on high quality datasets originating from a pixelated detector. Through the fine grained control over the entire analysis process the effects of image processing and template simulation parameters on the result could be evaluated. In addition, a two stage indexation procedure was implemented for mapping the orientation of nano-sized precipitates embedded in a crystalline matrix, a workflow that could be adapted by many other researchers. All the workflows in this paper are fully documented and reproducible and can serve as templates for other open, custom and transparent analysis workflows.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

We acknowledge funding through BiGmax (https://www.bigmax.mpg.de/), the Max Planck research network on big-data-driven materials science. We also want to sincerely thank all the contributors to HyperSpy and Pyxem, who have dedicated their time to building free analysis libraries that will benefit all future researchers and lay the foundation for open science in electron microscopy. In particular we wish to acknowledge the original creator of the Pyxem library D. Johnstone, whose pioneering work this work built on. P.C. acknowledges the support of his studentship from the EPSRC. H.W.Å. acknowledges NTNU for financial support through the NTNU Aluminium Product Innovation Centre.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ultramic.2022.113517.

## References

[1] E. Rauch, M. Véron, J. Portillo, D. Bultreys, Y. Maniette, S. Nicolopoulos, Microsc. Anal. 22 (6) (2008) S5–S8.
[2] E.F. Rauch, J. Portillo, S. Nicolopoulos, D. Bultreys, S. Rouvimov, P. Moeck, Z. Kristallogr. 225 (2–3) (2010) 103–109, http://dx.doi.org/10.1524/zkri.2010.1205.
[3] J. Jeong, N. Cautaerts, G. Dehm, C.H. Liebscher, Microsc. Microanal. 27 (5) (2021) 1102–1112, http://dx.doi.org/10.1017/S1431927621012538.
[4] S. Zaefferer, Cryst. Res. Technol. 46 (6) (2011) 607–628, http://dx.doi.org/10.1002/crat.201100125.
[5] D. Viladot, M. Véron, M. Gemmi, F. Peiró, J. Portillo, S. Estradé, J. Mendoza, N. Llorca-Isern, S. Nicolopoulos, J. Microsc. 252 (1) (2013) 23–34.
[6] F. Mompiou, M. Legros, Scr. Mater. 99 (2015) 5–8.
[7] J. Brons, G. Thompson, Jom 66 (1) (2014) 165–170.
[8] A. Kobler, A. Kashiwar, H. Hahn, C. Kübel, Ultramicroscopy 128 (2013) 68–81.
[9] C. Ophus, Microsc. Microanal. 25 (3) (2019) 563–582.
[10] N. Cautaerts, E.F. Rauch, J. Jeong, G. Dehm, C.H. Liebscher, Scr. Mater. 201 (2021) 113930.
[11] S.K. Lam, A. Pitrou, S. Seibert, Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 2015, pp. 1–6.
[12] M. Rocklin, Proceedings of the 14th Python in Science Conference, 126, Citeseer, 2015.
[13] R. Nishino, S.H.C. Loomis, 31st Conference on Neural Information Processing Systems, 2017, p. 151.
[14] D.N. Johnstone, P. Crout, M. Nord, J. Laulainen, S. Høgås, E. Opheim, B. Martineau, T. Bergh, S. Smeets, C. Francis, E. Prestat, A. Ross, S. Collins, I. Hjorth, Mohsen, D. Jannis, T. Furnival, E. Jacobsen, A. Herzing, H.W. Ånes, J. Morzy, T. Doherty, A. Iqbal, T. Ostasevicius, M. von Lany, R. Tovey, T. Poon, pyxem/pyxem: pyxem 0.12.1, Zenodo, 2020, http://dx.doi.org/10.5281/zenodo.3976823.
[15] F. de la Peña, T. Ostasevicius, V.T. Fauske, P. Burdet, P. Jokubauskas, M. Nord, M. Sarahan, E. Prestat, D.N. Johnstone, J. Taillon, et al., Microsc. Microanal. 23 (S1) (2017) 214–215.
[16] T. Kluyver, B. Ragan-Kelley, F. Pérez, B.E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J.B. Hamrick, J. Grout, S. Corlay, et al., Jupyter notebooks-a publishing format for reproducible computational workflows, vol. 2016, 2016.
[17] P. Crout, D.N. Johnstone, S. Høgås, B. Martineau, isabelwood100, J. Laulainen, H.W. Ånes, N. Cautaerts, S. Collins, S. Smeets, E. Jacobsen, J. Morzy, E. Prestat, phillipcrout, T. Doherty, AgBorrelli, T. Ostasevicius, R. Tovey, EirikOpheim, T. Bergh, pyxem/diffsims: diffsims 0.4.2, Zenodo, 2021, http://dx.doi.org/10.5281/zenodo.4697299.
[18] D.N. Johnstone, B.H. Martineau, P. Crout, P.A. Midgley, A.S. Eggeman, J. Appl. Crystallogr. 53 (5) (2020) 1293–1298.
[19] H.W. Ånes, P. Crout, P. Harrison, D.N. Johnstone, N. Cautaerts, S. Høgås, pyxem/orix: orix 0.8.1, Zenodo, 2022, http://dx.doi.org/10.5281/zenodo.6078059.
[20] S. Smeets, X. Zou, W. Wan, J. Appl. Crystallogr. 51 (5) (2018) 1262–1273.
[21] R. Bücker, P. Hogan-Lamarre, R. Miller, Front. Mole. Biosci. 8 (2021) 415.
[22] C. Ophus, S.E. Zeltmann, A. Bruefach, A. Rakowski, B.H. Savitzky, A.M. Minor, M.C. Scott, Microsc. Microanal. (2021) 1–14.
[23] H.J. Bunge, Mathematische Methoden Der Texturanalyse: Mit 86 Abbildungen Und 31 Tabellen, Sowie 7 Abbildungen Und 9 Tabellen Im Anhang, Akademie-Verlag, 1969.
[24] E.J. Kirkland, Advanced Computing in Electron Microscopy, Springer, 1998.
[25] I. Lobato, D. Van Dyck, Acta Crystallogr. Sect. A: Found. Adv. 70 (6) (2014) 636–649.
[26] E. Rauch, L. Dupuy, Arch. Metall. Mater. 50 (2005) 87–99.
[27] L. Palatinus, P. Brázda, M. Jelínek, J. Hrdá, G. Steciuk, M. Klementová, Acta Crystallogr. Sect. B: Struct. Sci. Cryst. Eng. Mater. 75 (4) (2019) 512–522.
[28] E. Rauch, M. Véron, Microsc. Microanal. 25 (S2) (2019) 1922–1923, http://dx.doi.org/10.1017/s1431927619010341.
[29] G. Tzimiropoulos, V. Argyriou, S. Zafeiriou, T. Stathaki, IEEE Trans. Pattern Anal. Mach. Intell. 32 (10) (2010) 1899–1906.
[30] G. Wu, S. Zaefferer, Ultramicroscopy 109 (11) (2009) 1317–1325.
[31] T. Oellers, V.G. Arigela, C. Kirchlechner, G. Dehm, A. Ludwig, ACS Combinatorial Sci. 22 (3) (2020) 142–149.
[32] N. Cautaerts, P. Harrison, din14970/TVIPSconverter: tvipsconverter v0.1.3, Zenodo, 2020, http://dx.doi.org/10.5281/zenodo.4288857.
[33] N. Cautaerts, din14970/pyxem_template_matching_workflows: Version submission, Zenodo, 2021, http://dx.doi.org/10.5281/zenodo.5701051.
[34] J. Jeong, N. Cautaerts, C. Liebscher, G. Dehm, Precession electron diffraction dataset from nanocrystaline Cu-Ag (FCC) alloy collected on pixelated TVIPS detector, Zenodo, 2021, http://dx.doi.org/10.5281/zenodo.5595292.
[35] A. Valery, E.F. Rauch, L. Clément, F. Lorut, J. Microsc. 268 (2) (2017) 208–218, http://dx.doi.org/10.1111/jmi.12599.
[36] N. Cautaerts, R. Delville, E. Stergar, J. Pakarinen, M. Verwerft, Y. Yang, C. Hofer, R. Schnitzer, S. Lamm, P. Felfer, D. Schryvers, Acta Mater. 197 (2020) 184–197, http://dx.doi.org/10.1016/j.actamat.2020.07.022.
[37] N. Cautaerts, J. Jeong, C. Liebscher, G. Dehm, Nanobeam electron diffraction dataset from ion irradiated DIN 1.4970 austenitic stainless steel with G-phase precipitates collected on pixelated TVIPS detector, Zenodo, 2021, http://dx.doi.org/10.5281/zenodo.5597738.
[38] B.H. Martineau, D.N. Johnstone, A.T. van Helvoort, P.A. Midgley, A.S. Eggeman, Adv. Struct. Chem. Imaging 5 (1) (2019) 1–14.
[39] A. Morawiec, E. Bouzy, H. Paul, J.-J. Fundenberger, Ultramicroscopy 136 (2014) 107–118.
[40] J. Madsen, T. Susi, Microsc. Microanal. 26 (S2) (2020) 448–450.
[41] A. Clausen, D. Weber, K. Ruzaeva, V. Migunov, A. Baburajan, A. Bahuleyan, J. Caron, R. Chandra, S. Halder, M. Nord, et al., J. Open Source Softw. 5 (50) (2020) 2006.
[42] A. Clausen, D. Weber, K. Ruzaeva, V. Migunov, A. Baburajan, A. Bahuleyan, J. Caron, R. Chandra, S. Dey, S. Halder, B.D. Levin, M. Nord, C. Ophus, S. Peter, J. Schyndel van, J. Shin, S. Sunku, K. Müller-Caspary, R.E. Dunin-Borkowski, LiberTEM/LiberTEM: 0.5.1, Zenodo, 2020, http://dx.doi.org/10.5281/zenodo.3982290.