Universiteit
Leiden
The Netherlands

Superlattices in van der Waals materials: a low-energy electron microscopy study

Jong, T.A. de

Citation

Jong, T. A. de. (2022, November 3). *Superlattices in van der Waals materials: a low-energy electron microscopy study*. *Casimir PhD Series*. Retrieved from https://hdl.handle.net/1887/3485753

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3485753

Note: To cite this publication please use the final published version (if applicable).

A

GEOMETRIC PHASE ANALYSIS

This appendix provides a more elaborate description of Geometric Phase Analysis / Spatial lock-in and adaptive GPA.

A.1 DEFORMATIONS OF A LATTICE

We perform lock-in measurements on images that clearly display a periodic lattice. In STM, this implies we can use any topography of sufficient quality that displays the crystal lattice. The idea is to use a lock-in measurement in order to find a transformation of coordinates between the measured, "distorted" image and its pristine, undeformed equivalent (in this work, a perfect triangular lattice). Defining the measured and pristine image as $T_m(\mathbf{r})$, $T_r(\mathbf{r}')$ respectively, both with measurement coordinates $\mathbf{r} = (x, y) \in \mathbb{R}^2$ and lattice coordinates $\mathbf{r}' = (x', y') \in \mathbb{R}^2$, the following relation holds:

$$T_m(\mathbf{r}) = T_r\left(\mathbf{r} + \mathbf{u}(\mathbf{r})\right) = T_r\left(\mathbf{f}(\mathbf{r})\right) = T_r(\mathbf{r}') = T_m\left(\mathbf{f}^{-1}(\mathbf{r}')\right)$$

where the transformation from measurement coordinates to lattice coordinates is given by:

$$\mathbf{f}(\mathbf{r}) = \mathbf{r} + \mathbf{u}(\mathbf{r}) = \mathbf{r}' \tag{A.1}$$

Here, $\mathbf{u}(\mathbf{r})$ is called the displacement field, connecting the measurement coordinates to the lattice coordinates, as is well-established in continuum mechanics. For convenience, we also define the inverse displacement:

$$\mathbf{u}'(\mathbf{r}') := \mathbf{f}^{-1}(\mathbf{r}') - \mathbf{r}' = \mathbf{r} - \mathbf{r}'$$

Note that by substitution, we have the following relation between forward and inverse displacement:

$$\mathbf{u}'(\mathbf{r}') = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{r})) - (\mathbf{r} + \mathbf{u}(\mathbf{r})) = -\mathbf{u}(\mathbf{r})$$

Parts of this appendix have been published in the Supplementary materials of Phys. Rev. Research **3**, 013153 [141] and Nat. Commun. **13**, 70 (2022) [135]. Tjerk Benschop made a significant contribution to this appendix.

With this, we can express the pristine image at lattice coordinates in terms of the measured image:

$$\begin{aligned} T_r(\mathbf{r}') &= T_m\left(\mathbf{f}^{-1}(\mathbf{r}')\right) = T_m(\mathbf{r}' + \mathbf{u}'(\mathbf{r}')) \\ &= T_m\left(\mathbf{r}' - \mathbf{u}(\mathbf{r})\right) \\ &= T_m\left(\mathbf{r}' - \mathbf{u}(\mathbf{r}' - \mathbf{u}(\mathbf{r}))\right) \\ &\approx T_m\left(\mathbf{r}' - (\mathbf{u}(\mathbf{r}') - (\nabla\mathbf{u})(\mathbf{r}' - \mathbf{r}))\right) \\ &= T_m\left(\mathbf{r}' - (\mathbf{u}(\mathbf{r}') + (\nabla\mathbf{u})\mathbf{u}(\mathbf{r}))\right) \\ &= T_m\left(\mathbf{r}' - \mathbf{u}(\mathbf{r}') + (\nabla\mathbf{u})\mathbf{u}'(\mathbf{r}'))\right) \end{aligned}$$

Therefore, if we can determine $\mathbf{u}(\mathbf{r})$, and thereby $\mathbf{u}'(\mathbf{r}')$, we can reconstruct the pristine image. This is the idea of the Lawler–Fujita reconstruction algorithm [143]. In their original paper, Lawler–Fujita uses $\mathbf{u}'(\mathbf{r}') = -\mathbf{u}(\mathbf{r}')$, which is a good approximation if $\mathbf{u}$ varies slowly.

A.2 PROPERTIES OF THE DEFORMATION

The displacement field $\mathbf{u}(\mathbf{r})$ as defined above, fully describes the deformation of the lattice, but does not directly provide insight into the relevant properties. To that end, we first define the Jacobian of the transformation $\mathbf{f}$:

$$J \equiv \nabla\mathbf{f} = \mathbb{1} + \nabla\mathbf{u}$$

, where $\nabla\mathbf{u}$ is the Jacobian of the displacement field, in continuum mechanics terminology the deformation gradient tensor, and in canonical terms defined as follows:

$$\nabla\mathbf{u} = \begin{pmatrix} \frac{du_x}{dx} & \frac{du_x}{dy} \\ \frac{du_y}{dx} & \frac{du_y}{dy} \end{pmatrix}$$

In order to fully characterise the deformation of the lattice, we decompose J in its polar form:

$$J = WP = WV^\top DV\,, \tag{A.2}$$

where W is the rotation matrix corresponding to the rotation of the full lattice and the matrix P describes the local anisotropy and scaling. P is further decomposed in the rotation matrix V indicating the orientation of the axis of anisotropy (i.e. the axis of largest scaling, with the axis of smallest scaling perpendicular to it) and the diagonal scaling matrix $D = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix}$, where by convention and implementation $d_1 \geq d_2$ holds for any position $\mathbf{r}$.

The geometric mean of these directional scaling factors is equal to the square root of the determinant of D and therefore of J: $\sqrt{d_1 d_2} = \sqrt{\det(J)}$. As this corresponds to the local scaling of the wavelength of the moiré lattice, we can use this to quantify the local twist angle:

$$\lambda(\mathbf{r}) = \sqrt{d_1 d_2}\frac{4\pi}{\sqrt{3}|\mathbf{q}_j|} \tag{A.3}$$

Where $|\mathbf{q}_j|$ is the length of the chosen reference vectors. This local wavelength is then converted to a local twist angle using the well-known expression:

$$\theta(\mathbf{r}) = 2\arcsin\left(\frac{2\lambda(\mathbf{r})}{a}\right)$$

, where $a = 2.46$Å is the lattice constant of graphene and $\theta(\mathbf{r})$ the local twist angle.

A quantification of the local anisotropy is given by the ratio $\kappa = d_1/d_2$ and the angle between the anisotropy axis and the x-axis is finally calculated from V: $\psi = \arctan\left(\frac{V_{xy}}{V_{xx}}\right)$.

In our practical implementation, the singular value decomposition (SVD) is used to obtain the decomposition in equation A.2 for each point in the image, and numpy's `atan2` is used to find the right quadrant of the angles from the signs of V_{xx} and V_{xy}.

A.3 DETERMINATION OF THE DISPLACEMENT FIELD $\mathbf{u}(\mathbf{r})$

In order to determine $\mathbf{u}(\mathbf{r})$ for a certain image, we perform a lock-in measurement. To clarify, we can represent any (nearly) periodic image as:

$$T_m(\mathbf{r}) = T_0\sum_j e^{i\mathbf{q}_j\cdot(\mathbf{r}+\mathbf{u}(\mathbf{r}))} = T_0\sum_j e^{i(\mathbf{q}_j\cdot\mathbf{r}+\phi_j)} \tag{A.4}$$

where $\phi_j = \mathbf{q}_j\cdot\mathbf{u}(\mathbf{r})$ is the position-dependent phase of the lattice. The summation runs over the reciprocal lattice vectors $\mathbf{q}_j$ ($j \in \{1,2,3\}$ for a hexagonal lattice), T_0 is the constant indicating the amplitude of the modulation and $\mathbf{u}(\mathbf{r})$ is again the displacement field.

The phase is measured using standard lock-in procedure: The existing image is mixed with a reference image containing a specific plane wave. If we choose the periodicity of this reference wave sufficiently close to that of the lattice in the image itself, we can then low-pass filter the mixed image and end up with a phase map for a specific wave. For clarification:

$$\cos(\mathbf{q}_j\cdot\mathbf{r}+\phi_j)e^{-iq_j\cdot\mathbf{r}} = \frac{e^{i\phi_j}}{2}\left(1+e^{-2i(\mathbf{q}_j\cdot\mathbf{r}+\phi_j)}\right) \mapsto \frac{1}{2}e^{i\phi_j}$$

where the cosine in the first term denotes the (real-valued) measured image, whereas the complex exponential denotes the reference wave and $\mapsto$ denotes low-pass filtering in order to get rid of the last term between brackets, corresponding to a rotating wave approximation. Alternatively, for a gaussian low-pass filter, this corresponds to a real space gaussian integration window of the lock-in.

By taking the (pointwise) angle of the complex, filtered product image, we end up with the phase map. In particular, this phase map contains information about the displacements of each pixel in the measured image $T_m(\mathbf{r})$ with respect to the pristine reference lattice $T_r(\mathbf{r})$ along the wave vector $\mathbf{q}_j$ used for the lock-in procedure. This procedure is repeated for at least one additional reciprocal lattice vector. The two phase maps are then used to find the displacement field $\mathbf{u}(\mathbf{r})$. From the definition of $\mathbf{u}(\mathbf{r})$ (eq. A.1), the following holds: $\mathbf{r}' = \mathbf{r}+\mathbf{u}(\mathbf{r})$. Multiplying this equation by the reciprocal lattice

vectors, we get a system of equations expressing the projection of the distortion onto the reciprocal lattice vectors:

$$\mathbf{q}_j \cdot \mathbf{r}' = \mathbf{q}_j \cdot \mathbf{r} + \phi_j,\ j \in 1,2,3$$

Selecting only $j \in \{1,2\}$, we have in matrix notation:

$$Q = \begin{pmatrix} -\mathbf{q}_1 - \\ -\mathbf{q}_2 - \end{pmatrix} = \begin{pmatrix} q_{1x} & q_{1y} \\ q_{2x} & q_{2y} \end{pmatrix}$$

such that we can write for $\phi = \binom{\phi_1}{\phi_2}$:

$$Q\mathbf{r}' = Q\mathbf{r} + \phi\,. \tag{A.5}$$

Multiplying by Q^{-1}, we find $\mathbf{r}' = \mathbf{r} + Q^{-1}\phi$, and therefore $\mathbf{u}(\mathbf{r}) = Q^{-1}\phi(\mathbf{r})$.

A.4 ADDITIONAL NOTES ON CHOICE OF REFERENCE VECTORS

SELECTING TWO REFERENCE VECTORS

To obtain $\mathbf{u}(\mathbf{r})$ as described above, we only used the phase of the lock-in signal of two reference vectors. For a triangular/hexagonal lattice, *a priori* three possible choices of which two reference vectors to use are possible from the three linear independent references vectors as fitted to the FFT of the image. To select which two vectors to use for the reconstruction of $\mathbf{u}(\mathbf{r})$, we either selected the ones with the largest average lock-in amplitude, or by inspecting the phase-unwrapped images and selecting the ones where no remaining phase slips occurred.

USING MORE THAN TWO REFERENCE VECTORS

In principle, information is lost when only selecting the phase of the lock-in signal of two reference vectors to obtain $\mathbf{u}(\mathbf{r})$. In low signal-to-noise ratio situations, it could be beneficial to use all the information. Equation A.5 also holds for more than two phases and reference vectors. Although Q is not a square matrix in this case, a solution can be obtained for each pixel using linear least squares minimization of the following equivalent equation:

$$Q\mathbf{u}(\mathbf{r}) = \phi(\mathbf{r})$$

Where additionally the amplitude of the lock-in signals can be used as weights to the minimization problem.

ISOTROPY

Enforcing the reference lattice to be isotropic can be done either in advance, by enforcing isotropic reference wavevectors (as applied in this work) or alternatively, after the initial lock-in step, by adding an additional linear phase $\Delta\phi_j = \Delta\mathbf{q}_j \cdot \mathbf{r}$ to the obtained phase, where $\Delta\mathbf{q}_j$ is the difference between the used reference wavevector and the isotropic wavevector.

The advantage of the latter method would be a slightly improved signal-to-noise ratio, as the smoothing window can be centered around the actual average wavevector occurring in the image instead of the ideal, equal-length, 60 degree rotated ones.

A.5 ADAPTIVE GPA

Regular GPA is limited in the wave vector deviations (with respect to the reference wave vector) it can measure, due to the limitations in spectral leakage. This is no problem when applied to atomic lattices, as the expected deviations are very small there. However, due to the moiré magnification of small lattice distortions, it does become a limiting factor when applying GPA to small twist angle moiré lattices.

To overcome this limitation, we extended the GPA algorithm to use adaptive reference wave vectors, based on the combination of two ideas and related to earlier work in laser fringe analysis [173]: First, a GPA phase calculated with respect to one reference vector can always be converted to the GPA phase with respect to another reference vector by adding a phase corresponding to the phase difference between the reference vectors. Second, a larger lock-in amplitude corresponds to a better fit between the reference vector and the data.

The adaptive GPA algorithm therefore works as follows: The spatial lock-in signal is calculated for a grid of wave vectors around a base reference vector, converting the GPA phase to reference the base reference vector every time. For each pixel, the spatial lock-in signal with the highest amplitude is selected as the final signal.

It was realized that to deduce the deformation properties, reconstruction to a globally consistent phase (requiring 2D phase unwrapping), as reported previously [141], is not strictly needed, making it possible to circumvent the problems associated with 2D phase unwrapping. Instead, the gradient of each GPA phase was calculated, requiring only local 1D phase unwrapping (i.e. assuming the derivative of the phase in both the x and y direction will never be more than π per pixel, an assumption in practice always met). Subsequently, these three GPA gradients are converted to the displacement gradient tensor (in real space coordinates), estimating the transformation via weighted least squares, using the local spatial lock-in amplitudes as weights.

As an added benefit, this entire procedure is local, i.e. not depending on pixels beyond nearest neighbors in any way except for the initial Gaussian convolution in determining the GPA. This reduces the effect of artefacts in the image to a minimum local area around each artefact (where for in the 2D phase unwrapping they have a global influence on the phases).

However, when the gradient is computed based on phase values stemming from two different GPA reference vectors, i.e. at the edge of their valid/optimal regions, artefacts appear due to their relatively large absolute error. To prevent this, the local gradient of the phase with the highest lock-in magnitude is stored alongside the lock-in signal itself in the GPA algorithm. This way, the gradient is calculated based on a single reference phase, propagating only the much smaller relative/derivative error between the two signals instead of the absolute error.

As mentioned in Chapter 6, even adaptive GPA has its limits. In particular, too large deviations from the base reference vector can not be resolved correctly, causing an erroneous, lower, extracted deviation, as is visible in the lower right of Figure 6.3**e**). As the deformation becomes too large, e.g. towards the folds in the TBG, the highest lock-in amplitude will occur at a different moiré peak or at the near-zero components of the fourier transform, causing an incorrect value to be extracted.

A.6 DECOMPOSITION OF THE DISPLACEMENT FIELD.

Kerelsky et al. [174] use the following idea to extract twist angle θ_T, strain magnitude ϵ and strain direction θ_s from reciprocal moiré lattice vectors K_{is}: These difference vectors of the constituting atomic lattices are written in terms of a rotated and a strained lattice vector each:

$$K_{is} = k_{ir} - k_{is} = R(\theta_T)k_i - S(\theta_s,\epsilon)k_i$$

where k_i are the original lattice vectors. Kerelsky et al. assume k_0 to be along the x-axis, and get around this by taking amplitudes, discarding any global rotation. Here, we do however introduce that global rotation, by a multiplication with $R(\xi)$:

$$K_{is} = (R(\theta_T) - S(\theta_s,\epsilon))\,R(\xi)k_i$$

Eihter of these expressions can, and indeed by Kerelsky et al. is, numerically fitted to the found amplitudes or k-vectors for each triangle. However, from GPA analysis we most naturally obtain a Jacobian transformation J_{ac} of the moiré k-vectors with respect to some specific set of reference vectors with predefined strain and rotations:

$$K_{is} = (J+I)K_{i0} = J_{ac}K_{i0} = J_{ac}\,(R(\theta_{T0}) - S(\theta_{s0},\epsilon_0))\,R(\xi_0)k_i := J_{ac}A_0R(\xi_0)k_i$$

Note that we can force $\epsilon_0 = 0 \rightarrow S(\theta_{s0},\epsilon_0) = I$.
This simplifies to:

$$J_{ac}A_0R(\xi_0)k_i = (R(\theta_T) - S(\theta_s,\epsilon))\,R(\xi)k_i$$

The linear transformation is uniquely described by its effect on two points in k-space, so their matrix representations should be equal:

$$\begin{aligned} J_{ac}A_0R(\xi_0) &= (R(\theta_T) - S(\theta_s,\epsilon))\,R(\xi) \\ J_{ac}A_0 &= (R(\theta_T) - S(\theta_s,\epsilon))\,R(\xi - \xi_0) \end{aligned}$$

The left hand side is a known quantity at each position, the right hand side remains to be numerically fitted or extracted. This is implemented in `pyGPA` using `scipy.optimize` and `numba` to just-in-time compile the fitting code [88, 246].

Alternatively, we could formulate a symmetric expression with two strains, but without allowing for further joint rotation of the lattices:

$$K_{i0} = (R(\theta_{T0}/2) - R(-\theta_{T0}/2))\,R(\xi_0)k_i := B_0R(\xi_0)k_i$$

$$\begin{aligned} J_{ac}B_0R(\xi_0)k_i &= (S(\theta_b,\epsilon_b)R(\theta_T/2) - S(\theta_a,\epsilon_a)R(-\theta_T/2))\,R(\xi_0)k_i \\ J_{ac}B_0 &= (S(\theta_b,\epsilon_b)R(\theta_T/2) - S(\theta_a,\epsilon_a)R(-\theta_T/2)) \end{aligned}$$

REFERENCES

88. T. A. de Jong. *pyGPA* 2021. doi:10.5281/zenodo.5589555.

135. T. A. de Jong, T. Benschop, X. Chen, et al. Imaging moiré deformation and dynamics in twisted bilayer graphene. *Nature Communications* **13,** 70. doi:10.1038/s41467-021-27646-1 (2022).

141. T. Benschop*, T. A. de Jong*, P. Stepanov*, et al. Measuring local moiré lattice heterogeneity of twisted bilayer graphene. *Physical Review Research* **3,** 013153. doi:10.1103/PhysRevResearch.3.013153 (2021).

143. M. J. Lawler, K. Fujita, J. Lee, et al. Intra-unit-cell electronic nematicity of the high-Tc copper-oxide pseudogap states. *Nature* **466,** 347–351. doi:10.1038/nature09169 (2010).

173. Q. Kemao. Two-dimensional windowed Fourier transform for fringe pattern analysis: Principles, applications and implementations. *Optics and Lasers in Engineering* **45,** 304–317. doi:10.1016/j.optlaseng.2005.10.012 (2007).

174. A. Kerelsky, L. J. McGilly, D. M. Kennes, et al. Maximized electron interactions at the magic angle in twisted bilayer graphene. *Nature* **572,** 95–100. doi:10.1038/s41586-019-1431-9 (2019).

246. S. K. Lam, A. Pitrou & S. Seibert. *Numba* in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15* (ACM Press, 2015). doi:10.1145/2833157.2833162.

B

LEEM STITCHING

To achieve stitching of images without inducing any additional deformation, a custom stitching algorithm tailored towards such LEEM data, was developed, working as follows:

To compensate sample stage inaccuracy, nearest neighbor (by sample stage coordinates) images are compared, finding their relative positions by cross-correlation. Using an iterative procedure, calculating cross correlations of overlapping areas at each step, the absolute positions of all images are found. Images are then combined in a weighted fashion, with the weight sloping to zero at the edges of each image, to smooth out any mismatch due to residual image warping. The full stitching algorithm is implemented in `Python`, available as a `Jupyter Notebook`[42].

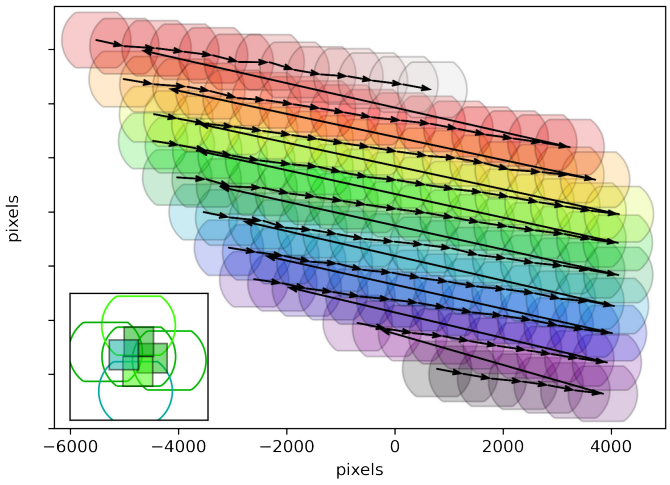

Figure B.1: *Illustration of the sample stage scanning for stitched overview images. Black arrows indicate the direction of the sample stage movement.* ***inset,*** *Illustration of the square overlapping regions of neighboring images used to determine relative positions.*

It is designed for use with ESCHER LEEM images. For those images, their positions are known approximately in terms of *stage coordinates*, i.e. the positions as reported by

This appendix has been published prior in the Supplementary material Nat. Commun. **13**, 70 (2022) [135].

the sensors in the sample stage. It should however generalize to any set of overlapping images where relative positions of the images are known in some coordinate system which can approximately be transformed to coordinates in terms of pixels by an affine transformation (rotation, translation, mirroring).

The algorithm consists of the following steps:

1. Using the stage coordinates for each image, obtain a nearest neighbour graph with the nearest `n_neighbors` neighbouring images for each image.
2. Obtain an initial guess for the transformation matrix between stage coordinates and pixel coordinates, by one of the following options:
    1. Copying a known transformation matrix from an earlier run of a comparable dataset.
    2. Manually overlaying some nearest neighbor images from the center of the dataset, either refining the estimate, or making a new estimate for an unknown dataset
3. Calculate an initial estimate of the pixel coordinates of the images by applying the corresponding transformation to the stage coordinates
4. Apply a gaussian filter with width `sigma` to the original dataset and apply a magnitude sobel filter. Optionally scale down the images by an integer factor `z` in both directions to be able to reduce `fftsize` by the same factor, without reducing the sample area compared.
5. Iterate the following steps until the calculated image positions have converged to within `sigma`:
    1. Obtain a nearest neighbour graph with per image the nearest `n_neighbors` neighbouring images from the current estimate of the pixel coordinates and calculate the difference vectors between each pair of nearest neighbours.
    2. For each pair of neighboring images:
        i. Calculate the cross-correlation between areas estimated to be in the center of the overlap of size `fftsize*fftsize` of the filtered data. If the estimated area is outside the valid area of the image defined by `mask/radius`, take an area as close to the intended area but still within the valid area as possible.
        ii. Find the location of the maximum in the cross-correlation. This corresponds to the correction to the estimate of the difference vector between the corresponding image position pair.
        iii. Calculate the weight of the match by dividing the maximum in the cross-correlation by the square root of the maximum of the auto-correlations.
    3. Compute a new estimate of the difference vectors by adding the found corrections. Reconvert to a new estimate of pixel coordinates by minimizing the squared error in the system of equations for the positions, weighing by modified weights, either:

i. $w_{mod} = w - w_{min}$ for $w > w_{min}$, $w = 0$ else, with w_{min} the maximum lower bound such that the graph of nearest neighbours with non-zero weights is still connected

ii. Only use the 'maximum spanning tree' of weights, i.e. minus the minimum spanning tree of minus the weights, such that only the n best matches are used.

6. (Optional) Refine the estimate of the transformation matrix, using all estimated difference vectors with a weight better than $w_{min\ est}$ and restart from step 3.

7. Repeat step 4. and 5. until `sigma` is satisfactory small. Optionally repeat a final time with the original data if the signal to noise of the original data permits.

8. Select only the images for stitching where the average of the used weights (i.e. where $w > w_{min}$) is larger than q_{thresh} for an appropriate value of q_{thresh}.

9. (Optional) For those images, match the intensities by calculating the intensity ratios between the overlap areas of size `fftsize*fftsize` and perform a global optimization.

10. Define a weighting mask, 1 in the center and sloping linearly to zero at the edges of the valid region, over a width of `bandwidth` pixels, as illustrated in Figure B.2.

11. Per block of output `blocksize*blocksize`, select all images that have overlap with the particular output block, multiply each by the weighting mask and shift each image appropriately. Divide by an equivalently shifted stack of weighting masks. As such information at the center of images gets prioritized, and transitions get smoothed.

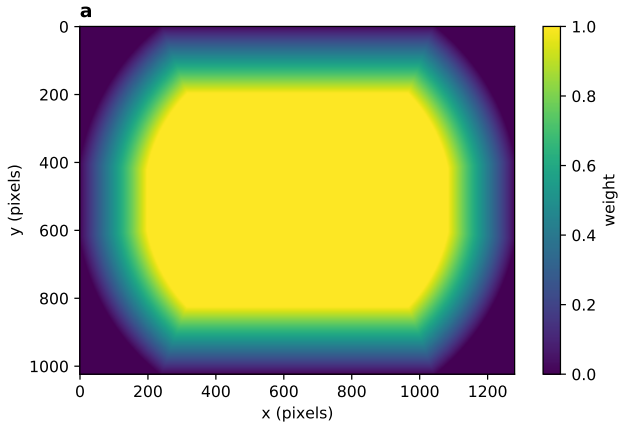

Figure B.2: a, *Weight mask used to merge images. A linear slope of the weight towards the edges of the round microchannel plate detector is used to smoothly merge images.*

B.1 CONSIDERATIONS

For square grids with a decent amount of overlap, it makes sense to put `n_neighbors` to 5 (including the image itself), however, for larger overlaps or datasets where an extra dimension is available (such as landing energy), it can be appropriate to increase the number of nearest neighbors to which each image is matched.

Parameters and intermediate results of the iteration are saved in an `xarray` and saved to disk for reproducibility.

B.2 PARALLELIZATION

Using `dask`, the following steps are parallelized:

- step 5B , where each pair of images can be treated independently. In practice parallelization is performed over blocks of subsequent images with their nearest neighbours. This could be improved upon in two ways: firstly by treating each pair only once, and secondly by making a nicer selection of blocks of images located close to each other in the nearest neighbor graph. This would most likely require another (smarter) data structure than the nearest neighbour indexing matrix used now.
- Step 6 is quite analogous to 5B and is parallelized similarly.
- Step 11 is parallelized on a per block basis. To optimize memory usage, results are directly streamed to a `zarr` array on disk.
- The minimizations are parallelized by `scipy` natively.

REFERENCES

42. T. A. de Jong. *Quantitative Data Analysis for spectroscopic LEEM* version v0.2.0. 2021. doi:10.5281/zenodo.3539538.

135. T. A. de Jong, T. Benschop, X. Chen, et al. Imaging moiré deformation and dynamics in twisted bilayer graphene. *Nature Communications* **13,** 70. doi:10.1038/s41467-021-27646-1 (2022).

C

ADDITIONAL DATA OF THE TWISTED BILAYER GRAPHENE SAMPLE

Parts of this appendix have been published in the Supplementary materials of T. A. de Jong, T. Benschop, X. Chen, E. E. Krasovskii, M. J. A. de Dood, R. M. Tromp, M. P. Allan and S. J. van der Molen, Nat. Commun. **13**, 70 (2022) [135]. Xingchen Chen performed the AFM measurements in Section C.4.

C.1 ADDITIONAL LEEM IMAGES/CROPS

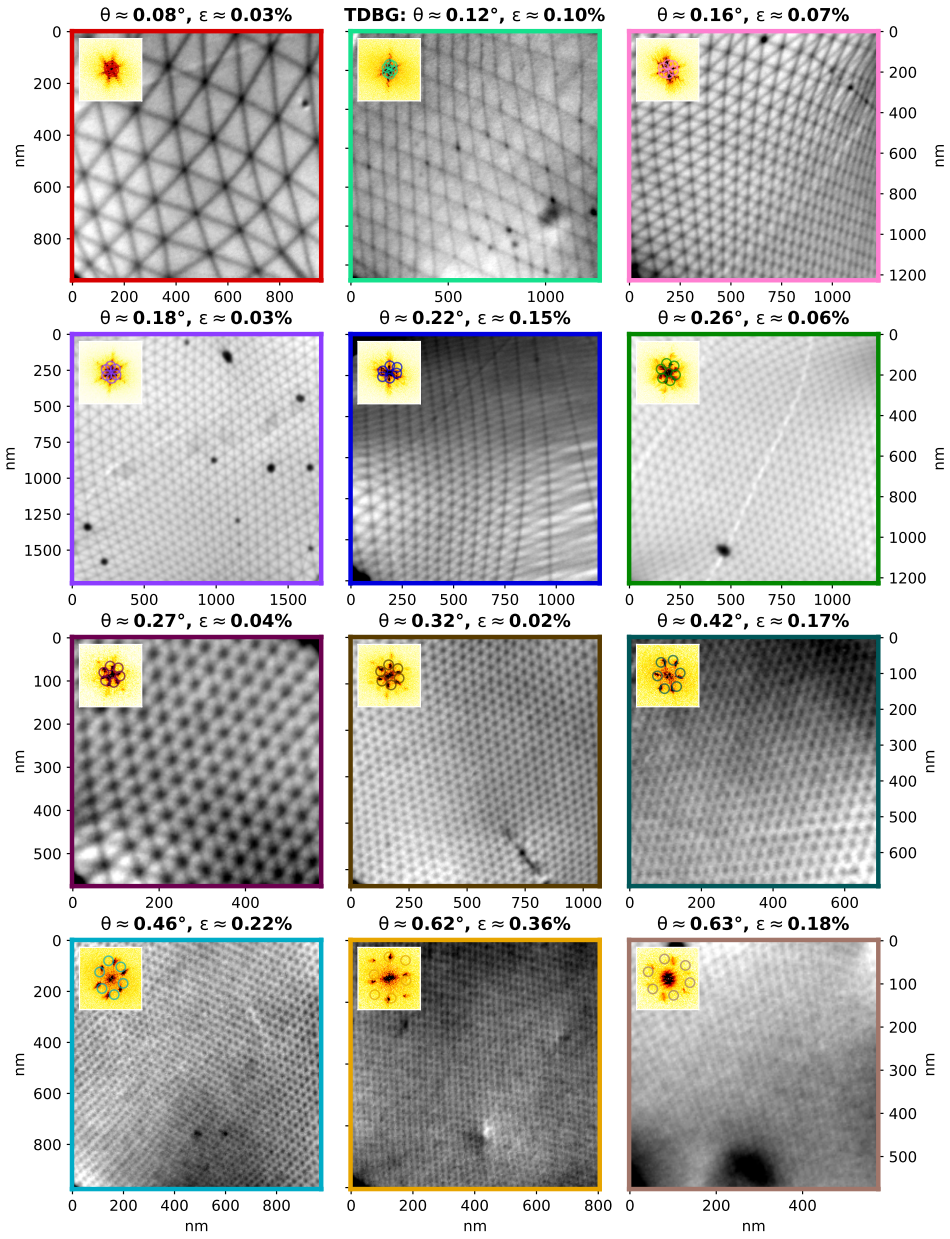

Figure C.1: *A wider range of images found in the sample from Chapter 6, as used to determine the histograms of twist angles and strain in Figure 6.3. Insets show FFT's with the detected moiré peaks*

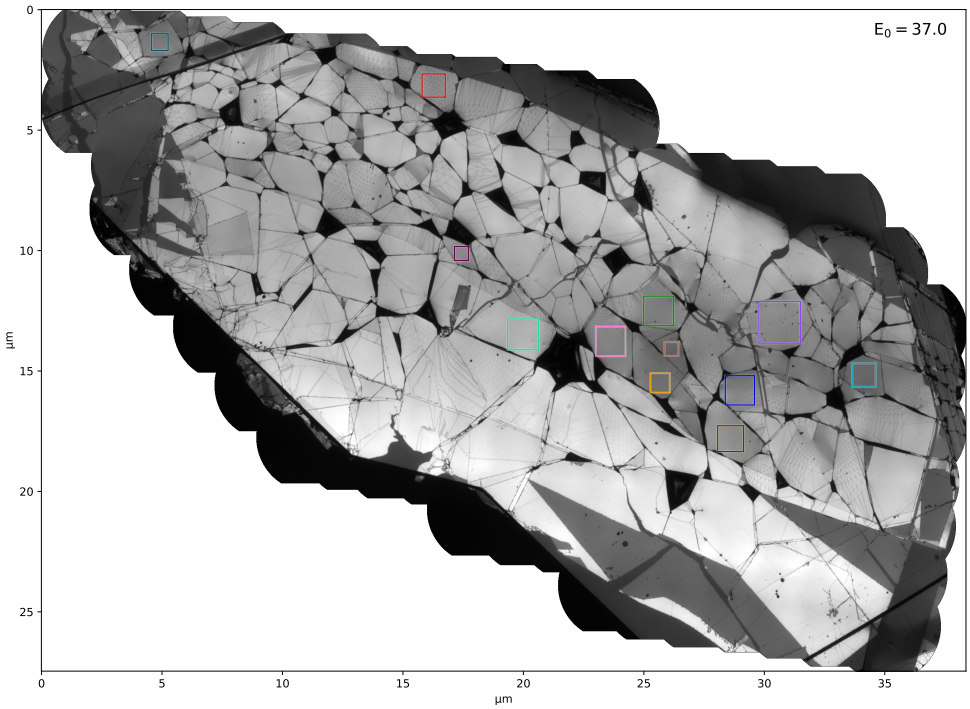

Figure C.2: *Locations of the crops in Figure C.1 indicated in the full overview (data is a super set of the data in Figure 6.1* ***e*** *in Chapter 6).*

C.2 ADDITIONAL FIGURES ON DYNAMICS

Supplementary Video 1, the full movie showing the dynamics of a larger Field of View compared to Figure 6.5 in Chapter 6, in real space LEEM data, difference data and GPA-extracted displacement field is available in the Supplementary information of the published paper, Ref. [135] at https://doi.org/10.1038/s41467-021-27646-1.

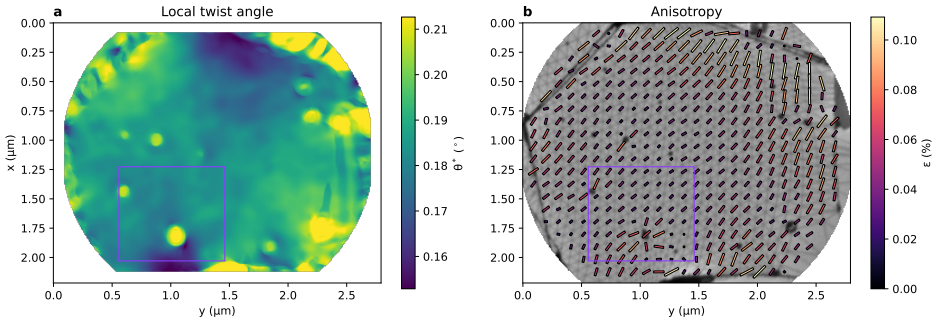

Figure C.3: a, *Local twist angle as extracted with GPA of the area imaged in Supplementary Video 1.* ***b,*** *Local strain magnitude and direction as extracted with GPA of the same area. Purple rectangles indicate the area depicted in Figure 6.5 in Chapter 6.*

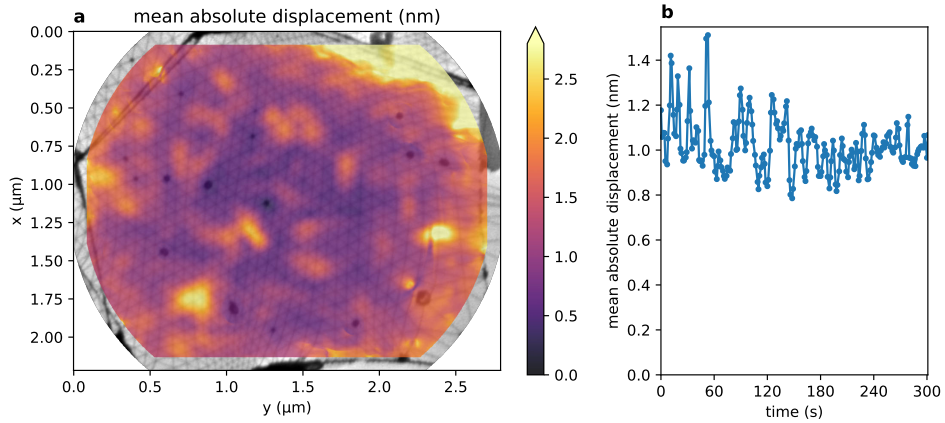

Figure C.4: a, *(Temporal) Mean absolute displacement from mean position during Supplementary Video 1.* ***b,*** *Spatial mean absolute displacement from mean position during Supplementary Video 1 as a function of time for a center area.*

C.3 ADDITIONAL DISLOCATIONS

More dislocations than single one shown in Figure 6.4 in Chapter 6 were found on the same sample. One additional dislocation in a low twist angle area is shown in Figure C.5**a** together with the GPA phases, and next to a larger area around the dislocation in Figure 6.4. More dislocations in less clean areas of the sample are shown in Figure C.6. The original dislocation had moved when re-examining the sample a few days later. The dislocation before and after movement, together with a reconstruction based on the decomposition of the displacement field as described in Appendix A.6 is shown in Figure C.7.

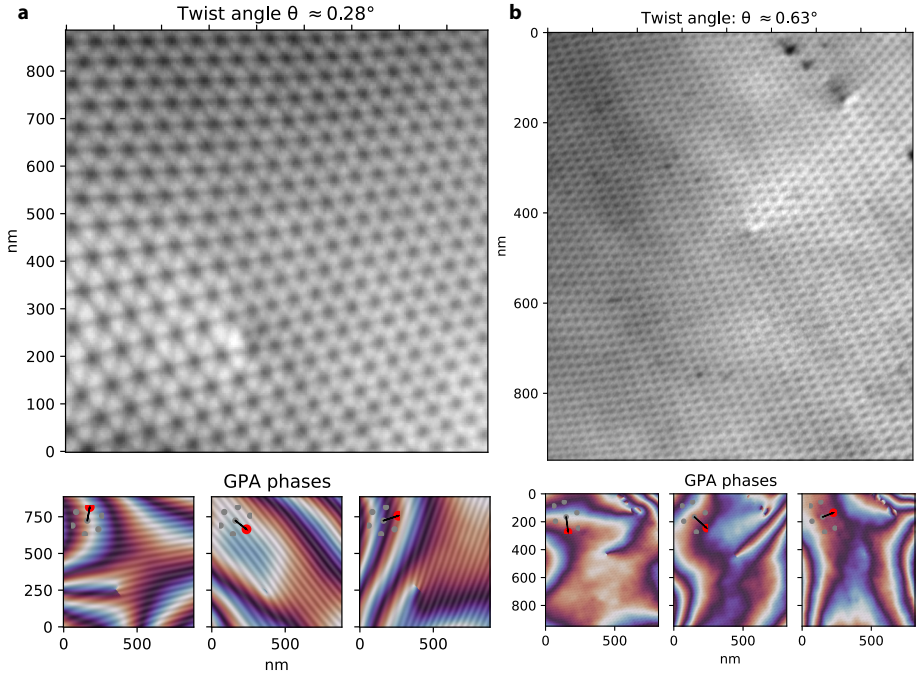

Figure C.5: a, *Additional edge dislocation found on the sample at a lower twist angle.* ***b,*** *Larger area around edge dislocation in Figure 6.4* ***f*** *in Chapter 6. In both case GPA phases are also displayed.*

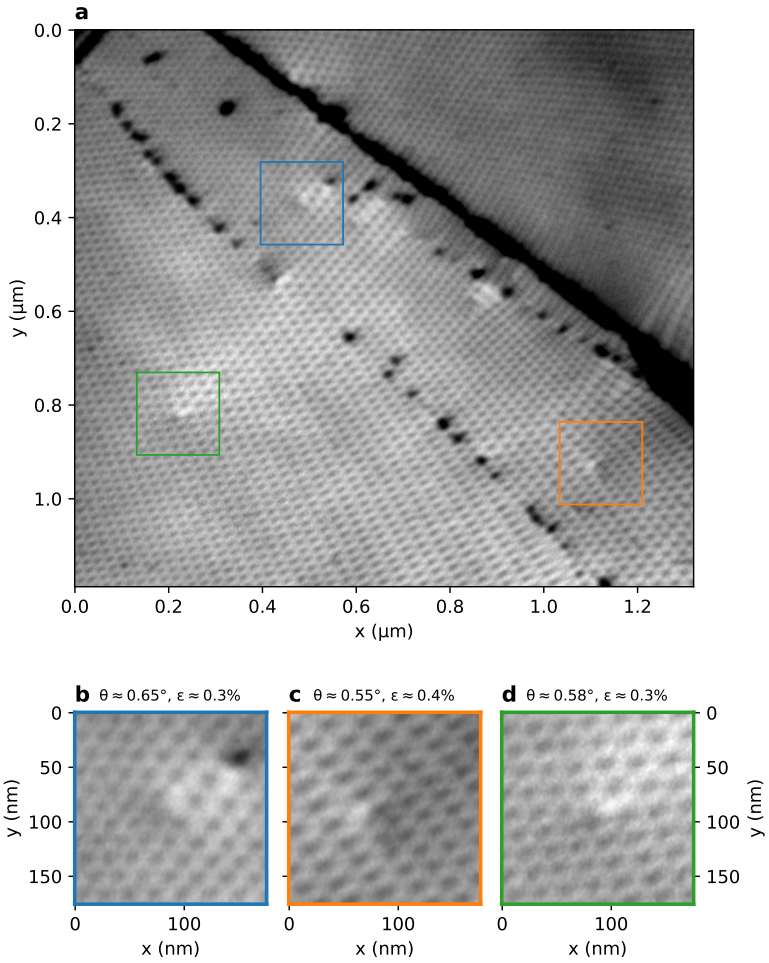

Figure C.6: *More dislocations in the vicinity of the dislocation shown in Figure C.5* ***b****.* ***d,*** *corresponds to the dislocation there. θ as extracted from the shown area here is a bit lower as unit cell area tends to be a bit larger near the dislocation.*

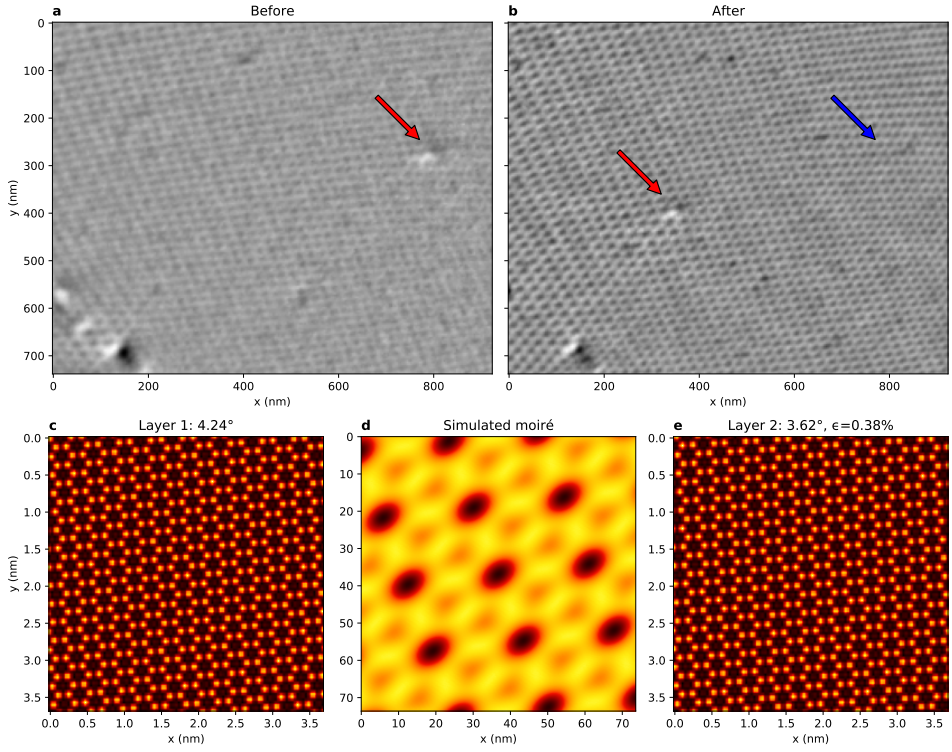

Figure C.7: Movement of dislocation. ***a,*** *Dislocation in its original location, indicated by red arrow.* ***b,*** *Image of the same area as in* ***a,****, but imaged two days later. The dislocation has moved, as indicated by the red arrow. The former location is indicated with a blue arrow.* ***c-e,*** *Rendering of the individual atomic lattices and the resulting moiré lattice from the extracted lattice parameters, showing the atomic lattice directions.*

C.4 AFM COMPARISON

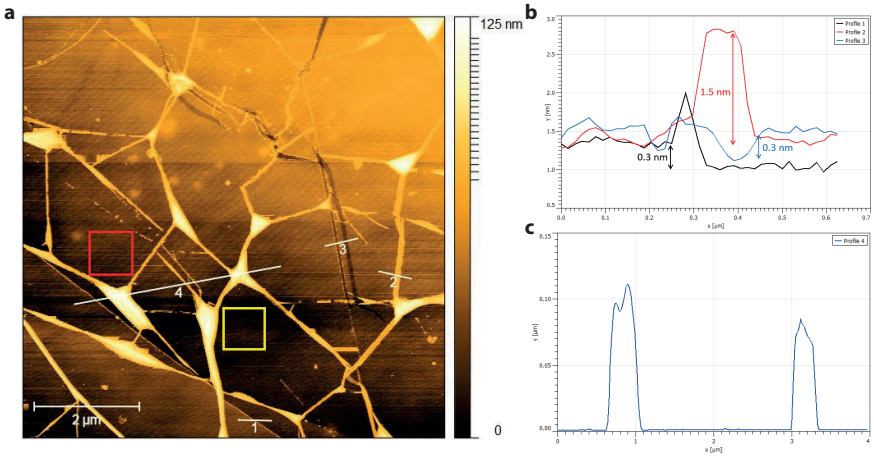

Figure C.8: ***a,*** *Atomic Force Microscopy overview of sample area. Locations of line profiles and detailed topographies in Figure C.9 are indicated.* ***b,c,*** *Line cuts along the cuts indicated in* ***a****.*

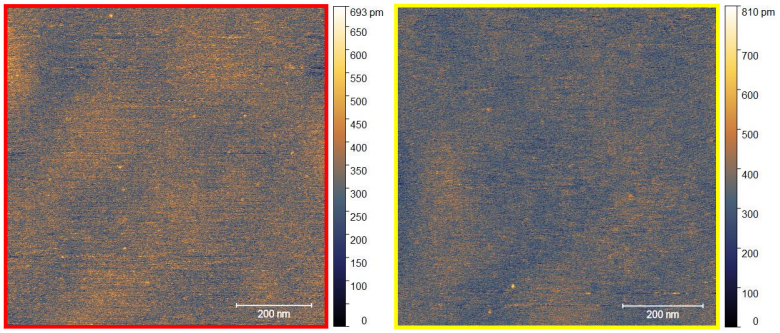

Figure C.9: ***a,*** *Atomic Force Microscopy of the dislocation area in Figure C.5****a****. Area is indicated in red in Figure C.8****a****.* ***b,*** *Atomic Force Microscopy of the dislocation area in Figure C.5****b****. Area is indicated in yellow in Figure C.8****a****.*

To further characterize the surface properties of the sample, an AFM (JPK, NanoWizard 3) measurement was performed in AC tapping mode following the LEEM measurements. Predominantly, the results show a very flat and clean graphene surface between folds, indicating annealing at 500 °C in UHV had successfully removed the polymer residue left on the surface.

In profile 1, the terrace height sees a difference of 0.3 nm, demonstrating the graphene layer count goes down by one at this location. This corresponds to the layer counts extracted from the LEEM spectra.

Profile 2 to 4 shows 3 different kinds of defects in the bilayer graphene region. The ridge at location 2 seems to be a neat folding of both the bilayer graphene flake (1.5 nm in

height, four layers of graphene), whereas profile 4 shows wrinkles that are up to 120 nm tall. This is also reflected by the distinct patterns in the LEEM bright field overview image, respectively. While the wrinkles merely appear black, the ridge resembles more like a unique layer count domain. Profile 3 shows two tears within the one layer of graphene, corresponding nicely to the defect region observed in LEEM where monolayer graphene shines through.

The zoomed-in small scale measurements marked by the red and yellow box shows the topography on top of two dislocations observed in LEEM. As shown in Figure C.9, no distinctive feature was observed at either dislocation. The topography, however, shows an exceptionally flat surface with a height variation (peak-to-peak) of less than 1 nm.

C.5 SAMPLE HEATING

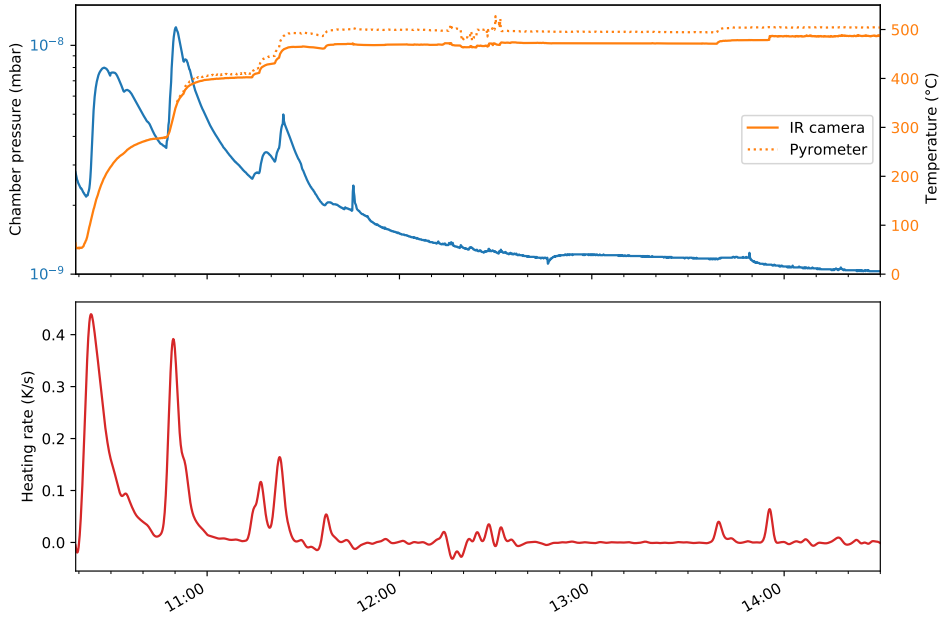

Figure C.10: *Sample temperature, heating rate and chamber pressure as measured by pyrometer and IR camera during initial heating.*

D

ADDITIONAL TaS_2 FIGURES

This appendix contains extra figures with spectra and other information supporting the work in Chapter 8.

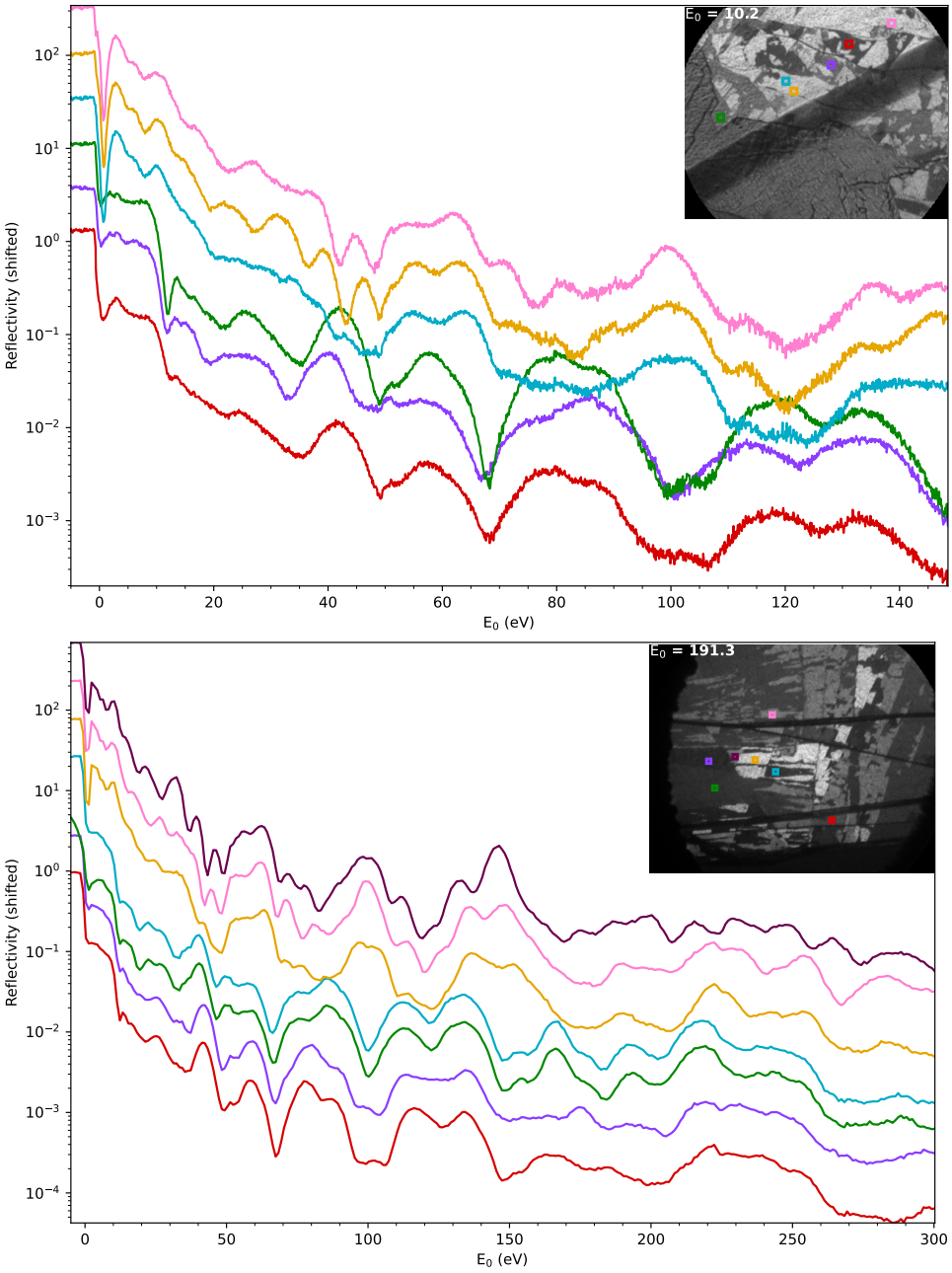

Figure D.1: More BF-LEEM spectra of mixed polytypes. *Curves are offset for clarity, with 2H-like curves below 3 1T-like curves in each panel. Data in the lower panel and in Figure D.2 was taken after the sample was kept in vacuum for 80 days.*

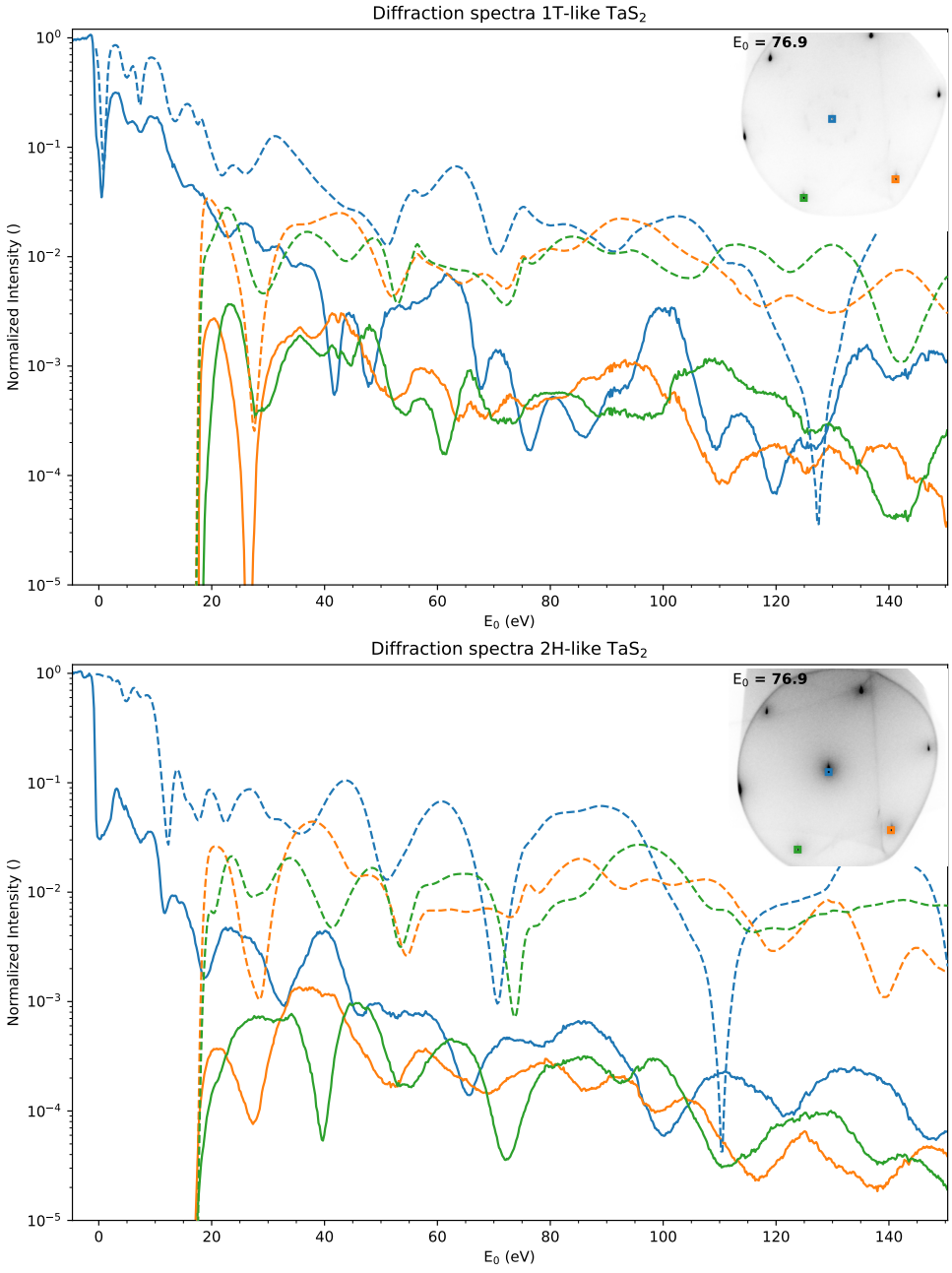

Figure D.2: ***(top),*** *Diffraction spectrum of an 1T-like polytype, most similar to the brown curve in Figure 8.7, i.e. fourth from the top in that figure and not the most similar to the pristine 1T.* ***(bottom),*** *Diffraction spectrum of 2H-like polytype, most similar to the green and turquoise curves in tyhe lower panel of Figure D.1. Dashed curves are corresponding ab-initio calculations for the bulk polytypes without taking into account the CDW.*

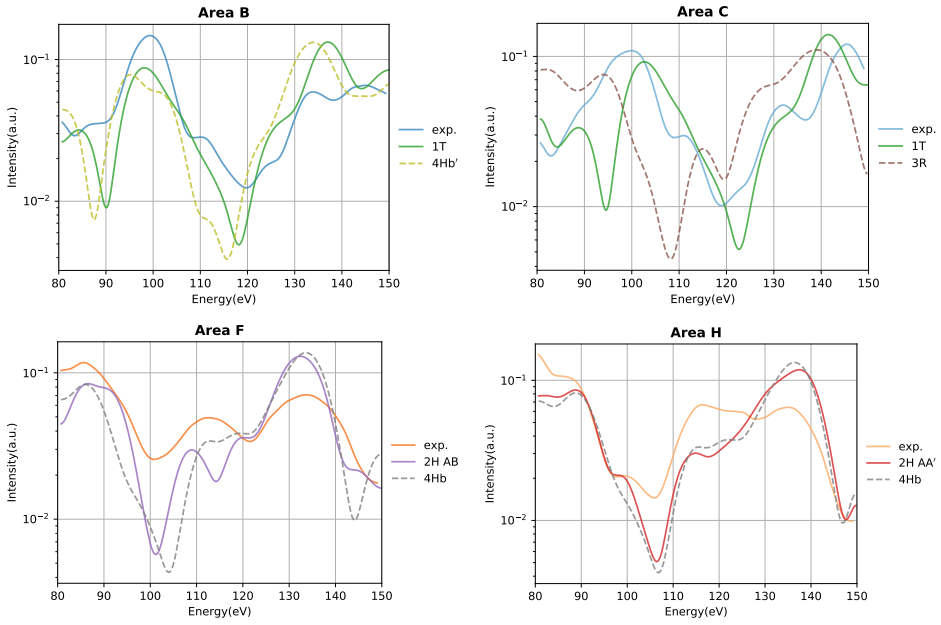

Figure D.3: TensorLEED calculations for selected data from Figure 8.7. *Taken from Ref. [232]. For each area the closest matching structure and second closest match (dashed) are shown. Here,* ***4Hb*** *denotes alternating Oc and Pr layers, terminating in a Pr layer (i.e. a 2H-like layer on top),* ***4Hb'*** *the same, but terminating in a Oc layer,* ***2H AB*** *denotes the* $2H^{\triangle}$ *polytype, and* ***2H AA'*** *corresponds to* $2H^{\hexagon}$*.*

B: *corresponds to ocre spectrum in Figure 8.7.*

C: *dark brown spectrum (fourth from top).*

F: *mint green spectrum (third from bottom).*

H: *bright red spectrum, (bottom spectrum).*

Although the match between experiment and calculations is far from perfect (best Pendry R-factor of 0.31), this comparison does support the hypothesis of 1T-like terminated and 2H-like terminated polytype structures as discussed in the main text.

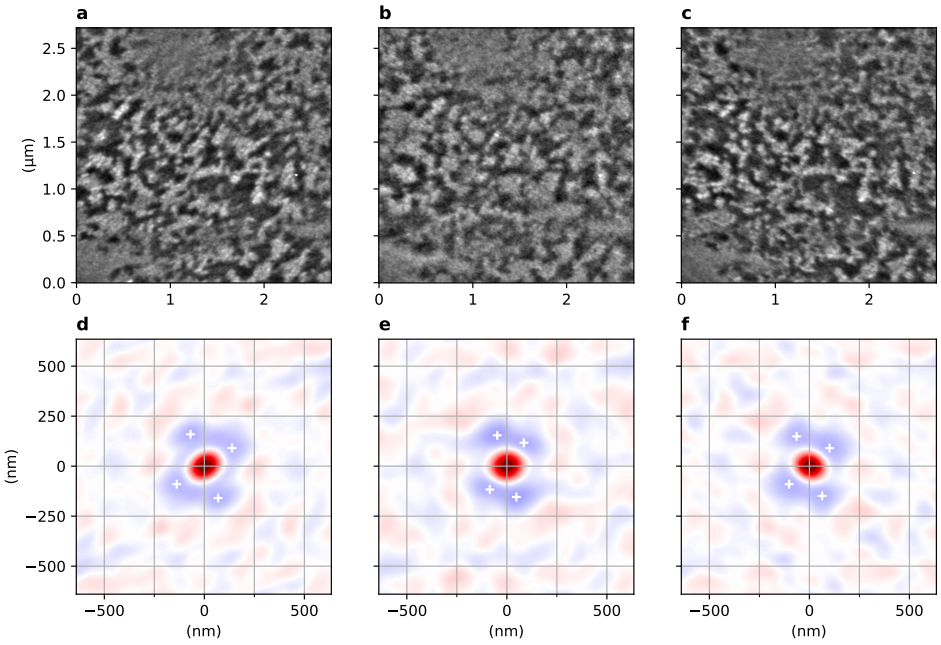

Figure D.4: Auto-correlation of DF-LEEM images of the tC-CDW. a-c, *Three DF-LEEM images of the same area using different CDW spots.* ***d-f,*** *Auto-correlations of these images. From the minima in the cross-correlations, we extract a characteristic domain size between 135 and 175 nm.*

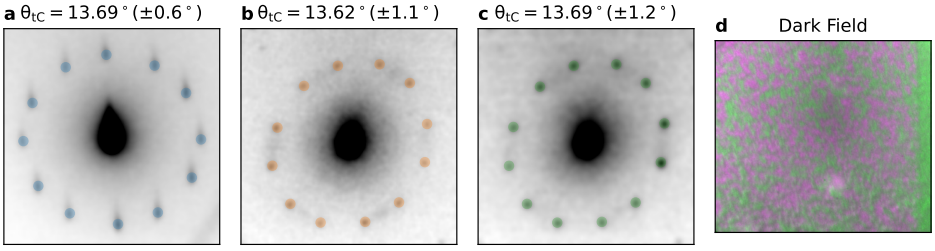

Figure D.5: Comparison with data measured by Philip Schädlich at Chemnitz. a, *Same as Figure 8.9* ***e****.* ***b,c,*** *Diffraction images taken in Chemnitz with the detected peaks indicated.* ***d,*** *Composite Dark Field image taken in the same area as* ***b,c****. The indicated margin on the angles is the standard deviation of the six angles within a single diffraction image. The fact that the data from independently prepared samples in Chemnitz give such a similar mean angle suggests the error on the mean angle is however much lower than this standard deviation, as averaging over all six cancels out most diffraction pattern distortions.*

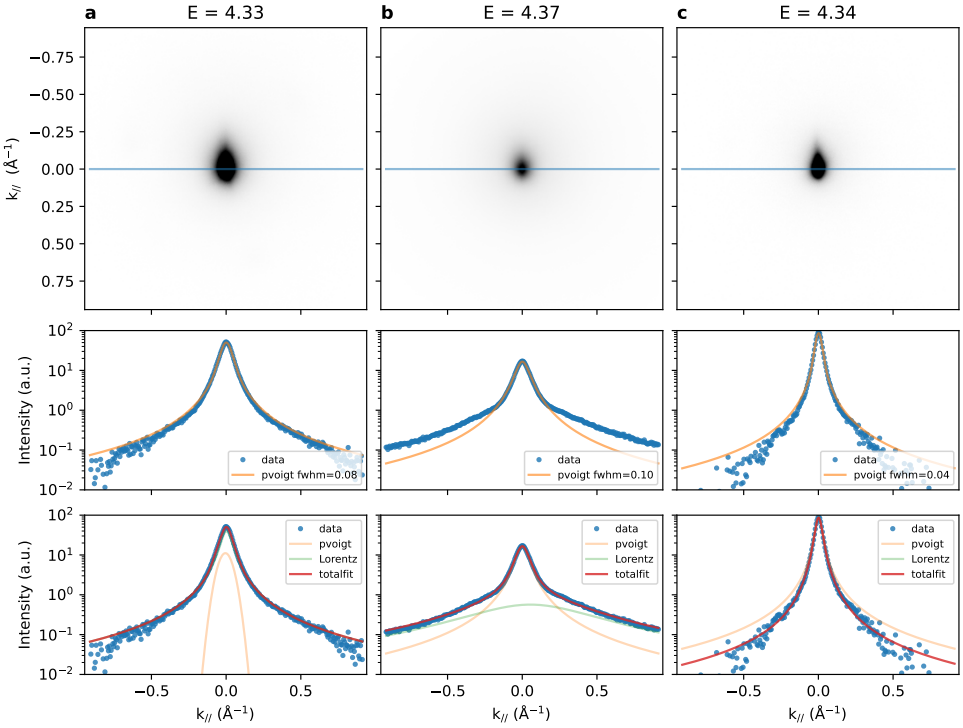

Figure D.6: 2H and 1T Diffraction profile fits. ***a-c,*** *Diffraction images, corresponding to Figure 8.8* ***a-c****, but at the landing energy of the dark field images in Figure 8.10. For the 1T-like areas in a and c, a single pseudovoigt peak is enough to fit the profile (center row). For the 2H-like area in b however, a broad background intensity is present, which can be sufficiently fit by a broad Lorentzian background (bottom row). This broad background causes 2H-like areas to appear bright in the Dark Field images, as the total intensity of this background in the aperture is larger than the total intensity of the CDW diffraction spot in the 1T-like areas.*